"All-in-One is All You Need."

ALL-IN-ONE

#1 CCIE LAB STUDY GUIDE

Cisco® CCIE™ LAB Study Guide

SECOND EDITION

Completely updated to cover all new material on Cisco CCIE Lab exam

Contains technology overviews and over 115 hands-on network-based scenarios

Ideal as an exam preparation tool and an on-the-job reference manual

All configuration examples from the book on CD

STEPHEN HUTNIK &
MICHAEL SATTERLEE, CCIE™, #3080

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

All−in−One Cisco CCIE Lab Study Guide, Second Edition

Stephen Hutnik &
Michael Satterlee

Osborne/**McGraw−Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

1234567890 DOC DOC 01987654321

Book p/n 0−07−212759−7 and CD p/n 0−07−212758−9
parts of

ISBN 0−07−212760−0

Publisher
Brandon A. Nordin

Vice President & Associate Publisher
Scott Rogers

Executive Editor
Steven Elliot

Project Editor
Monika Faltiss

Acquisitions Coordinator
Alex Corona

Technical Editor
Peter Mokros

Copy Editor
Robert Campbell
Dennis Weaver

Compositor and Indexer
MacAllister Publishing Services, LLC

DEDICATION

I dedicate this book my wife MaryJo and my parents Jim and Gladys Satterlee.Their love and encouragement have enabled me to write this book. — Mike

To Robin, Bobby, Andrew, Lauren, and Jennifer for all of their love and support. — Steve

ACKNOWLEDGMENTS

Many thanks to our editor Steve Elliot.

Thanks to Monika Faltiss project editor.

We want to thank George Kovachi from Adtran for lending us an Atlas 800 ISDN switch.

ABOUT THE AUTHORS

Stephen Hutnik, CCNA, is a Senior Network Engineer at AT&T Global Network Services, where he is responsible for development, testing, and training for the Global backbone of the AT&T Network. He is also an adjunct Professor of Telecommunications at Pace University, and is the co−author of the *Cisco CCIE Lab Practice Kit.*

Michael Satterlee, CCIE, is a Senior Network Architect at AT&T Global Network Services, where he is responsible for the architecture and design of a Next Generation VPN services. He is a co−author of the *Cisco CCIE Lab Practice Kit.*

ABOUT THE TECH REVIEWERS

Pete Mokros is a Cisco− and Check Point−certified engineering professional with a Fortune 500 global technology company. Currently, his work focuses on TCP/IP and network security. He is active in many areas of the information technology field and has been involved in research projects on the Internet since 1992. He holds B.A. degrees in Computer Science and Mathematics from Macalester College.

Chapter 1: Take the Lab Once and Pass

Overview

Our goal when writing the CCIE Lab Study Guide was to provide a book that was 100 percent hands−on. Many of the Cisco−related books on the market only include parts of the routers configuration and do not provide enough information so the reader can completely build and test the configurations. We feel this book is unique. The second edition of this book includes 33 new labs, bringing the total number of labs to 119. Four completely new chapters focus on key areas such as MPLS, IPSec, Bridging, and Voice technology. We hope you enjoy reading this book as much as we enjoyed writing it.

CCIE Lab Exams

The CCIE lab exam is a challenging, hands−on assessment of your inter−networking skills. It costs $1,250 in the United States and stretches over two days. Before you can sign up for the lab exam, you must pass the CCIE qualification exam. Unlike the computer−administered exams, CCIE lab exams are only offered through Cisco locations. The exams are standardized among sites, and selecting the location is a matter of geographical preference.

CCIE Routing and Switching Lab Locations

- San Jose, California, USA
- Research Triangle Park, North Carolina, USA
- Halifax, Nova Scotia, Canada
- Chatswood, NSW, Australia
- Brussels, Belgium
- Beijing, China
- Sao Paulo, Brazil
- Bangalore, India
- Seoul, Korea
- Tokyo, Japan
- Singapore
- Johannesburg, South Africa

Each test candidate will receive his/her own rack and patch panel. You will also receive a set of Cisco documentation to use throughout the exam. You cannot bring any other notes or documentation into the exam with you.

Your first task will be to create a network to specification. This will take up all of the first day and half of the second. Halfway through the second day, while you are out of the room, the exam proctor will insert faults into your network, and you will have to find and fix them — as well as be able to document the problems and their resolutions.

There are a total of 100 possible points on the exam. To pass, you must achieve a score of 80 or better. You must achieve a passing score on each section of the exam to be allowed to progress to the next. For example, a perfect score on the first day would be 45 points. You have to earn at least 30 of them to be allowed to return for the first part of day two. Table 1−1 shows the scoring breakdown.

Table 1−1: CCIE Lab Exam

DAY	TASK	POINTS		

			TOTAL SO FAR	MINIMUM SCORE TO CONTINUE
1	build	45	45	30
2 (part I)	build	30	75	55
2 (part II)	troubleshooting	25	100	80 or better to pass

The lab starting time varies depending upon location, but will be somewhere between 8:00 A.M. and 9:00 A.M. each day and run for 7 1/2 hours. There is a half hour break for lunch. A proctor will be in the room to clarify questions and handle any emergencies that may arise, but basically you are on your own.

The failure rate for this exam is high. According to Cisco, only about 20 percent of the candidates pass it on the first attempt. On average, CCIE candidates require two to three lab exams before they earn a passing score. Think of your first time through as a learning experience, and if you manage to pass, that is a bonus. There is no limit on the number of times you can retake the exam.

As with all certification exams, lab exam content and structure are subject to change, so when you are ready to consider taking the lab exam, it's best to get the latest information from Cisco. Cisco's Web site contains specific instructions about how to prepare for each of the CCIE lab and qualification exams.

It cannot be stressed enough that you must get lots of hands−on practice if you hope to pass this exam. If you do not have equipment to practice on at work, you will have to set up a home lab or find another way to gain access to the equipment.

Format of the Book

This book is geared toward a wide audience. The technology introductions at the beginning of each chapter will provide the user with a detailed explanation of networking protocols and technologies.

Students studying for their CCIE will find the 119 sample labs and over 350 router configurations a valuable study reference. Those people that are fortunate enough to have access to several routers will be able to actually go through each lab step by step.

All of the 119 labs in this book are self−contained with fully debugged configurations and step−by−step instructions. All of the labs were tested, and the output shown in each lab was actually taken from the working configurations.

Each lab was created using the least number of routers possible, so the reader who wishes to actually go through each lab can do so with the least amount of equipment.

Chapter Format

The format of all the chapters in this book are similar.

1. Each chapter starts with an introduction section, outlining the topic to be discussed.
2. A detailed technology overview of the topic is then presented. Readers should read through this overview to make sure that they have a thorough understanding of the topic.
3. All of the commands that are discussed in the chapter are then listed, and their use and function are then discussed.
4. The chapters labs are then presented.
5. Troubleshooting information on the topic is then presented.
6. A conclusion wraps up the chapter.

Throughout each chapter you will find the following features:

Note Notes highlight important information.

Tip Tips offer guidance to help the reader better understand the material and succeed on the exam.

Lab Format

Each of the 119 labs have the same format. All of the labs are numbered in order, starting at Lab #1 and ending at Lab #119. The format is as follows:

1. The list of equipment needed to perform the lab is reviewed.
2. The lab objectives and configuration objectives are discussed in the Configuration Overview section.
3. Any notes related to the configuration are listed.
4. A detailed drawing of the lab is shown.
5. Configurations for all routers in the lab are listed. The configurations are taken directly from the routers that were used to perform the lab while writing this book. Any interfaces that were not used to perform the lab are not shown in the configuration listing (to conserve space).
6. Step−by−step monitoring and testing instructions are given to verify that the lab setup is functioning properly.

CD−ROM

The included CD−ROM contains all the configurations that are presented in this book. Readers can cut and paste the configurations into their setups, thereby saving time.

The file−naming convention for the files on the CD−ROM includes the lab number and the router name used in the lab. For example, the file LAB75A.txt contains the configuration for RouterA in Lab #75.

Chapter 2: Terminal Servers

Overview

Topics Covered in This Chapter

- Out−of−band network management
- Basic terminal server configuration
- Configuring IP host tables
- Absolute versus relative line numbers
- Changing the default escape character
- Troubleshooting a terminal server

Introduction

This chapter explores configuring and troubleshooting one of Cisco's access services, terminal server. Cisco routers provide four "access services" that are supported on the Cisco 2500 router series: models 2509, 2510, 2511, and 2512 (remote node service, terminal services, protocol translation, and asynchronous remote access routing). This chapter will discuss terminal services. The remaining access services will be covered later in the book.

The terminal server used for this configuring is a 2511RJ, which provides 16 asynchronous serial ports. The terminal server provides access to all of our test routers via reverse telnet. Reverse telnet is the process of using telnet to make connection out an asynchronous port.

The test routers console port will be connected directly to one of the 16 asynchronous interfaces on the 2511RJ, using a standard Cisco console rolled cable. The test router will be accessed using a reverse telnet connection. To make a reverse telnet connection, you telnet to any active IP address on the box followed by the 200x, where x is the port number that you wish to access (**Telnet 1.1.1.1 2001**).

Out−of−Band Network Management

Figure 2−1 depicts a remote site that does not use a terminal server to access the routers on the network. Therefore, each router requires a separate modem connection in order to manage the device out−of−band.

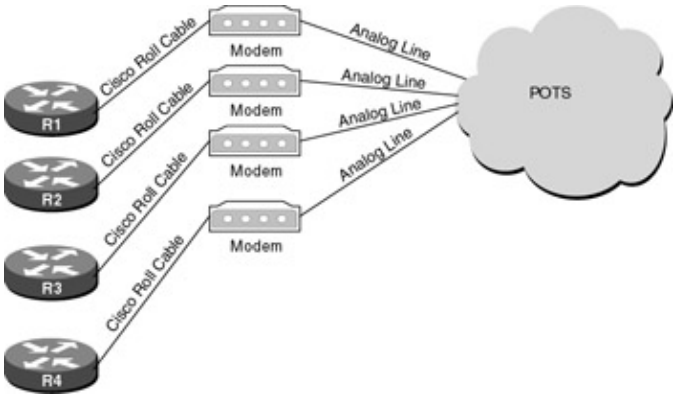


Figure 2−1: Out−of−band network management without a terminal server

In Figure 2−2, all devices are accessed through the terminal server. Notice that only one analog line and one modem are needed to manage all of the local devices. Not only does this simplify network management, it also greatly reduces the cost.

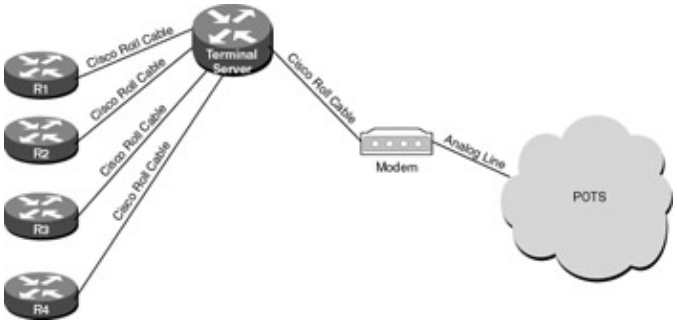


Figure 2−2: Out−of−band network management using a terminal server

Commands Discussed in This Chapter

- **ip host name** [*tcp−port−number*] *address1*
- **no exec**
- **transport input {all}**

Definitions

ip host: This global configuration command is used to define a static host name−to−address mapping in the router's host cache.

no exec: This interface configuration command is used to disable EXEC processing for the specified line.

transport input: This interface configuration command is used to specify an incoming transport protocol. Cisco routers do not accept incoming network connections to asynchronous ports (TTY lines) by default. You have to specify an incoming transport protocol, or specify **transport input all**, before the line will accept incoming connections.

Lab #1: Basic Terminal Server Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, one of which is a terminal server (2511)
- A PC running a terminal emulation program
- One Cisco rolled cable

Connecting the Terminal Server

Connect R1's console cable to the asyc port 1 of the terminal server using a standard Cisco rolled cable.

Basic Terminal Server Configuration

The terminal server is very simple to set up and requires minimal configuration. In the sample configuration shown in Figure 2−3, notice that the only commands used are **transport input all** and **no exec**. A loopback interface is used, since it provides a reliable interface for reverse telneting because it is always up; however, any active interface can be used.

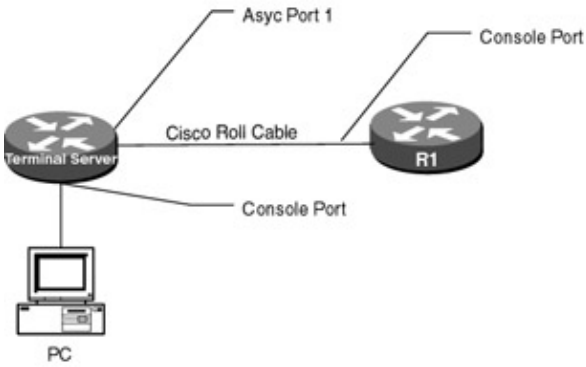


Figure 2−3: Lab #1 basic terminal server configuration
The command **transport input all** specifies the input transport protocol. By default on IOS 11.1 and later, the transport input is set to **none**, but prior to 11.1 the default was **all**. If the input transport protocol is left **none**, you will receive an error stating that the connection is refused by remote host.

```
terminal_server# telnet 1.1.1.1 2001 ← (Reverse Telnet)
Trying 1.1.1.1, 2001 ...
% Connection refused by remote host
```

The command **no exec** allows only outgoing connections for the line. This prevents the terminal server from starting an EXEC process if the attached device sends out unsolicited data. By default, if the port receives unsolicited data, an EXEC process is started that makes the line unavailable. This can be monitored using the **debugging modem** command and then showing the line that is attached to the device.

```
TTY1: EXEC creation ← (Output from debug)
```

As soon as the EXEC process is created, the line becomes unavailable; the star to the left of the line number indicates this.

```
terminal_server#  show line 1
 Tty Ty
   Tx/Rx     A Modem    Roty  AccO AccI  Uses    Noise   Overruns
*  1 TTY     9600/9600  − −   −    − −   12      1127    871/2644
á (Indicates the line is active)
```

Terminal Server Configuration

```
terminal_server#
Current configuration:
!
version 11.2
service timestamps log uptime
no service udp−small−servers
no service tcp−small−servers
!
hostname terminal_server
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface Ethernet0
 ip address 2.2.2.2 255.255.255.0
!
interface Serial0
  no ip address
!
no ip classless
!
line con 0
line 1 16
 no exec ← Disables EXEC processing
```

```
 transport input all ← Specifies the input transport protocol
line aux 0
line vty 0 4
 login
!
end
```

Connecting to a Port

To connect to a device attached to a terminal server, simply telnet to any active IP address on the box followed by 20xx, where xx is the port number you are connecting to. What follows is an example on how you would reverse−telnet to port 1 of the terminal server.

```
Telnet 1.1.1.1  2001 ← (01 is the port number)
         á (IP Address of the Loopback interface)
```

Mapping a Host Name to an IP Address

The Cisco IOS software maintains a table of host names and their corresponding addresses. As with a DNS server, you can statically map host names to IP addresses. This is very useful and saves a lot of keystrokes when you have multiple devices connected to the terminal server.

The following global configuration command defines router1 as connecting to port 1:

```
             (Port Number)
                   ↓
IP host router1 2001 1.1.1.1
          á              á
      (Host Name)     (IP Address Loopback 0)
```

Absolute Versus Relative Line Numbers

When configuring a line, you can specify an absolute line number or a relative line number. For example, on the terminal server used in Lab #1, absolute line 17 is Aux port 0. For the 16 asynchronous ports on the terminal server, the absolute and relative line numbers are the same.

```
terminal_server#show users all
                                  Line      User      Host(s)     Idle Location
(Indicates an active session) →*  0 con 0             Idle        00:00:00
                                  1 tty 1                          00:00:00
                                  2 tty 2                          00:00:00
                                  3 tty 3                          00:00:00
                                  4 tty 4                          00:00:00
                                  5 tty 5                          00:00:00
                                  6 tty 6                          00:00:00
                                  7 tty 7                          00:00:00
                                  8 tty 8                          00:00:00
                                  9 tty 9                          00:00:00
                                 10 tty 10                         00:00:00
                                 11 tty 11                         00:00:00
                                 12 tty 12                         00:00:00
                                 13 tty 13                         00:00:00
                                 14 tty 14                         00:00:00
                                 15 tty 15                         00:00:00
                                 16 tty 16                         00:00:00
      (Absolute Line Number) → 17 aux 0   ← (Relative number)  00:00:00
                                 18 vty 0                          00:00:00
```

Exiting a Reverse Telnet Session

Once you have configured your terminal server and made a reverse telnet connection to the attached device, how do you get back to the terminal server? Well, the answer is, you type the escape character sequence, which by default is CTRL−SHIFT−6, followed by X (the combination written as CTRL^X).

The escape character can be changed to any ASCII value with the **terminal escape−character** command. Each line on the terminal server can have a different escape character; for example, you can arrange that when you telnet to the router, the escape character will be CTRL−W and when you are connected to the console port, the escape character will be the default (CTRL−SHIFT−6).

The following configuration sets the escape character on VTY 0 to CTRL−W and the escape character on the console port to the default (CTRL−SHIFT−6).

```
terminal_server#show run

Current configuration:
!
version 11.2

no service udp−small−servers
no service tcp−small−servers
!
hostname terminal_server
!
enable password cisco
!
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface Ethernet0
 ip address 2.2.2.2 255.255.255.0
!
interface Serial0
 no ip address
!
no ip classless
!
line con 0

line 1 16
 no exec
 transport input all
 line aux 0
line vty 0
 password cisco
 login
 escape−character 23 ← (Escape Character Ctrl−W)
line vty 1 4
 no login
!
end
```

Figure 2−4 is a good example of when it would be necessary to use multiple escape characters on a router. User Mahar connected to the console port of RouterA wishes to telnet to RouterB and then reverse−telnet to RouterC, which is connected to asynchronous port 1 of RouterB. The problem arises when user Mahar wishes to break out of the reverse telnet; if the default escape character sequence, CTRL^X, is used, the session will be returned to RouterA and not RouterB. This is because RouterA responds to the same default break character. The solution is to configure the escape character on the VTY interface on RouterB to something different.

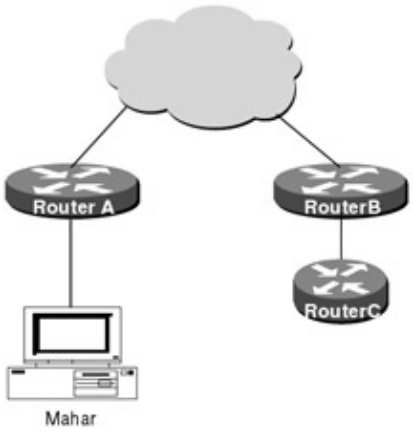


Figure 2−4: Changing the default escape character

Troubleshooting

Displaying Active Sessions

The escape sequence breaks you out of the telnet session; however, that session will still remain open. To display all open connections, use the **show sessions;** what follows is the output from the command.

The asterisk (*) indicates the current terminal session; if you were to hit the ENTER key, you would be connected to RouterA. If you wanted to reestablish the connection to RouterB, you would simply type **2**, which is the connection number.

```
terminal_server#show sessions

Conn    Host         Address      Byte   Idle    Conn Name
*  1    routera      1.1.1.1      0       0      routera
   2    routerb      1.1.1.1      0       0      routerb
   3    routerc      1.1.1.1      0       0      routerc
```

The following list describes other terms used in the example:

Conn: The connection number used to reference the session; for example, if you wished to reestablish the session to RouterC, you would type **3** at the command line.

Host: The remote host to which the router is connected through a telnet session.

Address: The IP address of the remote host; in our case, since we are reverse−telneting, this is the IP address of our loopback interface.

Byte: The number of unread bytes displayed for the user to receive.

Idle: The interval (in minutes) since data was last sent on the line.

Conn Name: The assigned name of the connection.

Switching Between Sessions

Several concurrent sessions can be open at once. To switch between sessions by escaping one session and resuming a previously opened session, perform the following:

Step 1: Escape out of the current session by pressing the escape sequence.

Step 2: Issue the **show sessions** command. All open sessions associated with the current terminal line are displayed.

Step 3: Enter the session number to make the connection. The following example resumes connection 2.

```
terminal_server# 2
[Resuming connection 2 to routerb ... ]
```

Disconnecting a Session

To disconnect an active reverse telnet session, use the **disconnect** command. To disconnect a session, perform the following steps:

Step1: Issue **show sessions** command. All open sessions associated with the current terminal line are displayed.

Step2: Issue the command **disconnect x** where x is the session number that you wish to terminate.

The following example disconnects session 2.

```
terminal_server# disconnect 2
Closing connection to routerb [confirm]
```

Clearing a Line

At times it may become necessary to return a terminal line to idle state; to do this, use the command **clear line**. The example that follows will clear line 1.

```
terminal_server# clear line 1
[confirm]
 [OK]
```

The following is the output from the **debug modem** command after the **clear line** command was issued. Notice that the carrier is dropped and the line is now idle. The **debug modem** command shows the modem line activity on an access server.

```
                                        terminal_server#
(TTY1 is Line 1) → TTY1: Carrier Drop
                                        TTY1: Line reset by "TTY Daemon"
                                        TTY1: Modem: READY->READY
```

Displaying the Status of a Line

To show the status of any line, use the **show line x** command, where x is the number of the line that you wish to view. This command is very useful in troubleshooting a terminal server connection.

The following sample output from the **show line** command shows that line 1 is an asynchronous terminal port with a transmit and receive rate of 9600 bps. Also shown is the modem state, terminal screen width and length, capabilities, status, and much more. All significant lines are described in detail here:

```
terminal_server#show line 1

      Tty   Typ     Tx/Rx    A Modem  Roty AccO   AccI  Uses   Noise   Overruns
   *  1    TTY     9600/9600  -  -     -    -      -      13    15      0/0
Line 1, Location: "", Type: ""
Length: 24 lines, Width: 80 columns
Baud rate (TX/RX) is 9600/9600, no parity, 2 stopbits, 8 databits
Status: Ready, Connected, Active
Capabilities: EXEC Suppressed
Modem state: Ready
Special Chars: Escape  Hold  Stop  Start  Disconnect  Activation
         BREAK     none  -   -     none
Timeouts:  Idle EXEC      Idle Session    Modem Answer  Session Dispatch
```

```
           00:10:00       never                none     not set
 Idle Session Disconnect Warning
  never
Modem type is unknown.
Session limit is not set.
Time since activation: 00:00:09
Editing is enabled.
History is enabled, history size is 10.
DNS resolution in show commands is enabled
Full user help is disabled
Allowed transports are pad v120 telnet rlogin.  Preferred is telnet.
No output characters are padded
No special data dispatching characters
Modem hardware state: CTS* DSR*  DTR RTS
```

Table 2−1 shows several different fields and their descriptions.

Table 2−1: Show Line Field Names and Their Descriptions

Field	Description
Tty	Line number. In this case, 1.
Typ	Type of line. In this case, a TTY — asynchronous terminal port, which is active, as noted by the asterisk.
Tx/Rx	Transmit rate/receive rate of the line, which is set to 9600/9600 .
A	Indicates whether autobaud has been configured for the line. The hyphen indicates that it has not been configured.
Modem	Type of modem signals configured for the line; in this case, there is none.
Roty	Rotary group configured for the line; in our case, this is none.
AccO, AccI	Is the number of the Output or Input access list configured on the line; in our case, there is none.
Uses	Number of connections established to or from the line since the system was restarted.
Noise	Number of times noise has been detected on the line since the system restarted.
Overruns	The number of overruns or overflows that have occurred on the specified line since the system was restarted.
Status	State of the line. In our case, the line is connected and active.
Capabilities	Indicates current terminal capabilities; in our case, we are suppressing EXEC processing.
Time since activation	Time that the session has been active; our session has been active for 9 seconds.
Transport methods	Current set transport method; our transport is set to all.

Conclusion

Terminal servers permit asynchronous devices to be accessed and managed out of band.

In the past, in order to manage a group of remote devices, multiple modems and call directors were needed. Through the uses of terminal servers, multiple devices can be accessed via one dialup or IP connection, greatly reducing the complexity and cost associated with asynchronous management.

Chapter 3: ISDN

Overview

Topics Covered in This Chapter

- ISDN technology overview
- ISDN configuration and switch basics
- Backup interfaces
- Floating static routes
- Dialer profiles
- ISDN PRI configuration
- Snapshot routing
- OSPF Demand Circuit
- ISDN troubleshooting
- PPP Callback
- Dialer Watch

Introduction

ISDN is a popular technology used for high−speed switched access and dial backup applications. This chapter will examine ISDN technology and will present seven hands−on ISDN labs using Cisco routers.

ISDN Technology Overview

ISDN is a circuit−switched digital data service. ISDN was originally envisioned as a way to offer enhanced voice and data services. In recent years, ISDN has been primarily used in three roles:

1. High−speed Internet access for home users — A standard ISDN BRI circuit can achieve a speed of 128 Kbps. Figure 3−1 shows three different users dialed into an ISDN network and being connected to the Internet through an access server. The first user is dialed in via a V.90 analog modem. The ability to place a call from an analog phone circuit to a digital ISDN circuit is an attractive advantage of ISDN. The second user is dialed into the access server with an ISDN BRI circuit at a speed of 64K. The third user is dialed into the access server at a speed of 128K. This user is connected to two 64K channels and his ISDN device is able to combine the two 64K circuits into a single 128K circuit referred to as a multilink bundle.

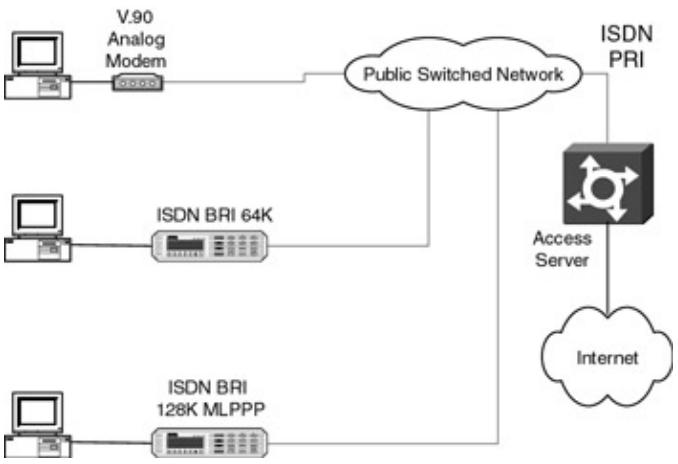


Figure 3−1: ISDN Dial Backup example

2. Terminating circuits for large−scale dial access servers — An ISDN PRI circuit is the standard method used for terminating many dial users onto an access server. In Figure 3−1, we see an ISDN PRI circuit connecting the public network to the access server. An ISDN PRI circuit is a high−speed ISDN circuit that can accommodate up to 23 simultaneous users. Having an ISDN PRI circuit connected to the access server is more flexible and cost effective than having 23 individual phone lines coming into the access server.
3. Dial backup — Both ISDN BRI and ISDN PRI circuits are used to provide a backup data path between routers. In Figure 3−2, we see four users connected to a frame relay cloud. Three of the users have Cisco 3600 access routers while one user has a Cisco 7200 core router. Under normal conditions all four of the routers are connected to the frame relay network. When any of the frame relay circuits fail on a 3600 router, that router will make an ISDN call into the 7200 router. This scenario is what is frequently referred to as a dial around the cloud scenario. This means that all of the routers have a primary circuit that goes through the frame relay cloud. The backup path is an ISDN call around the frame relay cloud, using PPP or MLPPP.

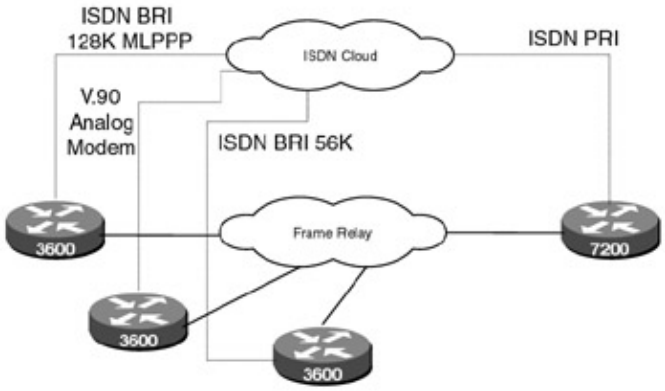


Figure 3−2: ISDN switch overview

ISDN Switches

The main component in an ISDN circuit is an ISDN switch. User equipment at both ends of a circuit connects to an ISDN switch. The ISDN switch is usually a voice switch with a special line card installed in it. The main job of the ISDN switch is to create an end−to−end data or voice circuit between two endpoints; in the case of Figure 3−3, these endpoints are RouterA and RouterB. In the U.S., the most prevalent switches are built by either Lucent Technologies or Nortel. The Lucent switches are the 5ESS models and the Nortel switches are DMS models. As shown in Figure 3−3, the D channel of the ISDN circuit only exists between the ISDN user device and the ISDN switch. Layer 2 aliveness messages such as the Receiver Ready (RR) message that is sent every 10 seconds occurs between the ISDN user device and the ISDN switch.

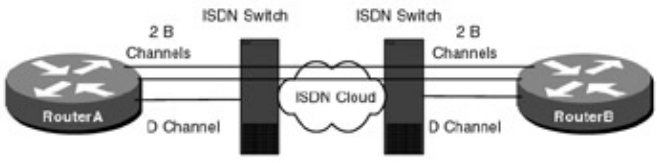


Figure 3−3: ISDN switch overview

A BRI circuit will use one of three types of local signaling, also referred to as call control, between the ISDN user device and the ISDN switch. This call control defines the data format for the D channel. The three call control types used in the U.S. are:

1. National ISDN — A standard call control agreed upon by all ISDN switch manufacturers. National ISDN simplifies ISDN configuration because you no longer need to know exactly what type of switch you are connected to. Both the Lucent 5ESS and the Nortel DMS switches support National ISDN call control.

2. Lucent — A custom Lucent call control.
3. Nortel — A custom Nortel call control.

People are often confused between the physical switch type and the switch signaling. A Lucent 5ESS for example can support both Lucent and National ISDN call signaling on the D channel.

The switch type must be known when bringing up an ISDN PRI circuit because PRI circuits use either Lucent or Nortel call signaling control.

ISDN BRI

An ISDN BRI (Basic Rate Interface) circuit offers a maximum data rate of 128 Kbps. As shown in Figure 3−4, an ISDN BRI circuit consists of one or two data channels running at 56K or 64K, called bearer channels. Single B−channel BRI service is referred to as 1B+D BRI. Two B−channel BRI service is referred to as 2B+D service. Each of the B channels is usually assigned a unique directory number. A directory number, also referred to as a DN, is similar to a phone number. The directory number is used to dial into the BRI channel. Both B channels can also share a single directory number. This type of service is known as a hunt group since the first incoming call will connect to the first B channel and the second incoming call will connect to the second B channel. Each B channel can be provisioned for data only, voice only, or both voice and data.

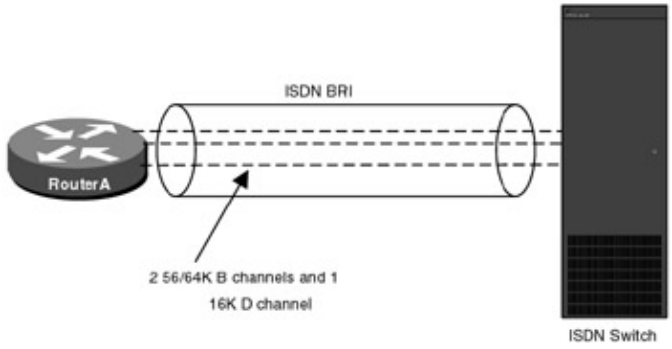


Figure 3−4: ISDN BRI circuit

Each B channel of an ISDN BRI is also assigned a Service Profile Identifier (SPID). The SPID is used when the ISDN end−user device initially synchronizes to the ISDN switch. The analyzer traces later in this chapter will show how the SPID gets sent to the ISDN switch.

As an example, a BRI circuit ordered with 2 B channels and data−only capability might be assigned the following parameters:

B channel #	Directory Number (DN)	SPID	Capability
1	9148313510	91483135100101	Data
2	9148313511	91483135110101	Data

An ISDN BRI circuit also includes a 16K signaling channel, called the D channel. The D channel is responsible for synchronization between the user and the ISDN switch as well as for call setup and teardown. The D channel of an ISDN BRI circuit can also be used to transmit X.25 packet traffic in some applications; this type of BRI service is often referred to as 0B+D BRI.

An ISDN BRI circuit in the U.S. is usually delivered on a single twisted−pair wire. This is referred to as a U interface ISDN circuit. In Europe and other countries, an ISDN circuit is delivered on two twisted−pair wires and is referred to as an ST interface ISDN circuit. Many of the Cisco routers with built−in ISDN interfaces have an ST interface. In order to convert the U interface circuit from the carrier to an ST interface circuit that the router can handle, an external NT1 (network terminating unit) is needed. This is depicted in Figure 3−5. An ISDN circuit is usually delivered by the Telco on an 8−pin RJ−45 jack.

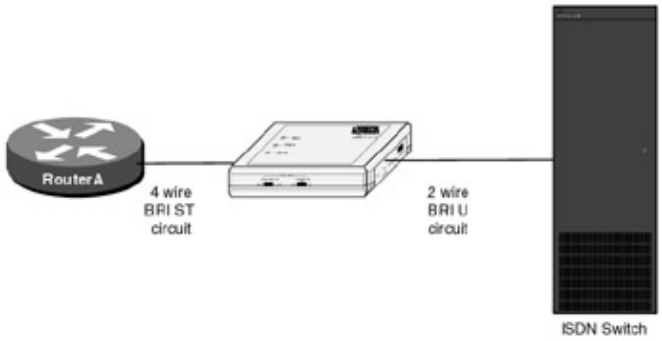


Figure 3−5: ISDN BRI U and ST interfaces

ISDN PRI

An ISDN PRI (Primary Rate Interface) is delivered on a T1 circuit. As shown in Figure 3−6, a PRI consists of 23 data channels (bearer channels) running at 56K or 64K each. Each B channel can be provisioned for data only, voice only, or both voice and data. A PRI also contains a single 64K signaling channel, called the D channel. The D channel is responsible for synchronization between the user and the ISDN switch as well as for call setup and teardown.

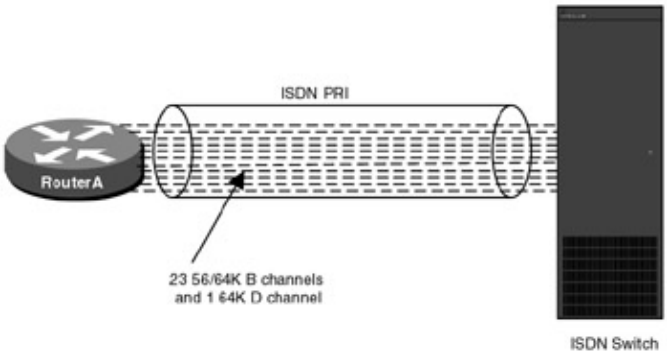


Figure 3−6: ISDN PRI

Unlike a BRI, each of the B channels on a PRI usually shares the same directory number. This means that all calls coming into the PRI will be hunted onto the first available B channel. Each B channel can be provisioned for data only, voice only, or both voice and data.

An ISDN PRI circuit does not have a SPID associated with it.

ISDN Bearer Capability

Each B (bearer) channel of an ISDN BRI or an ISDN PRI can be used in one of two modes: voice, or data. If the B channel is configured to only carry data traffic, it is referred to as having circuit−switched data (CSD) capability. If the B channel is configured to only carry voice traffic, it is referred to as having circuit−switched voice (CSV) capability. A B channel that is configured to carry both voice and data traffic is referred to as having CSV and CSD capabilities.

The ISDN Protocol Stack

As shown in Figure 3−7, ISDN provides a physical transport for upper−layer protocols. On an ISDN data circuit, you will have a layer 2 datalink encapsulation such as PPP, MLPPP, HDLC, or frame relay. Encapsulated in the layer 2 datalink frame will be a layer 3 network layer protocol such as IP, IPX, or AppleTalk.

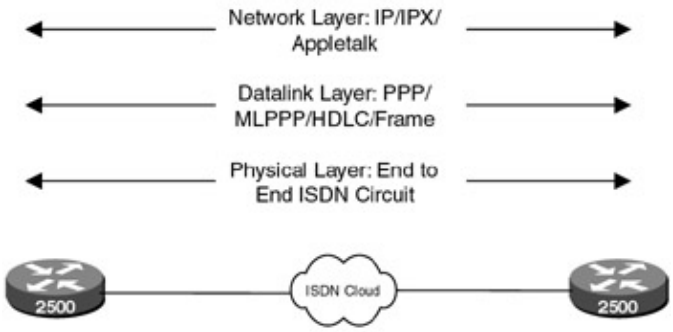


Figure 3−7: ISDN transport

Layer 1

As shown in Figure 3−8, ISDN is defined as a three−layer protocol stack. Layer 1 of the ISDN stack is responsible for the physical transmission of the data on the ISDN circuit.

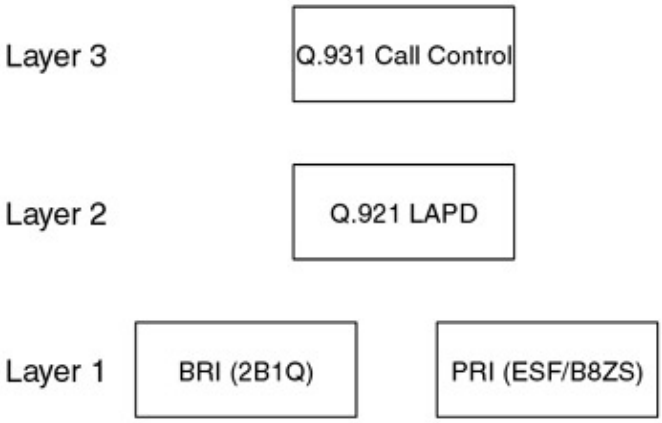


Figure 3−8: ISDN protocol stack

An ISDN BRI U interface circuit is carried over a single twisted pair utilizing 2B1Q data encoding. Data is first framed in a 240−bit frame consisting of 216 data bits and 24 bits of overhead. Eight of these 240 bit frames are then combined into a 1,920−bit superframe.

An ISDN PRI circuit is carried on a T1 circuit, using the same layer 1 Extended Super Frame (ESF) framing and Bipolar Eight Zero Substitution (B8ZS) line coding as a standard T1.

Layer 2

All traffic that flows on the D channel of a BRI or a PRI is encapsulated in an LAPD frame. The LAPD frame, shown in Figure 3−9, is very similar in structure to an HDLC frame. The LAPD signaling is formally specified in the Q.921 specification. These LAPD frames can contain:

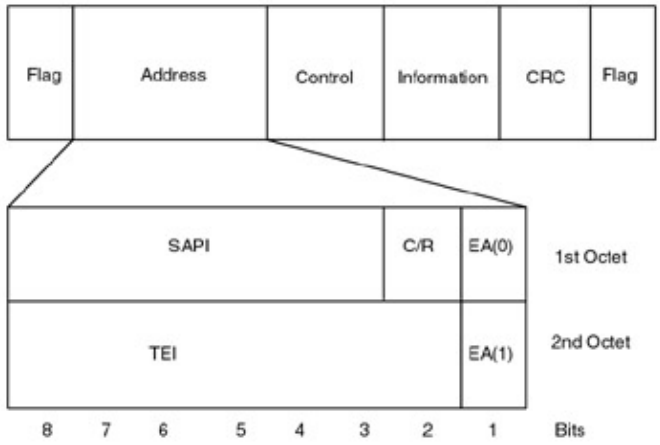


Figure 3−9: ISDN LAPD frame

- Call setup information
- Link establishment, status, and maintenance information
- X.25 packet data

As shown in Figure 3−9, an LAPD frame contains the following fields:

- Flag — Every LAPD frame starts and stops with an 8−bit flag character, which is a 7E.
- Address — This 2−byte field is used to identify if the frame is carrying call control, management overhead, or X.25 traffic. It also indicates if the frame is a command frame or a response to a command.
- Control — The Control field is used to indicate if the LAPD frame is an information frame, a supervisory frame, or an unnumbered frame.
- Information — This field can carry up to 260 bytes of call control or address messages.

Layer 2 Link Layer Establishment

When an ISDN device is first connected to the network, a synchronization process is initiated by the user equipment to synchronize to the network. As seen in Figure 3−10, six packets are exchanged between the router and the ISDN switch to establish the link as active and send an acceptable SPID (in the case of a BRI) to the switch. A trace of each of these packets is shown below.

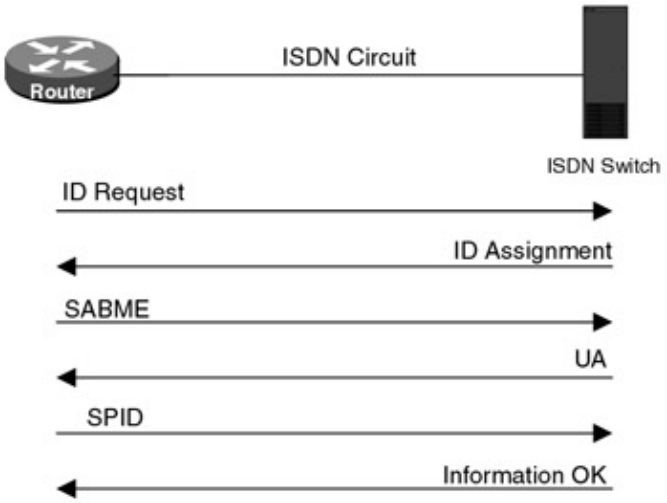


Figure 3−10: ISDN layer 2 establishment

ID Request

The router sends an ID Request message to the switch.

```
Port A DTE ID=72, 03/02/99, 09:37:53.655500
Length=10, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                    64767
SAPI                                  Layer 2 Management 63
Cmd/Resp                                                    0
Ext Bit                                More Addr Octet 0
TEI                                                       127
Ext Bit                               Final Addr Octet 1
Ctrl                                                 UI 03h
P                                                           0
TEI Management Entity ID                                   15
Reference number                                        B34Fh
Message Type                                 ID REQUEST 01h
Action Indicator                                         7Fh
Ext Bit                                                     1
FCS                                                 Good EE0Dh
```

ID Assigned

The switch responds with an ID Assigned message.

```
Port A DCE ID=84, 03/02/99, 09:38:12.393575
Length=10, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                    65279
SAPI                                    Layer 2 Management 63
Cmd/Resp                                                    1
Ext Bit                                    More Addr Octet 0
TEI                                                       127
Ext Bit                                   Final Addr Octet 1
Ctrl                                                UI 03h
P                                                           0
TEI Management Entity ID                                   15
Reference number                                        A237h
Message Type                                 ID ASSIGNED 02h
Action Indicator                                         40h
Ext Bit                                                     1
FCS                                                 Good 102Fh
```

SABME

The router responds to the switch with a SABME message.

```
Port A DTE ID=85, 03/02/99, 09:38:12.400850
Length=5, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                      129
SAPI                                  Call Control Procedures 0
Cmd/Resp                                                    0
Ext Bit                                    More Addr Octet 0
TEI                                   Auto Assign User Eqip 64
Ext Bit                                   Final Addr Octet 1
Ctrl                                                SABME 7Fh
P                                                           1
FCS                                                 Good A8D8h
```

UA

The switch responds with a UA message.

```
Port A DCE ID=86, 03/02/99, 09:38:12.429075
Length=5, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                      129
SAPI                                  Call Control Procedures 0
Cmd/Resp                                                    0
Ext Bit                                    More Addr Octet 0
TEI                                   Auto Assign User Eqip 64
Ext Bit                                   Final Addr Octet 1
Ctrl                                                  UA 73h
F                                                           1
FCS                                                 Good C412h
```

SPID

The router can now send its SPID to the ISDN switch. This SPID must match exactly with the SPID that is entered into the database of the switch.

```
Port A DTE ID=87, 03/02/99, 09:38:12.447850
Length=22, Good FCS
```

```
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                          129
SAPI                                        Call Control Procedures 0
Cmd/Resp                                                        0
Ext Bit                                            More Addr Octet 0
TEI                                         Auto Assign User Eqip 64
Ext Bit                                           Final Addr Octet 1
Ctrl                                                      I 0000h
NS                                                              0
NR                                                              0
P                                                               0
National ISDN
Protocol Discriminator                                Call Control 08h
Reference Flag                                  Msg from Origination 0
Call Reference Value [len=0]                                    0
Message Type                                        INFORMATION 7Bh
INFO ELEMENT Service Profile ID [Code=58] [Len=11]
INFO ELEMENT Service Profile ID                          89953010101
                                                                á
                                    The SPID is sent to the ISDN switch
FCS                                                      Good CAF5h
```

Information

The switch responds with an Information frame to the router after the SPID has been successfully received.

```
Port A DCE ID=89, 03/02/99, 09:38:12.506075
Length=13, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                          641
SAPI                                        Call Control Procedures 0
Cmd/Resp                                                        1
Ext Bit                                            More Addr Octet 0
TEI                                         Auto Assign User Eqip 64
Ext Bit                                           Final Addr Octet 1
Ctrl                                                      I 0002h
NS                                                              0
NR                                                              1
P                                                               0
National ISDN
Protocol Discriminator                                Call Control 08h
Reference Flag                                  Msg from Origination 0
Call Reference Value [len=0]                                    0
Message Type                                        INFORMATION 7Bh
INFO ELEMENT End Point ID [Code=59] [Len=2]
EXT                                                             1
User service identifier                                        37
EXT                                                             1
Interpreter                                                     0
Interpreter terminal identifier                                 1
FCS                                                      Good 22D0h
```

Layer 2 Link Layer Status Checks

Every 10 seconds, the ISDN switch will send an RR packet (Receiver Ready) to the router and will expect an immediate reply. The following two packets show what this RR exchange looks like.

RR Sent from the ISDN Switch to the Router

The ISDN switch sends an RR (Receiver Ready) frame to the router.

```
Port A DCE ID=4, 03/01/99, 10:41:29.238050
```

```
Length=6, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                          641
SAPI                                        Call Control Procedures 0
Cmd/Resp                                                        1
Ext Bit                                            More Addr Octet 0
TEI                                         Auto Assign User Eqip 64
Ext Bit                                           Final Addr Octet 1
Ctrl                                                    RR 0103h
NR                                                              1
PF                                                              1
FCS                                                   Good DBB8h
```

RR Reply Sent from the Router to the ISDN Switch

The router immediately responds with an RR (Receiver Ready) frame to the ISDN switch.

```
Port A DTE ID=5, 03/01/99, 10:41:29.246300
Length=6, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                          641
SAPI                                        Call Control Procedures 0
Cmd/Resp                                                        1
Ext Bit                                            More Addr Octet 0
TEI                                         Auto Assign User Eqip 64
Ext Bit                                           Final Addr Octet 1
Ctrl                                                    RR 0103h
NR                                                              1
PF                                                              1
FCS                                                   Good DBB8h
```

ISDN Layer 3 Signaling

The ISDN layer 3 protocols are used for establishing, maintaining, and disconnecting calls between the user equipment and the network. Various control messages are passed between the user and the network for these purposes. Figure 3−11 shows how a call is placed. As seen in Figure 3−11, some of the signaling is local — between the router and the ISDN switch on either side of the circuit — while other signaling flows end to end between both routers.

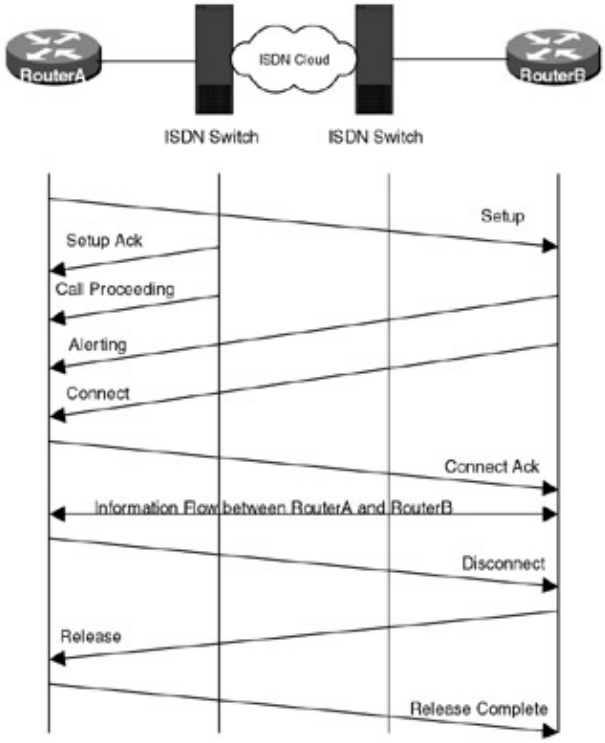


Figure 3−11: ISDN Layer 3 call control

The following traces show a call setup and disconnect sequence between RouterA and RouterB (reference Figure 3−11 for a frame−by−frame explanation).

Setup Message from Router to Switch

RouterA sends a Setup message to the RouterB. This message specifies several key parameters:

- An unrestricted digital channel is requested.
- A 64Kbits/second call is requested.
- The calling number of the router placing the call is 8995301.
- The number of the router being called is 8993601:

```
Port A DTE ID=22, 03/02/99, 16:38:08.917600
Length=37, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                        129
SAPI                                    Call Control Procedures 0
Cmd/Resp                                                      0
Ext Bit                                        More Addr Octet 0
TEI                                     Auto Assign User Eqip 64
Ext Bit                                        Final Addr Octet 1
Ctrl                                                  I 1216h
NS                                                            9
NR                                                           11
P                                                             0
National ISDN
Protocol Discriminator                                Call Control 08h
Reference Flag                                  Msg from Origination 0
Call Reference Value [len=1]                                  1
Message Type                                             SETUP 05h
INFO ELEMENT Bearer Capability [Code=4] [Len=2]
Coding Std                                                 CCITT 0
Transfer Capab                                Unrestricted Digital 8
EXT                                                           1
Transfer Mode                                           Circuit 0
Transfer Rate (D Channel)                                64 kb/s 16
EXT                                                           1
```

```
INFO ELEMENT Channel ID [Code=24] [Len=1]
EXT                                                                  1
Interface ID Present                                        Implicit 0
Interface Type                                             Basic Rate 0
Pref/Excl                                                Preferred Chan 0
D−channel Ind                                                       No 0
Info Channel Select                                        Any channel 3
INFO ELEMENT Keypad [Code=44] [Len=8]
Info (IA5 Char)                                                 98993601
INFO ELEMENT Calling Number [Code=108] [Len=8]
Type                                                 Subscriber (Local) 4
Plan                                                 ISDN − E.164/E.163 1
EXT                                                                  1
Num Digits (IA5 Char)                                            8995301
FCS                                                            Good 9BCDh
```

Setup Acknowledgement

A Setup Acknowledge message is sent from the local ISDN switch to RouterA.

```
Port A DCE ID=24, 03/02/99, 16:38:08.986850
Length=13, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                               641
SAPI                                          Call Control Procedures 0
Cmd/Resp                                                             1
Ext Bit                                                 More Addr Octet 0
TEI                                            Auto Assign User Eqip 64
Ext Bit                                               Final Addr Octet 1
Ctrl                                                          I 1614h
NS                                                                  11
NR                                                                  10
P                                                                    0
National ISDN
Protocol Discriminator                                    Call Control 08h
Reference Flag                                        Msg to Origination 1
Call Reference Value [len=1]                                         1
Message Type                                                SETUP ACK 0Dh
INFO ELEMENT Channel ID [Code=24] [Len=1]
EXT                                                                  1
Interface ID Present                                        Implicit 0
Interface Type                                             Basic Rate 0
Pref/Excl                                                Exclusive Chan 1
D−channel Ind                                                       No 0
Info Channel Select                                                 B1 1
FCS                                                            Good 15AFh
```

Call Proceeding

A Call Proceeding message is sent from the local ISDN switch to RouterA.

```
Port A DCE ID=26, 03/02/99, 16:38:11.006325
Length=10, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                               641
SAPI                                          Call Control Procedures 0
Cmd/Resp                                                             1
Ext Bit                                                 More Addr Octet 0
TEI                                            Auto Assign User Eqip 64
Ext Bit                                               Final Addr Octet 1
Ctrl                                                          I 1814h
NS                                                                  12
NR                                                                  10
P                                                                    0
```

```
National ISDN
Protocol Discriminator                                  Call Control 08h
Reference Flag                                      Msg to Origination 1
Call Reference Value [len=1]                                           1
Message Type                                    CALL PROCEEDING 02h
FCS                                                          Good 6E62h
```

Alerting

An Alerting message is sent from RouterB to RouterA.

```
Port A DCE ID=28, 03/02/99, 16:38:11.231825
Length=13, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                               641
SAPI                                              Call Control Procedures 0
Cmd/Resp                                                               1
Ext Bit                                                  More Addr Octet 0
TEI                                               Auto Assign User Eqip 64
Ext Bit                                                Final Addr Octet 1
Ctrl                                                          I 1A14h
NS                                                                  13
NR                                                                  10
P                                                                    0
National ISDN
Protocol Discriminator                                  Call Control 08h
Reference Flag                                      Msg to Origination 1
Call Reference Value [len=1]                                           1
Message Type                                          ALERTING 01h
INFO ELEMENT Signal [Code=52] [Len=1]
INFO ELEMENT Signal                                 Ring Back Tone On 01h
FCS                                                          Good D6D6h
```

Connect

A Connect message is sent from RouterB to RouterA.

```
Port A DCE ID=30, 03/02/99, 16:38:11.276950
Length=13, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                               641
SAPI                                              Call Control Procedures 0
Cmd/Resp                                                               1
Ext Bit                                                  More Addr Octet 0
TEI                                               Auto Assign User Eqip 64
Ext Bit                                                Final Addr Octet 1
Ctrl                                                          I 1C14h
NS                                                                  14
NR                                                                  10
P                                                                    0
National ISDN
Protocol Discriminator                                  Call Control 08h
Reference Flag                                      Msg to Origination 1
Call Reference Value [len=1]                                           1
Message Type                                            CONNECT 07h
INFO ELEMENT Signal [Code=52] [Len=1]
INFO ELEMENT Signal                                        Tones Off 3Fh
FCS                                                          Good AEE1h
```

Connect Acknowledge

A Connect Acknowledge message is sent from RouterA to RouterB.

```
Port A DTE ID=32, 03/02/99, 16:38:11.306700
Length=10, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                        129
SAPI                                        Call Control Procedures 0
Cmd/Resp                                                      0
Ext Bit                                            More Addr Octet 0
TEI                                         Auto Assign User Eqip 64
Ext Bit                                           Final Addr Octet 1
Ctrl                                                    I 141Eh
NS                                                           10
NR                                                           15
P                                                             0
National ISDN
Protocol Discriminator                                Call Control 08h
Reference Flag                                  Msg from Origination 0
Call Reference Value [len=1]                                  1
Message Type                                      CONNECT ACK 0Fh
FCS                                                     Good 7443h
```

Once the ISDN call has been established, traffic can flow between RouterA and RouterB. The call has established a 64−Kbps data pipe between the two routers. The call can be terminated by either router. Call termination is initiated by one of the routers sending a Disconnect message. The following disconnect sequence shows an example where RouterA initiates the disconnect sequence.

Disconnect

RouterA sends a Disconnect message to RouterB.

```
Port A DTE ID=112, 03/02/99, 16:42:11.593225
Length=14, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                        129
SAPI                                        Call Control Procedures 0
Cmd/Resp                                                      0
Ext Bit                                            More Addr Octet 0
TEI                                         Auto Assign User Eqip 64
Ext Bit                                           Final Addr Octet 1
Ctrl                                                    I 161Eh
NS                                                           11
NR                                                           15
P                                                             0
National ISDN
Protocol Discriminator                                Call Control 08h
Reference Flag                                  Msg from Origination 0
Call Reference Value [len=1]                                  1
Message Type                                      DISCONNECT 45h
INFO ELEMENT Cause [Code=8] [Len=2]
Coding Std                                              CCITT 0
Location                                                 User 0
EXT                                                           1
Class                                               Normal Event 1
Value                                       Normal Call Clearing 16
EXT                                                           1
FCS                                                     Good 84A0h
```

Release

RouterB sends a Release message to RouterA.

```
Port A DCE ID=114, 03/02/99, 16:42:11.667225
Length=10, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
```

```
MOD 128:
Addr                                                    641
SAPI                                  Call Control Procedures 0
Cmd/Resp                                                  1
Ext Bit                                      More Addr Octet 0
TEI                                   Auto Assign User Eqip 64
Ext Bit                                     Final Addr Octet 1
Ctrl                                                I 1E18h
NS                                                       15
NR                                                       12
P                                                         0
National ISDN
Protocol Discriminator                             Call Control 08h
Reference Flag                                 Msg to Origination 1
Call Reference Value [len=1]                              1
Message Type                                      RELEASE 4Dh
FCS                                                Good 57B7h
```

Release Complete

RouterA sends a Release Complete message to RouterB.

```
Port A DTE ID=116, 03/02/99, 16:42:11.701225
Length=10, Good FCS
LAPD − LINK ACCESS PROCEDURE D CHANNEL
MOD 128:
Addr                                                    129
SAPI                                  Call Control Procedures 0
Cmd/Resp                                                  0
Ext Bit                                      More Addr Octet 0
TEI                                   Auto Assign User Eqip 64
Ext Bit                                     Final Addr Octet 1
Ctrl                                                I 1820h
NS                                                       12
NR                                                       16
P                                                         0
National ISDN
Protocol Discriminator                             Call Control 08h
Reference Flag                               Msg from Origination 0
Call Reference Value [len=1]                              1
Message Type                                RELEASE COMPLETE 5Ah
FCS                                                Good C1C2h
```

ISDN Configuration

When configuring ISDN on your network, you will need some basic information about the ISDN circuit, which must be supplied by the carrier. For an ISDN BRI, you will need to know:

- The Directory Number (DN) of each B channel
- The SPID of each B channel
- The D−channel signaling protocol being used

For an ISDN PRI, you will need to know:

- The D−channel signaling protocol being used
- What the directory number of the PRI is

ISDN with Non−ISDN−Equipped Routers

A router does not have to have an integrated BRI or PRI interface to take advantage of ISDN services. An external device, known as an ISDN terminal adapter, allows users to connect to an ISDN service with a router that does not have an ISDN interface. An ISDN terminal adapter has a V.35 interface on the user DTE side and one or more ISDN interfaces on the network side. The ISDN terminal adapter can be programmed to dial

the far−end router in a variety of ways. The most popular dial method is usually DTR dialing. With DTR dialing, the ISDN terminal adapter will call a preprogrammed far−end number when the attached router raises the V.35 DTR interface. Figure 3−12 shows an ISDN terminal adapter network diagram.

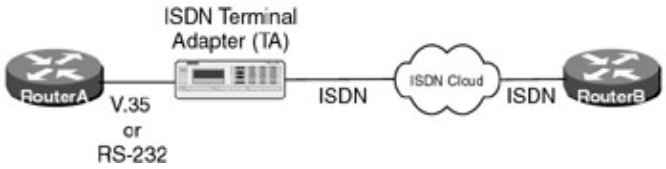


Figure 3−12: ISDN terminal adapter

Commands Discussed in This Chapter

- **backup delay** *enable−delay disable−delay*
- **backup interface** *type number*
- **controller t1** *number*
- **dialer callback−secure**
- **dialer−group** *group−number*
- **dialer idle−timeout** *seconds*
- **dialer−list** *dialer−group* **list** *access−list−number*
- **dialer−list** *dialer−group* protocol *protocol−name* **{permit | deny** | list *access−list−number* | *access−group*}
- **dialer load−threshold** *load* [outbound | inbound | either]
- **dialer map** *protocol next−hop−address* [**name** *hostname*] [**broadcast**] [*dial−string*]
- **dialer pool** *number*
- **dialer pool−member** *number*
- **dialer remote−name** *username*
- **dialer string** *dial−string*
- **dialer watch−group** *group−number*
- **dialer watch−list** *group−number* **ip** *ip−address address−mask*
- **debug isdn q921**
- **debug isdn q931**
- **debug ppp authentication**
- **interface bri** *number*
- **interface dialer** *number*
- **ip ospf demand−circuit**
- **isdn spid1** *spid−number [ldn]*
- **isdn spid 2** *spid−number [ldn]*
- **isdn switch−type** *switch−type*
- **ppp callback {accept | request}**
- **ppp multilink**
- **pri−group** [**timeslots** *range*]
- **show dialer**
- **show interfaces bri** *number:bchannel*
- **show isdn service**
- **show isdn status**
- **show ppp multilink**
- **show snapshot**
- **snapshot client** *active−time quiet−time* **[suppress−statechange−updates] [dialer]**
- **snapshot server** *active−time* **[dialer]**

Definitions

backup delay: This interface configuration command defines how much time, in seconds, will elapse before activating a backup interface when a primary interface fails. The command also specifies, in seconds, how much time will elapse before dropping the backup interface after the primary interface returns to an active state.

backup interface: This interface configuration command causes the associated interface to be defined as a backup interface. The backup interface will remain in standby mode until it is activated.

controller t1: This interface command defines a channelized T1 interface that is used to carry an ISDN PRI circuit.

dialer callback−secure: This interface command ensures that the initial call is always disconnected at the receiving end and the return call is made only if the username is configured for callback.

dialer−group: This interface configuration command causes the interface to be associated with a specific dialer list.

dialer idle−timeout: This interface command specifies the idle time before the ISDN line is disconnected. Any interesting traffic will reset this timer.

dialer−list: This global command applies an access list to a specified interface and specifies what type of traffic is interesting, and will initiate an ISDN call to be made.

dialer load−threshold: This interface command defines a load threshold that must be met before the router will initiate an additional call to a destination.

dialer map: This interface command configures an ISDN interface to call a far−end device. It associates a next−hop address with an ISDN phone number.

dialer pool: This interface command can only be applied to a dialer interface. It specifies what dialer pool to use to connect to a specific destination.

dialer pool−member: This interface command configures a physical interface to be a member of a dialer pool.

dialer remote−name: This interface command only applies to a dialer interface. It specifies the authentication name of a router on a remote network.

dialer string: This interface configuration command specifies a phone number to be used when making an ISDN call. This command is used with dialer profiles.

dialer watch−group: This interface command enables dialer watch and assigns an access list of watched routes.

dialer watch−list: This global command is used to add the list of routes that dialer watch will monitor.

debug isdn q921: This debug command causes the router to display layer 2 ISDN call control information.

debug isdn q931: This debug command causes the router to display layer 3 ISDN call control information.

debug ppp authentication: This debug command causes the router to display CHAP or PAP authentication negotiation status.

interface bri: This global command defines a BRI interface and enters interface configuration mode.

interface dialer: This global command defines a dialer interface. A dialer interface is used to apply a single interface configuration to one of more physical interfaces.

ip ospf demand−circuit: This interface command configures the specific interface to treat OSPF as a demand−type circuit.

isdn spid1: This interface configuration command defines the SPID value for BRI channel B1 on a BRI interface.

isdn spid 2: This interface configuration command defines the SPID value for BRI channel B2 on a BRI interface.

isdn switch−type: This global configuration command defines the D channel signaling protocol being used on an ISDN interface.

ppp callback: This interface command enables the router to act as a PPP callback client.

ppp multilink: This interface configuration command causes the defined interface to try to negotiate a multilink PPP datalink protocol.

pri−group: This controller configuration command defines a channelized T1 interface to be used as an ISDN PRI circuit.

show dialer: This exec command displays diagnostic and status information for dial on demand interfaces.

show interfaces bri: This exec command is used to display information on an ISDN BRI D channel and B channels.

show isdn service: This exec command displays information about an ISDN PRI interface on the router.

show isdn status: This exec command displays information about the ISDN circuits defined on the router.

show ppp multilink: This exec command displays information about any multilink bundles that are active on the router.

show snapshot: This exec command displays information about snapshot routing.

snapshot client: This interface command configures a router as a snapshot client.

snapshot server: This interface command configures a router as a snapshot server.

IOS Requirements

The labs in this chapter were run with IOS 11.2. Some features, such as dialer profiles, are only available in IOS 11.2 and higher.

ISDN Switch Configuration

The Adtran Atlas 800 is being used in this chapter as our ISDN switch. The Atlas 800 can be populated with both ISDN BRI and ISDN PRI cards. The advantage of using the Atlas 800 is its ease of use and adherence to ISDN standards. In addition, the Atlas' small form factor makes it a portable unit that can be set up on a desktop. The base unit of the Atlas 800 comes with two ISDN PRI circuits. Additional BRI cards and PRI cards can be added. The Atlas 800 chassis can hold up to eight additional cards. Each BRI card contains eight ISDN BRI U interface circuits. Each PRI card contains four ISDN PRI circuits.

The Atlas 800 chassis has a 10BaseT Ethernet connection that is used for system management. The Atlas 800 also has a control port that can be used for local terminal access. The Atlas 800 is managed via a simple menu system. The management screens are the same whether you telnet to the Atlas via the Ethernet port or directly connect to the unit via the rear control port. Additional information on the Atlas 800 can be found on the Adtran Web page at **http://www.adtran.com/**.

The Atlas 800 is simple to configure. Following are the key configuration screens for setting up these labs:

The System Info screen is shown below. Key system information such as firmware levels and system uptime can be found on this screen.

```
CCIE LAB STUDY GUIDE/System Info
System Info     System Name
CCIE LAB STUDY GUIDE
System Status  | System Location           Adtran ATLAS 800
System Config  | System Contact            Adtran ATLAS 800
System Utility | Firmware Revision      ATLAS 800   Rev. H 09/18/98 09:11:41
Modules        | System Uptime           0 days   1 hours  36 min   55 secs
Dedicated Maps | Startup Mode            Power cycle
Dial Plan      | Current Time/Date  (24h)  Saturday March  6  16:55:12  1999
               | Installed Memory    Flash:1048576 bytes  DRAM:8388608 bytes

               | Serial Number              847B8304
               | Boot ROM Rev               C 11/18 /97
```

The Modules screen displays all active modules that are installed in the Atlas 800. We see from the following screen print that our system has four occupied slots. Slot 0 is the system controller. The system controller is part of the base chassis. Slot 1 contains a four−port ISDN PRI card. Slot 5 contains an eight−port ISDN BRI card. Slot 8 contains a four−port V.35 card. For these labs, we will only be using the PRI and BRI cards.

```
CCIE LAB STUDY GUIDE/Modules
System Info    | Slt   Type   Menu   Alarm   Test   State     Status     Rev
System Status  |  0  Sys Ctrl  [+]    [OK]   [OFF]  ONLINE    Online      T
System Config  |  1  T1/PRI−4  [+]    [OK]   [OFF]  ONLINE    Online      A
System Utility |  2  EMPTY                             ONLINE   Empty       −
Modules        |  3  EMPTY                             ONLINE   Empty       −
Dedicated Maps |  4  EMPTY                             ONLINE   Empty       −
Dial Plan      |  5  UBRI−8    [+]    [OK]   [OFF]   ONLINE   Online      C
               |  6  EMTPY      ONLINE  Empty        −
               |  7  EMPTY                             ONLINE   Empty       −
               |  8  V35Nx−4   [+]    [OK]   [OFF]    ONLINE   Online     M
```

The Dial Plan screen is used to configure directory numbers (DNs) for each of the BRI circuits. We see from the following screen print that BRI #1 in slot #5 has been assigned 8995101 as its number for channel B1. BRI #2 in slot #5 has been assigned 8995201 as its number for channel B1.

```
CCIE LAB STUDY GUIDE/Dial Plan/User Term
Network Term |  #    Slot/Svc  Port/Link  Sig  In#Accept  Out#Rej  Ifce Config
User Term    |  1    5)UBRI−8   1)BRI 5/1      [8995101]    [+]     [8995101]
Global Param |  2    5)UBRI−8   2)BRI 5/2      [8995201]    [+]     [8995201]
```

The Interface Configuration screen is used to set the switch type for each circuit. We see from the following screen that National ISDN switch type has been selected for this circuit.

```
CCIE LAB STUDY GUIDE/Dial Plan/User Term[1]/Interface Configuration
Incoming Number Accept List | Switch Type          National ISDN
Outgoing Number Reject List | SPID list            [8995101]
Interface Configuration     | Strip MSD            None
                            | Source ID            0
                            | Outgoing Caller ID   Send as provided
```

The SPID List screen is used to configure the SPID and call capability for each B channel. We see from the following screen print that the SPID for channel B1 has been set to 5101 while the SPID for channel B2 has been set to 5102. Each of these two B channels are enabled for 64K, 56K, audio, and speech capabilities.

```
CCIE LAB STUDY GUIDE/Dial Plan/User Term[1]/Interface Configuration/SPID list
SPID list| #    Phone #            SPID #      Calls  D64    D56   Audio  Speech
         | 1    8995101             5101         2   Enable Enable Enable Enable
         | 2    8995102             5102         2   Enable Enable Enable Enable
```

Lab #2: ISDN Basics and Switch Basics

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a single ISDN BRI interface
- Cisco IOS 11.2 or higher
- Two ISDN BRI circuits
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This configuration will explore basic ISDN commands and functionality.

RouterA and RouterB are connected as shown in Figure 3−13.

Figure 3−13: ISDN basics and switch basics
A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. ISDN commands are highlighted in bold.

RouterA

```
RouterA#show run
Building configuration...

Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
username RouterB password 7 030752180500
isdn switch-type basic-ni1 ← Set D channel call control
!
interface Serial0/0
 no ip address
!
```

```
interface BRI0/0
 ip address 196.1.1.1 255.255.255.0
 encapsulation ppp
 isdn spid1 5101 8995101 ← Set SPID for both B channels
 isdn spid2 5102 8995102
 dialer idle-timeout 90 ← Set interesting traffic timeout
 dialer map ip 196.1.1.2 name RouterB broadcast 8995201 ← Define next hop
                                                          address and dial
                                                          string
 dialer load-threshold 1 ← Threshold for adding additional B channels
 dialer-group 1 ← Associate with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
RouterB#show run
Building configuration...

Current configuration:
!
version 11.2
service timestamps debug datetime localtime
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
username RouterA password 7 030752180500
isdn switch-type basic-ni1 ← Set D channel call control
!
interface BRI0/0
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
 isdn spid1 5201 8995201 ← Set the SPID for both B channels
 isdn spid2 5202 8995202
 dialer idle-timeout 90 ← Set interesting traffic timeout
 dialer map ip 196.1.1.1 name RouterA ← Define a next hop address
 dialer-group 1 ← Associate with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
```

```
end
```

Monitoring and Testing the Configuration

Let's start by connecting our terminal to RouterA. Type the **show isdn status** command. This command enables you to view the operational status of the BRI circuit that is terminated on the router. The command's output displays layers 1, 2, and 3 ISDN status. We see from the following screen print that layer 1 is active on this circuit. This means that the router senses the 2B1Q line coding of the BRI circuit. Layer 2 status indicates that a TEI has been assigned for both B channels. We also see that the SPIDs for both B channels have been sent to the switch and are valid.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
      ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 8(established)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 8(established)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 0
 Total Allocated ISDN CCBs = 0
```

Type the **show interface bri 0/0** command. We see that the interface is in an up/up(spoofing) state. This means that the D channel is active on the BRI circuit. The D channel is said to be spoofing because it is a local control circuit between the router and the ISDN switch.

```
RouterA#show interface bri 0/0
BRI0/0 is up, line protocol is up (spoofing)
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

On this ISDN BRI circuit, the actual BRI interface is referenced as bri 0/0. As we saw above, this refers to the D channel of the ISDN circuit. The status of each of the two B channels can also be displayed. This is accomplished by using the **show interface bri 0/0:1** and the **show interface bri 0/0:2** commands for channels B1 and B2, respectively. We see that both of these interfaces are in the down/down state. This is because there are no active calls on the BRI circuit at this time.

```
RouterA#show interface bri 0/0:1
BRI0/0:1 is down, line protocol is down
  Hardware is QUICC BRI with U interface
.
.
RouterA#show interface bri 0/0:2
BRI0/0:2 is down, line protocol is down
  Hardware is QUICC BRI with U interface
.
.
```

The **show dialer** command displays the dial and timer status of each B channel that is defined on the router.

We see that there have been 0 attempted calls to 8995201 since the router was last booted. We also see that each B channel will drop its call after 90 seconds of inactivity.

```
RouterA#show dialer

BRI0/0 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8995201                  0          0    never                    −
0 incoming call(s) have been screened.

BRI0/0:1 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is idle

BRI0/0:2 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is idle
```

The **show ppp multilink** command will display any active MLPPP bundles on the router. Since there are no active calls, we see that there are no active bundles at this time.

```
RouterA#show ppp multi
No active bundles
```

Now let's connect to RouterB. The BRI circuit on RouterB is also configured and active as indicated by the **show isdn status** command shown below.

```
RouterB#show isdn status
The current ISDN Switchtype = basic−ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 8(established)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 8(established)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 0
    Number of active calls = 0
    Number of available B−channels = 2
 Total Allocated ISDN CCBs = 0
```

The **show dialer** command shows that RouterB will drop a B channel after 90 seconds of inactivity.

```
RouterB#show dialer

BRI0/0 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
0 incoming call(s) have been screened.

BRI0/0:1 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is idle
```

```
BRI0/0:2 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is idle
```

The BRI interface status for RouterB is identical to the status of the BRI interfaces that we just examined on Router A. The bri 0/0 interface refers to the D channel of the circuit. This is shown to be in an up/up(spoofing state), indicating that the D channel is active.

```
RouterB#show interface bri 0/0
BRI0/0 is up, line protocol is up (spoofing)
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.2/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Now let's reconnect to RouterA. Try to ping the ISDN interface of RouterB at IP address 196.1.1.2. Notice that this ping causes RouterA to place two calls to RouterB.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:

%LINK−3−UPDOWN: Interface BRI0/0:1, changed state to up
← Call #1
%LINK−3−UPDOWN: Interface Virtual−Access1, changed state to up
%LINEPROTO−5−UPDOWN: Line protocol on Interface BRI0/0:1, changed
state to up
%LINEPROTO−5−UPDOWN: Line protocol on Interface Virtual−Access1,
changed state to up
%LINK−3−UPDOWN: Interface BRI0/0:2, changed state to up
← Call #2
%LINEPROTO−5−UPDOWN: Line protocol on Interface BRI0/0:2, changed
state to up
%ISDN−6−CONNECT: Interface BRI0/0:2 is now connected to 8995201
RouterB
```

The ping will be partially successful because some of the ping packets will be sent when the call is being made.

```
..!!!
Success rate is 60 percent (3/5), round−trip min/avg/max = 20/21/24 ms
```

After the call has connected, another ping should be 100 percent successful.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 20/22/24 ms
```

The **show dialer** command will now indicate that two successful calls have been made. We see that interface bri 0/0:1 and interface bri 0/0:2 are both online. The dial reason for the call occurring on BRI channel B1 is shown as being traffic from 196.1.1.1 to 196.1.1.2. This was our ping to 196.1.1.2. The dial reason for BRI channel B2 was multilink bundle overload. This overload is determined by the **dialer load−threshold** command in the router's configuration. The time until each B channel is disconnected is also displayed. Any interesting traffic will reset these numbers to the idle timer.

```
RouterA#show dialer

BRI0/0 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8995201                  2          0    00:00:10       successful
0 incoming call(s) have been screened.

BRI0/0:1 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: ip (s=196.1.1.1, d=196.1.1.2) ← Reason for dialing was our ping
Time until disconnect 76 secs ← Disconnect time
Current call connected 00:00:11
Connected to 8995201 (RouterB)

BRI0/0:2 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: Multilink bundle overloaded
Time until disconnect 77 secs
Current call connected 00:00:12
Connected to 8995201 (RouterB)
```

Issuing the **show ppp multilink** command will now show that we have an MLPPP bundle consisting of two members. The two members are bri 0/0:1 and bri 0/0:2.

```
RouterA#show ppp multi

Bundle RouterB, 2 members, Master link is Virtual−Access1
Dialer Interface is BRI0/0
  0 lost fragments, 0 reordered, 0 unassigned, sequence 0x24/0x26 rcvd/sent
  0 discarded, 0 lost received, 1/255 load

Member Links: 2
BRI0/0:1 ← These two interfaces make up the MLPPP bundle
BRI0/0:2
```

The D channel of the BRI, shown as bri 0/0, will still indicate that it is in a spoofing state.

```
RouterA#show interface bri 0/0
BRI0/0 is up, line protocol is up (spoofing)
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Each of the B channels on the BRI will now be in an up/up state. Notice that both of these B channels have negotiated the multilink protocol with the far−end router.

```
RouterA#show interface bri 0/0:1
BRI0/0:1 is up, line protocol is up
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open, multilink Open ← MLPPP
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
```

```
  5 minute output rate 0 bits/sec, 0 packets/sec
     1191 packets input, 67466 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1198 packets output, 69418 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     6 carrier transitions

RouterA#show interface bri 0/0:2
BRI0/0:2 is up, line protocol is up
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open, multilink Open ← MLPPP
.
.
```

The **show ppp multi**, **show isdn status**, and **show interface bri** commands on RouterB will display similar output with respect to what we just examined on RouterA.

After the idle timeout period (90 seconds without interesting traffic), the call will disconnect. The following screen print shows the call being brought down and both B channels on the BRI (bri 0/0:1 and bri 0/0:2) being changed to a down state.

```
%LINEPROTO−5−UPDOWN: Line protocol on Interface
Virtual−Access1, changed state to down
%LINK−3−UPDOWN: Interface Virtual−Access 1,
changed state to down
%ISDN−6−DISCONNECT: Interface BRI0/0:1  disconnected from
8995201 RouterB, call lasted 99 seconds
%ISDN−6−DISCONNECT: Interface BRI0/0:2  disconnected from
8995201 RouterB, call lasted 96 seconds
%LINK−3−UPDOWN: Interface BRI0/0:1, changed state to down
%LINK−3−UPDOWN: Interface BRI0/0:2, changed state to down
%LINEPROTO−5−UPDOWN: Line protocol on Interface
BRI0/0:1, changed state to down
%LINEPROTO−5−UPDOWN: Line protocol on Interface
 BRI0/0:2, changed state to down
```

Lab #3: Backup Interfaces

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a single ISDN BRI interface, a single serial interface, and an Ethernet interface
- Two ISDN BRI circuits
- Cisco IOS 11.2 or higher
- A Cisco DCE/DTE V.35 crossover cable. If a crossover cable is not available, you can use a Cisco DCE cable connected to a Cisco DTE cable.
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This configuration demonstrates how to use a backup interface for ISDN dial backup. A backup interface is a designated interface that remains in standby mode until the primary interface goes down. A backup interface

can be either an ISDN interface or a serial interface such as a V.35 port. When a V.35 port acts as a backup interface, the V.35 port is usually connected to an external ISDN terminal adapter or analog modem.

The two routers are connected as shown in Figure 3−14. RouterA and RouterB are connected to an Adtran Atlas 800 ISDN switch. When the serial connection between RouterA and RouterB is broken, RouterA will dial RouterB over the BRI circuit.

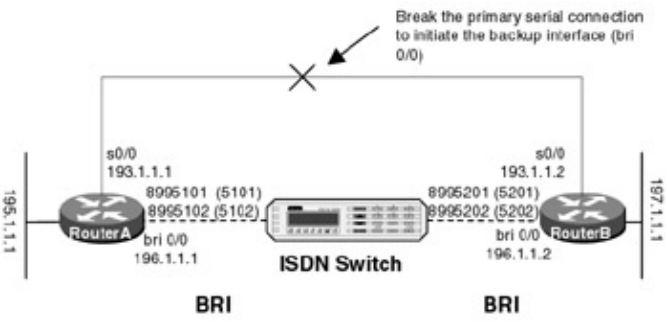


Figure 3−14: Backup interfaces

A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

Note An item to keep in mind during this lab exercise is that a drawback of backup interfaces is their testability. The only way to initiate a call from a backup interface is to bring down the primary circuit, thus affecting the customer's traffic. We will see in the next chapter, on floating static routes, that floating statics allow the ISDN interface to be tested while customer traffic continues to flow.

Note The s0/0 interface on RouterA is configured as a DCE interface and supplies clocking to the s0/0 interface of RouterB.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. ISDN commands are highlighted in bold.

RouterA

```
RouterA#show run
Building configuration...

Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
username RouterB password 7 070C285F4D06
isdn switch-type basic-ni1 ← Set D channel call control
!
interface Ethernet0/0
 ip address 195.1.1.1 255.255.255.0
```

```
 no keepalive
!
interface Serial0/0
 backup delay 5 20 ← Go to backup 5 seconds after loss, return 20 seconds
                     after primary returns
 backup interface BRI0/0 ← The BRI interface is the backup interface
 ip address 193.1.1.1 255.255.255.0
 encapsulation ppp
 clockrate 64000
!
interface BRI0/0
 ip address 196.1.1.1 255.255.255.0
 encapsulation ppp
 isdn spid1 5101 8995101 ← Set SPIDs for both B channels
 isdn spid2 5102 8995102
 dialer idle-timeout 90 ← Disconnect 90 seconds after no interesting packets
 dialer map ip 196.1.1.2 name RouterB broadcast 8995201 ← Define next hop
                                                          address and dial
                                                          string
 dialer load-threshold 1 ← Set threshold for adding additional B channels
 dialer-group 1 ← Associate interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink ← Type to negotiate a multilink PPP session
!
router rip
 network 195.1.1.0
 network 193.1.1.0
 network 196.1.1.0
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Router B

```
RouterB#show run
Building configuration...

Current configuration:
!
version 11.2
service timestamps debug datetime localtime
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
username RouterA password 7 094F471A1A0A
isdn switch-type basic-ni1 ← Set D channel call control
!
interface Ethernet0/0
 ip address 197.1.1.1 255.255.255.0
 no keepalive
!
interface Serial0/0
```

```
 ip address 193.1.1.2 255.255.255.0
 encapsulation ppp
!
interface BRI0/0
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
 isdn spid1 5201 8995201 ← Set the SPID value for both B channels
 isdn spid2 5202 8995202
 dialer idle-timeout 90 ← Set the interesting traffic timeout
 dialer map ip 196.1.1.1 name RouterA broadcast ← Define a next hop address
 dialer-group 1 ← Associate this interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink
!
router rip
 network 193.1.1.0
 network 197.1.1.0
 network 196.1.1.0
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Verify that interface s 0/0 is in an up/up state. Notice that interface bri 0/0 is designated as the backup interface for s 0/0.

```
RouterA#show interface s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 193.1.1.1/24
  Backup interface BRI0/0, kickin load not set, kickout load not set
      failure delay 5 sec, secondary disable delay 20 sec
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open
  Open: IPCP, CDP
  Last input 00:00:08, output 00:00:08, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/1 (active/max active)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     489 packets input, 63338 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 1 giants, 0 throttles
     4 input errors, 1 CRC, 2 frame, 0 overrun, 0 ignored, 1 abort
     518 packets output, 51626 bytes, 0 underruns
     0 output errors, 0 collisions, 40 interface resets
 0 output buffer failures, 0 output buffers swapped out
     8 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

We see that routes to RouterB are being learned from RIP via s 0/0 on RouterA.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       193.1.1.0/24 is directly connected, Serial0/0
C       193.1.1.2/32 is directly connected, Serial0/0
C    195.1.1.0/24 is directly connected, Ethernet0/0
R    196.1.1.0/24 [120/1] via 193.1.1.2, 00:00:05, Serial0/0
R    197.1.1.0/24 [120/1] via 193.1.1.2, 00:00:05, Serial0/0
```

Type **show interface bri 0/0** to display the BRI interface status. Notice that the interface is in a standby mode. The BRI interface in our previous labs was in an up/up (spoofing) condition. A backup interface is treated differently and stays in standby mode until a call is made.

```
RouterA#show interface bri 0/0
BRI0/0 is standby mode, line protocol is down
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Make sure that you can ping the far−end router (RouterB) at IP address 193.1.1.2 by using the **ping 193.1.1.2** command.

```
RouterA#ping 193.1.1.2

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 193.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 28/30/32 ms
```

Now let's connect to RouterB. We see that RouterB has learned a route to the 195.1.1.0 network (RouterA) via interface S0/0.

```
RouterB#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       193.1.1.1/32 is directly connected, Serial0/0
C       193.1.1.0/24 is directly connected, Serial0/0
R    195.1.1.0/24 [120/1] via 193.1.1.1, 00:00:02, Serial0/0
C    196.1.1.0/24 is directly connected, BRI0/0
C    197.1.1.0/24 is directly connected, Ethernet0/0
```

Make sure that S 0/0 is in an up/up state by entering the **show interface s 0/0** command.

```
RouterB#show interface s 0/0
Serial0/0 is up, line protocol is up
```

```
  Hardware is QUICC Serial
  Internet address is 193.1.1.2/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open
  Open: IPCP, CDP
.
.
```

We see that the BRI interface on RouterB is in an up/up (spoofing) state. Recall that the BRI interface on RouterA is in a standby mode because RouterA is using a backup interface.

```
RouterB#show interface bri 0/0
BRI0/0 is up, line protocol is up (spoofing) ← D channel is active
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.2/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Now let's reconnect to RouterA. Turn on ISDN call control debugging with the **debug isdn q931** command.

```
RouterA#debug isdn q931
ISDN Q931 packets debugging is on
```

Turn on PPP authentication tracing with the **debug ppp authen** command.

```
RouterA#debug ppp authen
PPP authentication debugging is on
```

You can check which debug commands are enabled with the **show debug** command.

```
RouterA#show debug
PPP:
  PPP authentication debugging is on
ISDN:
  ISDN Q931 packets debugging is on
```

Now we are going to start a ping between RouterA and RouterB. While the ping is running we will pull the serial cable out of interface S0/0 of RouterA. This will cause RouterA to activate the backup interface (bri0/0). Start an extended ping as shown below. The IP address that we will ping is 197.1.1.1, which is the Ethernet interface on RouterB. When the ping begins, the traffic will flow between RouterA and RouterB over the serial interface connecting the two routers. When the serial interface connection is broken, the traffic will start to flow over the ISDN interface.

```
RouterA#ping
Protocol [ip]:
Target IP address: 197.1.1.1
Repeat count [5]: 100000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000, 100-byte ICMP Echos to 197.1.1.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.....  ← Pull the cable

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to
down ← The serial interface will be declared down
%LINK-3-UPDOWN: Interface Serial0/0, changed state to down
```

```
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to down
%LINK-3-UPDOWN: Interface BRI0/0, changed state to up ← The backup interface
is activated
```

Following is the ISDN Q931 call control and CHAP authentication that takes place between RouterA and RouterB when a call is made.

```
ISDN BR0/0: TX ->  INFORMATION pd = 8  callref = (null)
        SPID Information i = '5101'
ISDN BR0/0: RX <-  INFORMATION pd = 8  callref = (null)
        ENDPOINT IDent i = 0xF081
ISDN BR0/0: Received EndPoint ID.
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x0E
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201' ← RouterA calls RouterB on B channel #1
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x8E
        Channel ID i = 0x89
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x8E
        Channel ID i = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x0E
PPP BRI0/0:1: Send CHAP Challenge id=7 ← CHAP Challenge B channel #1
PPP BRI0/0:1: CHAP Challenge id=7 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=7
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=7 received from RouterB
PPP BRI0/0:1: Send CHAP Success id=7 ← CHAP passed on B channel #1
PPP BRI0/0:1: remote passed CHAP authentication.
PPP BRI0/0 1: Passed CHAP authentication with remote
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
PPP Virtual-Access1: treating connection as a callin
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1,
changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:1 is now connected to 8995201
RouterB
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: TX ->  INFORMATION pd = 8  callref = (null)
        SPID Information i = '5102'
ISDN BR0/0: RX <-  INFORMATION pd = 8  callref = (null)
        ENDPOINT IDent i = 0xF082
ISDN BR0/0: Received EndPoint ID
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x0F
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201' ← RouterA calls RouterB on B channel #2
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x8F
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x8F
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up
PPP BRI0/0:2: treating connection as a callout
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x0F
PPP BRI0/0:2: Send CHAP Challenge id=7 ← CHAP Challenge for B channel #2
PPP BRI0/0:2: CHAP Challenge id=7 received from RouterB
PPP BRI0/0:2: Send CHAP Response id=7
PPP BRI0/0:2: CHAP response received from RouterB
PPP BRI0/0:2: CHAP Response id=7 received from RouterB
PPP BRI0/0:2: Send CHAP Success id=7 ← CHAP passed on B channel #2
PPP BRI0/0:2: remote passed CHAP authentication
PPP BRI0/0:2: Passed CHAP authentication with remote
```

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2,
changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to
8995201 RouterB
```

Once the backup interface is active, the pings will start to work again.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 98 percent (1594/1619), round-trip min/avg/max = 20/24/40 ms
```

When the ping has completed, type the **show isdn status** command to display the call status for the router's BRI interface. Notice that there are two active layer 3 calls. This is the two B channels that are now connected.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     2 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 2
     CCB:callid=8011, sapi=0, ces=1, B-chan=1
     CCB:callid=8013, sapi=0, ces=2, B-chan=2
 Total Allocated ISDN CCBs = 2
```

Type the **show ppp multilink** command to verify that the two connected B channels on the BRI have been bundled together into an MLPPP bundle. We see from the command's output that both channels bri 0/0:1 and bri 0/0:2 are connected in an MLPPP bundle.

```
RouterA#show ppp multi

Bundle RouterB, 2 members, Master link is Virtual-Access1
Dialer Interface is BRI0/0
  0 lost fragments, 11 reordered, 0 unassigned, sequence 0x99F/0x9A1 rcvd/sent
  0 discarded, 0 lost received, 29/255 load

Member Links: 2
BRI0/0:1
BRI0/0:2
```

Type the **show dialer** command to display statistics on the current ISDN call. Notice that the second B channel was brought up due to a multilink overload.

```
RouterA#show dialer

BRI0/0 - dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8995201                 14          5    00:00:37       successful
0 incoming call(s) have been screened.

BRI0/0=:1 - dialer type = ISDN
```

```
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: ip (s=196.1.1.1, d=255.255.255.255)
Time until disconnect 89 secs
Current call connected 00:00:38
Connected to 8995201 (RouterB)

BRI0/0=:2 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: Multilink bundle overloaded ← Dialer load threshold
                                           statement determines when another
                                           channel should be brought up
Time until disconnect 50 secs
Current call connected 00:00:39
Connected to 8995201 (RouterB)
```

Both of the B channels on the BRI will now be in an up/up state. Display the B−channel status with the **show interface bri 0/0:1** and **show interface bri 0/0:2** commands.

```
RouterA#show interface bri 0/0:1
BRI0/0:1 is up, line protocol is up ← B channel #1
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 19/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open, multilink Open
.
.
RouterA#show interface bri 0/0:2
BRI0/0:2 is up, line protocol is up ← B channel #2
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 15/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open, multilink Open
.
.
```

Interface s0/0 is now in a down/down state. It was this interface going down that caused the backup interface, bri0/0, to go into an active state and dial the far−end router.

```
RouterA#show interface s 0/0
Serial0/0 is down, line protocol is down
  Hardware is QUICC Serial
  Internet address is 193.1.1.1/24
  Backup interface BRI0/0, kickin load not set, kickout load not set
      failure delay 5 sec, secondary disable delay 20 sec
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Closed
  Closed: IPCP, CDP
  Last input 00:02:20, output 00:02:20, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75 /0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64 /0 (size/threshold/drops)
     Conversations  0/1  (active/max active)
     Reserved Conversations 0/0  (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     1075 packets input, 120020 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 1 giants, 0 throttles
     5 input errors, 1 CRC, 3 frame, 0 overrun, 0 ignored, 1 abort
     1113 packets output, 100942 bytes, 0 underruns
     0 output errors, 0 collisions, 45 interface resets
```

```
  0 output buffer failures, 0 output buffers swapped out
     11 carrier transitions
     DCD=up  DSR=up  DTR=down  RTS=down  CTS=up
```

Let's examine the routing table with the **show ip route** command. Notice that we have now learned a RIP route to the 197.1.1.0 network via 196.1.1.2. 197.1.1.0 is the Ethernet network on RouterB. 196.1.1.2 is the IP address of the BRI interface on RouterB. Our routing table now reflects the fact that our primary serial interface, s0/0, is down and our backup interface, bri0/0, is now our only connection to RouterB. Routes learned via s0/0 will be deleted as soon as that interface goes down.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

C    195.1.1.0/24 is directly connected, Ethernet0/0
     196.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       196.1.1.0/24 is directly connected, BRI0/0
C       196.1.1.2/32 is directly connected, BRI0/0
R    197.1.1.0/24 [120/1] via 196.1.1.2, 00:00:05, BRI0/0
```

Now we are going to start a ping and reconnect our serial cable to RouterB. Recall from the configuration of RouterA that the backup interface will activate 5 seconds after S0/0 goes down and will go inactive 20 seconds after S0/0 becomes active again.

Router A Configuration for interface S0/0

```
interface Serial0/0
 backup delay 5 20 ← The backup interface activates 5 seconds after S0/0 goes
                     down and deactivates 20 seconds after S0/0 goes back up
 backup interface BRI0/0
 ip address 193.1.1.1 255.255.255.0
 encapsulation ppp
 clockrate 64000
```

Start an extended ping to 197.1.1.1. Remember from the IP routing table that our traffic will flow over the bri0/0 interface.

```
RouterA#ping
Protocol [ip]:
Target IP address: 197.1.1.1
Repeat count [5]: 100000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000, 100−byte ICMP Echos to 197.1.1.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

During the ping, reconnect the serial cable to the s0/0 interface of RouterA. You will see interface s0/0 go to an up state, and several seconds later, the bri backup interface will go down.

```
%LINK−3−UPDOWN: Interface Serial0/0, changed state to
up ← s0/0 comes back up
PPP Serial0/0: treating connection as a dedicated line
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial0/0,
```

```
changed state to up
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down
← Backup interface

goes down
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed
state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed
state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to down
%ISDN-6-DISCONNECT: Interface BRI0/0:1  disconnected from unknown,
call lasted 163 seconds
%ISDN-6-DISCONNECT: Interface BRI0/0:2  disconnected from unknown,
call lasted 130 seconds
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
%LINK-5-CHANGED: Interface BRI0/0, changed state to standby mode
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to down

.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 99 percent (2243/2245), round-trip min/avg/max = 20/26/64 ms
```

Verify that the ISDN call has terminated with the **show isdn status** command. Notice that there are no layer 3 calls currently active.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     DEACTIVATED
    Layer 2 Status:
     TEI = 64, State = TEI_ASSIGNED
     TEI = 65, State = TEI_ASSIGNED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s) ←
The call has completed
    Activated dsl 0 CCBs = 0
 Total Allocated ISDN CCBs = 0
```

MLPPP status can be verified with the **show ppp multilink** command. Since all the ISDN BRI channels have disconnected, there will not be any active MLPPP bundles on the router.

```
RouterA#show ppp multi
No active bundles
```

Interface s0/0 should now be in an up/up state.

```
RouterA#show interface s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 193.1.1.1/24
  Backup interface BRI0/0, kickin load not set, kickout load not set
      failure delay 5 sec, secondary disable delay 20 sec
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
```

```
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open
  Open: IPCP, CDP
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/1  (active/max active)
     Reserved Conversations 0/0  (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 5000 bits/sec, 5 packets/sec
     2015 packets input, 216161 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 1 giants, 0 throttles
     5 input errors, 1 CRC, 3 frame, 0 overrun, 0 ignored, 1 abort
     2471 packets output, 240038 bytes, 0 underruns
     0 output errors, 0 collisions, 46 interface resets
  0 output buffer failures, 0 output buffers swapped out
     14 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Let's take a look at the routing table for our router. Notice that the route to the Ethernet interface (197.1.1.0) on RouterB is now being learned via 193.1.1.2, which is the serial interface on RouterB. The routes that were learned via the bri interface on network 196.1.1.0 have now been deleted from the routing table. In general, routes that are learned via a specific interface are deleted if that interface goes down.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       193.1.1.0/24 is directly connected, Serial0/0
C       193.1.1.2/32 is directly connected, Serial0/0
C    195.1.1.0/24 is directly connected, Ethernet0/0
R    196.1.1.0/24 [120/1] via 193.1.1.2, 00:00:09, Serial0/0
R    197.1.1.0/24 [120/1] via 193.1.1.2, 00:00:09, Serial0/0
```

Lab #4: Floating Static Routes

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a single ISDN BRI interface, a single serial interface, and an Ethernet interface
- Two ISDN BRI circuits
- A Cisco V.35 DCE/DTE crossover cable. If you can not get a crossover cable, you can use a Cisco DCE cable connected to a Cisco DTE cable.
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This configuration demonstrates how to use a floating static route for ISDN dial backup. A floating static route is a statically defined route that has a higher administrative cost than routes learned through the dynamic routing algorithm that is being run. For example, in this lab we will use RIP as our dynamic routing algorithm. Routes that are learned via RIP have an administrative distance of 120. In this lab, we will define a floating static route that has an administrative distance of 121. A route with a higher administrative distance than other routes will not be installed in the routing table until other routes to that same destination with lower administrative costs are removed.

ISDN backup with floating static routes uses static routes with high administrative distance in the following way. Two routes will exist to a given destination. One route will be learned dynamically via the primary leased line circuit. The second route will be defined statically and will be via an ISDN interface. This second route will have a higher administrative distance than the first route. This static route will only take effect when the dynamic route is deleted from the routing table. When this static route takes effect, it will cause the router to place an ISDN call. When the dynamic route is added back to the routing table, it will take precedence over the static route because it has a lower administrative distance. This will cause the ISDN circuit to disconnect the ISDN call.

The two routers are connected as shown in Figure 3−15. RouterA and RouterB are connected to an Adtran Atlas 800 ISDN switch. A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

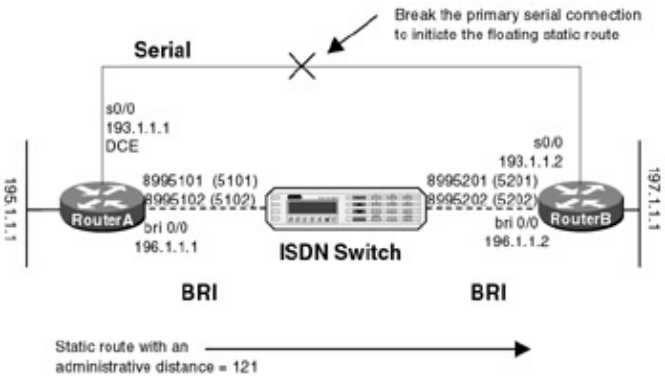


Figure 3−15: Floating static routes

Note Floating static routes have a big advantage over backup interfaces. When you use a floating static route, you can test the ISDN interface by pinging the far end of the ISDN circuit. This can be done while customer traffic is still traversing the router. A backup interface is more difficult to test since the primary interface has to be either brought down or have the backup statement removed from it.

Note RouterA is acting as a DCE on interface s0/0, providing clocking to RouterB.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. ISDN commands are highlighted in bold.

RouterA

```
RouterA#show run
Building configuration...
```

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
username RouterB password 7 070C285F4D06
isdn switch-type basic-ni1 ← Set D channel call control
!
interface Ethernet0/0
 ip address 195.1.1.1 255.255.255.0
 no keepalive
!
interface Serial0/0
 ip address 193.1.1.1 255.255.255.0
 encapsulation ppp
 clockrate 64000
!
 interface BRI0/0
 ip address 196.1.1.1 255.255.255.0
 encapsulation ppp
 isdn spid1 5101 8995101 ← Set SPID for both B channels
 isdn spid2 5102 8995102
 dialer idle-timeout 90 ← Disconnect 90 seconds after last interesting packet
 dialer map ip 196.1.1.2 name RouterB broadcast 8995201 ← Associate next hop
                                                          address and dialer
                                                          string
 dialer load-threshold 1 ← Define the threshold for adding additional B
                           channels
 dialer-group 1 ← Associate interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink
!
router rip
 network 195.1.1.0
 network 193.1.1.0
!
no ip classless
ip route 0.0.0.0 0.0.0.0 196.1.1.2 121 ← Floating Static route pointing to
                                         the BRI interface on RouterB
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
RouterB#show run
Building configuration...

Current configuration:
!
version 11.2
service timestamps debug datetime localtime
```

```
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
username RouterA password 7 094F471A1A0A
isdn switch-type basic-ni1 ← Set D channel call control
!
interface Ethernet0/0
 ip address 197.1.1.1 255.255.255.0
 no keepalive
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.0
 encapsulation ppp
!
 interface BRI0/0
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
 isdn spid1 5201 8995201 ← Set the SPID for both B channels
 isdn spid2 5202 8995202
 dialer idle-timeout 90 ← Define the interesting traffic timeout
 dialer map ip 196.1.1.1 name RouterA ← Define a next hop address
 dialer-group 1 ← Associate this interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink
!
router rip
 network 193.1.1.0
 network 197.1.1.0
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by examining the routing table of RouterA. We see two interesting items. First, notice that RouterA has learned a route to the Ethernet interface of RouterB (197.1.1.0). This route is via 193.1.1.2, which is the serial connection between RouterA and RouterB. The route is dynamically learned via RIP and has a default administrative distance of 120. Second, there is a static default route to 196.1.1.2 with an administrative distance of 121. This route will not take effect until the first route is deleted from the routing table.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is 196.1.1.2 to network 0.0.0.0

     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       193.1.1.0/24 is directly connected, Serial0/0
C       193.1.1.2/32 is directly connected, Serial0/0
```

```
C    195.1.1.0/24 is directly connected, Ethernet0/0
C    196.1.1.0/24 is directly connected, BRI0/0
R    197.1.1.0/24 [120/1] via 193.1.1.2, 00:00:12, Serial0/0
S*   0.0.0.0/0 [121/0] via 196.1.1.2 ← Floating Static with administrative
                                       distance of 121
```

Verify that the ISDN interface is ready to place a call by typing the **show isdn status** command. Notice that RouterA has sent a SPID to the ISDN switch for each of the two B channels on the BRI interface. Also, notice that there are no active layer 3 calls.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid ← SPID #1 is ok
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid ← SPID #2 is ok
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s) ← No active calls on the router
    Activated dsl 0 CCBs = 1
     CCB:callid=0, sapi=0, ces=1, B-chan=0
 Total Allocated ISDN CCBs = 1
```

Type **show interface bri 0/0** to view the status of the bri interface on the router. The up/up(spoofing) status indicates that the D channel between RouterA and the ISDN switch is active.

```
RouterA#show interface bri 0/0
BRI0/0 is up, line protocol is up (spoofing) ← D channel is active
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Type **show dialer** to view the status of the dialer on RouterA. We see that both B channels are in an idle state. We also see that this interface will dial 8995201 to place its call.

```
RouterA#show dialer

BRI0/0 - dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8995201                  6          0    00:28:40       successful
0 incoming call(s) have been screened.

BRI0/0:1 - dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle

BRI0/0:2 - dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle
```

Now let's connect to RouterB. View the routing table for RouterB with the **show ip route** command. RouterB has a route to the Ethernet interface (195.1.1.0) on RouterA via the serial 0/0 interface.

```
RouterB#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is 193.1.1.1 to network 0.0.0.0

     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       193.1.1.1/32 is directly connected, Serial0/0
C       193.1.1.0/24 is directly connected, Serial0/0
R    195.1.1.0/24 [120/1] via 193.1.1.1, 00:00:07, Serial0/0
C    196.1.1.0/24 is directly connected, BRI0/0
C    197.1.1.0/24 is directly connected, Ethernet0/0
R*   0.0.0.0/0 [120/1] via 193.1.1.1, 00:00:07, Serial0/0
```

Verify that the ISDN interface on RouterB is ready to receive a call with the **show isdn status** command. We see that the SPID for both B channels has been sent and that there are no active calls on RouterB.

```
RouterB#show isdn status
The current ISDN Switchtype = basic−ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 1
     CCB:callid=0, callref=0, sapi=0, ces=1, B−chan=0
    Number of active calls = 0
    Number of available B−channels = 2
 Total Allocated ISDN CCBs = 1
```

The **show dialer** command for RouterB reveals that the interface is configured to accept a call since there is no dial string associated with the interface.

```
RouterB#show dialer

BRI0/0 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
0 incoming call(s) have been screened.

BRI0/0:1 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is idle

BRI0/0:2 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
```

```
Dialer state is idle
```

Verify that the D channel is active on the bri0/0 interface of RouterB with the **show interface bri 0/0** command. The up/up(spoofing) status of the interface indicates that the D channel is active.

```
RouterB#show interface bri 0/0
BRI0/0 is up, line protocol is up (spoofing) ← Active D channel
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.2/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Let's reconnect to RouterA. Turn on PPP authentication debugging and ISDN Q931 debugging with the **debug ppp authen** and **debug isdn q931** commands. You can verify what debug commands are enabled with the **show debug** command.

```
RouterA#show debug
PPP:
  PPP authentication debugging is on
ISDN:
  ISDN Q931 packets debugging is on
```

A major advantage of a floating static route over a backup interface is the fact that a floating static route can be tested via a ping command while traffic flows over the primary interface. Let's activate the ISDN circuit with the **ping 196.1.1.2** command. Notice that the ping command causes the ISDN interface to activate and both B channels to connect.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:

ISDN BR0/0: TX -> SETUP pd = 8 callref = 0x07
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <- CALL_PROC pd = 8 callref = 0x87
        Channel ID i = 0x89
ISDN BR0/0: RX <- CONNECT pd = 8 callref = 0x87
        Channel ID i = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up ← Channel #1 is active
PPP BRI0/0:1: treating connection as a callout
ISDN BR0/0: TX -> CONNECT_ACK pd = 8 callref = 0x07
PPP BRI0/0:1: Send CHAP Challenge id=4 ← Chap challenge for channel #1
PPP BRI0/0:1: CHAP Challenge id=4 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=4
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=4 received from RouterB
PPP BRI0/0:1: Send CHAP Success id=4
PPP BRI0/0:1: remote passed CHAP authentication
PPP BRI0/0:1: Passed CHAP authentication with remote
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
PPP Virtual-Access1: treating connection as a callin
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed
state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
ISDN BR0/0: TX -> SETUP pd = 8 callref = 0x08
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <- CALL_PROC pd = 8 callref = 0x88
        Channel ID i = 0x8A
```

```
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x88
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up  ← Channel #2 is active
PPP BRI0/0:2: treating connection as a callout
RouterA#SDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x08
PPP BRI0/0:2: Send CHAP Challenge id=4 ← Chap challenge for channel #2
PPP BRI0/0:2: CHAP Challenge id=4 received from RouterB
PPP BRI0/0:2: Send CHAP Response id=4
PPP BRI0/0:2: CHAP response received from RouterB
PPP BRI0/0:2: CHAP Response id=4 received from RouterB
PPP BRI0/0:2: Send CHAP Success id=4
PPP BRI0/0:2: remote passed CHAP authentication
PPP BRI0/0:2: Passed CHAP authentication with remote
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2,
changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to
8995201 RouterB

..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max
= 20/22/24 ms
```

As we see, the ping was successful. The first two ping packets were lost because the interface was still coming up when the packets were being sent.

Let's make sure that an MLPPP bundle has been established. Type the **show ppp multilink** command. Notice that there is an active MLPPP bundle consisting of two members.

```
RouterA#show ppp multi

Bundle RouterB, 2 members, Master link is Virtual-Access1
Dialer Interface is BRI0/0
  0 lost fragments, 0 reordered, 0 unassigned, sequence 0x1A/0x1C
rcvd/sent
  0 discarded, 0 lost received, 1/255 load

Member Links: 2 ← MLPPP bundle consisting of both B channels on bri0/0
BRI0/0:1
BRI0/0:2
```

The **show isdn status** command will verify that there are two active calls on the router. Each of the B channels on the BRI is considered a separate call.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     2 Active Layer 3 Call(s) ← Both B channels on the BRI are connected
    Activated dsl 0 CCBs = 3
     CCB:callid=0, sapi=0, ces=1, B-chan=0
     CCB:callid=8007, sapi=0, ces=1, B-chan=1
```

```
     CCB:callid=8008, sapi=0, ces=2, B-chan=2
 Total Allocated ISDN CCBs = 3
```

After the test ping has completed, RouterA will wait 90 seconds before it brings down the ISDN circuit. This 90−second period is defined by the **dialer idle−timeout 90** statement in the configuration for RouterA.

```
%LINK-3-UPDOWN: Interface Virtual-Access1,
changed state to down
%ISDN-6-DISCONNECT: Interface BRI0/0:1  disconnected
from 8995201 RouterB, call lasted 210 seconds
%ISDN-6-DISCONNECT: Interface BRI0/0:2  disconnected
from 8995201 RouterB, call lasted 207 seconds
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x09 ← Disconnect B Channel #1
        Cause i = 0x8090 - Normal call clearing
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x0A ← Disconnect B Channel #2

        Cause i = 0x8090 - Normal call clearing
ISDN BR0/0: RX <-  RELEASE pd = 8  callref = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x09
ISDN BR0/0: RX <-  RELEASE pd = 8  callref = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x0A
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1,
changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2,
changed state to down
```

Now let's test the floating static route configuration. We are going to start a ping to the Ethernet interface of RouterB at IP address 197.1.1.1. After the ping starts, we will pull the serial cable connected to interface s0/0 of RouterA.

```
RouterA#ping ← Start an extended ping to the Ethernet interface of RouterB
Protocol [ip]:
Target IP address: 197.1.1.1
Repeat count [5]: 100000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000, 100-byte ICMP Echos to 197.1.1.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ← After the ping has started pull the
                                        cable on the s0/0 interface of RouterA
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to
down ← s0/0 will go down when the cable is pulled
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x09 ← RouterA is making a call to
                                                  RouterB on B channel #1
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x89
        Channel ID i = 0x89
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x89
        Channel ID i = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x09
PPP BRI0/0:1: Send CHAP Challenge id=5 ← Chap challenge for B channel #1
PPP BRI0/0:1: CHAP Challenge id=5 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=5
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=5 received from RouterB
PPP BRI0/0:1: Send CHAP Success id=5
```

```
PPP BRI0/0:1: remote passed CHAP authentication
PPP BRI0/0:1: Passed CHAP authentication with remote ← Chap successful for
                                                       B channel #1
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
PPP Virtual-Access1: treating connection as a callin
%LINK-3-UPDOWN: Interface Serial0/0, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x0A ← RouterA is making a call to
                                                  RouterB on B channel #2
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x8A
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x8A
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up
PPP BRI0/0:2: treating connection as a callout
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x0A
PPP BRI0/0:2: Send CHAP Challenge id=5 ← Chap challenge for B channel #2
PPP BRI0/0:2: CHAP Challenge id=5 received from RouterB
PPP BRI0/0:2: Send CHAP Response id=5
PPP BRI0/0:2: CHAP response received from RouterB
PPP BRI0/0:2: CHAP Response id=5 received from RouterB
PPP BRI0/0:2: Send CHAP Success id=5
PPP BRI0/0:2: remote passed CHAP authentication
PPP BRI0/0:2: Passed CHAP authentication with remote ← Chap successful for
                                                       B channel #1
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to 8995201 RouterB

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.
Success rate is 99 percent (2721/2725), round-trip min/avg/max
= 20/23/44 ms
```

Notice from the above ping results that the dial backup occurred very quickly. We only lost four pings during the time that interface s0/0 went down and we were able to make an ISDN backup call on interface bri0/0.

Let's view the routing table with the **show ip route** command. Notice that the RIP route to RouterB has been deleted from the routing table. This occurred because interface s0/0 went to a down state when the cable was pulled. Connectivity to RouterB is now being achieved via the static route that we defined with an administrative distance of 121.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is 196.1.1.2 to network 0.0.0.0

C    195.1.1.0/24 is directly connected, Ethernet0/0
     196.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       196.1.1.0/24 is directly connected, BRI0/0
C       196.1.1.2/32 is directly connected, BRI0/0
S*   0.0.0.0/0 [121/0] via 196.1.1.2
```

Let's type the **show ppp multilink** command to verify that we have established an MLPPP bundle between RouterA and RouterB.

```
RouterA#show ppp multi

Bundle RouterB, 2 members, Master link is Virtual-Access1
Dialer Interface is BRI0/0
  0 lost fragments, 17 reordered, 0 unassigned, sequence 0x1470/0x1474
rcvd/sent
  0 discarded, 0 lost received, 43/255 load

Member Links: 2 ← Both B channels are in an MLPPP bundle
BRI0/0:1
BRI0/0:2
```

Let's look at the status of the dialer by typing the **show dialer** command. Notice that the dial reason is listed for each of the two B channels of the BRI. The dial reason indicates what interesting traffic caused the ISDN circuit to place a call to the far-end router. The reason that B channel #1 dialed RouterB was due to the ping. This is written as **(s=196.1.1.1, d=197.1.1.1),** which means that traffic was sent to 197.1.1.1 (the Ethernet interface of RouterB) from 196.1.1.1 (the bri interface of RouterA). The reason that B channel #2 dialed was due to the MLPPP bundle being overloaded. This is determined by the **dialer load-threshold 1** statement in the configuration of RouterA.

```
RouterA#show dialer

BRI0/0 - dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8995201                 10          0    00:01:25       successful
0 incoming call(s) have been screened.

BRI0/0:1 - dialer type = ISDN ← B channel #1
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is physical layer up
Dial reason: ip (s=196.1.1.1, d=197.1.1.1) ← Ping to RouterB
Time until disconnect 71 secs
Current call connected 00:01:26
Connected to 8995201 (RouterB)

BRI0/0:2 - dialer type = ISDN ← B channel #2
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is physical layer up
Dial reason: Multilink bundle overloaded ← Dialer load threshold
Time until disconnect 73 secs
Current call connected 00:01:27
Connected to 8995201 (RouterB)
```

Now let's start another ping to the Ethernet interface of RouterB. This time we will reconnect the serial cable to RouterA while the ping is running.

```
RouterA#ping ← Start a ping to 197.1.1.1
Protocol [ip]:
Target IP address: 197.1.1.1
Repeat count [5]: 100000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000, 100-byte ICMP Echos to 197.1.1.1, timeout is 2 seconds:

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

%LINK-3-UPDOWN: Interface Serial0/0, changed state to up ← Reconnect the serial
                                                           cable between
                                                           RouterA and RouterB
PPP Serial0/0: treating connection as a dedicated line
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed
state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to down
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
%ISDN-6-DISCONNECT: Interface BRI0/0:1  disconnected from 8995201
RouterB, call lasted 210 seconds
%ISDN-6-DISCONNECT: Interface BRI0/0:2  disconnected from 8995201
RouterB, call lasted 207 seconds
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x09 ← B channel #1
                                                       disconnected
        Cause i = 0x8090 - Normal call clearing
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x0A ← B channel #2
                                                       disconnected
        Cause i = 0x8090 - Normal call clearing
ISDN BR0/0: RX <-  RELEASE pd = 8  callref = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x09
ISDN BR0/0: RX <-  RELEASE pd = 8  callref = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x0A
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed state to down


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!.
Success rate is 99 percent (4068/4070), round-trip min/avg/max = 20/26/60 ms
```

Notice that the router quickly switched the data path back to the s0/0 interface. We only lost two pings while the ISDN line disconnected and RouterA switched back to the s0/0 interface.

Let's make sure that the routing table for RouterA reflects the fact that interface s0/0 is now up and active. Type the **show ip route** command to view the routing table for RouterA. Notice that route to 197.1.1.0, which is learned via 193.1.1.2 from RIP, has reappeared in the routing table.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is 196.1.1.2 to network 0.0.0.0

     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       193.1.1.0/24 is directly connected, Serial0/0
C       193.1.1.2/32 is directly connected, Serial0/0
C    195.1.1.0/24 is directly connected, Ethernet0/0
C    196.1.1.0/24 is directly connected, BRI0/0
R    197.1.1.0/24 [120/1] via 193.1.1.2, 00:00:15, Serial0/0
S*   0.0.0.0/0 [121/0] via 196.1.1.2
```

Lab #5: Dialer Profiles

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a single ISDN BRI interface
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

Dialer profiles allow the configuration of physical interfaces to be separated from the logical configuration required for a call. Dialer profiles also allow the logical and physical configurations to be bound together dynamically on a per−call basis.

Dialer profiles consist of:

- A *dialer pool* of physical interfaces to be used by the dialer interface
- A *dialer interface* (a logical entity) configuration including one or more dial strings, each of which is used to reach one destination network

When dialer profiles are used for DDR configuration, the physical interface on the router has no configuration settings except link encapsulation, SPID information, and the dialer pools the interface belongs to.

RouterA and RouterB are configured as shown in Figure 3−16.

Figure 3−16: Dialer profiles

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. ISDN commands are highlighted in bold.

RouterA

```
RouterA#show run
Building configuration...

Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
```

```
!
enable password cisco
!
username RouterB password 7 070C285F4D06
isdn switch-type basic-ni1 ← Set D channel call control
!
interface BRI0/0
 no ip address
 encapsulation ppp
 isdn spid1 5101 8995101 ← Set SPIDs for both B channels
 isdn spid2 5102 8995102
 dialer pool-member 1 ← This interface will be part of dialer pool #1
 no fair-queue
 ppp authentication chap
 ppp multilink ← Request a PPP multilink session
!
interface Dialer0
 ip address 196.1.1.1 255.255.255.0
 encapsulation ppp
 dialer remote-name RouterB ← Hostname of far end router
 dialer idle-timeout 90 ← Disconnect the call 90 seconds after the last
                          interesting packet is received
 dialer string 8995201 ← Define number to dial to reach far end
 dialer load-threshold 1 ← Define threshold for adding additional B channels
 dialer pool 1 ← This is dialer pool #1
 dialer-group 1 ← Associate this interface with dialer-list 1
!
no ip classless
ip route 196.1.1.2 255.255.255.255 Dialer0
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
RouterB#show run
Building configuration...

Current configuration:
!
version 11.2
service timestamps debug datetime localtime
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
username RouterA password 7 094F471A1A0A
isdn switch-type basic-ni1 ← Set the D channel call control
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.0
 encapsulation ppp
!
interface BRI0/0
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
```

```
 isdn spid1 5201 8995201 ← Set the SPID for both B channels
 isdn spid2 5202 8995202
 dialer idle-timeout 90 ← Define the interesting traffic timeout
 dialer map ip 196.1.1.1 name RouterA ← Define a next hop address
 dialer-group 1 ← Associate the interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterB and verifying that the ISDN circuit is up and active. Type the **show isdn status** command to view the status of the BRI circuit on RouterB. We see that the SPID for both B channels has been sent to the ISDN switch and is valid.

```
RouterB#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid ← B channel #1
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid ← B channel #2
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 1
     CCB:callid=0, callref=0, sapi=0, ces=1, B-chan=0
    Number of active calls = 0
    Number of available B-channels = 2
 Total Allocated ISDN CCBs = 1
```

Let's verify the status of the bri interface and make sure that the D channel is active between RouterB and the ISDN switch. Type **show interface bri 0/0** to view the status of the D channel. The up/up (spoofing) state indicates that the D channel is up and active.

```
RouterB#show interface bri 0/0
BRI0/0  is up, line protocol is up (spoofing) ← Active D channel
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.2/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Now let's connect to RouterA and verify that the BRI interface is ready to place a call. Type the **show isdn status** command to view the status of the BRI interface. We see that a SPID for B channel #1 and B channel

#2 has been successfully sent to the switch.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 8(established)
         spid1 configured, spid1 sent, spid1 valid ← B channel #1
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 8(established)
         spid2 configured, spid2 sent, spid2 valid ← B channel #2
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 0
 Total Allocated ISDN CCBs = 0
```

Let's verify that the D channel for the BRI interface on RouterA is active. Type the **show interface bri 0/0** command to view the D channel. The up/up (spoofing) state of the interface indicates that the D channel between RouterA and the ISDN switch is active.

```
RouterA#show interface bri 0/0
BRI0/0  is up, line protocol is up (spoofing) ← Active D channel
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

Let's turn on PPP authentication and ISDN call control debugging with the **debug ppp authen** command and the **debug isdn q931** commands. Active debug commands can be displayed with the **show debug** command. Remember that you also need to type the **term mon** command if you are not connected to the router's console port.

```
RouterA#show debug
PPP:
  PPP authentication debugging is on
ISDN:
  ISDN Q931 packets debugging is on
Dial on demand:
  Dial on demand events debugging is on
```

Let's ping the BRI interface on RouterB at IP address 196.1.1.2. Notice that this ping will activate the BRI interface.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:

BRI0/0: Dialing cause ip (s=196.1.1.1, d=196.1.1.2)
BRI0/0: Attempting to dial 8995201
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x0B ← Placing call on B channel #1
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x8B
        Channel ID i = 0x89
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x8B
```

```
        Channel ID i = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
%DIALER-6-BIND: Interface BRI0/0:1 bound to profile Dialer0
PPP BRI0/0:1: treating connection as a callout
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x0B
PPP BRI0/0:1: Send CHAP Challenge id=5
PPP BRI0/0:1: CHAP Challenge id=5 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=5
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=5 received from RouterB
PPP BRI0/0:1: Send CHAP Success id=5
PPP BRI0/0:1: remote passed CHAP authentication
PPP BRI0/0:1: Passed CHAP authentication with remote
%DIALER-6-BIND: Interface Virtual-Access1 bound to profile Dialer0
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
PPP Virtual-Access1: treating connection as a callin
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
BRI0/0: Attempting to dial 8995201
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x0C ← Placing call on B channel #2
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x8C
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x8C
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up
%DIALER-6-BIND: Interface BRI0/0:2 bound to profile Dialer0
PPP BRI0/0:2: treating connection as a callout
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x0C
PPP BRI0/0:2: Send CHAP Challenge id=4
PPP BRI0/0:2: CHAP Challenge id=4 received from RouterB
PPP BRI0/0:2: Send CHAP Response id=4
PPP BRI0/0:2: CHAP response received from RouterB
PPP BRI0/0:2: CHAP Response id=4 received from RouterB
PPP BRI0/0:2: Send CHAP Success id=4
PPP BRI0/0:2: remote passed CHAP authentication.
PPP BRI0/0:2: Passed CHAP authentication with remote.
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to 8995201 RouterB

..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 20/21/24 ms
```

The **show isdn status** command will verify that we have two active calls on RouterA.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 8(established)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 8(established)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     2 Active Layer 3 Call(s) ← Both B channels on the BRI are active
    Activated dsl 0 CCBs = 2
```

```
    CCB:callid=800D, sapi=0, ces=1, B-chan=1
    CCB:callid=800E, sapi=0, ces=2, B-chan=2
 Total Allocated ISDN CCBs = 2
```

The dialer status indicates that the reason for the current call on B channel #1 was our ping. This is shown as IP traffic from a source of 196.1.1.1 (RouterA) to a destination of 196.1.1.2 (RouterB). The reason for the call on B channel #2 was the multilink bundle overload. This is determined by the **dialer load threshold** statement in the configuration for RouterA.

```
RouterA#show dialer

BRI0/0 - dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
0 incoming call(s) have been screened.

BRI0/0:1 - dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is physical layer up
Dial reason: ip (s=196.1.1.1, d=196.1.1.2) ← Ping from RouterA to RouterB
Interface bound to profile Dialer0
Time until disconnect 58 secs
Current call connected 00:00:28
Connected to 8995201 (RouterB)

BRI0/0:2 - dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is physical layer up
Dial reason: Multilink bundle overloaded ← Dialer load threshold
Interface bound to profile Dialer0
Time until disconnect 60 secs
Current call connected 00:00:29
Connected to 8995201 (RouterB)

Dialer0 - dialer type = DIALER PROFILE
Load threshold for dialing additional calls is 1
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is data link layer up

Dial String      Successes   Failures    Last called   Last status
8995201                 5          0    00:00:32       successful   Default
```

You can verify that RouterA and RouterB are communicating over an MLPPP by typing the **show ppp multilink** command. We see that there is an active multilink bundle consisting of two B channels.

```
RouterA#show ppp multi

Bundle RouterB, 2 members, Master link is Virtual-Access1
Dialer Interface is Dialer0
  0 lost fragments, 0 reordered, 0 unassigned, sequence 0x10/0x12 rcvd/sent
  0 discarded, 0 lost received, 1/255 load

Member Links: 2 ← 2 B channels in the MLPPP bundle
BRI0/0:1
BRI0/0:2
```

Lab #6: ISDN BRI to ISDN PRI

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, one of which must have a BRI interface and the other must have a PRI interface
- One ISDN PRI circuit and one ISDN BRI circuit
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This configuration will demonstrate a BRI connected router calling into a PRI connected router. PRI connected routers are used for large−scale access servers. A typical topology is to have multiple spoke sites that are BRI connected dialing into a PRI−enabled hub router.

The two routers are connected as shown in Figure 3−17. RouterA and RouterB are connected to an Adtran Atlas 800 ISDN switch.

Figure 3−17: ISDN BRI to ISDN PRI

A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

Note A PRI is different from a BRI. A PRI is carried on a T1 circuit and consists of 23 B channels, each carrying 56K or 64K of user traffic. A PRI has a 64K D channel used for signaling between the user device and the ISDN switch. A BRI consists of two B channels, each carrying 56K or 64K of user traffic. A BRI has a 16K D channel used for signaling between the user device and the ISDN switch.

Note A PRI ISDN circuit does not have a SPID associated with each B channel.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. ISDN commands are highlighted in bold.

RouterA

```
RouterA#show run
Building configuration...

Current configuration:
!
version 11.2
no service udp−small−servers
```

```
no service tcp-small-servers
!
hostname RouterA
!
!
username RouterB password 7 121A0C041104
isdn switch-type basic-ni1 ← Set D channel call control
!
interface BRI0/0
 ip address 196.1.1.1 255.255.255.0
 encapsulation ppp
 isdn spid1 8995101 5101 ← Set SPID value for both B channels
 isdn spid2 8995102 5102
 dialer idle-timeout 30 ← Define how many seconds to disconnect after last
                          interesting packet
 dialer map ip 196.1.1.7 name RouterB broadcast 8991000 ← Associate next hop
                                                          address with a dial
                                                          string
 dialer load-threshold 1 ← Define threshold to add additional B channels
 dialer-group 1 ← Associate interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink ← Request a PPP multilink session
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
RouterB#show run
Building configuration...

Current configuration:
!
version 11.2
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
username RouterA password 0 cisco
isdn switch-type primary-5ess ← Set D channel call control for the PRI
!
controller T1 0
 framing esf ← Set T1 Extended Superframe Framing
 linecode b8zs ← Set T1 line coding
 pri-group timeslots 1-24 ← Define entire T1 to belong to the PRI
!
interface Serial0:23 ← D channel of the PRI
 ip address 196.1.1.7 255.255.255.0
 encapsulation ppp
 no ip mroute-cache
 isdn incoming-voice modem
 dialer idle-timeout 900 ← Set the interesting traffic timeout
 dialer-group 1 ← Associate the interface with dialer-list 1
 no fair-queue
 no cdp enable
```

```
 ppp authentication chap
 ppp multilink
!
interface Dialer1
 no ip address
 encapsulation ppp
 dialer in-band
 dialer idle-timeout 900
 dialer-group 1
 no fair-queue
 ppp authentication chap
!
no ip classless
ip route 192.1.5.0 255.255.255.0 196.1.1.1
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 exec-timeout 60 0
 password cisco
 login
line aux 0
line vty 0 4
 login
!
end
```

Notice that the configuration for the PRI interface on RouterB is quite different from a BRI configuration. When configuring a PRI, you must first define a T1 controller interface and specify the proper T1 framing and line coding. The PRI gets configured as a serial interface:23, specifying the D channel of the PRI interface.

Monitoring and Testing the Configuration

Let's start by connecting to RouterA and verifying that the BRI circuit is up and active. Type the **show isdn status** command to display the status of the ISDN interface. We see that a SPID has been sent for both B channels to the ISDN switch and has been validated. We also see under the layer 3 status that there are no active calls on the router at this time.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid ← B channel #1
         Endpoint ID Info: epsf = 0, usid = B, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid ← B channel #2
         Endpoint ID Info: epsf = 0, usid = C, tid = 1
    Layer 3 Status:
     0 Active Layer 3 Call(s) ← No active calls
    Activated dsl 0 CCBs = 2
     CCB:callid=0, sapi=0, ces=1, B-chan=0
     CCB:callid=0, sapi=0, ces=1, B-chan=0
 Total Allocated ISDN CCBs = 2
```

The status of the D channel of the BRI circuit can be displayed by typing **show interface bri 0/0**. We see that the interface is in an up/up (spoofing) state, which indicates that the D channel is active.

```
RouterA#show interface bri 0/0
BRI0/0  is up, line protocol is up (spoofing) ← D channel of BRI
```

```
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
.
.
```

We see from the **show dialer** command that the BRI interface will dial 8991000 to place its calls. The idle timer is set to 30 seconds. This value is set by the **dialer idle−timeout 30** statement in the configuration for RouterA. It tells the router to disconnect a call 30 seconds after the last interesting packet has been received.

```
RouterA#show dialer

BRI0/0 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8991000                  5          0    00:07:48       successful
0 incoming call(s) have been screened.

BRI0/0:1 − dialer type = ISDN
Idle timer (30 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is idle

BRI0/0:2 − dialer type = ISDN
Idle timer (30 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is idle
```

Now let's connect to RouterB and view some PRI statistics. We will see that monitoring a PRI ISDN circuit is slightly different than monitoring a BRI circuit. Type the **show isdn status command**. We see that although there are 23 B channels, we will only get one Multiple_Frame_Established message. We also see that there are no indications of valid SPIDs being sent. This is because an ISDN PRI circuit does not have any SPIDs associated with it.

```
RouterB#show isdn status
The current ISDN Switchtype = primary−5ess
ISDN Serial0:23 interface
    Layer 1 Status:
        ACTIVE
    Layer 2 Status:
        TEI = 0, State = MULTIPLE_FRAME_ESTABLISHED
    Layer 3 Status:
        0 Active Layer 3 Call(s) ← No active calls at this time
    Activated dsl 0 CCBs = 0
    Total Allocated ISDN CCBs = 0
```

A PRI ISDN interface has an additional monitoring command. Type the **show isdn service** command. This command displays the B channel status of the entire PRI circuit. The state line shows which channels are currently connected, while the channel line shows which channels can accept or make a call. Possible states are:

- Idle
- Busy
- Reserved
- Restart
- Maint

We see that each B channel is in an idle state (State=0). The last eight channels are in a state of 3, which is reserved. These channels are used for an E1 PRI, which is the European counterpart to a T1. An E1 PRI has 31 B channels as opposed to the T1, which only has 23.

```
RouterB#show isdn service
PRI Channel Statistics:
ISDN Se0:23, Channel (1-31)
  Activated dsl 0
  State (0=Idle 1=Propose 2=Busy 3=Reserved 4=Restart 5=Maint)
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3
  Channel (1-31) Service (0=Inservice 1=Maint 2=Outofservice)
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

To display the status of the D channel of the PRI, use the **show interfaces 0:23** command. The s 0 corresponds to the **controller T1 0** command in the configuration for RouterB.

```
RouterB#show interface s 0:23
Serial0:23 is up, line protocol is up (spoofing) ← D channel of PRI
  Hardware is DSX1 ← T1 interface
  Internet address is 196.1.1.7/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
  Last input 00:00:08, output 00:00:08, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     1284 packets input, 5949 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1285 packets output, 5541 bytes, 0 underruns
     0 output errors, 0 collisions, 5 interface resets
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
  Timeslot(s) Used:24, Transmitter delay is 0 flags
```

The **show dialer** command will display the status of the dialer on the router. We notice some differences from how this command looks when used on a BRI interface. When used on a PRI, the command will display the status of each of the 23 B channels.

```
RouterB#show dialer

Dialer1 - dialer type = IN-BAND SYNC NO-PARITY
Idle timer (900 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)

Dial String      Successes   Failures    Last called   Last status

Serial0:0 - dialer type = ISDN
Idle timer (900 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle

Serial0:1 - dialer type = ISDN
Idle timer (900 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle

.
.
.
.
.


Serial0:22 - dialer type = ISDN
Idle timer (900 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle
```

```
Serial0:23 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
0 incoming call(s) have been screened.
```

Turn on PPP authentication and ISDN Q931 call control debugging with the **debug ppp authen** command and the **debug isdn q931** command. The status of what debug commands are active can be displayed by typing the **show debug** command. Remember to use the **term mon** command to display the debug output if you are not connected to the console port of the router.

```
RouterA#show debug
PPP:
  PPP authentication debugging is on
ISDN:
  ISDN Q931 packets debugging is on
```

Now let's try to ping RouterB at IP address 196.1.1.7. We see that an ISDN call is made as soon as we start our ping.

```
RouterA#ping 196.1.1.7

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 196.1.1.7, timeout is 2 seconds:

ISDN BR0/0: TX −>  SETUP pd = 8  callref = 0x06 ← B channel #1
        Bearer Capability I = 0x8890
        Channel ID I = 0x83
        Keypad Facility I = '98991000'
ISDN BR0/0: RX <−  SETUP_ACK pd = 8  callref = 0x86
        Channel ID I = 0x89.
ISDN BR0/0: RX <−  CALL_PROC pd = 8  callref = 0x86
ISDN BR0/0: RX <−  CONNECT pd = 8  callref = 0x86
        Signal I = 0x3F − Tones off
%LINK−3−UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
ISDN BR0/0: TX −>  CONNECT_ACK pd = 8  callref = 0x06
PPP BRI0/0:1: Send CHAP Challenge id=5 ← Chap successful on B channel #1
PPP BRI0/0:1: CHAP Challenge id=6 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=6
PPP BRI0/0:1: Passed CHAP authentication with remote
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=5 received from RouterB
PPP BRI0/0:1: Send CHAP Success id=5
PPP BRI0/0:1: remote passed CHAP authentication.
%LINK−3−UPDOWN: Interface Virtual−Access1, changed state to up
PPP Virtual−Access1: treating connection as a callin
%LINEPROTO−5−UPDOWN: Line protocol on Interface BRI0/0:1, changed state to up
%LINEPROTO−5−UPDOWN: Line protocol on Interface Virtual−Access1, changed state to up
ISDN BR0/0: TX −>  SETUP pd = 8  callref = 0x07 ← B channel #2
        Bearer Capability I = 0x8890
        Channel ID I = 0x83
        Keypad Facility I = '98991000'
ISDN BR0/0: RX <−  SETUP_ACK pd = 8  callref = 0x87
        Channel ID I = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%ISDN−6−CONNECT: Interface BRI0/0:1 is now connected to 98991000 RouterB
ISDN BR0/0: RX <−  CALL_PROC pd = 8  callref = 0x87
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: RX <−  CONNECT pd = 8  callref = 0x87
        Signal I = 0x3F − Tones off
ISDN BR0/0: Event: incoming ces value = 2
%LINK−3−UPDOWN: Interface BRI0/0:2, changed state to up
PPP BRI0/0:2: treating connection as a callout
ISDN BR0/0: TX −>  CONNECT_ACK pd = 8  callref = 0x07
PPP BRI0/0:2: Send CHAP Challenge id=2
```

```
PPP BRI0/0:2: CHAP Challenge id=3 received from RouterB
PPP BRI0/0:2: Send CHAP Response id=3
PPP BRI0/0:2: Passed CHAP authentication with remote
PPP BRI0/0:2: CHAP response received from RouterB
PPP BRI0/0:2: CHAP Response id=2 received from RouterB
PPP BRI0/0:2: Send CHAP Success id=2 ← Chap successful on B channel #1
PPP BRI0/0:2: remote passed CHAP authentication.
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to 98991000 RouterB


..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 32/32/32 ms
RouterA#
```

Notice that the first two pings were not successful. This is because the router was still in the process of establishing the ISDN call. Another ping to 196.1.1.7 would be 100−percent successful.

Let's verify that we have an MLPPP bundle between RouterA and RouterB. Type the **show ppp multilink** command to view the status of the MLPPP Link. We see that there are two B channels in the MLPPP bundle. These are the two B channels of the BRI.

```
RouterA#show ppp multi

Bundle RouterB, 2 members, Master link is Virtual-Access1
Dialer Interface is BRI0/0
  0 lost fragments, 0 reordered, 0 unassigned, sequence 0xA/0xA rcvd/sent
  0 discarded, 0 lost received, 1/255  load

Member Links: 2 ← B channel #1 and B channel #2
BRI0/0:1
BRI0/0:2
```

The **show isdn status** command can also be used to verify that we have two active calls on the router.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0 interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = B, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = C, tid = 1
    Layer 3 Status:
     2 Active Layer 3 Call(s) ← Active call on both B channels
    Activated dsl 0 CCBs = 4
     CCB:callid=0, sapi=0, ces=1, B-chan=0
     CCB:callid=0, sapi=0, ces=1, B-chan=0
     CCB:callid=8006, sapi=0, ces=1, B-chan=1
     CCB:callid=8007, sapi=0, ces=2, B-chan=2
 Total Allocated ISDN CCBs = 4
```

After the idle timeout period of 30 seconds, RouterA will disconnect the ISDN call. This period is defined by the **dialer idle−timeout 30** statement on RouterA's configuration.

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface
Virtual-Access1, changed state to down
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
```

```
%ISDN-6-DISCONNECT: Interface BRI0/0:1  disconnected from 98991000
RouterB, call lasted 55 seconds
%ISDN-6-DISCONNECT: Interface BRI0/0:2  disconnected from 98991000
RouterB, call lasted 51 seconds
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x06 ← RouterA sends a
                                                       disconnect to the ISDN
                                                       switch for B channel #1
        Cause i = 0x8090 - Normal call clearing
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x07 ← RouterA sends a
                                                       disconnect to the ISDN
                                                       switch for B channel #2
        Cause i = 0x8090 - Normal call clearing
ISDN BR0/0: RX <-  RELEASE pd = 8  callref = 0x86
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x06
ISDN BR0/0: RX <-  RELEASE pd = 8  callref = 0x87
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x07
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1,
changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2,
changed state to down
```

Now let's connect to RouterB and see what an incoming call looks like on the ISDN PRI interface. You can either attach a second terminal to RouterB so that you can place a call on RouterA with a ping and monitor RouterB at the same time or you can log the terminal output on RouterB to a log file.

Make sure that PPP authentication and ISDN Q931 call control debugging are enabled on RouterB by typing the **debug ppp authen** and **debug isdn q931** commands. You can verify what debug commands are enabled on the router by typing the **show debug** command. Remember to use the **term mon** command to display the debug output if you are not connected to the console port of the router.

```
RouterB#show debug
PPP:
  PPP authentication debugging is on
ISDN:
  ISDN Q931 packets debugging is on
```

The following is a trace on RouterB while a call is coming in from RouterA.

The D channel of the PRI is referenced as Se0:23

```
      ↓<ΛINE/>ISDN Se0:23: RX <-  SETUP pd = 8  callref = 0x0C
        Bearer Capability i = 0x8890
        Channel ID i = 0xA98393
        Calling Party Number i = '!', 0x80, '8995201' ← Calling number
        Called Party Number i = 0xA1, '8991000' ← Called number
```

The first call that comes into the PRI connects to channel 18

```
                              ↓
%LINK-3-UPDOWN: Interface Serial0:18, changed state to up
Se0:18 PPP: Treating connection as a callin
ISDN Se0:23: TX ->  CALL_PROC pd = 8  callref = 0x800C
        Channel ID i = 0xA98393
ISDN Se0:23: TX ->  CONNECT pd = 8  callref = 0x800C
        Channel ID i = 0xA98393
ISDN Se0:23: RX <-  CONNECT_ACK pd = 8  callref = 0x0C
Se0:18 PPP: Phase is AUTHENTICATING, by both
Se0:18 CHAP: O CHALLENGE id 7 len 29 from "RouterB"
Se0:18 CHAP: I CHALLENGE id 6 len 28 from "RouterA"
Se0:18 CHAP: Waiting for peer to authenticate first
Se0:18 CHAP: I RESPONSE id 7 len 28 from "RouterA"
Se0:18 CHAP: O SUCCESS id 7 len 4
Se0:18 CHAP: Processing saved Challenge, id 6
```

```
Se0:18 CHAP: O RESPONSE id 6 len 29 from "RouterB"
Se0:18 CHAP: I SUCCESS id 6 len 4
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
Vi1 PPP: Treating connection as a callin
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0:18, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed
state to up
ISDN Se0:23: RX <-  SETUP pd = 8  callref = 0x35
        Bearer Capability i = 0x8890
        Channel ID i = 0xA98394
        Calling Party Number i = '!', 0x80, '8995201'
        Called Party Number i = 0xA1, '8991000'

The second call that comes into the PRI connects to channel 19
                              ↓
%LINK-3-UPDOWN: Interface Serial0:19, changed state to up
Se0:19 PPP: Treating connection as a callin
ISDN Se0:23: TX ->  CALL_PROC pd = 8  callref = 0x8035
        Channel ID i = 0xA98394
ISDN Se0:23: TX ->  CONNECT pd = 8  callref = 0x8035
        Channel ID i = 0xA98394
ISDN Se0:23: RX <-  CONNECT_ACK pd = 8  callref = 0x35
Se0:19 PPP: Phase is AUTHENTICATING, by both
Se0:19 CHAP: O CHALLENGE id 4 len 29 from "RouterB"
Se0:19 CHAP: I CHALLENGE id 3 len 28 from "RouterA"
Se0:19 CHAP: Waiting for peer to authenticate first
Se0:19 CHAP: I RESPONSE id 4 len 28 from "RouterA"
Se0:19 CHAP: O SUCCESS id 4 len 4
Se0:19 CHAP: Processing saved Challenge, id 3
Se0:19 CHAP: O RESPONSE id 3 len 29 from "RouterB"
Se0:19 CHAP: I SUCCESS id 3 len 4
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0:19, changed state to up
%ISDN-6-CONNECT: Interface Serial0:19 is now connected to 9148993601 RouterA
```

The **show ppp multilink** command on RouterB will reveal that two B channels are active in an MLPPP bundle.

```
RouterB#show ppp multi

Bundle RouterA, 2 members, Master link is Virtual-Access1
Dialer Interface is Serial0:23
  0 lost fragments, 0 reordered, 0 unassigned, sequence 0x0/0x0 rcvd/sent
  0 discarded, 0 lost received, 1/255 load

Member Links: 2
Serial0:18
Serial0:19
```

Type the **show dialer maps** command to view the dynamic dialer map that gets created when RouterA dials into RouterB.

```
RouterB#show dialer maps
Dynamic dialer map ip 196.1.1.1 name RouterA () on Serial0:23
```

The **show isdn service** command shows us that there are two active B channels on the PRI. The active channels are denoted by a 2 in the appropriate channel position of the PRI.

```
RouterB#show isdn service
PRI Channel Statistics:
ISDN Se0:23, Channel (1-31)
  Activated dsl 0
  State (0=Idle 1=Propose 2=Busy 3=Reserved 4=Restart 5=Maint)
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 3 3 3 3 3 3 3 3
  Channel (1-31) Service (0=Inservice 1=Maint 2=Outofservice)
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

When the call disconnects on the PRI, we see that channels 18 and 19 receive a disconnect message from the ISDN switch. Remember that the far−end router (RouterA) is disconnecting the call so RouterB will receive a Disconnect message from the network.

```
%LINK−3−UPDOWN: Interface Virtual−Access1, changed state to down
ISDN Se0:23: RX <−  DISCONNECT pd = 8  callref = 0x0C ← Disconnect for
                                                        Channel 18
        Cause i = 0x8090 − Normal call clearing
%ISDN−6−DISCONNECT: Interface Serial0:18  disconnected from 9148993601 RouterA,
call lasted 32 seconds
%LINK−3−UPDOWN: Interface Serial0:18, changed state to down
ISDN Se0:23: TX −>  RELEASE pd = 8  callref = 0x800C
ISDN Se0:23: RX <−  DISCONNECT pd = 8  callref = 0x35 ← Disconnect for
                                                        Channel 19
        Cause i = 0x8090 − Normal call clearing
%ISDN−6−DISCONNECT: Interface Serial0:19  disconnected from 9148993601 RouterA,
call lasted 27 seconds
%LINK−3−UPDOWN: Interface Serial0:19, changed state to down
ISDN Se0:23: TX −>  RELEASE pd = 8  callref = 0x8035
ISDN Se0:23: RX <−  RELEASE_COMP pd = 8  callref = 0x0C
ISDN Se0:23: RX <−  RELEASE_COMP pd = 8  callref = 0x35
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial0:18, changed state to down
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial0:19, changed state to down
%LINEPROTO−5−UPDOWN: Line protocol on Interface Virtual−Access1, changed state to down
```

Once the PRI call is disconnected, the **show isdn service** command output will reveal that there are no connected B channels on the PRI.

```
RouterB#show isdn service
PRI Channel Statistics:
ISDN Se0:23, Channel (1−31)
  Activated dsl 0
  State (0=Idle 1=Propose 2=Busy 3=Reserved 4=Restart 5=Maint)
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3
  Channel (1−31) Service (0=Inservice 1=Maint 2=Outofservice)
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Since the PRI interface on this router contains a full T1 CSU, you can type the **show cont t 0** command to view the status of the T1 ESF registers on the interface. Data is broken into the previous 24 hours of performance information. The 24−hour statistics are broken up into 96 intervals, each representing 15 minutes of error information.

```
RouterB#show cont t 0
T1 0 is up.
  No alarms detected.
  Framing is ESF, Line Code is B8ZS, Clock Source is Line.
  Data in current interval (256 seconds elapsed):
     0 Line Code Violations, 0 Path Code Violations
     0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
     0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
  Data in Interval 1: ← A single 15 minute interval
     0 Line Code Violations, 0 Path Code Violations
     0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
     0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
  Data in Interval 2:
     0 Line Code Violations, 0 Path Code Violations
     0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
     0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
 .
 .
 .
 .

  Total Data (last 13 15 minute intervals):
     0 Line Code Violations, 0 Path Code Violations,
```

```
1 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins,
0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 1 Unavail Secs
```

Lab #7: Snapshot Routing

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a BRI interface
- Two ISDN BRI circuits
- Cisco IOS 11.3 or higher
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This lab will demonstrate snapshot routing. Snapshot routing allows an ISDN hub and spoke network to be built without configuring and maintaining static routes. Snapshot is only supported for distance vector routing protocols such as RIP and IGRP for IP traffic. Without snapshot routing, running RIP on an ISDN interface would mean that every 30 seconds a call would be made (assuming a call was not already up) to exchange updates. Snapshot defines an active period and a quiet period. During the active period, a RIP−enabled snapshot router will exchange routing updates. If there are no active calls, the snapshot router will initiate an ISDN call during the active period to send a routing update. During the quiet period, a snapshot router will not initiate a call to send a routing update. Snapshot routing freezes entries in the routing table during the quiet period. The active and quiet periods are user defined. The minimum active period is 5 minutes and the minimum quiet period is 8 minutes.

Any calls that bring up the ISDN interface will also reset the snapshot routing process to the beginning of a new active period.

The two routers are connected as shown in Figure 3−18. RouterA and RouterB are connected to an Adtran Atlas 800 ISDN switch.

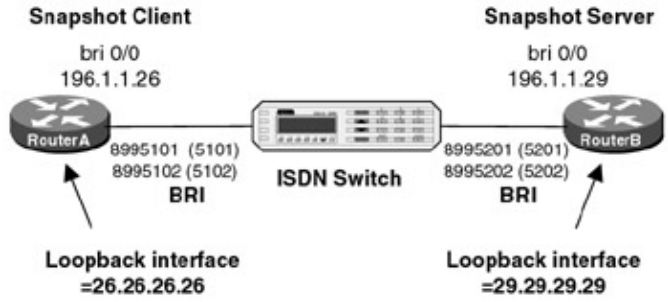


Figure 3−18: Snapshot routing

A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

Note Some versions of the IOS do not support snapshot routing with MLPPP. Do not use a ppp multilink statement in your router configuration.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. ISDN commands are highlighted in bold.

RouterA

```
RouterA#show run
Building configuration...

Current configuration:
!
version 11.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
username RouterB password 0 cisco
isdn switch-type basic-ni ← Set the D channel call control
!
!
!
interface Loopback0
 ip address 26.26.26.26 255.255.255.0
!
interface BRI0/0
 ip address 196.1.1.26 255.255.255.0
 encapsulation ppp
 dialer map snapshot 1 name RouterB broadcast 8995201 ← Define the dial string
                                                         for snapshot updates
 dialer map ip 196.1.1.29 name RouterB broadcast 8995201 ← Define next hop
                                                            address and dial
                                                            string
 dialer-group 1 ← Associate this interface with dialer-list 1
 isdn switch-type basic-ni
 isdn spid1 5101 8995101 ← Define the SPID for both B channels
 isdn spid2 5102 8995102
 snapshot client 5 8 dialer ← Define this router as a snapshot client.
                              The active time is 5 minutes and the quiet time
                              is 8 minutes.
 no fair-queue
 no cdp enable
 ppp authentication chap
!
interface Serial0/0
 no ip address
!
router rip
 network 26.0.0.0
 network 196.1.1.0
!
no ip classless
!
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
```

```
 login
!
end
```

RouterB

```
RouterB#show run
Building configuration...

Current configuration:
!
version 11.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
enable password cisco
!
username RouterA password 7 060506324F41
isdn switch-type basic-ni ← Set the D channel call control
!
!
!
interface Loopback0
 ip address 29.29.29.29 255.255.255.0
!
interface BRI0/0
 ip address 196.1.1.29 255.255.255.0
 encapsulation ppp
 dialer map ip 196.1.1.26 name RouterA broadcast ← Define the next hop address
 dialer-group 1 ← Associate this interface with dialer-list 1
 isdn switch-type basic-ni
 isdn spid1 5101 8995101 ← Define the SPID for both B channels
 isdn spid2 5102 8995102
 snapshot server 5 dialer ← Define this router as a snapshot server. The active
                              time of 5 minutes must match the active time on the
                              snapshot client
 no fair-queue
 no cdp enable
 ppp authentication chap
hold-queue 75 in
!
router rip
 network 29.0.0.0
 network 196.1.1.0
!
no ip classless
!
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterB and verifying that the ISDN circuit is up and active. Type the **show isdn**

status command to view the ISDN BRI status information.

```
RouterB#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI0/0  interface
        dsl 0, interface ISDN Switchtype = basic-ni
    Layer 1 Status:
        ACTIVE
    Layer 2 Status:
        TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
        TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
        TEI 64, ces = 1, state = 5(init)
            spid1 configured, spid1 sent, spid1 valid
            Endpoint ID Info: epsf = 0, usid = 0, tid = 1
        TEI 65, ces = 2, state = 5(init)
            spid2 configured, spid2 sent, spid2 valid
            Endpoint ID Info: epsf = 0, usid = 1, tid = 1
    Layer 3 Status:
        0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 0
    Total Allocated ISDN CCBs = 0
```

RouterB is provisioned for snapshot routing. Type the **show snap** command to view snapshot information. We see that RouterB is a snapshot server.

```
RouterB#show snap
BRI0/0 is up, line protocol is up Snapshot server
  Options: dialer support
  Length of active period:          5 minutes
   For ip address: 196.1.1.26
    Current state: active, remaining time: 1 minute
    Connected dialer interface:
       BRI0/0:1
```

Now let's connect to RouterA. Verify that the ISDN circuit is active with the **show isdn status** command.

```
RouterA#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI0/0  interface
        dsl 0, interface ISDN Switchtype = basic-ni
    Layer 1 Status:
        ACTIVE
    Layer 2 Status:
        TEI = 80, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
        TEI = 89, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
        TEI 80, ces = 1, state = 5(init)
            spid1 configured, spid1 sent, spid1 valid
            Endpoint ID Info: epsf = 0, usid = 0, tid = 1
        TEI 89, ces = 2, state = 5(init)
            spid2 configured, spid2 sent, spid2 valid
            Endpoint ID Info: epsf = 0, usid = 1, tid = 1
    Layer 3 Status:
        0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 0
    Total Allocated ISDN CCBs = 0
```

The **show dialer maps** command will display information about any dialer maps configured on the router. We see that RouterA has two dialer maps configured. The first dialer map is a snapshot dialer map used for snapshot routing. The second dialer map is the map used for defining the next hop address to RouterB.

```
RouterA#show dialer maps
Static dialer map snapshot 1 name RouterB broadcast (8995201) on BRI0/0
Static dialer map ip 196.1.1.29 name RouterB broadcast (8995201) on BRI0/0
```

We see from the **show snap** command on RouterA that RouterA is a snapshot client. RouterA is currently in the quiet state. This means that Router A will not initiate an ISDN call to send out RIP routing updates.

```
RouterA#show snap
BRI0/0 is up, line protocol is up Snapshot client
  Options: dialer support
  Length of active period:          5 minutes
  Length of quiet period:           8 minutes
  Length of retry period:           8 minutes
   For dialer address 1
    Current state: quiet, remaining: 6 minutes
```

The quiet period is defined to be 8 minutes. During the quiet period, connect to RouterB and examine its routing table. We see that the route to the loopback on RouterA (26.0.0.0/8 [120/1] via 196.1.1.26, 00:04:31, BRI0/0) is being aged, but is not being deleted from the routing table. Without snapshot, the route would be deleted as soon as the BRI disconnected. With snapshot, the route is kept in the routing table and is not deleted.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

C    196.1.1.0/24 is directly connected, BRI0/0
R    26.0.0.0/8 [120/1] via 196.1.1.26, 00:04:31, BRI0/0
     29.0.0.0/24 is subnetted, 1 subnets
C       29.29.29.0 is directly connected, Loopback0
```

We see that the route ages to 7 minutes and 58 seconds. Notice that it is still in the routing table. Without snapshot, a route would have been removed from the routing table if an update had not been received for this amount of time.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

C    196.1.1.0/24 is directly connected, BRI0/0
R    26.0.0.0/8 [120/1] via 196.1.1.26, 00:07:58, BRI0/0
     29.0.0.0/24 is subnetted, 1 subnets
C       29.29.29.0 is directly connected, Loopback0
```

The snapshot timers will continue to decrement. After 6 more minutes, the timer will show zero minutes.

```
RouterA#show snap
BRI0/0 is up, line protocol is upSnapshot client
  Options: dialer support
  Length of active period:          5 minutes
  Length of quiet period:           8 minutes
  Length of retry period:           8 minutes
   For dialer address 1
    Current state: quiet, remaining: 0 minutes
```

After the quiet period expires, snapshot will enter the active period. RouterA will now initiate an ISDN call to send out routing updates.

```
21:09:39: %LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
21:09:39: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
21:09:39: RT: network 196.1.1.0 is now variably masked
21:09:39: RT: add 196.1.1.29/32  via 0.0.0.0, connected metric [0/0]
21:09:39: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed stat
e to up
21:09:39: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, chang
ed state to up
21:09:45: %ISDN-6-CONNECT: Interface BRI0/0:1 is now connected to 8995201
RouterB
```

The **show snap** command now shows that RouterA is in the 5-minute active period during which it will send out RIP updates. If the ISDN circuit is not connected, snapshot will initiate the ISDN circuit to place the call.

```
RouterA#show snap
BRI0/0 is up, line protocol is up Snapshot client
  Options: dialer support
  Length of active period:          5 minutes
  Length of quiet period:           8 minutes
  Length of retry period:           8 minutes
   For dialer address 1
    Current state: active, remaining/exchange time: 5/0 minutes
    Connected dialer interface:
       BRI0/0:1
      Updates received this cycle:  ip
```

Now that snapshot is in the active state, reconnect to RouterB and view the routing table with the **show ip route** command. Notice that the route to RouterA is still in the table but it has now been updated in the last 6 seconds. Since snapshot is in the active state, it is now sending RIP updates again.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     196.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       196.1.1.0/24 is directly connected, BRI0/0
C       196.1.1.26/32 is directly connected, BRI0/0
R    26.0.0.0/8 [120/1] via 196.1.1.26, 00:00:06, BRI0/0
     29.0.0.0/24 is subnetted, 1 subnets
C       29.29.29.0 is directly connected, Loopback0
```

Lab #8: OSPF Demand Circuits

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a single ISDN BRI interface. One of the routers also needs an Ethernet interface
- Cisco IOS 11.2 or higher
- Two ISDN BRI circuits

- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This configuration will demonstrate an OSPF demand circuit. With OSPF demand circuits, periodic hellos are suppressed and periodic refreshes of LSAs are not flooded over the ISDN link. The ISDN link will be brought up initially to exchange routing information. After initial route exchanges, the link will only be activated when there is a change in the routing topology. Without demand circuits, the OSPF periodic hello and LSA updates will keep the ISDN circuit active even though there are no changes to the routing table.

RouterA and RouterB are connected as shown in Figure 3−19.

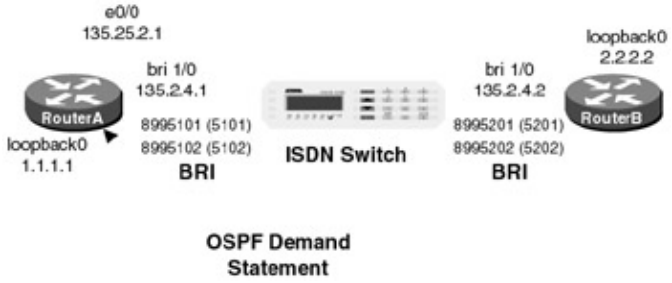


Figure 3−19: OSPF demand circuits

A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. OSPF demand commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
!
username RouterB password 0 cisco
!
ip subnet-zero
!
lane client flush
isdn switch-type basic-ni
cns event-service server
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface Ethernet0/0
```

```
 ip address 135.25.2.1 255.255.252.0
 no keepalive
!
interface BRI1/0
 ip address 135.2.4.1 255.255.252.0
 encapsulation ppp
 ip ospf demand-circuit ← RouterA is configured with the ip ospf demand-circuit
                          command
 dialer map ip 135.2.4.2 name RouterB broadcast 8995201
 dialer load-threshold 255 either
 dialer-group 1
 isdn switch-type basic-ni
 isdn spid1 5101 8995101
 isdn spid2 5102 8995102
 ppp authentication chap
!
router ospf 64
 network 1.0.0.0 0.255.255.255 area 0
 network 135.0.0.0 0.255.255.255 area 0
!
ip classless
no ip http server
!
access-list 100 permit ip any any
access-list 100 permit icmp any any
dialer-list 1 protocol ip list 100
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
username RouterA password 0 cisco
!
ip subnet-zero
!
lane client flush
isdn switch-type basic-ni
cns event-service server
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
interface BRI1/0
 ip address 135.2.4.2 255.255.252.0
 encapsulation ppp
 dialer map ip 135.2.4.1 name RouterA broadcast
 dialer load-threshold 255 either
 dialer-group 1
 isdn switch-type basic-ni
 isdn spid1 5201 8995201
 isdn spid2 5202 8995202
 ppp authentication chap
```

```
!
router ospf 64
 network 2.0.0.0 0.255.255.255 area 0
 network 135.0.0.0 0.255.255.255 area 0
!
ip classless
no ip http server
!
dialer-list 1 protocol ip permit
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Verify that the ISDN circuit is up and active with the **show isdn status** command. We see that both SPIDs have been sent to the switch and are valid. Also notice that there are no active calls on RouterA. This is important to note since we are running OSPF over the ISDN interface. We will see shortly that our routing table is maintaining active OSPF routes without keeping the ISDN circuit active at all times.

```
RouterA# show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI1/0  interface
     dsl 8, interface ISDN Switchtype = basic-ni
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 0
    The Free Channel Mask:  0x80000003
    Total Allocated ISDN CCBs = 0
```

Let's view the routing table on RouterA with the **show ip route** command. We see that RouterA has learned about the 2.2.2.2 network via the BRI interface. Recall that the BRI interface is not currently active (no calls exist on the router). When RouterA initially powers on, the ISDN circuit will activate so that OSPF routes can be exchanged. After the initial exchange of routes, OSPF demand will bring down the ISDN call. The before OSPF demand circuit keeps the routing table entries active even though the ISDN circuit is not active. OSPF keepalive messages are suppressed.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set
```

```
     1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1is directly connected, Loopback0
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/1563] via 135.2.4.2, 00:06:07, BRI1/0
     135.25.0.0/22 is subnetted, 1 subnets
C       135.25.0.0 is directly connected, Ethernet0/0
     135.2.0.0/22 is subnetted, 1 subnets
C       135.2.4.0 is directly connected, BRI1/0
```

Let's get some information on the OSPF configuration of RouterA with the **show ip ospf interface bri 1/0** command. We see that the interface is configured as a demand circuit. We also see that the OSPF hello messages are being suppressed. RouterA is keeping its OSPF adjacencies even though the ISDN circuit is not active.

```
RouterA#sh ip ospf int bri 1/0
BRI1/0 is up, line protocol is up (spoofing)
  Internet Address 135.2.4.1/22, Area 0
  Process ID 64, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost: 1562
  Configured as demand circuit.
  Run as demand circuit.
  DoNotAge LSA allowed.
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:00
  Index 3/3 , flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 2.2.2.2  (Hello suppressed)
  Suppress hello for 1 neighbor(s)
```

Now let's connect to RouterB. Verify that the ISDN circuit is active on RouterB with the **show isdn status** command. Also notice that there are no active calls on RouterB.

```
RouterB#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI1/0  interface
     dsl 8, interface ISDN Switchtype = basic-ni
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI Not Assigned, ces = 2, state = 3(await establishment)
         spid2 configured, spid2 NOT sent, spid2 NOT valid
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 0
    The Free Channel Mask:  0x80000003
    Total Allocated ISDN CCBs = 0
```

Let's view the routing table on RouterB with the **show ip route** command. Notice that RouterB is learning about the 1.1.1.1 and the 135.25.0.0 network via the ISDN interface. Notice that these routes are still being maintained in the routing table even though the ISDN interface is not active.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
```

```
       * − candidate default, U − per−user static route, o − ODR
       P − periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/32 is subnetted, 1 subnets
O       1.1.1.1 [110/1563] via 135.2.4.1, 00:06:42, BRI1/0
     2.0.0.0/32 is subnetted, 1 subnets
C       2.2.2.2 is directly connected, Loopback0
     135.25.0.0/22 is subnetted, 1 subnets
O       135.25.0.0 [110/1572] via 135.2.4.1, 00:06:42, BRI1/0
     135.2.0.0/22 is subnetted, 1 subnets
C       135.2.4.0 is directly connected, BRI1/0
```

The **show ip ospf neighbor** command shows us that that RouterB and RouterA are still neighbored even though the ISDN circuit is not active. Also notice that there is no OSPF dead time associated with this neighbor.

```
RouterB#sh ip ospf neigh

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   FULL/  −           −        135.2.4.1       BRI1/0
```

We also see that RouterB is configured to run as a demand circuit. Recall that the **ip ospf demand−circuit** command was only entered into RouterA's configuration, not RouterB's. This is because the **ip ospf demand−circuit** command only needs to be entered on one side of the link.

```
RouterB#sh ip ospf int bri 1/0
BRI1/0 is up, line protocol is up (spoofing)
  Internet Address 135.2.4.2/22, Area 0
  Process ID 64, Router ID 2.2.2.2, Network Type POINT_TO_POINT, Cost: 1562
  Run as demand circuit.
  DoNotAge LSA allowed.
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:04
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 1.1.1.1  (Hello suppressed)
  Suppress hello for 1 neighbor(s)
```

Now let's reconnect to RouterA. Let's see how OSPF demand circuit handles a change to the network topology. First, enable PPP authentication debugging with the **debug ppp authentication** command. Next, go into global configuration mode and shut down the e0/0 interface.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#int e 0/0
RouterA(config−if)#shut
RouterA(config−if)#^Z
```

After exiting out of configuration mode, you will see RouterA dial RouterB. The debug trace is shown in the following.

```
00:47:12: ISDN BR1/0: TX −>  SETUP pd = 8  callref = 0x4E ← RouterA calls
                                                            RouterB
00:47:12:         Bearer Capability i = 0x8890
00:47:12:         Channel ID i = 0x83
00:47:12:         Keypad Facility i = '8995201'
00:47:12: ISDN BR1/0: RX <−  CALL_PROC pd = 8  callref = 0xCE
00:47:12:         Channel ID i = 0x89
```

```
00:47:13: ISDN BR1/0: RX <-  CONNECT pd = 8  callref = 0xCE
00:47:13:         Channel ID i = 0x89
00:47:13: ISDN BR1/0: TX ->  CONNECT_ACK pd = 8  callref = 0x4E
00:47:13: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
00:47:13: BR1/0:1 PPP: Treating connection as a callout
00:47:13: BR1/0:1 CHAP: O CHALLENGE id 10 len 23 from "RouterA"
00:47:13: BR1/0:1 CHAP: I CHALLENGE id 10 len 23 from "RouterB"
00:47:13: BR1/0:1 CHAP: O RESPONSE id 10 len 23 from "RouterA"
00:47:13: BR1/0:1 CHAP: I SUCCESS id 10 len 4
00:47:13: BR1/0:1 CHAP: I RESPONSE id 10 len 23 from "RouterB"
00:47:13: BR1/0:1 CHAP: O SUCCESS id 10 len 4
00:47:14: %LINK-5-CHANGED: Interface Ethernet0/0, changed state to
administratively down
00:47:14: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1,
changed state to up
00:47:15: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0,
changed state to down
00:47:19: %ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 8995201
RouterB ← The ISDN call is now connected and the
updated routing table can be exchanged
```

The **show dialer** command on RouterA reveals that there is an active call. Notice that the dial reason is destination traffic to 224.0.0.5. This is OSPF traffic. Since we shut down our Ethernet interface, OSPF demand activated the ISDN circuit in order to update the routing table on RouterB.

```
RouterA#show dialer

BRI1/0 - dialer type = ISDN

Dial String      Successes   Failures    Last DNIS   Last status
8995201                 18         74    00:00:15       successful
0 incoming call(s) have been screened.
0 incoming call(s) rejected for callback.

BRI1/0:1 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is data link layer up
Dial reason: ip (s=135.2.4.1, d=224.0.0.5) ← OSPF demand caused the ISDN circuit to dial
Time until disconnect 108 secs
Connected to 8995201 (RouterB)

BRI1/0:2 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle
```

Now reconnect to RouterB. The **show ip route** command will reveal that the 135.25.0.0 network (E0/0 on RouterA) is no longer in the routing table.

```
RouterB#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/32 is subnetted, 1 subnets
O       1.1.1.1 [110/1563] via 135.2.4.1, 00:00:14, BRI1/0
     2.0.0.0/32 is subnetted, 1 subnets
C       2.2.2.2 is directly connected, Loopback0
     135.2.0.0/16 is variably subnetted, 2 subnets, 2 masks
```

```
C       135.2.4.1/32 is directly connected, BRI1/0
C       135.2.4.0/22 is directly connected, BRI1/0
```

Lab #9: PPP Callback

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a single ISDN BRI interface
- Cisco IOS 11.2 or higher
- Two ISDN BRI circuits
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This lab will demonstrate the PPP callback function. RouterA is the callback client and RouterB is the callback server. PPP callback is used to provide enhanced security in a dial network. The PPP callback client calls the PPP callback server, the call is authenticated, and the PPP callback server then calls back the PPP callback client. We will see in this lab that a call from RouterA to RouterB will be disconnected and then RouterB will dial back RouterA.

RouterA and RouterB are connected as shown in Figure 3−20.

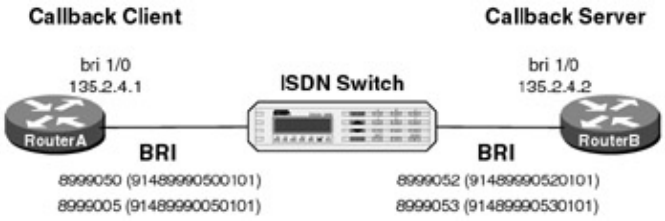


Figure 3−20: PPP callback

A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. PPP callback commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
service timestamps debug datetime
no service password-encryption
no service udp-small-servers
```

```
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
username RouterB password 0 cisco
no ip domain-lookup
isdn switch-type basic-ni1
!
interface BRI1/0
 ip address 135.2.4.1 255.255.252.0
 encapsulation ppp
 isdn spid1 91489990500101 8999050
 isdn spid2 91489990050101 8999005
 dialer map ip 135.2.4.2 name RouterB broadcast 8999052
 dialer load-threshold 1 outbound
 dialer-group 1
 no fair-queue
 no cdp enable
 ppp callback request
 ppp authentication chap
 ppp multilink
 hold-queue 75 in
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
ip route 135.2.0.0 255.255.0.0 135.2.4.2 120
access-list 100 deny  ospf any any
access-list 100 permit ip any any
access-list 100 permit icmp any any
no cdp run
!
dialer-list 1 protocol ip list 100
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
username RouterA password 0 cisco
no ip domain-lookup
isdn switch-type basic-ni1
!
interface BRI1/0
 ip address 135.2.4.2 255.255.252.0
 encapsulation ppp
 isdn spid1 91489990520101 8999052
 isdn spid2 91489990530101 8999053
```

```
 dialer callback-secure
 dialer map ip 135.2.4.1 name RouterA class dial1 broadcast 8999050
 dialer load-threshold 100 either
 dialer-group 1
 no fair-queue
 no cdp enable
 ppp callback accept
 ppp authentication chap callin
 ppp multilink
 hold-queue 75 in
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
map-class dialer dial1
 dialer callback-server username
access-list 100 deny  ospf any any
access-list 100 permit ip any any
access-list 100 permit icmp any any
no cdp run
!
dialer-list 1 protocol ip list 100
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA and enabling PPP protocol, authentication, and ISDN call control debugging with the **debug ppp authentication**, **debug ppp negotiation**, and **debug isdn q931** commands.

```
RouterA#debug ppp authentication
RouterA#debug ppp negotiation
RouterA#debug isdn q931
```

Now activate the ISDN link by pinging RouterB at 135.2.4.2. We see below that RouterA places a call to RouterB.

```
RouterA#ping 135.2.4.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 135.2.4.2, timeout is 2 seconds:

*Mar  7 23:43:23: ISDN BR1/0: TX ->  SETUP pd = 8  callref = 0x32 ← RouterA
                                                                    calls
                                                                    RouterB
*Mar  7 23:43:23:         Bearer Capability i = 0x8890
*Mar  7 23:43:23:         Channel ID i = 0x83
*Mar  7 23:43:23:         Keypad Facility i = '8999052'
*Mar  7 23:43:24: ISDN BR1/0: RX <-  CALL_PROC pd = 8  callref = 0xB2
*Mar  7 23:43:24:         Channel ID i = 0x89
*Mar  7 23:43:24: ISDN BR1/0: RX <-  CONNECT pd = 8  callref = 0xB2
%LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
*Mar  7 23:43:24: BR1/0:1 PPP: Treating connection as a callout
*Mar  7 23:43:24: BR1/0:1 PPP: Phase is ESTABLISHING, Active Open
*Mar  7 23:43:24: BR1/0:1 LCP: O CONFREQ [Closed] id 32 len 27
*Mar  7 23:43:24: BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  7 23:43:24: BR1/0:1 LCP:    MagicNumber 0xF4B761AD (0x0506F4B761AD)
```

```
*Mar  7 23:43:24: BR1/0:1 LCP:    Callback 0  (0x0D0300)
*Mar  7 23:43:24: BR1/0:1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:24: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015231)
*Mar  7 23:43:24: ISDN BR1./0: TX ->  CONNECT_ACK pd = 8  callref = 0x32
*Mar  7 23:43:24: BR1/0:1 LCP: I CONFREQ [REQsent] id 57 len 24
*Mar  7 23:43:24: BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  7 23:43:24: BR1/0:1 LCP:    MagicNumber 0x052DFD6C (0x0506052DFD6C)
*Mar  7 23:43:24: BR1/0:1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:24: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015232)
*Mar  7 23:43:24: BR1/0:1 LCP: O CONFACK [REQsent] id 57 len 24
*Mar  7 23:43:24: BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  7 23:43:24: BR1/0:1 LCP:    MagicNumber 0x052DFD6C (0x0506052DFD6C)
*Mar  7 23:43:24: BR1/0:1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:24: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015232)
*Mar  7 23:43:24: BR1/0:1 LCP: I CONFACK [ACKsent] id 32 len 27
*Mar  7 23:43:24: BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  7 23:43:24: BR1/0:1 LCP:    MagicNumber 0xF4B761AD (0x0506F4B761AD)
*Mar  7 23:43:24: BR1/0:1 LCP:    Callback 0  (0x0D0300)
*Mar  7 23:43:24: BR1/0:1 LCP:    MRRU 1524 .(0x110405F4)
*Mar  7 23:43:24: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015231)
*Mar  7 23:43:24: BR1/0:1 LCP: State is Open
*Mar  7 23:43:24: BR1/0:1 PPP: Phase is AUTHENTICATING, by both
*Mar  7 23:43:24: BR1/0:1 CHAP: O CHALLENGE id 5 len 23 from "RouterA"
*Mar  7 23:43:24: BR1/0:1 CHAP: I CHALLENGE id 29 len 23 from "RouterB"
*Mar  7 23:43:24: BR1/0:1 CHAP: O RESPONSE id 29 len 23 from "RouterA"
*Mar  7 23:43:24: BR1/0:1 CHAP: I SUCCESS id 29 len 4
*Mar  7 23:43:24: BR1/0:1 CHAP: I RESPONSE id 5 len 23 from "RouterB"
*Mar  7 23:43:24: BR1/0:1 CHAP: O SUCCESS id 5 len 4
*Mar  7 23:43:24: BR1/0:1 PPP: Phase is VIRTUALIZED
*Mar  7 23:43:24: Vi1 PPP: Phase is DOWN, Setup
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to up
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
*Mar  7 23:43:24: Vi1 PPP: Treating connection as a callout
*Mar  7 23:43:24: Vi1 PPP: Phase is ESTABLISHING, Active Open
*Mar  7 23:43:24: Vi1 LCP: O CONFREQ [Closed] id 1 len 27
*Mar  7 23:43:24: Vi1. LCP:    AuthProto CHAP (0x0305C22305)
*Mar  7 23:43:24: Vi1 LCP:    MagicNumber 0xF4B7621A (0x0506F4B7621A)
*Mar  7 23:43:24: Vi1 LCP:    Callback 0  (0x0D0300)
*Mar  7 23:43:24: Vi1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:24: Vi1 LCP:    EndpointDisc 1 Local (0x1305015231)
*Mar  7 23:43:24: Vi1 PPP: Phase is UP
*Mar  7 23:43:24: Vi1 IPCP: O CONFREQ [Closed] id 1 len 10
*Mar  7 23:43:24: Vi1 IPCP:    Address 135.2.4.1 (0x030687020401)
*Mar  7 23:43:24: ISDN BR1/0: RX <-  DISCONNECT pd = 8  callref = 0xB2
← RouterB hangs up on RouterA
*Mar  7 23:43:24:         Cause i = 0x8090 - Normal call clearing
%LINK-3-UPDOWN: Interface BRI1/0:1, changed state to down
*Mar  7 23:43:24: BR1/0:1 PPP: Phase is TERMINATING
*Mar  7 23:43:24: BR1/0:1 LCP: State is Closed
*Mar  7 23:43:24: BR1/0:1 PPP: Phase is DOWN
*Mar  7 23:43:24: ISDN BR1/0: TX ->  RELEASE pd = 8  callref = 0x32
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
*Mar  7 23:43:24: Vi1 IPCP: State is Closed
*Mar  7 23:43:24: Vi1 PPP: Phase is TERMINA.TING
*Mar  7 23:43:24: Vi1 LCP: State is Closed
*Mar  7 23:43:24: Vi1 PPP: Phase is DOWN
*Mar  7 23:43:24: ISDN BR1/0: RX <-  RELEASE_COMP pd = 8  callref = 0xB2
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to down.
Success rate is 0 percent (0/5)
```

Notice how the call from RouterA to RouterB has been disconnected by RouterB.

The following trace shows how RouterB places the callback call to RouterA.

```
RouterA#
*Mar  7 23:43:41: ISDN BR1/0: RX <-  SETUP pd = 8  callref = 0x4F ← RouterB
                                                                    calls
```

RouterA back

```
*Mar  7 23:43:41:         Bearer Capability i = 0x8890
*Mar  7 23:43:41:         Channel ID i = 0x89
*Mar  7 23:43:41:         Signal i = 0x40 − Alerting on − pattern 0
*Mar  7 23:43:41:         Calling Party Number i = '!', 0x83, '9148999052'
*Mar  7 23:43:41:         Called Party Number i = 0xC1, '8999050'
*Mar  7 23:43:41:         Locking Shift to Codeset 5
*Mar  7 23:43:41:         Codeset 5 IE 0x2A  i = 0x808001039E05, 'From',
0x8B0C, '914 899−9052', 0x8001, '<'
%LINK−3−UPDOWN: Interface BRI1/0:1, changed state to up
*Mar  7 23:43:41: BR1/0:1 PPP: Treating connection as a callin
*Mar  7 23:43:41: BR1/0:1 PPP: Phase is ESTABLISHING, Passive Open
*Mar  7 23:43:41: BR1/0:1 LCP: State is Listen
*Mar  7 23:43:41: ISDN BR1/0: TX −>  CONNECT pd = 8  callref = 0xCF
*Mar  7 23:43:41:         Channel ID i = 0x89
*Mar  7 23:43:41: ISDN BR1/0: RX <−  CONNECT_ACK pd = 8  callref = 0x4F
*Mar  7 23:43:41:         Channel ID i = 0x89
*Mar  7 23:43:41:         Signal i = 0x4F − Alerting off
*Mar  7 23:43:41: BR1/0:1 LCP: I CONFREQ [Listen] id 58 len 19
*Mar  7 23:43:41: BR1/0:1 LCP:    MagicNumber 0x052E3F25 (0x0506052E3F25)
*Mar  7 23:43:41: BR1/0:1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:41: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015232)
*Mar  7 23:43:41: BR1/0:1 LCP: O CONFREQ [Listen] id 33 len 24
*Mar  7 23:43:41: BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  7 23:43:41: BR1/0:1 LCP:    MagicNumber 0xF4B7A380 (0x0506F4B7A380)
*Mar  7 23:43:41: BR1/0:1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:41: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015231)
*Mar  7 23:43:41: BR1/0:1 LCP: O CONFACK [Listen] id 58 len 19
*Mar  7 23:43:41: BR1/0:1 LCP:    MagicNumber 0x052E3F25 (0x0506052E3F25)
*Mar  7 23:43:41: BR1/0:1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:41: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015232)
*Mar  7 23:43:41: BR1/0:1 LCP: I CONFACK [ACKsent] id 33 len 24
*Mar  7 23:43:41: BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  7 23:43:41: BR1/0:1 LCP:    MagicNumber 0xF4B7A380 (0x0506F4B7A380)
*Mar  7 23:43:41: BR1/0:1 LCP:    MRRU 1524 (0x110405F4)
*Mar  7 23:43:41: BR1/0:1 LCP:    EndpointDisc 1 Local (0x1305015231)
*Mar  7 23:43:41: BR1/0:1 LCP: State is Open
*Mar  7 23:43:41: BR1/0:1 PPP: Phase is AUTHENTICATING, by this end
*Mar  7 23:43:41: BR1/0:1 CHAP: O CHALLENGE id 6 len 23 from "RouterA"
*Mar  7 23:43:41: BR1/0:1 CHAP: I RESPONSE id 6 len 23 from "RouterB"
*Mar  7 23:43:41: BR1/0:1 CHAP: O SUCCESS id 6 len 4
*Mar  7 23:43:41: BR1/0:1 PPP: Phase is VIRTUALIZED
*Mar  7 23:43:41: Vi1 PPP: Phase is DOWN, Setup
%LINK−3−UPDOWN: Interface Virtual−Access1, changed state to up
*Mar  7 23:43:41: Vi1 PPP: Treating connection as a callin
*Mar  7 23:43:41: Vi1 PPP: Phase is ESTABLISHING, Passive Open
*Mar  7 23:43:41: Vi1 LCP: State is Listen
*Mar  7 23:43:41: Vi1 PPP: Phase is UP
*Mar  7 23:43:41: Vi1 IPCP: O CONFREQ [Closed] id 1 len 10
*Mar  7 23:43:41: Vi1 IPCP:    Address 135.2.4.1 (0x030687020401)
*Mar  7 23:43:41: Vi1 IPCP: I CONFREQ [REQsent] id 1 len 10
*Mar  7 23:43:41: Vi1 IPCP:    Address 135.2.4.2 (0x030687020402)
*Mar  7 23:43:41: Vi1 IPCP: O CONFACK [REQsent] id 1 len 10
*Mar  7 23:43:41: Vi1 IPCP:    Address 135.2.4.2 (0x030687020402)
*Mar  7 23:43:41: Vi1 IPCP: I CONFACK [ACKsent] id 1 len 10
*Mar  7 23:43:41: Vi1 IPCP:    Address 135.2.4.1 (0x030687020401)
*Mar  7 23:43:41: Vi1 IPCP: State is Open
*Mar  7 23:43:41: BR1/0  IPCP: Install route to 135.2.4.2
%LINEPROTO−5−UPDOWN: Line protocol on Interface BRI1/0:1, changed state to up
%LINEPROTO−5−UPDOWN: Line protocol on Interface Virtual−Access1, changed state
to up
```

Verify that the call has been connected by pinging RouterB from RouterA.

```
RouterA#ping 135.2.4.2
```

```
Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 135.2.4.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 28/30/32 ms
```

Now connect to RouterB. The **show dialer** command will reveal that the ISDN circuit is active and that the dial reason was a callback return call. Notice that we are using both B channels of the BRI circuit; the above debug output only showed a single channel so that the debug output would be easier to read.

```
RouterB#show dialer

BRI1/0 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8999050                  6          0    00:00:42       successful
0 incoming call(s) have been screened.

BRI1/0:1 − dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: Callback return call
Time until disconnect 58 secs
Connected to 8999050 (RouterA)

BRI1/0:2 − dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: Callback return call
Time until disconnect 77 secs
Connected to 8999050 (RouterA)
```

Lab #10: Dialer Watch

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have a single ISDN BRI interface and a serial interface
- Cisco IOS 12.0 or higher
- Two ISDN BRI circuits
- A PC running a terminal emulation program for console port connection on the routers

Configuration Overview

This configuration will demonstrate dialer watch. Dialer watch is a backup feature that integrates dial backup with routing capabilities. Before the implementation of dialer watch, there were three ways that a dial backup connection could be initiated:

1. Interesting packets defined using DDR
2. Back up interfaces caused by a connection loss on a primary interface
3. Traffic thresholds being exceeded using a dialer load threshold

Dialer watch works by defining a set of routes that will be monitored. When any of these routes are no longer in the routing table, the router will initiate a dial session.

RouterA and RouterB are connected as shown in Figure 3−21.

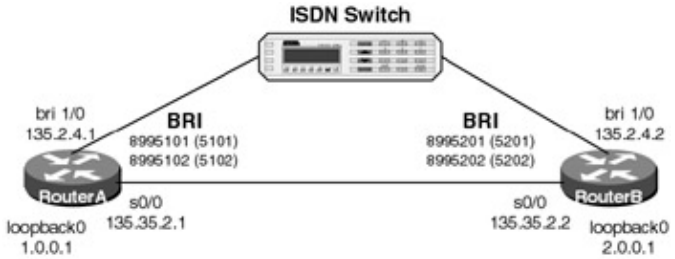


Figure 3−21: Dialer watch

A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. Dialer watch commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterA
!
!
username RouterB password 0 cisco
!
ip subnet−zero
!
lane client flush
isdn switch−type basic−ni
cns event−service server
!
interface Loopback0
 ip address 1.0.0.1 255.0.0.0
!
interface Serial0/0
 ip address 135.35.2.1 255.255.0.0
 encapsulation ppp
 no fair−queue
 clockrate 800000
 ppp multilink
!
interface BRI1/0
 ip address 135.2.4.1 255.255.0.0
 encapsulation ppp
 dialer map ip 2.0.0.0 name RouterB broadcast 8995201
 dialer map ip 135.2.4.2 name RouterB broadcast 8995201
 dialer load−threshold 255 either
 dialer watch−group 1
 dialer−group 1
 isdn switch−type basic−ni
 isdn spid1 5101 8995101
```

```
 isdn spid2 5102 8995102
 ppp authentication chap
!
router eigrp 1
 network 1.0.0.0
 network 135.0.0.0 0.255.255.255
 no auto-summary
!
ip classless
no ip http server
!
access-list 100 deny  eigrp any any
access-list 100 permit ip any any
access-list 100 permit icmp any any
dialer watch-list 1 ip 2.0.0.0 255.0.0.0
dialer-list 1 protocol ip list 100
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
!
username RouterA password 0 cisco
!
ip subnet-zero
!
lane client flush
isdn switch-type basic-ni
cns event-service server
!
interface Loopback0
 ip address 2.0.0.1 255.0.0.0
!
interface Serial0/0
 ip address 135.35.2.2 255.255.0.0
 encapsulation ppp
!
interface BRI1/0
 ip address 135.2.4.2 255.255.0.0
 encapsulation ppp
 dialer map ip 135.2.4.1 name RouterA
 dialer load-threshold 1 either
 dialer-group 1
 isdn switch-type basic-ni
 isdn spid1 5201 8995201
 isdn spid2 5202 8995202
 ppp authentication chap
!
router eigrp 1
 network 2.0.0.0
 network 135.0.0.0 0.255.255.255
```

```
 no auto-summary
!
ip classless
no ip http server
!
access-list 100 deny  eigrp any any
access-list 100 permit ip any any
access-list 100 permit icmp any any
dialer-list 1 protocol ip list 100
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Verify that the ISDN circuit is up and active with the **show isdn status** command.

```
RouterA#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI1/0  interface
     dsl 8, interface ISDN Switchtype = basic-ni
  Layer 1 Status:
     ACTIVE
  Layer 2 Status:
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 8(established)
       spid1 configured, spid1 sent, spid1 valid
       Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
       spid2 configured, spid2 sent, spid2 valid
       Endpoint ID Info: epsf = 0, usid = 70, tid = 2
  Layer 3 Status:
     0 Active Layer 3 Call(s)
  Activated dsl 8 CCBs = 0
  The Free Channel Mask: 0x80000003
  Total Allocated ISDN CCBs = 0
```

Next, use the **show ip route** command to verify that the 2.0.0.0 network is being learned via EIGRP over the serial interface connecting RouterA and RouterB. This is the network that we will be monitoring via dialer watch. Notice that the network is being advertised as 2.0.0.0/8. The **dialer watch-list** command must be configured to match this exact network entry.

```
RouterA#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
    D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
    N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
    E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
    i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
    * - candidate default, U - per-user static route, o - ODR
    P - periodic downloaded static route

Gateway of last resort is not set

C  1.0.0.0/8 is directly connected, Loopback0
D  2.0.0.0/8 [90/1889792] via 135.35.2.2, 01:03:33, Serial0/0
   135.35.0.0/16 is variably subnetted, 2 subnets, 2 masks
C    135.35.0.0/16 is directly connected, Serial0/0
```

```
C    135.35.2.2/32 is directly connected, Serial0/0
C  135.2.0.0/16 is directly connected, BRI1/0
```

Enable ISDN and dialer debugging with the **debug isdn q931** and **debug dialer events** commands.

```
RouterA#debug isdn q931
ISDN Q931 packets debugging is on
RouterA#debug dialer events
Dial on demand events debugging is on
```

Now connect to RouterB. Verify that the ISDN circuit on RouterB is up and active with the **show isdn status** command.

```
RouterB#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI1/0  interface
     dsl 8, interface ISDN Switchtype = basic-ni
  Layer 1 Status:
     ACTIVE
  Layer 2 Status:
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 5(init)
       spid1 configured, spid1 sent, spid1 valid
       Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
       spid2 configured, spid2 sent, spid2 valid
       Endpoint ID Info: epsf = 0, usid = 70, tid = 2
  Layer 3 Status:
     0 Active Layer 3 Call(s)
  Activated dsl 8 CCBs = 0
  The Free Channel Mask: 0x80000003
  Total Allocated ISDN CCBs = 0
```

Go into configuration mode on RouterB and shut down the loopback0 interface. The loopback0 interface is on network 2.0.0.0/8, this is the network that RouterA is monitoring via dialer watch. After typing the **shutdown** command, quickly reconnect to RouterA.

```
RouterB#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterB(config)#int loop0
RouterB(config-if)#shutdown
```

Once reconnected to RouterA, you will see RouterA place an ISDN call to RouterB. The following debug output will be seen:

```
[Resuming connection 1 to RouterA ...]
01:57:27: DDR: Dialer Watch: watch-group = 1
01:57:27: DDR:       network 2.0.0.0/255 .0.0.0 DOWN, ← Dialer watch initiates
                                                        a call to RouterB
                                                        because the 2.0.0.0
                                                        network has been
                                                        declared down
01:57:27: DDR:       primary DOWN
01:57:27: DDR: Dialer Watch: Dial Reason: Primary of group 1 DOWN
01:57:27: DDR: Dialer Watch: watch-group = 1,
01:57:27: DDR:       dialing secondary by dialer map 2.0.0.0 on BR1/0
01:57:27: BR1/0 DDR: Attempting to dial 8995201
01:57:27: ISDN BR1/0: TX -> SETUP pd = 8 callref = 0x06
01:57:27:     Bearer Capability i = 0x8890
01:57:27:     Channel ID i = 0x83
01:57:27:     Keypad Facility i = '8995201'
01:57:27: ISDN BR1/0: RX <- CALL_PROC pd = 8 callref = 0x86
01:57:27:     Channel ID i = 0x89
```

```
01:57:27: ISDN BR1/0: RX <- CONNECT pd = 8 callref = 0x86
01:57:27:     Channel ID i = 0x89
01:57:27: ISDN BR1/0: TX -> CONNECT_ACK pd = 8 callref = 0x06
01:57:27: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
01:57:27: BR1/0:1 DDR: Dialer Watch: resetting call in progress
01:57:27: BR1/0:1 DDR: dialer protocol up
01:57:28: DDR: Dialer Watch: watch-group = 1
01:57:28: DDR:       network 2.0.0.0/255.0.0.0 DOWN,
01:57:28: DDR:       primary DOWN
01:57:28: DDR: Dialer Watch: Dial Reason: Primary of group 1 DOWN
01:57:28: DDR: Dialer Watch: watch-group = 1,
01:57:28: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to up
01:57:33: %ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 8995201 RouterB
```

Now reconnect to RouterB. Bring the loopback0 interface back up with the **no shut** command. After bringing the loopback0 back up, reconnect to RouterA.

```
RouterB(config-if)#no shut
```

Once connected to RouterA, verify that the 2.0.0.0 network is once again being learned via the serial interface.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
    D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
    N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
    E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
    i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
    * - candidate default, U - per-user static route, o - ODR
    P - periodic downloaded static route

Gateway of last resort is not set

C  1.0.0.0/8 is directly connected, Loopback0
   2.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
D    2.0.0.0/8 [90/1889792] via 135.35.2.2, 00:01:25, Serial0/0
   135.35.0.0/16 is variably subnetted, 2 subnets, 2 masks
C    135.35.0.0/16 is directly connected, Serial0/0
C    135.35.2.2/32 is directly connected, Serial0/0
C  135.2.0.0/16 is directly connected, BRI1/0
```

A short period of time later the ISDN call will be disconnected once dialer watch recognizes the 2.0.0.0 network as being active.

```
01:59:27: ISDN BR1/0: RX <- DISCONNECT pd = 8 callref = 0x86
01:59:27:     Cause i = 0x8290 - Normal call clearing
01:59:27: ISDN BR1/0: TX -> RELEASE pd = 8 callref = 0x06
01:59:27: %ISDN-6-DISCONNECT: Interface BRI1/0:1 disconnected
from 8995201 RouterB, call lasted 119 seconds
01:59:27: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to down
01:59:27: DDR: Dialer Watch: watch-group = 1
01:59:27: DDR:       network 2.0.0.0/255.0.0.0 UP,
01:59:27: DDR:       primary UP
01:59:27: BR1/0:1 DDR: disconnecting call
01:59:27: ISDN BR1/0: RX <- RELEASE_COMP pd = 8 callref = 0x86
01:59:28: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1,
changed state to down
```

Lab #11: ISDN Troubleshooting

Equipment Needed

- Two Cisco routers, each of which must have a BRI interface
- Two ISDN BRI circuits
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection on the routers

The following equipment is needed to perform this lab exercise:

Configuration Overview

This lab will demonstrate key ISDN debug and troubleshooting techniques.

The two routers are connected as shown in Figure 3−22. RouterA and RouterB are connected to an Adtran Atlas 800 ISDN switch.

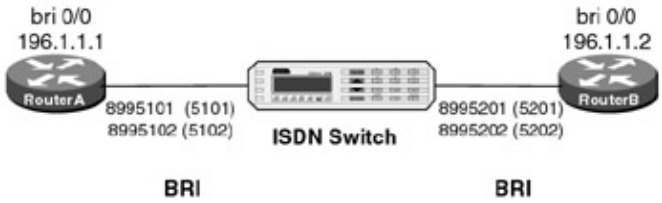


Figure 3−22: ISDN troubleshooting

A PC running a terminal emulation program should be connected to the console port of one of the routers using a Cisco rolled cable.

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. Information on configuring an ISDN desktop switch can be found in the ISDN switch configuration section earlier in this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. ISDN commands are highlighted in bold.

RouterA

```
RouterA#show run
Building configuration...

Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
username RouterB password 7 070C285F4D06
isdn switch-type basic-ni1 ← Set D channel call control
!
interface BRI0/0
 ip address 196.1.1.1 255.255.255.0
 encapsulation ppp
```

```
 isdn spid1 5101 8995101 ← Set SPID for both B channels
 isdn spid2 5102 8995102
 dialer idle-timeout 90 ← Disconnect call 90 seconds after last interesting packet
 dialer map ip 196.1.1.2 name RouterB broadcast 8995201 ← Associate a next hop
                                                          address with dial
                                                          string
 dialer load-threshold 1 ← Threshold for adding additional B channels
 dialer-group 1 ← Associate this interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink ← Request a PPP multilink session
!
 no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
RouterB#show run
Building configuration...

Current configuration:
!
version 11.2
service timestamps debug datetime localtime
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
username RouterA password 7 094F471A1A0A
isdn switch-type basic-ni1 ← Set the D channel call control
!
interface BRI0/0
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
 isdn spid1 5201 8995201 ← Set the SPID for both B channels
 isdn spid2 5202 8995202
 dialer idle-timeout 90 ← Set the interesting traffic timeout
 dialer map ip 196.1.1.1 name RouterA ← Define a next hop address
 dialer-group 1 ← Associate this interface with dialer-list 1
 no fair-queue
 ppp authentication chap
 ppp multilink
!
router rip
 network 192.1.1.0
!
no ip classless
dialer-list 1 protocol ip permit ← Define interesting traffic
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
```

```
end
```

Monitoring and Testing the Configuration

The **show isdn status** command will display the condition of the ISDN circuit. It allows you to view the layer 1, 2, and 3 status of the ISDN circuit. Layer 1 Active indicates that the Router sees 2B1Q framing on the ISDN circuit. Layer 2 status should show Multiple_Frame_Established. This indicates that the router and ISDN switch have made an initial handshake. The SPID status should show that each SPID has been sent and is valid. Remember that a 2B+D ISDN BRI circuit will usually have two SPIDs. An ISDN PRI circuit does not have any SPIDs.

```
RouterB#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 1
     CCB:callid=0, callref=0, sapi=0, ces=1, B-chan=0
    Number of active calls = 0
    Number of available B-channels = 2
 Total Allocated ISDN CCBs = 1
```

The **show controllers** command can also be used to verify that the ISDN circuit is active.

```
RouterA#sh controllers bri 0/0
BRI unit 0:BRI unit 0 with U interface:
Layer 1 internal state is ACTIVATED

Layer 1 U interface is ACTIVATED.
ISDN Line Information:
    Last C/I from ISDN transceiver:
        AI:Activation Indication
    Last C/I to ISDN transceiver:
        AI:Activation Indication
    Current EOC commands:
        RTN - Return to normal
    Received overhead bits:  AIB=1, UOA=1, FEBE=1, DEA=1, ACT=1
    Errors:  Receive [NEBE]=0, Transmit [FEBE]=0

Siemens 2091 read-only registers:
    ISTA 0 MOR FF MOSR 0 CIRI 3F CIRU 33
Siemens 2091 write-only registers - last values written:
    MASK 96 STCR 15 MOX FF DWU FF ADF2 28 RSVD 0 WB1U 0 WB2U 0
    WB1I 0 WB2I 0 MOCR A0 DWI FF CIWU F3 CIWI FF ADF 10 SWST D
D Channel Information:
idb at 0x609455C4, driver data structure at 0x6097BCF0
Siemens Chip Version 0x0
```

No ISDN Signal

Let's disconnect the ISDN cable from the router. Now type the **show isdn status** command. Notice that the layer 1 status is now deactivated. The router no longer sees the ISDN BRI signal coming from the ISDN

switch.

```
RouterB#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0 interface
    Layer 1 Status:
     DEACTIVATED ← No BRI signal from the ISDN switch
    Layer 2 Status:
     Layer 2 NOT Activated
    Spid Status:
     TEI Not Assigned, ces = 1, state = 1(terminal down)
         spid1 configured, spid1 NOT sent, spid1 NOT valid
     TEI Not Assigned, ces = 2, state = 1(terminal down)
         spid2 configured, spid2 NOT sent, spid2 NOT valid
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 0
    Number of active calls = 0
    Number of available B-channels = 2
 Total Allocated ISDN CCBs = 0
```

Wrong SPID

Now let's reconnect the ISDN cable to the router. After the ISDN circuit comes active again, we will change our SPID for both B channels to an incorrect value and see how the router reacts. Enter configuration mode with the **config term** command. Change the SPID for both B channels to an incorrect value.

```
RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#interface bri 0/0
RouterB(config-if)#isdn spid1 6201 8995201 ← Incorrect SPID value for B
                                              channel #1
RouterB(config-if)#isdn spid2 6202 8996202 ← Incorrect SPID Value for B
                                              channel #2
RouterB(config-if)#exit
RouterB(config)#exit
RouterB#
```

The router will try to send the new SPID value to the ISDN switch, and the ISDN switch will reject it because it is the wrong value. The following message will be displayed:

```
%ISDN-4-INVALID_SPID: Interface BR0/0, Spid1 was rejected
```

Type **show isdn status** to display the BRI interface status. We see that the router is reporting that it sent the SPID for B channel #1, but that it was not valid. This is correct since we changed the SPID for both B channels to an incorrect value.

```
RouterB#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI0/0  interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 6(not initialized)
      spid1 configured, spid1 sent, spid1 NOT valid ← ISDN switch rejected the
                                                      SPID that the router sent
     TEI Not Assigned, ces = 2, state = 1(terminal down)
         spid2 configured, spid2 NOT sent, spid2 NOT valid
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 0 CCBs = 0
    Number of active calls = 0
```

```
    Number of available B-channels = 2
 Total Allocated ISDN CCBs = 0
```

Now let's change the SPID for B channel #1 and B channel #2 back to the correct value. Type **config term** and change the SPIDs back to their original values as shown:

```
RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#interface bri 0/0
RouterB(config-if)#isdn spid1 5201 8995201 ← Correct SPID Value
RouterB(config-if)#isdn spid2 5202 8995202 ← Correct SPID Value
RouterB(config-if)#exit
RouterB(config)#exit
RouterB#show run
Building configuration . . .
```

Layer 2 Debugging

Now let's look at some layer 2 debugging capabilities of the Cisco router. Enable layer 2 ISDN debugging by typing the **debug isdn q921** command. Remember to also type the **term mon** command if you are not connected to the console port of the router.

```
RouterB#debug isdn q921
ISDN Q921 packets debugging is on
```

The **debug isdn q921** command causes the router to display all layer 2 activity between itself and the ISDN switch. Recall from the ISDN technology introduction at the beginning of this chapter that layer 2 ISDN activity involves SPID and TEI negotiation as well as periodic 10-second aliveness messages (referred to as RR or Receiver Ready) that are initiated by the ISDN switch and immediately answered by the router. We see below that the first RR message is received by the router at time 00:03:20 and is immediately answered. Notice that there are two sets of RRs being sent from the ISDN switch. The first set is for TEI=64 and the second set is for TEI=65. This is for each of the two B channels of the BRI interface.

```
*Mar  1 00:03:20: ISDN BR0/0: RX <-  RRp sapi = 0  tei = 64 nr = 3
*Mar  1 00:03:20: ISDN BR0/0: TX ->  RRf sapi = 0  tei = 64  nr = 3

*Mar  1 00:03:27: ISDN BR0/0: RX <-  RRp sapi = 0  tei = 65 nr = 1
*Mar  1 00:03:27: ISDN BR0/0: TX ->  RRf sapi = 0  tei = 65  nr = 1

*Mar  1 00:03:30: ISDN BR0/0: RX <-  RRp sapi = 0  tei = 64 nr = 3
*Mar  1 00:03:30: ISDN BR0/0: TX ->  RRf sapi = 0  tei = 64  nr = 3

*Mar  1 00:03:37: ISDN BR0/0: RX <-  RRp sapi = 0  tei = 65 nr = 1
*Mar  1 00:03:37: ISDN BR0/0: TX ->  RRf sapi = 0  tei = 65  nr = 1
```

D-Channel Monitoring

Type **show interface bri 0/0** to display the status of the D channel of the BRI interface. The physical interface that the BRI is connected to is bri 0/0. When you display this interface with the **show interface bri 0/0** command, the router displays the D-channel status of the BRI circuit.

```
RouterA#show interface bri 0/0
BRI0/0  is up, line protocol is up (spoofing) ← D channel status
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
  Last input 00:00:00, output never, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
```

```
  5 minute output rate 0 bits/sec, 0 packets/sec
     156 packets input, 946 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     156 packets output, 1056 bytes, 0 underruns
     0 output errors, 0 collisions, 4 interface resets
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
```

To display the status of each individual B channel, you need to issue the **show interface bri 0/0:1** command for B channel #1 or **show interface bri 0/0:2** for B channel #2. We see that both of these interfaces are in a down/down state since there are no active calls on the routers at this time.

```
RouterA#show interface bri 0/0:1 ← B channel #1
BRI0/0:1 is down, line protocol is down
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Closed, multilink Closed
  Closed: IPCP, CDP
  Last input 00:01:04, output 00:01:04, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     37 packets input, 838 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     38 packets output, 1116 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     16 carrier transitions

RouterA#show interface bri 0/0:2 ← B channel #2
BRI0/0:2 is down, line protocol is down
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Closed, multilink Closed
  Closed: IPCP, CDP
  Last input 00:01:08, output 00:01:08, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     30 packets input, 740 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     32 packets output, 1032 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions
```

Layer 3 Debugging

Layer 3 debugging is turned on by issuing the **debug isdn q931** command on the router. Layer 3 ISDN traffic consists of the call setup and teardown for each B channel that is either making a call or receiving a call. CHAP authentication progress and status is monitored by entering the **debug ppp authen** command. Remember to also type the **term mon** command if you are not connected to the console port of the router.

Now that layer 3 debugging and PPP authentication debugging are enabled, let's place a call to RouterB by pinging IP address 196.1.1.2. Since Layer 3 debugging is turned on, we will see all of the call setup and

teardown. Since PPP authentication is turned on, we will see the progress of the CHAP authentication between RouterA and RouterB.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:

ISDN BR0/0: TX -> SETUP pd = 8 callref = 0x11 ← Call placed on B channel #1
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <- CALL_PROC pd = 8 callref = 0x91
        Channel ID i = 0x89
ISDN BR0/0: RX <- CONNECT pd = 8 callref = 0x91 ← Connect on B channel #1
        Channel ID i = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
ISDN BR0/0: TX -> CONNECT_ACK pd = 8 callref = 0x11
PPP BRI0/0:1: Send CHAP Challenge id=2 ← Chap challenge sent
PPP BRI0/0:1: CHAP Challenge id=2 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=2
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=2 received from RouterB
PPP BRI0/0:1: Send CHAP Success id=2 ← CHAP success B channel #1
PPP BRI0/0:1: remote passed CHAP authentication
PPP BRI0/0:1: Passed CHAP authentication with remote
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
PPP Virtual-Access1: treating connection as a callin
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
ISDN BR0/0: TX -> SETUP pd = 8 callref = 0x12 ← Call placed on B channel #2
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <- CALL_PROC pd = 8 callref = 0x92
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: RX <- CONNECT pd = 8 callref = 0x92 ← Connect on B channel #2
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up
PPP BRI0/0:2: treating connection as a callout
ISDN BR0/0: TX -> CONNECT_ACK pd = 8 callref = 0x12
PPP BRI0/0:2: Send CHAP Challenge id=2 ← Chap challenge sent
PPP BRI0/0:2: CHAP Challenge id=2 received from RouterB
PPP BRI0/0:2: Send CHAP Response id=2
PPP BRI0/0:2: CHAP response received from RouterB
PPP BRI0/0:2: CHAP Response id=2 received from RouterB
PPP BRI0/0:2: Send CHAP Success id=2 ← CHAP success B channel #2
PPP BRI0/0:2: remote passed CHAP authentication.
PPP BRI0/0:2: Passed CHAP authentication with remote.
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to 8995201 RouterB

..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 20/22/24 ms
RouterA#
```

The ping was only 60-percent successful because it was started while the router was still making its ISDN call and connecting.

The **show ppp multilink** command is used to display the status of any multilink PPP bundles on the router. We see from the output below that we have two B channels that are connected into a multilink bundle on RouterA.

```
RouterA#show ppp multilink

Bundle RouterB, 2 members, Master link is Virtual-Access1
Dialer Interface is BRI0/0
  0 lost fragments, 0 reordered, 0 unassigned, sequence 0x6/0x8 rcvd/sent
  0 discarded, 0 lost received, 1/255 load

Member Links: 2 ← 2 B channels in a multilink bundle
BRI0/0:1
BRI0/0:2
```

We see that when a call is active on the BRI interface, the D channel still showsup/up (spoofing).

```
RouterA#show interface bri 0/0
BRI0/0  is up, line protocol is up (spoofing) ←  D channel will show up / up
                                                 (spoofing) whether or not a
                                                 call is connected
  Hardware is QUICC BRI with U interface
  Internet address is 196.1.1.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
  Last input 00:00:03, output never, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     168 packets input, 1022 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     168 packets output, 1152 bytes, 0 underruns
     0 output errors, 0 collisions, 4 interface resets
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
```

We see that both B channels are now in an up/up state since we have an active call on both channels of the BRI.

```
RouterA#show interface bri 0/0:1
BRI0/0:1 is up, line protocol is up ← B channel #1
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open, multilink Open
  Last input 00:00:03, output 00:00:03, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     55 packets input, 1292 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     56 packets output, 1702 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     17 carrier transitions

RouterA#show interface bri 0/0:2
BRI0/0:2 is up, line protocol is up ← B channel #2
  Hardware is QUICC BRI with U interface
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open, multilink Open
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
```

```
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     42 packets input, 1116 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     45 packets output, 1554 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     3 carrier transitions
```

The output of the **show dialer** command will give you important information on how many successful calls and how many failed calls have been made. The command output will also show what phone number is being dialed to place calls. Under each B channel you will also see information on how long the call has been connected, the time until the call is disconnected, what number the call actually connected to, and the reason that the call was made.

```
RouterA#show dialer

BRI0/0 − dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8995201                  4         14    00:00:25       successful
0 incoming call(s) have been screened.

BRI0/0:1 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: ip (s5196.1.1.1, d5196.1.1.2) ← Reason for call
Time until disconnect 60 secs ← Time until disconnect
Current call connected 00:00:26 ← Current connect time
Connected to 8995201 (RouterB)← Number connected

BRI0/0:2 − dialer type = ISDN
Idle timer (90 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re−enable (15 secs)
Dialer state is physical layer up
Dial reason: Multilink bundle overloaded
Time until disconnect 62 secs
Current call connected 00:00:27
Connected to 8995201 (RouterB)
```

The **show dialer map** command is another important tool for troubleshooting layer 3 connectivity. This command will display all static and dynamic dialer map statements. We see that we have a static map defined on RouterA that tells us what number to call to get to RouterB at IP address 196.1.1.2.

```
RouterA#show dialer map
Static dialer map ip 196.1.1.2 name RouterB broadcast (8995201) on BRI0/0
```

The following screen print shows what will be output from the router when the router disconnects an ISDN call. ISDN Q931 debugging is enabled on this trace. You can tell that the router is sending the Disconnect message to the ISDN switch because the output message indicates **TX −> DISCONNECT**, which tells us that the router is transmitting a disconnect message to the ISDN switch.

```
%LINEPROTO−5−UPDOWN: Line protocol on Interface Virtual−Access1,
changed state to down
%LINK−3−UPDOWN: Interface Virtual−Access1, changed state to down
%ISDN−6−DISCONNECT: Interface BRI0/0:1  disconnected from 8995201 RouterB,
call lasted 94 seconds
%ISDN−6−DISCONNECT: Interface BRI0/0:2  disconnected from 8995201 RouterB,
call lasted 91 seconds
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x11 ← Disconnect B channel #1
```

```
        Cause i = 0x8090 − Normal call clearing
ISDN BR0/0: TX ->  DISCONNECT pd = 8  callref = 0x12 ← Disconnect B channel #2
        Cause i = 0x8090 − Normal call clearing
ISDN BR0/0: RX <−  RELEASE pd = 8  callref = 0x91
%LINK−3−UPDOWN: Interface BRI0/0:1, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x11
ISDN BR0/0: RX <−  RELEASE pd = 8  callref = 0x92
ISDN BR0/0: Event: incoming ces value = 2
%LINK−3−UPDOWN: Interface BRI0/0:2, changed state to down
ISDN BR0/0: TX ->  RELEASE_COMP pd = 8  callref = 0x12
%LINEPROTO−5−UPDOWN: Line protocol on Interface BRI0/0:1, changed state to down
%LINEPROTO−5−UPDOWN: Line protocol on Interface BRI0/0:2, changed state to down
```

CHAP Failure

Let's force a CHAP authentication failure on our test network and see how to debug this problem. We see from our configuration that configuring CHAP involves two main steps:

1. You need the command **PPP authentication chap** under the appropriate router interface.
2. You need a username that corresponds to the far−end router's hostname as well as a password that matches the password set on the far−end router's username statement.

Our current configuration for RouterA has the statement **username RouterB password 7 070C285F4D06**. The password is encrypted and is not visible in its clear text, but it was set to cisco when the configuration was first created.

Enter configuration mode by typing **config term** at the command prompt. Enter the command **username RouterB password 0 disco**. The corresponding username statement on RouterB contains the password cisco. Having a different CHAP password on RouterA and on RouterB will create a CHAP authentication failure when the two routers attempt to authenticate with each other.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#username RouterB password 0 disco
RouterA(config)#exit
RouterA#
```

Make sure that PPP authentication debugging is enabled by typing the **debug ppp authen** command.

```
RouterA#debug ppp authen
PPP authentication debugging is on
```

Now let's bring up the ISDN circuit by starting a ping from RouterA to RouterB at IP address 196.1.1.2. Notice in the debug output that RouterA declares a CHAP failure when it is trying to authenticate with RouterB. This example demonstrates that it is crucial to enable PPP authentication debugging when having a problem establishing an ISDN call.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:

%LINK−3−UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
PPP BRI0/0:1: Send CHAP Challenge id=3 ← Chap sent on B channel #1
PPP BRI0/0:1: CHAP Challenge id=3 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=3
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=3 received from RouterB
PPP BRI0/0:1: Send CHAP Failure id=3, MD compare failed ← CHAP failure on B
                                                          channel #1
%LINK−3−UPDOWN: Interface BRI0/0:1, changed state to down.
```

```
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
PPP BRI0/0:1: Send CHAP Challenge id=4 ← Chap sent on B channel #1
PPP BRI0/0:1: CHAP Challenge id=4 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=4
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=4 received from RouterB
PPP BRI0/0:1: Send CHAP Failure id=4, MD compare failed ← CHAP failure on B
                                                          channel #1
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down.
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
PPP BRI0/0:1: Send CHAP Challenge id=5 ← The far end router tries to CHAP
                                         several more times and then
                                         disconnects the call
PPP BRI0/0:1: CHAP Challenge id=5 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=5
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=5 received from RouterB
PPP BRI0/0:1: Send CHAP Failure id=5, MD compare failed
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down.
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
PPP BRI0/0:1: Send CHAP Challenge id=6
PPP BRI0/0:1: CHAP Challenge id=6 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=6
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=6 received from RouterB
PPP BRI0/0:1: Send CHAP Failure id=6, MD compare failed
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
PPP BRI0/0:1: treating connection as a callout
PPP BRI0/0:1: Send CHAP Challenge id=7
PPP BRI0/0:1: CHAP Challenge id=7 received from RouterB
PPP BRI0/0:1: Send CHAP Response id=7
PPP BRI0/0:1: CHAP response received from RouterB
PPP BRI0/0:1: CHAP Response id=7 received from RouterB
PPP BRI0/0:1: Send CHAP Failure id=7, MD compare failed
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to down ← Call gets disconnected
Success rate is 0 percent (0/5)
```

Now let's change the CHAP password back to its correct value on RouterA. Enter configuration mode and type the proper username command as shown:

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#username RouterB password 0 cisco ← Correct password
RouterA(config)#exit
RouterA#
```

Now let's verify that we can make a call and pass CHAP authentication. Ping RouterB at IP address 196.1.1.2.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:

%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to 8995201 RouterB

.!!!!
```

```
Success rate is 80 percent (4/5), round-trip min/avg/max = 32/33/36 ms
```

Wrong Dial Number

Now let's see how to debug the situation where you are dialing the wrong number. Recall from the portion of RouterA's configuration shown below that RouterA dials the number 8995201 to call RouterB.

```
RouterA#show run
Building configuration . . .

interface BRI0/0
 ip address 196.1.1.1 255.255.255.0
 encapsulation ppp
 isdn spid1 5101 8995101
 isdn spid2 5102 8995102
 dialer idle-timeout 90
 dialer map ip 196.1.1.2 name RouterB broadcast 8995201 ← Dial 8995201 to
                                                          call RouterB
 dialer load-threshold 1
 dialer-group 1
 no fair-queue
 ppp authentication chap
 ppp multilink
```

Let's change the dial number in RouterA's configuration. Enter configuration mode and delete the current dialer map and add a new dialer map that calls 8996000 instead of the correct number 8995201.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#interface bri 0/0
RouterA(config-if)#no dialer map ip 196.1.1.2 name RouterB broadcast 8995201
RouterA(config-if)#dialer map ip 196.1.1.2 name RouterB broadcast 8996000
RouterA(config-if)#exit                                                  á
RouterA(config)#exit                                          Incorrect number
RouterA#
```

Now turn on ISDN layer 3 call control debugging with the **debug isdn q931** command.

```
RouterA#debug isdn q931
ISDN Q931 packets debugging is on
```

Now let's ping RouterB at IP address 196.1.1.2. Remember that we will be dialing the wrong number to get to RouterB. Our Adtran Atlas 800 ISDN switch is not provisioned with 8996000 as a valid number.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:

ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x1A
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8996000' ← RouterA is dialing
8996000
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x9A
        Channel ID i = 0x89
ISDN BR0/0: RX <-  DISCONNECT pd = 8  callref = 0x9A
        Cause i = 0x8281 - Unallocated/unassigned number ← The ISDN switch
                                                           sends an immediate
                                                           disconnect message
                                                           to RouterA. The
                                                           cause code indicates
                                                           that we are dialing
                                                           an unassigned
```

```
                                                          number.
ISDN BR0/0: TX ->  RELEASE pd = 8  callref = 0x1A
ISDN BR0/0: RX <-  RELEASE_COMP pd = 8  callref = 0x9A.
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x1B ← Router dialed again
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8996000'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x9B
        Channel ID i = 0x89
ISDN BR0/0: RX <-  DISCONNECT pd = 8  callref = 0x9B
        Cause i = 0x8281 - Unallocated/unassigned number
ISDN BR0/0: TX ->  RELEASE pd = 8  callref = 0x1B
ISDN BR0/0: RX <-  RELEASE_COMP pd = 8  callref = 0x9B.
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x1C
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8996000'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x9C
        Channel ID i = 0x89
ISDN BR0/0: RX <-  DISCONNECT pd = 8  callref = 0x9C
        Cause i = 0x8281 - Unallocated/unassigned number
ISDN BR0/0: TX ->  RELEASE pd = 8  callref = 0x1C
ISDN BR0/0: RX <-  RELEASE_COMP pd = 8  callref = 0x9C.
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x1D
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8996000'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x9D
        Channel ID i = 0x89
ISDN BR0/0: RX <-  DISCONNECT pd = 8  callref = 0x9D
        Cause i = 0x8281 - Unallocated/unassigned number
ISDN BR0/0: TX ->  RELEASE pd = 8  callref = 0x1D
ISDN BR0/0: RX <-  RELEASE_COMP pd = 8  callref = 0x9D.
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x1E ← The router dials a total of 5
                                                  times before giving up
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8996000'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x9E
        Channel ID i = 0x89
ISDN BR0/0: RX <-  DISCONNECT pd = 8  callref = 0x9E
        Cause i = 0x8281 - Unallocated/unassigned number
ISDN BR0/0: TX ->  RELEASE pd = 8  callref = 0x1E
ISDN BR0/0: RX <-  RELEASE_COMP pd = 8  callref = 0x9E.
Success rate is 0 percent (0/5)
```

Now let's change the dialer map for RouterA back to the correct dial number. Enter configuration mode, delete the current incorrect dialer map, and add back the correct dial map.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#interface bri 0/0
RouterA(config-if)#no dialer map ip 196.1.1.2 name RouterB broadcast 8996000
RouterA(config-if)#dialer map ip 196.1.1.2 name RouterB broadcast 8995201
RouterA(config-if)#exit
RouterA(config)#exit
RouterA#
```

A ping to RouterB at IP address 196.1.1.2 will now be successful. Notice from the layer 3 trace below that RouterA is now dialing the correct number, 8995201.

```
RouterA#ping 196.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 196.1.1.2, timeout is 2 seconds:
```

```
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x1F
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201' ← Correct number
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0x9F
        Channel ID i = 0x89
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0x9F
        Channel ID i = 0x89
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x1F
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
ISDN BR0/0: TX ->  SETUP pd = 8  callref = 0x20
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201'
ISDN BR0/0: RX <-  CALL_PROC pd = 8  callref = 0xA0
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
ISDN BR0/0: RX <-  CONNECT pd = 8  callref = 0xA0
        Channel ID i = 0x8A
ISDN BR0/0: Event: incoming ces value = 2
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up
ISDN BR0/0: TX ->  CONNECT_ACK pd = 8  callref = 0x20
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:2, changed state to up
%ISDN-6-CONNECT: Interface BRI0/0:2 is now connected to 8995201 RouterB

.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 32/33/36 ms
```

Incoming Call Trace

Now let's look at what an incoming call looks like. Unlike an outgoing call, we see that the Setup message is received by the router. We also see that the router displays the number of the calling party. Notice that the first incoming call comes from 8995101 and the second incoming call comes from 8995102.

```
*Mar  1 00:55:48: ISDN BR0/0: RX <-  SETUP pd = 8  callref = 0x0E
*Mar  1 00:55:48:         Bearer Capability i = 0x8890
*Mar  1 00:55:48:         Channel ID i = 0x89
*Mar  1 00:55:48:         Calling Party Number i = '!', 0x80, '000'8995101'
*Mar  1 00:55:48:         Called Party Number i = 0xC1, '8995201'
%LINK-3-UPDOWN: Interface BRI0/0:1, changed state to up
*Mar  1 00:55:48: ISDN BR0/0: TX ->  CONNECT pd = 8  callref = 0x8E
*Mar  1 00:55:48:         Channel ID i = 0x89
*Mar  1 00:55:48: ISDN BR0/0: RX <-  CONNECT_ACK pd = 8  callref = 0x0E
*Mar  1 00:55:48: PPP BRI0/0:1: Send CHAP challenge id=10 to remote
*Mar  1 00:55:48: PPP BRI0/0:1: CHAP challenge from RouterA
*Mar  1 00:55:48: PPP BRI0/0:1: CHAP response received from RouterA
*Mar  1 00:55:48: PPP BRI0/0:1: CHAP response id=10 received from RouterA
*Mar  1 00:55:48: PPP BRI0/0:1: Send CHAP success id=10 to remote
*Mar  1 00:55:48: PPP BRI0/0:1: remote passed CHAP authentication.
*Mar  1 00:55:48: PPP BRI0/0:1: Passed CHAP authentication with remote.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0/0:1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
*Mar  1 00:55:50: ISDN BR0/0: RX <-  SETUP pd = 8  callref = 0x0F
*Mar  1 00:55:50:         Bearer Capability i = 0x8890
*Mar  1 00:55:50:         Channel ID i = 0x8A
*Mar  1 00:55:50:         Calling Party Number i = '!', 0x80, '000'8995102'
*Mar  1 00:55:50:         Called Party Number i = 0xC1, '8995201'
%LINK-3-UPDOWN: Interface BRI0/0:2, changed state to up
*Mar  1 00:55:50: ISDN BR0/0: TX ->  CONNECT pd = 8  callref = 0x8F
*Mar  1 00:55:50:         Channel ID i = 0x8A
*Mar  1 00:55:51: ISDN BR0/0: RX <-  CONNECT_ACK pd = 8  callref = 0x0F
*Mar  1 00:55:51: PPP BRI0/0:2: Send CHAP challenge id=5 to remote
```

```
*Mar  1 00:55:51: PPP BRI0/0:2: CHAP challenge from RouterA
*Mar  1 00:55:51: PPP BRI0/0:2: CHAP response received from RouterA
*Mar  1 00:55:51: PPP BRI0/0:2: CHAP response id=5 received from RouterA
*Mar  1 00:55:51: PPP BRI0/0:2: Send CHAP success id=5 to remote
*Mar  1 00:55:51: PPP BRI0/0:2: remote passed CHAP authentication.
*Mar  1 00:55:51: PPP BRI0/0:2: Passed CHAP authentication with remote.
%ISDN−6−CONNECT: Interface BRI0/0:1 is now connected to 0008995101 RouterA
%LINEPROTO−5−UPDOWN: Line protocol on Interface BRI0/0:2, changed state to up
%ISDN−6−CONNECT: Interface BRI0/0:2 is now connected to 0008995102 RouterA
```

Conclusion

ISDN is the most popular technology in use today for high−speed switched access and dial backup applications. This chapter demonstrated the ISDN capabilities

Chapter 4: Frame Relay

Overview

Topics Covered in This Chapter

- Frame Relay technology overview
- Configuring a Cisco router as a Frame Relay switch
- LMI autosense
- Configuring discard eligibility
- Frame Relay map statements
- Partial PVC mesh with full connectivity
- Frame Relay subinterfaces
- Frame Relay traffic shaping
- Troubleshooting Frame Relay

Introduction

Frame Relay is the most popular wide area backbone networking protocol today. The Cisco IOS has extensive support for Frame Relay networking. This chapter will examine Frame Relay technology in detail and will then explore how the Cisco IOS supports Frame Relay with eight hands−on labs.

Frame Relay Technology Overview

Frame Relay is a connection−oriented layer 2 transport protocol. It is closely related to X.25, but removes the error correction and retransmission built into X.25. Frame Relay assumes clean digital lines so that extensive error detection is not necessary. The following sections will take a detailed look at Frame Relay technology.

The Justification for Frame Relay

As networks grew larger during the 1980s, more and more leased lines were necessary to tie them together. These leased lines were used as point−to−point links between corporate data centers and offices. To ensure network integrity and redundancy, many sites had more than one leased line connecting them to other sites. Maximum redundancy was provided with a full mesh architecture where every site was connected to every other site. Mathematically, a fully meshed network of n sites requires $n\ (n - 1)/2$ leased lines. Figure 4−1 shows a sample network with five sites.

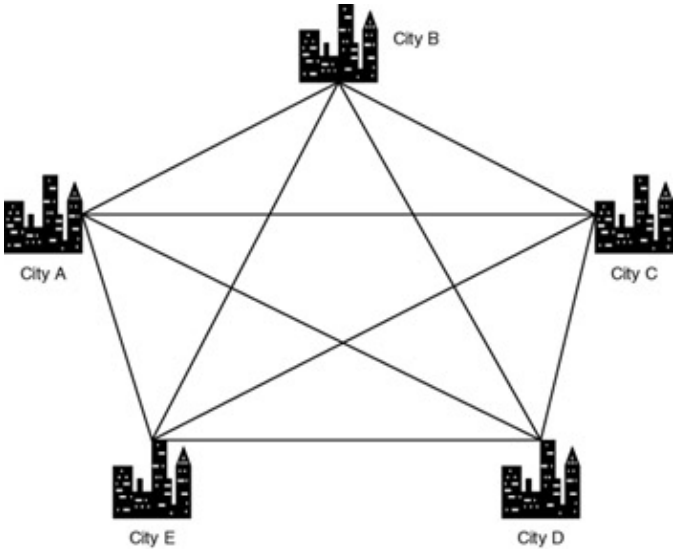


Figure 4−1: A fully meshed leased line network
Fully meshing this network requires 10 leased lines. This is calculated from the $n\ (n - 1)/2$ equation, where $n = 5$ as $5\ (5 - 1)/2 = 10$.

Figure 4−1 demonstrates the three main problems with trying to provide any−to−any connectivity on a large network:

1. As the network grows, the number of leased lines and router interfaces required for a fully meshed network grows by $n\ (n - 1)/2$. After a certain point, it becomes prohibitively expensive to implement a fully meshed network.
2. Excessive line charges are being paid, since the network owner also owns all the links end to end (leased lines).
3. The network is not efficient, since statistically not all lines will be utilized at once. Each site can only be transmitting to one other site at any given time, utilizing one leased line. All other leased lines will be idle during this period.

What Is Frame Relay

Figure 4−2 shows a solution to the three problems discussed in the previous section with a fully or partially meshed network.

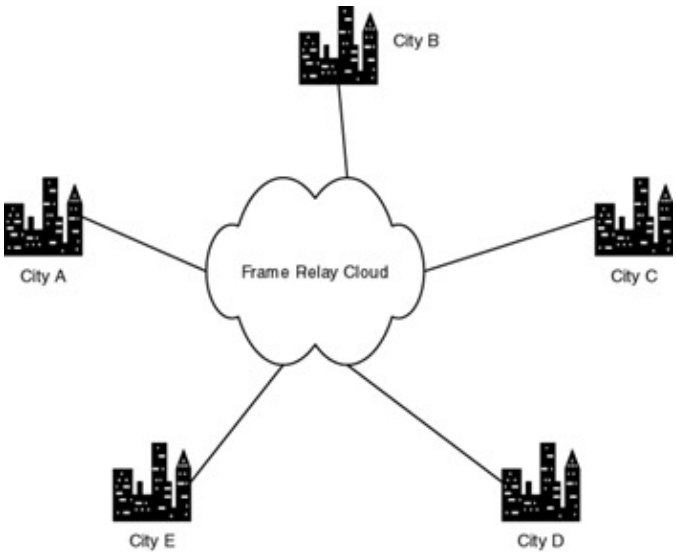


Figure 4−2: Sample Frame Relay network topology

Each site in the network has a single leased line into the network cloud. In Figure 4−2, there are five sites and five leased lines. Notice that the number of leased lines has been cut in half with respect to a fully meshed network. Line charges are also lower because you only pay for a circuit to the nearest Frame Relay switch instead of paying for an end−to−end leased line. This is the attraction of Frame Relay, the ability to take a partially or fully meshed network of leased lines and convert it to a network with only one leased line and router interface per site.

Frame Relay is a simple and streamlined protocol. An important feature of Frame Relay to keep in mind is that it is an access protocol. This means that Frame Relay only defines signaling and data formats between the router and the Frame Relay Switch.

We should note an important issue at this point. It can be argued that the network depicted in Figure 4−2 is less robust than the network depicted in Figure 4−1. This is because the network in Figure 4−2 has only one link going to each site, while the network in Figure 4−1 has four links going to each site. This problem is addressed in modern networks by also bringing an ISDN circuit to each of the sites on the network. ISDN architecture is discussed in more detail in Chapter 3, "ISDN." Suffice it to say at this point that a combination of Frame Relay and ISDN circuits at each site can provide the same level of network integrity as with a fully meshed network.

Frame Relay Terminology

Figure 4−3 shows some of the terminology used in a Frame Relay network. Two routers are connected via Frame Relay.

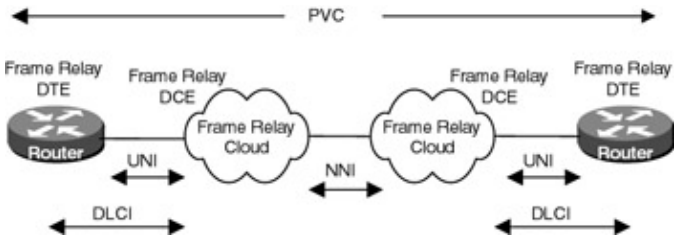


Figure 4−3: Frame Relay terminology

- Both routers are referred to as *Frame Relay DTE devices*.
- The Frame Relay switches that connect to the routers are referred to as *Frame Relay DCE devices.*
- The access rate is the physical speed of the link between the user and the Frame Relay switch.
- The circuit that connects these two routers is known as a PVC or permanent virtual circuit. This PVC consists of a unique DLCI or data link connection identifier at each end of the circuit.
- Links between different Frame Relay providers switch networks are referred to as NNIs or network−to−network interfaces links.
- The signaling protocol between the routers and the Frame Relay switch is referred to as the LMI or local management interface.
- The interface between the routers and the Frame Relay switches is referred to as the UNI or user network interface.

Frame Relay Addressing

Frame Relay is a layer 2 connection−oriented protocol. A Frame Relay circuit between two endpoints can either be permanent or switched. A *permanent* Frame Relay circuit is referred to as a *permanent virtual circuit* or a *PVC.* A switched Frame Relay circuit is referred to as a *switched virtual circuit* or *SVC.*

A point−to−point PVC connects two *endpoints.* Each endpoint refers to the PVC with a data link connection identifier or DLCI. These DLCI values are locally significant. This means that the DLCI value does not have to be unique between any other port on the Frame Relay network.

The primary job of a Frame Relay switch network is to receive traffic on one port and, by examining the

DLCI value associated with that traffic, send it out on the proper destination port with the proper DLCI value. The DLCI values on both sides of the circuit can be the same, or they can be different. If they are different, the Frame Relay switch network is responsible for doing the DLCI label switching.

Figure 4−4 shows an example of Frame Relay addressing. The figure shows four cities tied together via a Frame Relay network. In the diagram three PVCs are defined. Each PVC represents a permanent circuit connection between its two endpoints.

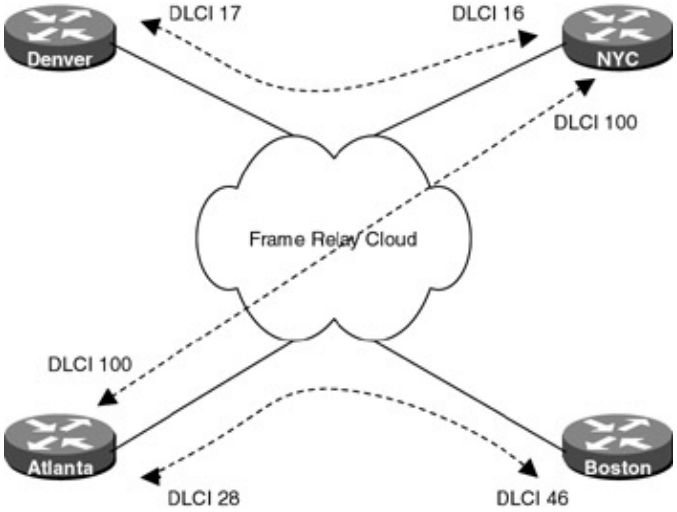


Figure 4−4: DLCI addressing example

These PVCs can be defined as follows:

- A PVC between Denver and NYC. This PVC consists of DLCI 17 on the Denver side and DLCI 16 on the NYC side. Any traffic sent into the Denver Frame Relay switch with a DLCI value of 17 will be sent out of the NYC Frame Relay switch with a DLCI value of 16. Likewise, any traffic sent into the NYC Frame Relay switch with a DLCI value of 16 will be sent out of the Denver Frame Relay switch with a DLCI value of 17.
- A PVC between Atlanta and NYC. This PVC consists of DLCI 100 on the Atlanta side and DLCI 100 on the NYC side. Any traffic sent into the Atlanta Frame Relay switch with a DLCI value of 100 will be sent out of the NYC Frame Relay switch with a DLCI value of 100. Likewise, any traffic sent into the NYC Frame Relay switch with a DLCI value of 100 will be sent out of the Atlanta Frame Relay switch with a DLCI value of 100. Notice that the DLCI value is 100 at both sides of the PVC. This is allowed since a DLCI is only locally significant.
- A PVC between Atlanta and Boston. This PVC consists of DLCI 28 on the Atlanta side and DLCI 46 on the Boston side. Any traffic sent into the Atlanta Frame Relay switch with a DLCI value of 28 will be sent out of the Boston Frame Relay switch with a DLCI value of 46. Likewise, any traffic sent into the Boston Frame Relay switch with a DLCI value of 46 will be sent out of the Atlanta Frame Relay switch with a DLCI value of 28.

Frame Relay Frame Format

One of Frame Relay's attractions is the fact that it is very efficient. With only two bytes of address overhead, you can transmit up to 8K of data. Figure 4−5 shows the format of the Frame Relay frame.

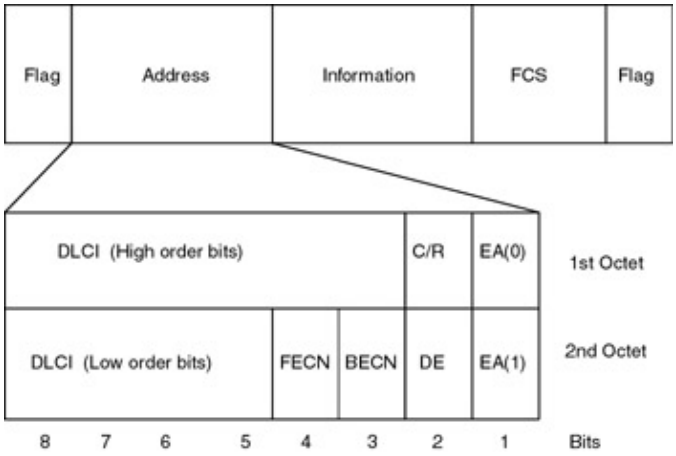


Figure 4−5: Frame Relay frame format

The fields in Figure 4−5 can be explained as follows:

Flag: Each Frame Relay frame starts and ends with at least one delimiter character of 7Eh. This bit sequence allows the receiver to synchronize on the start and end of a frame. A special bit destuffing algorithm exists that ensures that this 7Eh character cannot appear in the user traffic.

Address: The address field is either 2, 3, or 4 bytes long. A 2−byte address field is almost always used. The following fields are contained in a 2−byte address field:

- **DLCI:** This is a 10−bit field that contains the Data Link Connection Identifier value that identifies a virtual circuit. For a typical 2−byte Frame Relay header, this field can take on a value between 0 and 1023. Some DLCI values such as 0 and 1023 are reserved for Frame Relay management. Valid user DLCI values range from 16 through 991. Recall from the previous section that the DLCI value tells the Frame Relay switch the proper destination of traffic sent into the Frame Relay network.
- **CR:** This bit is known as the Command Response bit. This bit is not used in most Frame Relay implementations.
- **EA:** This is the Extended Address bit. This bit is used to indicate whether a 2−, 3−, or 4−byte header is being used. This bit is set to a 1 in the last byte of the header.
- **FECN:** This is the Forward Explicit Congestion Notification bit. It is used to tell the user receiving Frame Relay frames that congestion exists in the direction that the frame was sent. FECN will be more fully discussed in a later section.
- **BECN:** This is the Backward Explicit Congestion Notification bit. It is used to tell the user receiving Frame Relay frames that congestion exists in the reverse direction that the frame was sent. BECN will be more fully discussed in a later section.
- **DE:** This is the Discard Eligible bit. This bit can be set by either the Frame Relay DTE equipment (such as a router) or by the Frame Relay switch network. A frame with the DE bit set indicates that the frame can be discarded when the Frame Relay network becomes congested. Whether or not DE flagged frames are discarded depends on how the Frame Relay network has been provisioned.

User Data (Information): This field contains the actual user payload. The default Cisco Frame Relay frame size is 1500 bytes for a synchronous serial interface. Most Frame Relay equipment can pass larger frames. The largest frame size is 8192 bytes, although many Frame Relay devices cannot pass a frame that large. The main disadvantage of using frames larger than 4K bytes is that the Frame Check Sequence can no longer detect many types of data errors.

FCS (Frame Check Sequence): This is a 2−byte cyclic redundancy check that is calculated on the entire frame except the flags and the 2 bytes of FCS. A Frame Relay network will discard any frames that have a bad FCS. The FCS is only guaranteed for Frame Relay frames of 4K bytes or less, larger frames may not have their errors detected with the 2−byte CRC check that Frame Relay provides.

Frame Relay Congestion Control

Another feature of Frame Relay that makes it an efficient protocol is its simple congestion control. Three bits in the Frame Relay address field are used for congestion control: BECN, FECN, and DE. Figure 4−6 shows how the BECN and FECN bits can be set in a Frame Relay frame.

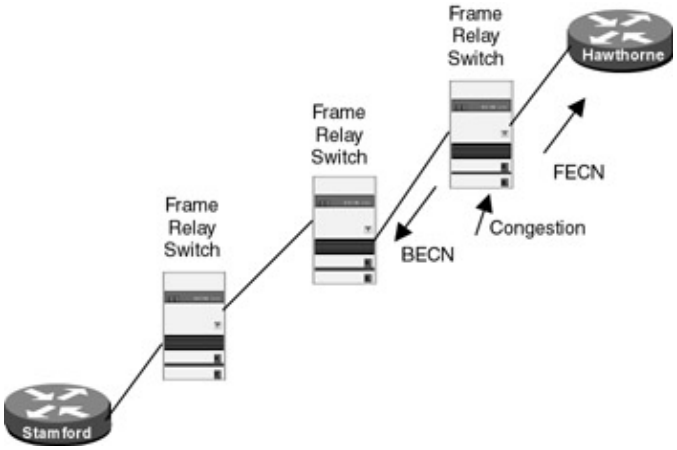


Figure 4−6: Frame Relay congestion example

A PVC is configured between Stamford and Hawthorne. Assume that the path from Hawthorne to Stamford is not congested but the path from Stamford to Hawthorne becomes congested. The Frame Relay network will set the FECN bit to a 1 in those frames that are going from Stamford to Hawthorne. The Frame Relay network will set the BECN bit to a 1 in those frames that are going from Hawthorne to Stamford. A router receiving these BECN or FECN tagged frames has no obligation to react to the fact that congestion exists in the network. Most end user devices are not able to react. With the introduction of IOS version 11.2, Cisco added support for reacting to the BECN bit in Frame Relay frames. This feature, enabled by default, causes the router to dynamically throttle outgoing traffic on a per−PVC basis. When BECN flagged frames stop coming into the router, the outbound traffic rate is once again increased.

The Discard Eligibility (DE) bit will be explained in more detail in the "Frame Relay Class of Service" section later in this chapter.

Frame Relay Error Handling

Frame Relay assumes that the circuits carrying the Frame Relay traffic will have very low error rates. Unlike X.25, Frame Relay switches do not retransmit errored frames. If a Frame Relay switch encounters a frame with a bad FCS (Frame Check Sequence), that frame is simply discarded.

Retransmission is the responsibility of the hosts, not the Frame Relay switch network or the router.

Frame Relay Class of Service

Every PVC on a Frame Relay network can be quantified by a set of standard Frame Relay measurements. These measurements constitute a service contract between the Frame Relay user and the Frame Relay provider. These items are:

Access Rate: The access rate is the speed of the circuit between the user and the Frame Relay network. The maximum data rate for traffic sent into the Frame Relay network is bounded by the access rate.

CIR: The CIR (Committed Information Rate) of a PVC is a measurement, in bits/second, of the maximum amount of traffic that the Frame Relay service provider agrees to carry through the network. The CIR value will usually be less than the access rate of the line. Keep in mind that several PVCs will be sharing the same physical link between the users' equipment and the Frame Relay switch.

Bc: Bc is Committed Burst Size. Bc is defined as the maximum number of bits that the Frame Relay network agrees to carry during time interval Tc. By definition Tc = Bc / CIR. CIR and Bc may seem similar, but notice that CIR is expressed in bits/sec, whereas Bc is expressed only in bits.

Be: Be is Excess Burst Size. It is a measurement of the maximum number of bits that the Frame Relay network will try to carry above the CIR value during time Tc.

Tc: Tc is the Committed Rate Measurement Interval. Tc by definition is equal to Bc/CIR. Tc is usually set to 1 second.

A Frame Relay switch needs to have a way to determine if the service contract is being met. The value of Tc tells the Frame Relay provider to take a traffic sample every Tc seconds. Most Frame Relay switches will mark any traffic above the CIR as discard eligible and will set the DE bit.

Figure 4−7 shows a graphical description of the Frame Relay class of service parameters. In this diagram Tc is set to 1 second. Four frames are sent during the 1 second interval. The CIR is set to 16 Kbps for this circuit; the Bc and Be are both set to 16Kb. The first three frames are not marked DE (Discard Eligible) because the total number of bits transmitted when any of the frames begins is less than 16Kb. The fourth frame is marked DE because when the fourth frame begins the total number of bits transmitted during the current one−second interval has already exceeded 16Kb.

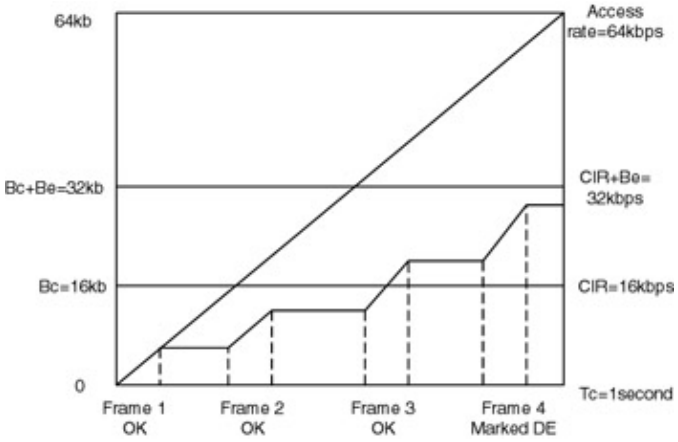


Figure 4−7: Frame Relay class of service

Local Management Interface

Frame Relay provides a simple signaling protocol between the Frame Relay switch and the Frame Relay DTE (router). This signaling protocol is known as the LMI (local management interface).

The LMI signaling protocol provides for notification of addition and deletion of PVCs, as well as periodic keep−alive messages between the Frame Relay switch and Frame Relay DTE.

There are three types of LMI. The LMI type must be set the same on the Frame Relay switch and the attached Frame Relay DTE (router). The Frame Relay provider will usually tell the customer which of the three types will be used on their network. The three LMI types are:

1. **ANSI Annex D** — Annex D uses DLCI 0 to pass status information between the Frame Relay switch and the Frame Relay DTE. Cisco refers to this LMI type as ANSI.
2. **CCITT Annex A** — Annex A also uses DLCI 0 to pass status information between the Frame Relay switch and the Frame Relay DTE. Annex A signaling also provides the CIR value of each PVC that is provisioned on the Frame Relay switch port providing status. Cisco refers to this LMI type as Q933A.
3. **LMI** — LMI uses DLCI 1023 to pass status information between the Frame Relay switch and the Frame Relay DTE. Cisco refers to this LMI type as Cisco.

LMI works in the following manner:

- Every 10 seconds the DTE requests a status from the Frame Relay switch. This status request will be sent to the switch over one of two reserved DLCIs, 0 or 1023, depending on which of the three LMI signaling protocols has been chosen.
- The Frame Relay switch will respond to the status request with a status response. This exchange is depicted in Figure 4−8. This exchange is referred to as a keep−alive or aliveness check, since its function is to tell both the DTE and the Frame Relay switch that there is still a communication path between them.

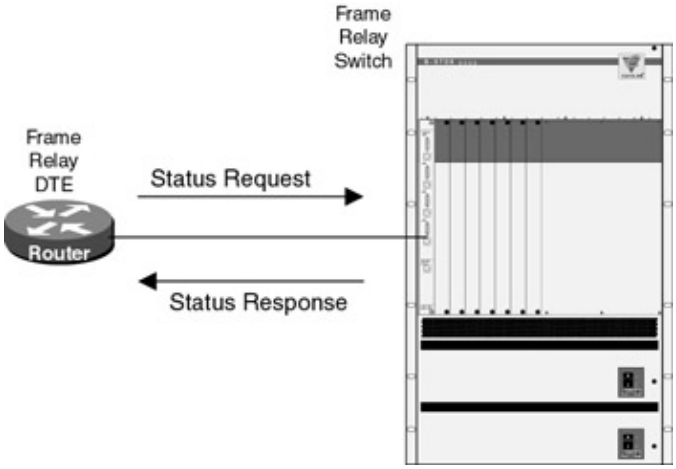


Figure 4−8: Frame Relay LMI

- Every sixth status request from the DTE is sent as a full status request. A full status request not only serves an an aliveness check, but also requests the Frame Relay switch to respond with a list of all DLCIs that have been defined on the port of the Frame Relay switch.
- The Frame Relay switch responds with a list of all defined DLCIs on the particular Frame Relay switch port.

Both the DTE and the Frame Relay switch maintain individual sequence numbers during their exchanges. These sequence numbers can be used to determine if an LMI sequence has been lost.

The following four frames show a typical LMI exchange between the router and a Frame Relay switch. The first frame (ID = 5) is a status request from the router to the Frame Relay switch. Notice that the DLCI is 0, which in this case is Annex D.

The second frame (ID = 6) is the reply from the switch to the first frame (ID = 5). Notice from the timestamp that the reply comes within 10 milliseconds.

The next frame (ID = 7) is a full status request from the router to the Frame Relay switch. The final frame (ID = 8) is the reply to frame #7. The switch replies to the full status request with a listing of all DLCIs that are configured on its port.

Status Request from the Router to the Frame Relay Switch

```
Port A RX2 ID=5, 01/19/99, 14:32:08.547325
Length=16, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                        00h
DLCI_lsb                                                         0h

DLCI                                                  Annex D or A 0
CR                                                               0
EA                                                               0
FECN                                                             0
BECN                                                             0
DE                                                               0
EA                                                               1
Q.933 FRAME RELAY SVC
```

```
Control                                                  UI Frame 03h
Protocol Discriminator                                Call Control 08h
Reference Flag                                     Msg from Origination 0
Call Reference Value [len=0]                                          0
Message Type                                            STATUS ENQ 75h
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                              (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                          Link Verification 01h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                         E6h
Last Rcvd Seq                                                       DAh
FCS                                                        Good FFC8h
```

Status Reply from the Frame Relay Switch to Router

```
Port A RX1 ID=6, 01/19/99, 14:32:08.563200
Length=16, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                            00h
DLCI_lsb                                                             0h
DLCI                                                    Annex D or A 0
CR                                                                   0
EA                                                                   0
FECN                                                                 0
BECN                                                                 0
DE                                                                   0
EA                                                                   1
Q.933 FRAME RELAY SVC
Control                                                  UI Frame 03h
Protocol Discriminator                                Call Control 08h
Reference Flag                                     Msg from Origination 0
Call Reference Value [len=0]                                          0
Message Type                                                STATUS 7Dh
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                              (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                          Link Verification 01h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                         DBh
Last Rcvd Seq                                                       E6h
FCS                                                        Good 004Ah
```

Full Status Request from the Router to the Frame Relay Switch

```
Port A RX2 ID=7, 01/19/99, 14:32:18.634050
Length=16, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                            00h
DLCI_lsb                                                             0h
DLCI                                                    Annex D or A 0
CR                                                                   0
EA                                                                   0
FECN                                                                 0
BECN                                                                 0
DE                                                                   0
EA                                                                   1
Q.933 FRAME RELAY SVC
Control                                                  UI Frame 03h
Protocol Discriminator                                Call Control 08h
Reference Flag                                     Msg from Origination 0
Call Reference Value [len=0]                                          0

Message Type                                            STATUS ENQ 75h
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                              (Codeset 5) 5h
```

```
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                                    Full Status 00h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                          E7h
Last Rcvd Seq                                                        DBh
FCS                                                          Good EACBh
```

Full Status Reply from the Frame Relay Switch to the Router

```
Port A RX1 ID=8, 01/19/99, 14:32:18.651225
Length=31, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                             00h
DLCI_lsb                                                              0h
DLCI                                                    Annex D or A 0
CR                                                                    0
EA                                                                    0
FECN                                                                  0
BECN                                                                  0
DE                                                                    0
EA                                                                    1
Q.933 FRAME RELAY SVC
Control                                                    UI Frame 03h
Protocol Discriminator                                  Call Control 08h
Reference Flag                                 Msg from Origination 0
Call Reference Value [len=0]                                          0
Message Type                                               STATUS 7Dh
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                               (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                                    Full Status 00h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                          DCh
Last Rcvd Seq                                                        E7h
IE − Annex D PVC Status [Code=7] [Len=3]
PVC DLCI MSB                                                         06h
PVC DLCI LSB                                                          9h
PVC DLCI                                                             105
N(ew)                                                             No 0h
D(eleted)                                                         No 0h
A(ctive)                                                         Yes 1h
IE − Annex D PVC Status [Code=7] [Len=3]
PVC DLCI MSB                                                         06h
PVC DLCI LSB                                                          Bh
PVC DLCI                                                             107
N(ew)                                                             No 0h
D(eleted)                                                         No 0h
A(ctive)                                                         Yes 1h
IE − Annex D PVC Status [Code=7] [Len=3]
PVC DLCI MSB                                                         07h
PVC DLCI LSB                                                          3h
PVC DLCI                                                             115
N(ew)                                                             No 0h
D(eleted)                                                         No 0h
A(ctive)                                                         Yes 1h
FCS                                                          Good 4629h
```

Asynchronous Status Updates

The previous section mentioned that Frame Relay LMI exchanges occur every 10 seconds between the router and the Frame Relay switch. The Frame Relay specification also provides for LMI updates that can be sent at any time from the Frame Relay switch to the router. These updates can be sent for a variety of reasons such as a PVC being added or a PVC being deleted. These type of updates are refered to as *asynchronous updates* because they are sent at any time, by contrast with the normal LMI exchanges, which are only sent at 10 second intervals. The following trace shows an asynchronous update being sent when a new PVC is added on

the switch.

The first frame (ID = 60) is a standard status request from the router to the Frame Relay switch. Notice that it occurs at time 16:40:57.56.

The second frame (ID = 61) is the reply to the first frame (ID = 60). This frame is sent from the Frame Relay switch to the router. Notice that it is sent at time 16:40:57.57, 10 milliseconds after the initial request.

The third frame (ID = 62) is an asynchronous update sent from the Frame Relay switch to the router. It was sent when a new DLCI (DLCI = 199) was configured on the Frame Relay switch. Notice that the update is not sent out on a 10−second boundary but is being sent between the normal 10−second LMI exchanges.

The fourth and final frame (ID = 64) is the next status request from the router to the Frame Switch. Notice that it occurs on the 10−second interval of the normal LMI exchanges.

Status Request from the Router to the Frame Relay Switch

```
Port A RX2 ID=60, 02/15/99, 16:40:57.561675
Length=16, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                        00h
DLCI_lsb                                                         0h
DLCI                                                 Annex D or A 0
CR                                                               0
EA                                                               0
FECN                                                             0
BECN                                                             0
DE                                                               0
EA                                                               1
Q.933 FRAME RELAY SVC
Control                                                UI Frame 03h
Protocol Discriminator                               Call Control 08h
Reference Flag                                   Msg from Origination 0
Call Reference Value [len=0]                                     0
Message Type                                          STATUS ENQ 75h
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                            (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                          Link Verification 01h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                     B3h
Last Rcvd Seq                                                   03h
FCS                                                   Good FC2Eh
```

Status Reply from the Frame Relay Switch to the Router

```
Port A RX1 ID=61, 02/15/99, 16:40:57.573700
Length=16, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                        00h
DLCI_lsb                                                         0h
DLCI                                                 Annex D or A 0
CR                                                               0
EA                                                               0
FECN                                                             0
BECN                                                             0
DE                                                               0
EA                                                               1
Q.933 FRAME RELAY SVC
Control                                                UI Frame 03h
Protocol Discriminator                               Call Control 08h
Reference Flag                                   Msg from Origination 0
Call Reference Value [len=0]                                     0
```

```
Message Type                                                  STATUS 7Dh
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                                (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                        Link Verification 01h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                     04h
Last Rcvd Seq                                                   B3h
FCS Good DB93h
```

Asynchronous Update from the Frame Relay Switch to the Router

```
Port A RX1 ID=62, 02/15/99, 16:41:01.011100
Length=17, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                        00h
DLCI_lsb                                                         0h
DLCI                                                 Annex D or A 0
CR                                                               0
EA                                                               0
FECN                                                             0
BECN                                                             0
DE                                                               0
EA                                                               1
Q.933 FRAME RELAY SVC
Control                                                UI Frame 03h
Protocol Discriminator                               Call Control 08h
Reference Flag                                 Msg from Origination 0
Call Reference Value [len=0]                                     0
Message Type                                           STATUS 7Dh
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                                (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                  Single PVC Async Status 02h
IE − Annex D PVC Status [Code=7] [Len=3]
PVC DLCI MSB                                                    0Ch
PVC DLCI LSB                                                     7h
PVC DLCI                                                       199
N(ew)                                                         No 0h
D(eleted)                                                     No 0h
A(ctive)                                                     Yes 1h
FCS                                                    Good 1320h
```

Status Request from the Router to the Frame Relay Switch

```
Port A RX2 ID=64, 02/15/99, 16:41:07.657950
Length=16, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                        00h
DLCI_lsb                                                         0h
DLCI                                                 Annex D or A 0
CR                                                               0
EA                                                               0
FECN                                                             0
BECN                                                             0
DE                                                               0
EA                                                               1
Q.933 FRAME RELAY SVC
Control                                                UI Frame 03h
Protocol Discriminator                               Call Control 08h
Reference Flag                                 Msg from Origination 0
Call Reference Value [len=0]                                     0
Message Type                                       STATUS ENQ 75h
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                                (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
```

```
IE − Annex D Report Type                                  Full Status 00h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                         B4h
Last Rcvd Seq                                                       04h
FCS                                                          Good 0F1Ch
```

Inverse Address Resolution Protocol (Inverse ARP)

Frame Relay inverse ARP provides a method to associate a far−end layer 3 network address with the local layer 2 DLCI.

When a Frame Relay circuit is initialized, a router attached to a Frame Relay switch does not have any address information except its own IP address. As shown in Figure 4−9, when the router sends a full status LMI request to the Frame Relay switch, the switch will respond with all DLCIs that are defined on the circuit. The router needs a way to find out what the IP address of the far−end router is on the other end of the DLCI. The router does this by sending an inverse ARP request out on each local DLCI that is defined on the circuit. The inverse ARP request travels to the far−end router, where it is received and replied to. The router that initiated the inverse ARP request receives the reply. This reply contains the IP address of the far−end router.

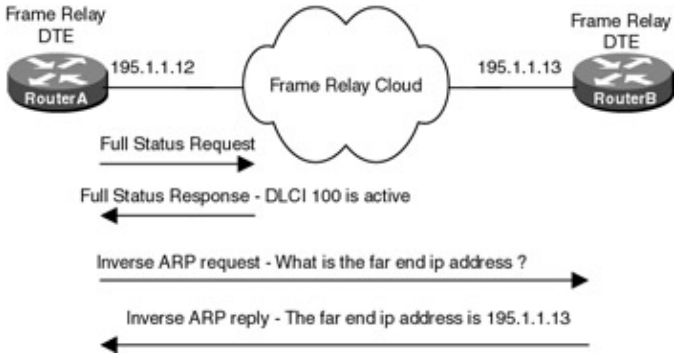


Figure 4−9: Frame Relay inverse ARP

The following trace shows how an inverse ARP exchange takes place between a router and a Frame Relay switch.

The first frame (ID = 6) is the inverse ARP request from the router to the Frame Relay network. Notice that the request is being sent over DLCI 100. This request will travel right through the Frame Relay switch and will be terminated by the router at the far end of the circuit. Also notice that the inverse ARP frame contains the IP address (195.1.1.12) of the sending router.

The second frame (ID = 7) is the reply packet to the original inverse ARP request (ID = 6). Notice that this frame contains a new IP address. This is the IP address of the router on the far end of the circuit (IP = 195.1.1.13).

Inverse ARP Request

```
Port A DTE ID=6, 01/25/99, 15:36:08.934350
Length=32, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                            06h
DLCI_lsb                                                             4h
DLCI                                                               100
CR                                                                   0
EA                                                                   0
FECN                                                                 0
BECN                                                                 0
DE                                                                   0
EA                                                                   1
NLPID − NETWORK LEVEL PROTOCOL ID
```

```
Control                                                LAPD UI Frame 03h
Pad                                                               00h
Network Level Protocol ID                                    SNAP 80h
SNAP Header OUI                                       Ethertype 000000h
Ethertype                                                     ARP 0806h
ARP − ADDRESS RESOLUTION PROTOCOL
Hardware Type                                         Frame Relay 000Fh
Protocol Type                                              DOD IP 0800h
Hrdwr Addr Len (in bytes)                                            2
Prtcl Addr Len (in bytes)                                            4
Operation Code                                      InARP Request 0008h
Sender Hardware Addr
 00000                    0000                ..
Sender Protocol Addr                                       195.1.1.12
Target Hardware Addr
 00000                    1841                .A
Target Protocol Addr                                          0.0.0.0
FCS                                                    Good 80E9h
```

Inverse ARP Reply

```
Port A DCE ID=7, 01/25/99, 15:36:08.942025
Length=32, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                          06h
DLCI_lsb                                                           4h
DLCI                                                              100
CR                                                                  0
EA                                                                  0
FECN                                                                0
BECN                                                                0
DE                                                                  0
EA                                                                  1
NLPID − NETWORK LEVEL PROTOCOL ID
Control                                                LAPD UI Frame 03h
Pad                                                               00h
Network Level Protocol ID                                    SNAP 80h
SNAP Header OUI                                       Ethertype 000000h
Ethertype                                                     ARP 0806h
ARP − ADDRESS RESOLUTION PROTOCOL
Hardware Type                                         Frame Relay 000Fh
Protocol Type                                              DOD IP 0800h
Hrdwr Addr Len (in bytes)                                            2
Prtcl Addr Len (in bytes)                                            4

Operation Code                                      InARP Response 0009h
Sender Hardware Addr
 00000                    0000                ..
Sender Protocol Addr                                       195.1.1.13
Target Hardware Addr
 00000                    1841                .A
Target Protocol Addr                                       195.1.1.12
FCS                                                    Good 84E4h
```

Cisco Frame Relay Capabilities

The Cisco IOS includes support for the standard Frame Relay features that were discussed in the previous sections. The IOS also includes support for additional advanced features such as Quality of Service support (QOS), Frame Relay switching, and subinterfaces.

Frame Relay Switching

A Cisco router can be configured as a Frame Relay switch. This capability is usually not used in a production network; it is mainly intended for use in test beds and example networks. The examples in this chapter will use a Cisco router configured as a Frame Relay switch.

IETF and Cisco Encapsulation

When a Frame Relay frame comes into a router, the router must have some way of knowing what higher−layer protocol is encapsulated in the Frame Relay frame. There are two methods on a Cisco router of encapsulating higher−layer traffic in a Frame Relay frame: Cisco proprietary and RFC 1490 encapsulation. Cisco proprietary encapsulation is the default encapsulation for Frame Relay traffic. IETF (RFC 1490) encapsulation should be used when the router will be communicating with another vendor's equipment. The following analyzer traces show an example of Cisco proprietary encapsulation and IETF (RFC 1490) encapsulation.

IETF Encapsulation The following frame shows an example of IETF encapsulation. Notice the control and NLPID (Network Level Protocol ID) bytes identifying what protocol is encapsulated in the Frame Relay frame.

```
Port A RX1 ID=7, 01/19/99, 15:26:24.045900
Length=66, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                    06h
DLCI_lsb                                                     Bh
DLCI                                                        107
CR                                                            0
EA                                                            0
FECN                                                          0
BECN                                                          0
DE                                                            0
EA                                                            1
NLPID − NETWORK LEVEL PROTOCOL ID
Control                                          LAPD UI Frame 03h
Network Level Protocol ID                                 IP CCh
IP − INTERNET PROTOCOL
Version                                                       4
IHL (in 32 bit words)                                         5
Precedence                                           Routine 0h
D(elay)                                                Normal 0
T(hroughput)                                           Normal 0
R(eliability)                                          Normal 0
Total Length (in octets)                                     60
Identification                                            525Fh
D(on't) F(ragment)                                         No 0
M(ore) F(ragments)                                         No 0
Fragment Offset (in 8 octets)                                 0
Time To Live (in seconds)                                    31
Protocol                                             ICMP 01h
Header Checksum                                           78E7h
Class A Source IP Address
Source Net ID                                                10
Source Host ID                                           656228
Source Addr                                        10.10.3.100
Class C Destination IP Address
Destination Net ID                                       131329
Destination Host ID                                          12
Destination Addr                                   194.1.1.12
ICMP − INTERNET CONTROL MESSAGE PROTOCOL
Type                                                 Echo 08h
Code                                                      00h
Checksum                                                2F5Ch
ID                                                      0100h
```

```
Sequence No                                                    1D00h
Dump
 00000                  6162  6364  6566  6768  696A         abcdefghij
 00010                  6B6C  6D6E  6F70  7172  7374         klmnopqrst
 00020                  7576  7761  6263  6465  6667         uvwabcdefg
 00030                  6869                                 hi
FCS                                                          Good D589h
```

Cisco Encapsulation — The following frame shows an example of Cisco encapsulation. Notice the two−byte ethertype identifying what protocol is encapsulated in the Frame Relay frame.

```
Port A RX1 ID=77, 01/19/99, 14:38:02.299325
Length=66, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                       06h
DLCI_lsb                                                        Bh
DLCI                                                          107
CR                                                              0
EA                                                              0
FECN                                                            0
BECN                                                            0
DE                                                              0
EA                                                              1
NLPID − NETWORK LEVEL PROTOCOL ID
Ethertype                                                DOD IP 0800h
IP − INTERNET PROTOCOL
Version                                                         4
IHL (in 32 bit words)                                           5
Precedence                                                Routine 0h
D(elay)                                                     Normal 0
T(hroughput)                                                Normal 0
R(eliability)                                               Normal 0
Total Length (in octets)                                       60
Identification                                              215Eh
D(on't) F(ragment)                                           No 0
M(ore) F(ragments)                                           No 0
Fragment Offset (in 8 octets)                                   0
Time To Live (in seconds)                                      31
Protocol                                                  ICMP 01h
Header Checksum                                             A9E8h
Class A Source IP Address
Source Net ID                                                  10
Source Host ID                                             656228
Source Addr                                           10.10.3.100
Class C Destination IP Address
Destination Net ID                                         131329
Destination Host ID                                            12
Destination Addr                                       194.1.1.12
ICMP − INTERNET CONTROL MESSAGE PROTOCOL
Type                                                      Echo 08h
Code                                                         00h
Checksum                                                    335Ch
ID                                                          0100h
Sequence No                                                 1900h
Dump
 00000                  6162  6364  6566  6768  696A         abcdefghij
 00010                  6B6C  6D6E  6F70  7172  7374         klmnopqrst
 00020                  7576  7761  6263  6465  6667         uvwabcdefg
 00030                  6869                                 hi
FCS                                                          Good 4264h
```

Traffic Shaping

Cisco introduced Frame Relay traffic shaping support in IOS version 11.2. *Traffic shaping* allows you to define the outbound and/or inbound traffic rate on a per−PVC basis. The traffic rate can be defined in terms of

either peak or average rate. This chapter includes a lab example showing how traffic shaping is configured.

DE Support

The DE bit in the Frame Relay header is used to indicate that a frame has a low priority and can be discarded when the Frame Relay network becomes congested. The Cisco IOS permits the user to specify which Frame Relay frames will have the DE bit set. The DE bit can be set based on upper−layer protocols, a specific TCP or UDP port, an access list number, a packet size, or packet fragmentation. This chapter includes a lab example showing how DE support is configured.

BECN Support

Cisco introduced BECN support in IOS version 11.2. BECN support is enabled by default on the router. When the router sees BECN set on incoming frames, it automatically throttles back outgoing traffic. Normal outgoing traffic levels are resumed when the router no longer sees incoming traffic with the BECN bit set.

Payload Compression

The Cisco IOS supports Frame Relay payload compression using the STAC compression algorithm. Payload compression differs from traditional link compression in that only the data portion of the Frame Relay frame is compressed. The Frame Relay header is left intact with payload compression.

LMI Autosense

LMI autosense support was added in IOS 11.2. It allow the router to automatically determine which of the three LMI protocols are configured on the Frame Relay switch that is attached to the Cisco router. LMI autosense works by having the router send out an LMI full status request to the attached Frame Relay switch in all three LMI formats: ANSI Annex D, CCITT Annex A, and LMI. When the router receives a response from the Frame Relay switch, it will decode the response and determine which of the three LMI formats are being used. This chapter includes a lab example showing how LMI autosense is configured.

Commands Discussed in This Chapter

- **clear frame−relay−inarp**
- **debug frame−relay lmi**
- **debug frame−relay packet**
- **encapsulation frame−relay [cisco | ietf]**
- **frame−relay bc {in | out}** ***bits***
- **frame−relay be {in | out}** ***bits***
- **frame−relay becn−response−enable**
- **frame−relay cir {in | out}** ***bps***
- **frame−relay class** *name*
- **frame−relay de−group** *group−number dlci*
- **frame−relay de−list** *list−number* **{protocol** *protocol* | **interface** *type number***}** *characteristic*
- **frame−relay interface−dlci** *dlci* [**broadcast**] [**ietf** | **cisco**]
- **frame−relay intf−type** [**dce** | **dte** | **nni**]
- **frame−relay inverse−arp** [*protocol*] [*dlci*]
- **frame−relay lmi−type {ansi** | **cisco** | **q933a}**
- **frame−relay map** *protocol protocol−address dlci* [**broadcast**] [**ietf** | **cisco** | **payload−compress packet by packet**]
- **frame−relay mincir {in | out}** ***bps***
- **frame−relay route** *in−dlci out−interface out−dlci*

- **frame−relay switching**
- **frame−relay traffic−shaping**
- **map−group** *group−name*
- **show frame−relay lmi** [*type number*]
- **show frame−relay map**
- **show frame−relay pvc** [*type number* [*dlci*]]
- **show frame−relay route**
- **show interfaces serial** *number*

Definitions

clear frame−relay−inarp: This exec command causes any frame−relay maps that were dynamically created through inverse ARP to be cleared.

debug frame lmi: This debug command will cause the router to display all LMI transactions between itself and the attached Frame Relay switch. This is a debug command. Remember to also enter the **term mon** command if you are not connected to the console port.

debug frame packet: This debug command causes the router to display Frame Relay packet traces for all Frame Relay traffic going through the router. This is a debug command. Remember to also enter the **term mon** command if you are not connected to the console port.

encapsulation frame−relay: This interface command enables Frame Relay encapsulation on an interface. The router supports two types of encapsulation, RFC 1490 and Cisco proprietary.

frame−relay bc: This map−class command specifies the outgoing or incoming committed burst size in bits for a Frame Relay PVC. If neither incoming or outgoing is specified, then the command applies to both directions.

frame−relay be: This map−class command specifies the outgoing or incoming excess burst size in bits for a Frame Relay PVC. If neither incoming or outgoing is specified, then the command applies to both directions.

frame−relay becn−response−enable: This map−class command causes the router to regulate the frame transmission rate on PVCs associated with a map class. The router regulates the traffic when it sees incoming frames with the BECN bit set.

frame−relay cir: This map−class command specifies the outgoing or incoming committed information rate (CIR) in bits per second for a Frame Relay PVC. If neither incoming or outgoing is specified, then the command applies to both directions.

frame−relay class: This interface command is used to associate a map class with an interface of a subinterface.

frame−relay de−group: This interface command is used to specify the DE (discard eligibility) group number that should be used for a particular DLCI.

frame−relay de−list: This global command defines a discard eligibility list that will specify what outgoing frames will have their DE bit set. Frames with their DE bit set can be used by a Frame Relay network to decide what traffic to discard during periods of congestion.

frame−relay interface−dlci: This interface command assigns a DLCI to a specific Frame Relay interface or subinterface on the router.

frame−relay intf−type: This interface command is used when the router will be acting as a Frame Relay switch. It is used to define the Frame Relay switch type.

frame−relay inverse−arp: This interface command enables inverse ARP on the router. Inverse ARP is enabled by default.

frame−relay lmi−type: This interface command is used to set the type of local management interface (LMI) signaling to be used on the router.

frame−relay map: This interface command is used to define a static map entry between a destination protocol address and the DLCI that is used to reach the destination protocol address.

frame−relay mincir: This map−class command specifies the outgoing or incoming minimum acceptable committed information rate (CIR) in bits per second for a Frame Relay PVC. If neither incoming or outgoing is specified, then the command applies to both directions.

frame−relay route: This command is used to configure a Frame Relay PVC between two interfaces on a router that is configured to act like a Frame Relay switch. This is an interface command.

frame−relay switching: This global command enables a router to perform Frame Relay switching functions.

frame−relay traffic−shaping: This interface command enables Frame Relay traffic shaping for all PVCs on a Frame Relay interface.

map−class frame−relay: This command creates a Frame Relay map class that is used to define Quality of Service (QOS) parameters for a Frame Relay PVC.

show frame−relay lmi: This exec command displays detailed statistics related to the local management interface (LMI) between the router and the Frame Relay switch.

show frame−relay map: This exec command displays all of the statically mapped and dynamically learned DLCI to protocol mappings on the router.

show frame−relay pvc: This exec command displays statistics about the Frame Relay PVCs that are defined to the router. When this command is entered with the optional DLCI number, any defined traffic shaping parameters will also be displayed.

show frame−relay route: This exec command is used on routers that are configured for Frame Relay switching. It will display all configured PVCs on the router and will show their status.

show interfaces serial: This exec command will display a varitey of information about the serial interfaces of the router. Several Frame Relay statistics such as LMI status and DTE/DCE configuration will be displayed.

IOS Requirements

All of the labs in this section used IOS version 11.2, except the Frame Relay traffic shaping lab which used IOS version 11.3. Cisco first supported Frame Relay with IOS 9.0.

Lab #12: Configuring a Cisco Routeras a Frame Relay Switch

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, one of which must have two serial ports. The other two routers must have one serial port
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection on the routers
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

Note The DCE side of the V.35 crossover cables must connect to the router FrameSwitch.

Configuration Overview

This configuration will demonstrate how to configure a Cisco router as a Frame Relay DTE and as a Frame Relay switch.

The three routers are connected as shown in Figure 4−10. RouterA and RouterB are configured as Frame Relay DTE devices. These two routers are connected to a router FrameSwitch that is acting as a Frame Relay switch. The router FrameSwitch is also configured to supply clock to both RouterA and RouterB. This is accomplished via the use of a Cisco DCE cable and the **clock rate** statement in the router configuration.

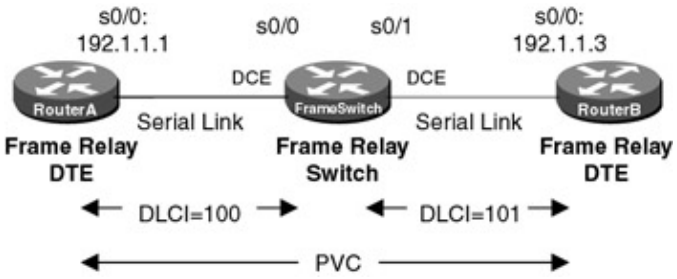


Figure 4−10: Basic Frame Relay switching

A PC running a terminal emulation program should be connected to the console port of one of the three routers using a Cisco rolled cable.

Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Router Configuration

The configurations for the three routers in this example are as follows. Frame Relay commands are highlighted in bold.

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
service timestamps debug datetime localtime
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
enable password cisco
!
interface Serial0/0
```

```
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay ← Configure interface for Frame Relay Encapsulation
 frame-relay lmi-type ansi ← Set LMI type to ANSI Annex D
!
router rip
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay ← Set the interface to Frame Relay encapsulation
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

The configurations for Routers A and B demonstrate a basic Frame Relay configuration on a Cisco router. You will need to set the interface for Frame Relay encapsulation via the **encapsulation frame-relay** command. You will also need to specify the type of LMI (local management interface) traffic between the router and the Frame Relay switch via the **frame-relay lmi-type** command.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
!
frame-relay switching ← Enable Frame Relay switching on the router
!
interface Serial0/0
 no ip address
 encapsulation frame-relay
```

```
 clockrate 38400 ← Generate a 38,400 bps clock to the DTE
 frame-relay lmi-type ansi ← Set LMI to Annex D
 frame-relay intf-type dce ← Set the interface type to a Frame Relay DCE
 frame-relay route 100 interface Serial0/1 101 ← Define PVC between ports S0/0 and S0/1
!
interface Serial0/1
 no ip address
 encapsulation frame-relay
 clockrate 19200
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 101 interface Serial0/0 100
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Notice that in this example the router FrameSwitch is acting as a Frame Relay switch. The ability for a Cisco router to act as a Frame Relay switch is an important aspect of Cisco's support for Frame Relay. This feature is particularly useful in testing situations such as these hands−on labs.

Frame Relay switching is enabled on the router by the global command **frame−relay switching**. In addition to the standard configuration commands defining Frame Relay encapsulation and LMI signaling, each interface on the router that will support switching needs to be defined as a Frame Relay DCE via the **frame−relay intf−type dce** command.

Frame Relay PVCs (Permanent Virtual Circuits) are defined with the **frame−relay route** command. As an example, let's look at interface S0/0 of the router FrameSwitch. We see the command: **frame−relay route 100 interface Serial0/1 101**. As shown in Figure 4−11, this command tells the router that any Frame Relay traffic coming into interface S0/0 with DLCI 100 should be sent out interface S0/1 with DLCI 101. Similarly, interface S0/1 has the command **frame−relay route 101 interface Serial0/0 100**. This command tells the router that any Frame Relay traffic coming into interface S0/1 with DLCI 101 should be sent out interface S0/0 with DLCI 100. These two **frame−relay route** commands create a Frame Relay PVC between ports S0/0 and S0/1 of the router FrameSwitch. The local DLCI on port S0/0 is DLCI 100 and the local DLCI on port S0/1 is DLCI 101.

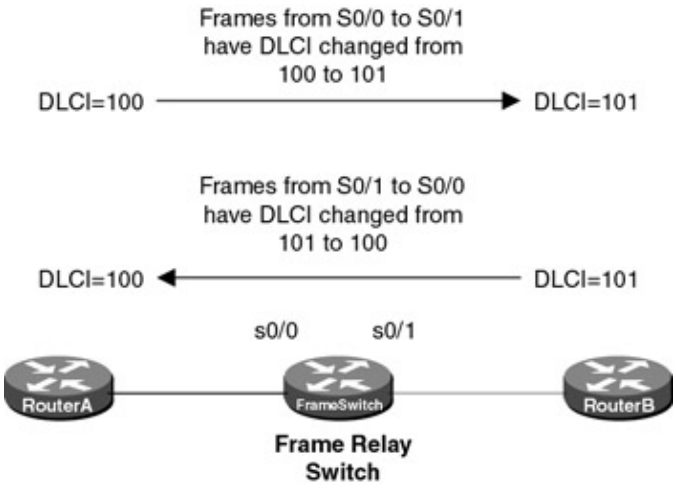


Figure 4−11: Frame Relay label switching example

Monitoring and Testing the Configuration

We can start to verify our configuration by connecting to the router FrameSwitch. The **show frame pvc** command is used to display all PVCs that are passing through the router. Several items in the output of the **show frame pvc** command indicate that the router FrameSwitch is acting as a Frame Relay switch. These items are shown in bold in the following screen print. The PVC statistics indicate that S0/0 and S0/1 are acting as Frame Relay DCE interfaces. DLCI usage is indicated as Switched. Finally, the field Num Pkts Switched shows how many frames have been switched through the interface.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DCE)

DLCI = 100, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 7             output pkts 1            in bytes 340
  out bytes 30             dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:07:57, last time pvc status changed 00:02:13
  Num Pkts Switched 7


PVC Statistics for interface Serial0/1 (Frame Relay DCE)

DLCI = 101, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 1             output pkts 7            in bytes 30
  out bytes 340            dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:07:06, last time pvc status changed 00:04:49
  Num Pkts Switched 1
```

Notice that the output of the **show frame pvc** command displays how many frames were sent/received with the FECN bit, BECN bit, or DE bit set. These performance parameters are key to determining if your Frame Relay network is experiencing congestion.

The **show frame pvc** command also displays how many frames have been sent into and out of an interface as well as how long the PVC has existed and when its status was last changed.

The **show frame lmi** command displays the status of the LMI interface between the router FrameSwitch and both directly connected routers, RouterA and RouterB. Again, we see that the router FrameSwitch is acting as a Frame Relay switch. The indication is the Frame Relay DCE message on the command output. Notice that the **show frame lmi** command displays information for all interfaces on the router that are set for Frame Relay encapsulation. In the case of the router FrameSwitch, these are interfaces S0/0 and S0/1.

```
FrameSwitch#sh frame lmi

LMI Statistics for interface Serial0/0  (Frame Relay DCE) LMI TYPE = ANSI
  Invalid Unnumbered info 0      Invalid Prot Disc 0
  Invalid dummy Call Ref 0       Invalid Msg Type 0
  Invalid Status Message 0       Invalid Lock Shift 0
  Invalid Information ID 0       Invalid Report IE Len 0
  Invalid Report Request 0       Invalid Keep IE Len 0
  Num Status Enq. Rcvd 32        Num Status msgs Sent 32
  Num Update Status Sent 0       Num St Enq. Timeouts 13


LMI Statistics for interface Serial0/1  (Frame Relay DCE) LMI TYPE = ANSI
  Invalid Unnumbered info 0      Invalid Prot Disc 0
  Invalid dummy Call Ref 0       Invalid Msg Type 0
  Invalid Status Message 0       Invalid Lock Shift 0
```

```
  Invalid Information ID 0       Invalid Report IE Len 0
  Invalid Report Request 0       Invalid Keep IE Len 0
  Num Status Enq. Rcvd 15        Num Status msgs Sent 15
  Num Update Status Sent 0       Num St Enq. Timeouts 20
```

The **show frame route** command is used when a router is configured as a Frame Relay switch. The command will display all DLCIs that are configured on the router. The sample output that follows from the router FrameSwitch shows that two DLCIs are configured on the router. The output of the command can be read as follows: Traffic coming into interface S0/0 with a DLCI value of 100 will get switched to interface S0/1 and will be given a DLCI value of 101. Traffic coming into interface S0/1 with a DLCI value of 101 will be switched to interface S0/0 and will be given a DLCI value of 100. The **show frame route** output also indicates that both of these DLCIs are currently in an active state.

```
FrameSwitch#sh frame route
Input Intf                  Input Dlci   Output Intf   Output Dlci   Status
Serial0/0                   100          Serial0/1     101           active
Serial0/1                   101          Serial0/0     100           active
```

The output of the s**how interface** command for the two Frame Relay encapsulated interfaces (S0/0 and S0/1) on the router FrameSwitch indicates several important items. We see that the interface is up and the line protocol is up, telling us that the Frame Relay protocol is running properly on this interface. Other important Frame Relay−related status information supplied in the output of the **show interface** command includes:

- Confirmation that the interface is set for Frame Relay encapsulation
- A summary of the LMI exchanges between the router and the Frame Switch
- Indications that the router interface is acting as a Frame Relay switch. These indications are highlighted in bold in the following screen prints
- Verification that the LMI signaling type is ANSI Annex D

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 34, LMI stat sent  34, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
  Broadcast queue 0/64, broadcasts sent/dropped 0/0, interface broadcasts 0
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/3  (active/max active)
     Reserved Conversations 0/0  (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     42 packets input, 872 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     49 packets output, 1423 bytes, 0 underruns
     0 output errors, 0 collisions, 39 interface resets
     0 output buffer failures, 0 output buffers swapped out
     42 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
FrameSwitch#sh int s 0/1
Serial0/1  is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 17, LMI stat sent  17, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
```

```
Broadcast queue 0/64, broadcasts sent/dropped 0/0, interface broadcasts 0
Last input 00:00:03, output 00:00:03, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0 (size/max/drops); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 0/64/0 (size/threshold/drops)
   Conversations  0/2  (active/max active)
   Reserved Conversations 0/0  (allocated/max allocated)
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
   18 packets input, 268 bytes, 0 no buffer
   Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
   0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
   37 packets output, 1553 bytes, 0 underruns
   0 output errors, 0 collisions, 14 interface resets
   0 output buffer failures, 0 output buffers swapped out
   36 carrier transitions
   DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Now connect to RouterA. RouterA is a Frame Relay DTE. The **show frame pvc** command indicates that DLCI 100 is active on interface S0/0. It also indicates that interface S0/0 is acting like a Frame Relay DTE. Notice that there is no indication of Num Pkts Switched, since this router is not emulating a Frame Switch.

```
RouterA#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 100, DLCI USAGE = LOCAL, PVC STATUS = Active, INTERFACE = Serial0/0

  input pkts 1             output pkts 11           in bytes 30
  out bytes 590            dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:07:14, last time pvc status changed 00:04:31
```

By default, a Cisco router will attempt to inverse ARP to resolve the DLCI(s) active on a given interface to the IP address on the other end of the PVC. The **show frame map** command shows the results of the inverse ARP. We will see in a future lab that static entries can also be made to the Frame Relay map table. The output for the **show frame relay map** command for RouterA is shown next. The output indicates that DLCI 100 is active on interface S0/0. It shows that the IP address of the interface on the far end of the PVC is 192.1.1.3. This is the IP address for the S0/0 interface of RouterB. The command output also indicates that this DLCI−to−IP address resolution occurred dynamically, as opposed to a static Frame Relay map, which will be discussed in a later lab.

```
RouterA#sh frame map
Serial0/0 (up): ip 192.1.1.3 dlci 100(0x64,0x1840), dynamic,
              broadcast,, status defined, active
```

Now verify that the LMI between RouterA and the Frame Relay switch is active with the **show frame lmi** command.

```
RouterA#sh frame lmi

LMI Statistics for interface Serial0/0 (Frame Relay DTE) LMI TYPE = ANSI
  Invalid Unnumbered info 0      Invalid Prot Disc 0
  Invalid dummy Call Ref 0       Invalid Msg Type 0
  Invalid Status Message 0       Invalid Lock Shift 0
  Invalid Information ID 0       Invalid Report IE Len 0
  Invalid Report Request 0       Invalid Keep IE Len 0
  Num Status Enq. Sent 173       Num Status msgs Rcvd 44
  Num Update Status Rcvd 0       Num Status Timeouts 129
```

Verify that the serial interface of RouterA is in an up state with the **show interface s0/0** command. The interface should be in an up/up state. This indicates that the interface is receiving carrier detect from the Frame Switch and that the Frame Relay LMI is active between RouterA and the switch. Notice that the LMI is DTE LMI, which means RouterA is a Frame Relay DTE.

```
RouterA#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 192.1.1.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  168, LMI stat recvd 39, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 8/0, interface broadcasts 8
  Last input 00:00:03, output 00:00:03, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/1  (active/max active)
     Reserved Conversations 0/0  (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     306 packets input, 11113 bytes, 0 no buffer
     Received 3 broadcasts, 0 runts, 0 giants, 0 throttles
     4 input errors, 0 CRC, 4 frame, 0 overrun, 0 ignored, 0 abort
     293 packets output, 9685 bytes, 0 underruns
     0 output errors, 0 collisions, 55 interface resets
     0 output buffer failures, 0 output buffers swapped out
     80 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Ping RouterB at 192.1.1.3 to verify end-to-end connectivity.

```
RouterA#ping 192.1.1.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 134/136/141 ms
```

Now let's connect to RouterB and verify that it has a Frame Relay map entry to RouterA. Notice that the map entry is dynamic, indicating that the entry was learned via Inverse Arp.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 100(0x64,0x1840), dynamic,
              broadcast,, status defined, active
```

Use the **show frame-relay pvc** command to verify that DLCI 100 is active on the S0/0 interface of RouterB.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 100, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 13            output pkts 1            in bytes 702
  out bytes 30             dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:05:21, last time pvc status changed 00:05:21
```

Verify that you can reach RouterA at 192.1.1.1 with a ping command.

```
RouterB#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 136/138/140 ms
```

Lab #13: Configuring LMI Autosense

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each of which must have at least one serial port
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection on the routers
- One Cisco DTE/DCE crossover cable. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

Configuration Overview

This configuration will demonstrate how a Cisco router running IOS 11.2 or greater can autoconfigure its LMI signaling protocol.

The two routers are connected as shown in Figure 4−12. RouterB is configured as a Frame Relay DTE device. The router FrameSwitch is configured as a Frame Relay switch. The router FrameSwitch is configured for Cisco LMI and RouterB is configured for LMI autosense. The router configured as a Frame Relay switch is also configured to supply clock to the DTE router. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router configuration.

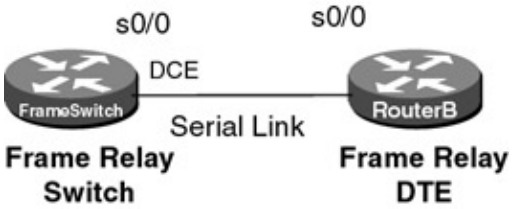


Figure 4−12: LMI autosense

Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Note Frame Relay autoconfigure is supported in IOS version 11.2 and higher.

Note The DCE side of the V.35 cable must connect to the router FrameSwitch.

Router Configuration

Start the lab by configuring both the routers for Frame Relay encapsulation. The router FrameSwitch should be configured as a Frame Relay switch, and RouterB should be configured as a Frame Relay DTE. Configure the FrameSwitch router for Cisco LMI by typing the command **frame−relay lmi−type cisco** under interface s0/0 and configure RouterB without a **frame−relay lmi−type** statement. Not including an LMI statement will cause the router to go into autoconfigure mode.

Frameswitch

```
Current configuration:
!
version 11.2
```

```
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
!
frame-relay switching ← Enable Frame Relay switching on this router
!
interface Serial0/0
 no ip address
 encapsulation frame-relay
 clockrate 64000 ← Generate a 64,000 bps clock to the DTE
 frame-relay intf-type dce ← Define this interface as a Frame Relay DCE
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Note that the **frame-relay lmi-type cisco** statement does not appear in the configuration for the router FrameSwitch. Since this is the default LMI type, the command does not show on the screen.

Router B

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Serial0/0 ← Do not set an LMI type. The router can autoconfig its LMI
                      setting
 ip address 195.1.1.33 255.255.255.0
 encapsulation frame-relay
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

The **show interface s0/0** command output issued on the router FrameSwitch is shown next. It verifies that the LMI type is Cisco, as shown in bold.

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  1, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 273, LMI stat sent  265, LMI upd sent  0, DCE LMI up
  LMI DLCI 1023  LMI type is CISCO  frame relay DCE
  Broadcast queue 0/64, broadcasts sent/dropped 0/0, interface broadcasts 0
  Last input 00:00:03, output 00:00:03, output hang never
```

```
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/1  (active/max active)
     Reserved Conversations 0/0  (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     273 packets input, 3602 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     5 input errors, 0 CRC, 4 frame, 0 overrun, 0 ignored, 1 abort
     265 packets output, 3495 bytes, 0 underruns
     0 output errors, 0 collisions, 99 interface resets
     0 output buffer failures, 0 output buffers swapped out
     51 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Now connect to RouterB. We will change the encapsulation from Frame Relay to PPP so that we can restart the LMI autoconfiguration process and capture it with a debug trace. Once in enabled mode, enter the command **config term**.

```
RouterB#config term
Enter configuration commands, one per line. End with CNTL/Z.
```

Under interface s0/0, you should shut down the interface and change its encapsulation to ppp:

```
RouterB(config)#int s 0/0
RouterB(config-if)#shut
RouterB(config-if)#encap ppp
RouterB(config)#exit
```

Turn on Frame Relay LMI debugging via the **debug frame-relay lmi** command. Remember to also type the **term mon** command if you are connected to the router by any other method than via the console port.

```
RouterB#debug frame-relay lmi
Frame Relay LMI debugging is on
Displaying all Frame Relay LMI data
```

Now enter configuration mode in order to activate interface s0/0 and change its encapsulation to Frame Relay.

```
RouterB#config term
Enter configuration commands, one per line. End with CNTL/Z.
```

Under interface s0/0, change the router's encapsulation to Frame Relay and take the interface out of shutdown mode. Exit out of configuration mode.

```
RouterB(config)#int s 0/0
RouterB(config-if)#encap frame-relay
RouterB(config-if)#no shut
RouterB(config-if)#exit
RouterB(config)#exit
```

A few seconds later, you will see four debug messages display on the screen. The first three messages are a result of RouterB sending out three LMI requests using Annex D, AnnexA, and Cisco LMI signaling. Notice that the first three debug outputs that follow are indicated as being in the direction out. This means that they are originating on RouterB and being sent toward the router FrameSwitch.

```
Serial0/0(out): StEnq, myseq 1, yourseen 0, DTE up ← Annex D request from
                                                      router to Frame switch
datagramstart = 0xD31EF4, datagramsize = 14
FR encap = 0x00010308
00 75 95 01 01 00 03 02 01 00
```

```
Serial0/0 (out): StEnq, myseq 1, yourseen 0, DTE up ← Annex A request from
                                                      router to Frame switch
datagramstart = 0xD31DB4, datagramsize = 13
FR encap = 0x00010308
00 75 51 01 00 53 02 01 00

Serial0/0 (out): StEnq, myseq 1, yourseen 0, DTE up ← Cisco LMI request from
                                                      router to Frame switch
datagramstart = 0xD31EF4, datagramsize = 13
FR encap = 0xFCF10309
00 75 01 01 00 03 02 01 00
```

After all three LMI requests are sent to the router FrameSwitch, a response, shown next, is sent back. Recall that the router FrameSwitch is set for LMI type Cisco, so it answers the third and last request from RouterB. The direction is indicated as in, meaning that this packet came from the router FrameSwitch into RouterB.

```
Serial0/0(in): Status, myseq 1 ← Frame switch reply to Cisco LMI status request
RT IE 1, length 1, type 0
KA IE 3, length 2, yourseq 1, myseq 1
```

Demonstrating the Configuration

Figure 4−13 shows the previous test lab with a TTC Fireberd 500 Internetwork Analyzer connected between routers FrameSwitch and RouterB. The TTC 500 was configured to trace all Frame Relay transactions between the two routers. The following screen print shows RouterB sending out three status enquiries spaced approximately 2 ms apart. The first status is in the Annex D format, the second is in the Annex A format, and the third is in the Cisco LMI format. Since the router FrameSwitch is set for Cisco LMI format, the router does not respond until it receives the last LMI request, which is also in the Cisco LMI format.

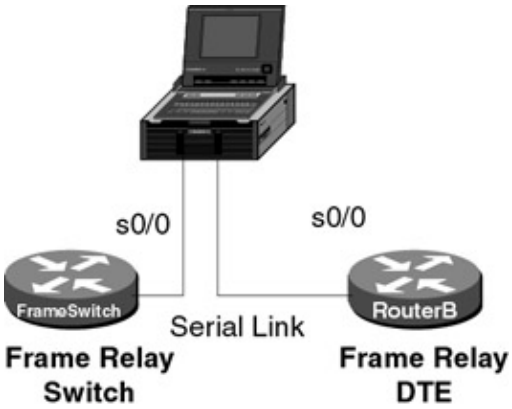


Figure 4−13: Monitoring LMI autosense

Annex D Request from RouterB to FrameSwitch

```
Port A DTE ID=24, 02/05/99, 16:39:54.153700

Length=16, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                          00h
DLCI_lsb                                                           0h
DLCI                                                  Annex D or A 0
CR                                                                 0
EA                                                                 0
FECN                                                               0
BECN                                                               0
DE                                                                 0
EA                                                                 1
Q.933 FRAME RELAY SVC
Control                                                    UI Frame 03h
Protocol Discriminator                                  Call Control 08h
Reference Flag                                 Msg from Origination 0
Call Reference Value [len=0]                                       0
```

```
Message Type                                                STATUS ENQ 75h
INFO ELEMENT Locking Shift [Code=18]
INFO ELEMENT Locking Shift                                  (Codeset 5) 5h
IE − Annex D Report Type [Code=1] [Len=1]
IE − Annex D Report Type                                 Full Status 00h
IE − Annex D Link Integrity Ver. [Code=3] [Len=2]
Current Seq                                                          01h
Last Rcvd Seq                                                        00h
FCS                                                          Good FD1Eh
```

Annex A Request from RouterB to FrameSwitch

```
Port A DTE ID=25, 02/05/99, 16:39:54.155825
Length=15, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                             00h
DLCI_lsb                                                              0h
DLCI                                                     Annex D or A 0
CR                                                                    0
EA                                                                    0
FECN                                                                  0
BECN                                                                  0
DE                                                                    0
EA                                                                    1
Q.933 FRAME RELAY SVC
Control                                                     UI Frame 03h
Protocol Discriminator                                   Call Control 08h
Reference Flag                                        Msg from Origination 0
Call Reference Value [len=0]                                          0
Message Type                                                STATUS ENQ 75h
INFO ELEMENT Report Type [Code=81] [Len=1]
INFO ELEMENT Report Type                                 Full Status 00h
INFO ELEMENT Link Integrity Ver. [Code=83] [Len=2]
Current Seq                                                          01h
Last Rcvd Seq                                                        00h
FCS                                                          Good B677h
```

Cisco LMI Request from RouterB to FrameSwitch

```
Port A DTE ID=26, 02/05/99, 16:39:54.158000
Length=15, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                             3Fh
DLCI_lsb                                                              Fh
DLCI                                                  LMI or CLLM 1023
CR                                                                    0
EA                                                                    0
FECN                                                                  0
BECN                                                                  0
DE                                                                    0
EA                                                                    1
LMI − LOCAL MANAGEMENT INTERFACE
Control                                                     UI Frame 03h
Protocol Discriminator                                           LMI 09h
Call Reference                                                       00h
Message Type                                           Status Enquiry 75h
Report Type IE [Code=1] [Len=1]
Report Type IE                                               Full Status 0
Keep Alive Sequence IE [Code=3] [Len=2]
Current Seq                                                          01h
Last Rcvd Seq                                                        00h
FCS                                                          Good DF4Ah
```

FrameSwitch Response to RouterB Cisco LMI Status Request

```
Port A DCE ID=33, 02/05/99, 16:39:54.164325
Length=15, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                        3Fh
DLCI_lsb                                                         Fh
DLCI                                                LMI or CLLM 1023
CR                                                              0
EA                                                              0
FECN                                                            0
BECN                                                            0
DE                                                              0
EA                                                              1
LMI – LOCAL MANAGEMENT INTERFACE
Control                                                UI Frame 03h
Protocol Discriminator                                      LMI 09h
Call Reference                                                00h
Message Type                                            Status 7Dh
Report Type IE [Code=1] [Len=1]
Report Type IE                                        Full Status 0
Keep Alive Sequence IE [Code=3] [Len=2]
Current Seq                                                   01h
Last Rcvd Seq                                                 01h
FCS                                                     Good EA76h
```

Lab #14: Configuring Cisco Discard Eligibility Support

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, two of which must have at least one serial port, and one of which must have two serial ports
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program used for console port connection to the routers
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

Configuration Overview

This configuration will demonstrate Cisco IOS support for setting the Discard Eligible bit on outgoing traffic. RouterB will be configured to set the DE bit on outgoing frames that are greater than 512 bytes in length.

The three routers are connected as shown in Figure 4−14. Two of the routers are configured as Frame Relay DTE devices. The third router is configured as a Frame Relay switch. The router configured as a Frame Relay switch is also configured to supply clock to the DTE routers. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router configuration.

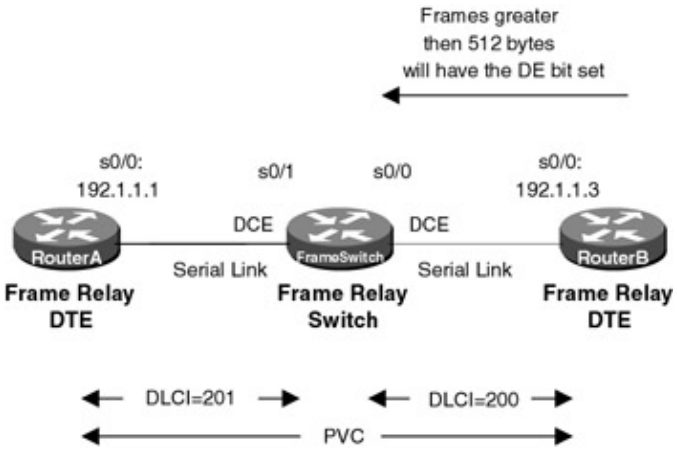


Figure 4−14: Cisco Discard Eligibility support

Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Note The DCE side of the V.35 cable must be on the router FrameSwitch.

Router Configuration

The configurations for the three routers in this example are as follows. Key Frame Relay commands are in bold.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname FrameSwitch
!
!
frame−relay switching ← Configure Frame Relay switching on this router
!
interface Serial0/0
 no ip address
 encapsulation frame−relay
 clockrate 64000 ← Generate a 64,000 bps clock to the DTE
 frame−relay lmi−type ansi ← Set LMI to Annex D
 frame−relay intf−type dce ← Set the interface type to a Frame Relay DCE
 frame−relay route 200 interface Serial0/1 201 ← Define a PVC between S0/0 and S0/1

!
interface Serial0/1
 no ip address
 encapsulation frame−relay
 clockrate 64000
 frame−relay lmi−type ansi
 frame−relay intf−type dce
 frame−relay route 201 interface Serial0/0  200
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay ← Set the interface encapsulation to Frame Relay
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
frame-relay de-list 3 protocol ip gt 512 ← Create a Frame Relay DE list. Any IP
                                           traffic greater than 512 bytes will
                                           be marked
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay
 frame-relay de-group 3 200 ← Apply the DE list to this interface for DLCI 200
 frame-relay lmi-type ansi
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Notice that the configuration for RouterB has two statements that will cause the DE bit to be set on outgoing traffic: **frame-relay de-list 3 protocol ip gt 512** and **frame-relay de-group 3 200**.

Monitoring and Testing the Configuration

Let's begin by connecting to the router FrameSwitch and verifying that it is working properly. Issue the **show frame pvc** command to display all DLCIs that are passing through the router. We see that DLCI 200 is active on port S0/0 and DLCI 201 is active on port S0/1. Several items indicate that ports S0/0 and S0/1 are acting as Frame Relay switch ports. These include the DLCI usage being referenced as switched, the interface being

referred to as a Frame Relay DCE, and the indication of Num Pkts Switched. The PVC status should indicate Active.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 845           output pkts 850          in bytes 143231
  out bytes 143751         dropped pkts 1           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 50            out DE pkts 0
  pvc create time 1d00h, last time pvc status changed 1d00h
  Num Pkts Switched 845
 PVC Statistics for interface Serial0/1  (Frame Relay DCE)

DLCI = 201, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 851           output pkts 845          in bytes 143781
  out bytes 143231         dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 1d00h, last time pvc status changed 1d00h
  Num Pkts Switched 850
```

Next issue the **show frame route** command. This command will display all active PVCs that are defined on the router. We see that two DLCIs are configured on this router, 200 and 201. The Frame Relay route table can be interpreted as follows: Any traffic coming into interface S0/0 with a DLCI of 200 will be sent out interface S0/1 with a DLCI of 201. Any traffic coming into interface S0/1 with a DLCI of 201 will be sent out interface S0/0 with a DLCI of 200. The status of both DLCIs should be active.

```
FrameSwitch#sh frame route
Input Intf       Input Dlci   Output Intf   Output Dlci   Status
Serial0/0        200          Serial0/1     201           active
Serial0/1        201          Serial0/0     200           active
```

The **show interface s0/0** and **show interface s0/1** commands will display the statuses of the serial interfaces on the router. Several important Frame Relay parameters are displayed by this command and are highlighted in bold including the interface encapsulation (Frame−Relay), the LMI status (LMI up), the fact that this port is acting as a Frame Relay DCE, the LMI signaling type (ANSI Annex D), and the LMI exchange counters.

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME−RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  1, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 34644, LMI stat sent  33688, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.

FrameSwitch#sh int s 0/1
Serial0/1  is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME−RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 8735, LMI stat sent  8735, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.
```

Now let's connect to RouterA. Display the results of the router's inverse ARP with the **show frame map** command. Notice how the router has resolved its local DLCI of 201 to an address of 192.1.1.3 at the far end of the PVC. Recall that 192.1.1.3 is the IP address of the S0/0 interface of RouterB.

```
RouterA#sh fra map
Serial0/0 (up): ip 192.1.1.3 dlci 201(0xC9,0x3090), dynamic,
              broadcast,, status defined, active
```

Verify end−to−end connectivity between RouterA and RouterB by pinging RouterB at 192.1.1.3:

```
RouterA#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/56/60 ms
```

Now let's connect to RouterB. The **show frame map** command should display an entry for RouterA's S0/0 interface at IP address 192.1.1.1.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
```

Clear the interface counters with the **clear counters** command.

```
RouterB#clear counters
Clear "show interface" counters on all interfaces [confirm]
%CLEAR−5−COUNTERS: Clear counter on all interfaces by console
```

Verify that all counters are cleared by typing the **show frame pvc** command. Notice how all packet counters have been reset to 0.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:39:11, last time pvc status changed 00:39:11
```

Do another ping to RouterA at IP address 192.1.1.1. Remember that a standard ping will issue five packets.

```
RouterB#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/56/60 ms
```

Now type the **show frame pvc** command again. Notice how the input and output packets are now 5. This value represents the five ping packets that were just sent.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0
```

```
  input pkts 5               output pkts 5               in bytes 520
  out bytes 520              dropped pkts 0              in FECN pkts 0
  in BECN pkts 0             out FECN pkts 0             out BECN pkts 0
  in DE pkts 0               out DE pkts 0
  pvc create time 00:39:19,  last time pvc status changed 00:39:19
```

Now let's issue an extended ping with 10 ping packtets sent, each ping packet being 500 bytes.

```
RouterB#ping
Protocol [ip]:
Target IP address: 192.1.1.1
Repeat count [5]: 10
Datagram size [100]: 500
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 10, 500-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!!!!!!
Success rate is 100 percent (10/10), round-trip min/avg/max = 256/257/260 ms
```

The **show frame pvc** command will now show 15 input and output packets. This is 10 more that the last time we examined the values. Remember that our extended ping was 10 packets. Notice that the command output shows no input or output DE packets.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 15              output pkts 15             in bytes 5560
  out bytes 5560             dropped pkts 0             in FECN pkts 0
  in BECN pkts 0             out FECN pkts 0            out BECN pkts 0
  in DE pkts 0               out DE pkts 0
  pvc create time 00:39:39, last time pvc status changed 00:39:39
```

Now let's do another extended ping. Ping the far-end address of RouterA at 192.1.1.1. This time we will use a datagram size of 512 bytes.

```
RouterB#ping
Protocol [ip]:
Target IP address: 192.1.1.1
Repeat count [5]: 10
Datagram size [100]: 512
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 10, 512-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!!!!!!
Success rate is 100 percent (10/10), round-trip min/avg/max = 260/263/264 ms
```

Now type the **show frame pvc** command. Notice that our input and output packets have increased by 10, from 15 to 25. Also notice that we now have 10 output DE packets. This occurred because we exceeded our 512-byte limit for non-DE frames going out of the router.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 25              output pkts 25             in bytes 10720
```

```
  out bytes 10720          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 10
  pvc create time 00:39:58, last time pvc status changed 00:39:58
```

The following analyzer trace shows what a frame looks like that has its DE bit set.

```
Port A DTE ID=54, 01/25/99, 16:19:25.718925

Length=106, Good FCS
FRAME RELAY PROTOCOL
DLCI_msb                                                        06h
DLCI_lsb                                                         4h
DLCI                                                            100
CR                                                                0
EA                                                                0
FECN                                                              0
BECN                                                              0
DE                                                                1
EA                                                                1
NLPID − NETWORK LEVEL PROTOCOL ID
Ethertype                                                DOD IP 0800h
IP − INTERNET PROTOCOL
Version                                                           4
IHL (in 32 bit words)                                             5
Precedence                                              Routine 0h
D(elay)                                                   Normal 0
T(hroughput)                                              Normal 0
R(eliability)                                             Normal 0
Total Length (in octets)                                        100
Identification                                                0212h
D(on't) F(ragment)                                             No 0
M(ore) F(ragments)                                             No 0
Fragment Offset (in 8 octets)                                     0
Time To Live (in seconds)                                       255
Protocol                                                ICMP 01h
Header Checksum                                               316Bh
Class C Source IP Address
Source Net ID                                                 196865
Source Host ID                                                   12
Source Addr                                             195.1.1.12
Class C Destination IP Address
Destination Net ID                                            196865
Destination Host ID                                              13
Destination Addr                                        195.1.1.13
ICMP − INTERNET CONTROL MESSAGE PROTOCOL
Type                                                      Echo 08h
Code                                                           00h
Checksum                                                      AA4Dh
ID                                                            0004h
Sequence No                                                   23FFh
Dump
 00000                  0000  0000  6BF1  4408  ABCD        ....k.D...
 00010                  ABCD  ABCD  ABCD  ABCD  ABCD        ..........
 00020                  ABCD  ABCD  ABCD  ABCD  ABCD        ..........
 00030                  ABCD  ABCD  ABCD  ABCD  ABCD        ..........
 00040                  ABCD  ABCD  ABCD  ABCD  ABCD        ..........
 00050                  ABCD  ABCD  ABCD  ABCD  ABCD        ..........
 00060                  ABCD  ABCD  ABCD  ABCD  ABCD        ..........
 00070                  ABCD                                      ..
FCS                                                      Good 2834h
```

Lab #15: Frame Relay Map Statements

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, two of which must have at least one serial port, and one of which must have two serial ports
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection to the routers
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

Configuration Overview

This configuration will demonstrate the use of the Frame Relay map statement. A Frame Relay map is used when connecting to a device that does not respond to an inverse ARP request. Since the device does not respond to inverse ARP, the router cannot automatically resolve the local DLCI to the far−end IP address. Configuring a Frame Relay map statement causes the router to install a static mapping to the far−end device. This static mapping contains the local DLCI and the far−end IP address. Frame Relay maps can be used for many other protocols, such as IPX.

The three routers are connected as shown in Figure 4−15. Two of the routers are configured as Frame Relay DTE devices. The third router is configured as a Frame Relay switch. The router configured as a Frame Relay switch is also configured to supply clock to the DTE routers. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router's configuration.

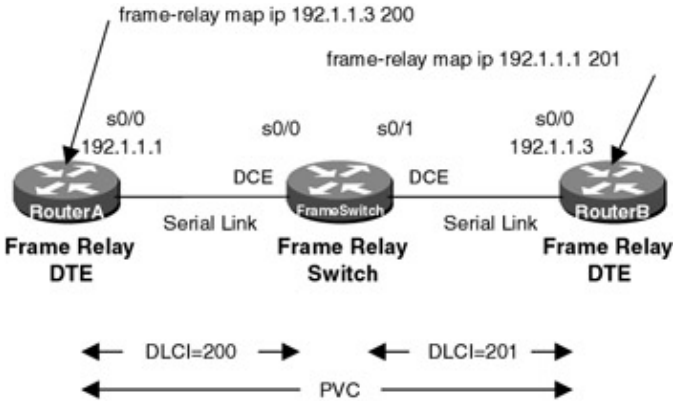


Figure 4−15: Frame Relay map statements

Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Note The DCE side of the V.35 crossover cables must be connected to the router FrameSwitch.

Router Configuration

The initial configurations for the three routers in this example are as follows. Key Frame Relay commands are in bold.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
```

```
hostname FrameSwitch
!
!
frame-relay switching ← Configure Frame Relay Switching on this router
!
interface Serial0/0
 no ip address
 encapsulation frame-relay ← Set the interface encapsulation to Frame Relay
 clockrate 64000
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
 frame-relay intf-type dce ← Set the interface type to a DCE
 frame-relay route 200 interface Serial0/1  201 ← Define a PVC between S0/0 and 0/1
!
interface Serial0/1
 no ip address
 encapsulation frame-relay
 clockrate 64000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 201 interface Serial0/0  200
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
service timestamps debug datetime localtime
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay
 no frame-relay inverse-arp ← Disable Frame Relay inverse ARP support on this
                              interface
 frame-relay lmi-type ansi
!
router rip
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay
 no frame-relay inverse-arp ← Disable Frame Relay inverse ARP support on this interface
 frame-relay lmi-type ansi
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Notice that RouterA and RouterB both have the statement **no frame-relay inverse-arp** under their serial 0/0 interfaces. This command will stop the router from sending out an inverse ARP request on its local DLCIs. Some networking devices do not respond to inverse ARP requests, so another way must be used to tell the router what far-end IP address corresponds to each local DLCI.

Monitoring and Testing the Configuration

Let's begin by connecting to the router FrameSwitch and verifying that it is working properly. Issue the **show frame pvc** command to display all DLCIs that are passing through the router.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:03:24, last time pvc status changed 00:02:40
  Num Pkts Switched 0

PVC Statistics for interface Serial0/1  (Frame Relay DCE)

DLCI = 201, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:02:45, last time pvc status changed 00:02:41
  Num Pkts Switched 0
```

We see that DLCI 200 is active on port S0/0 and DLCI 201 is active on port S0/1. Several items indicate that ports S0/0 and S0/1 are acting as Frame Relay switch ports. These include the DLCI usage being referenced

as switched, the interface being referred to as a Frame Relay DCE, and the indication of Num Pkts Switched. The PVC status should indicate Active.

Next issue the **show frame route** command. This command will display all active PVCs that are defined on the router.

```
FrameSwitch#sh frame route
Input Intf     Input Dlci    Output Intf     Output Dlci    Status
Serial0/0      200           Serial0/1       201            active
Serial0/1      201           Serial0/0       200            active
```

We see that two DLCIs are configured on this router, 200 and 201. The Frame Relay route table can be interpreted as follows: Any traffic coming into interface S0/0 with a DLCI of 200 will be sent out interface S0/1 with a DLCI of 201. Any traffic coming into interface S0/1 with a DLCI of 201 will be sent out interface S0/0 with a DLCI of 200. The status of both DLCIs should be active.

The **show interface s0/0** and **show interface s0/1** commands will display the statuses of the serial interfaces on the router. Several important Frame Relay parameters are displayed by this command and are highlighted in bold including the interface encapsulation (Frame−Relay), the LMI status (LMI up), the fact that this port is acting as a Frame Relay DCE, the LMI signaling type (ANSI Annex D), and the LMI exchange counters.

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME−RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 297, LMI stat sent  297, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.

FrameSwitch#sh int s 0/1
Serial0/1  is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME−RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 301, LMI stat sent  301, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.
```

Now let's connect to RouterA. Verify that RouterA uses DLCI 200 as its local DLCI:

```
RouterA#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:02:03, last time pvc status changed 00:01:23
  Num Pkts Switched 0
```

Display the results of the router's inverse ARP with the **show frame map** command.

```
RouterA#sh fra map
RouterA#
```

Notice how there is no output from the router. This is because we have disabled inverse ARP on this router. Even though the router learns about new DLCIs from the switch, it will still not inverse ARP on these DLCIs to learn the far−end IP address. In the case of RouterA, the router will not inverse ARP on DLCI 200.

Now turn on Frame Relay packet debugging on RouterA. Remember that debug messages only appear on the console. If you are telneted into the router or connected to the AUX port, you will also need to issue the **term mon** command.

```
RouterA#debug frame packet
Frame Relay packet debugging is on
```

Verify what debug items are enabled by typing the **show debug** command.

```
RouterA#sh debug
Frame Relay:
  Frame Relay packet debugging is on
```

Now try to ping RouterB at its address of 192.1.1.3. The ping to 192.1.1.3 fails.

```
RouterA#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:

*Mar 1 00:52:56: Serial0/0:Encaps failed−−no map entry link 7(IP)
*Mar 1 00:52:58: Serial0/0:Encaps failed−−no map entry link 7(IP).
*Mar 1 00:53:00: Serial0/0:Encaps failed−−no map entry link 7(IP).
*Mar 1 00:53:02: Serial0/0:Encaps failed−−no map entry link 7(IP).
*Mar 1 00:53:04: Serial0/0:Encaps failed−−no map entry link 7(IP).
Success rate is 0 percent (0/5)
```

Let's examine the output of the debug command. The ping command attempted to send five ICMP echo packets to 192.1.1.3. Each packet that was sent generated a debug statement saying that there was an encapsulation failure with no map entry link. The router cannot send the ping packet to 192.1.1.3 because it does not have a Frame Relay map to 192.1.1.3.

Now let's connect to RouterB. Verify that there is no Frame Relay map with the **show frame map** command.

```
RouterB#sh frame map
```

The **show frame pvc** command will verify that DLCI 201 is being used as the local DLCI. Remember that the router will not inverse ARP on this DLCI, since inverse ARP has been disabled.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 201, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:03:09, last time pvc status changed 00:03:09
  Num Pkts Switched 0
```

Turn on Frame Relay packet debugging on RouterB with the **debug frame packet** command.

```
RouterB#debug frame packet
Frame Relay packet debugging is on
```

Attempt to ping RouterA at IP address 192.1.1.1.

```
RouterB#ping 192.1.1.1
```

Notice that RouterB has the same problem as RouterA. Neither router knows how to reach the far−end router.

```
Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:

Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Success rate is 0 percent (0/5)
```

The problem of RouterA not being able to see RouterB and RouterB not being able to see RouterA can be fixed by adding a Frame Relay map in the configurations of RouterA and RouterB. Enter configuration mode and under interface s 0/0 type the **frame−relay map ip 192.1.1.1 201** command. This command tells RouterB that to reach the IP address of 192.1.1.1 it should encapsulate its Frame Relay traffic in DLCI 201 and send it out interface s 0/0.

```
RouterB#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterB(config)#int s 0/0
RouterB(config−if)#frame−relay map ip 192.1.1.1 201
RouterB(config−if)#exit
RouterB(config)#exit
```

Display the current Frame Relay maps with the **show frame map** command. Remember that most changes on the router take effect immediately. Notice how there is now a mapping between 192.1.1.1 and DLCI 201. Notice also that this map is a static map. The map is static because it was manually added in the configuration.

```
RouterB#sh fra map
Serial0/0 (up): ip 192.1.1.1 dlci 201(0xC9,0x3090), static,
              CISCO, status defined, active
```

Now let's try to ping RouterA at 192.1.1.1 with the **ping 192.1.1.1** command. Before typing the ping command, turn on Frame Relay packet debugging so that you can see the results of the ping.

```
RouterB#debug frame packet
Frame Relay packet debugging is on

RouterB#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:

Serial0/0 (o): dlci 201(0x3091), pkt type 0x800(IP), datagramsize 104.
Serial0/0 (o): dlci 201(0x3091), pkt type 0x800(IP), datagramsize 104.
Serial0/0 (o): dlci 201(0x3091), pkt type 0x800(IP), datagramsize 104.
Serial0/0 (o): dlci 201(0x3091), pkt type 0x800(IP), datagramsize 104.
Serial0/0 (o): dlci 201(0x3091), pkt type 0x800(IP), datagramsize 104.
Success rate is 0 percent (0/5)
```

The ping was not a success. None of the five ICMP echo packets sent to RouterA were returned. But notice the output of the debug trace. Each of the five ICMP packets sent to RouterA were encapsulated in DLCI 201. This is correct, since we now have a Frame Relay map that associates the IP address of RouterA (192.1.1.1) with DLCI 201. Why, then, did the ping not work ? RouterB knows how to get to RouterA. Why did the ICMP packets not get returned ? The answer is that even though RouterB has a Frame Relay map to RouterA, RouterA does not know how to get back to RouterB. When the ping is sent from RouterB to RouterA, it is being sent to RouterA via DLCI 201 as per the Frame Relay map. But when RouterA has to send the ICMP

packet back to RouterB, it does not know how to send it. The solution is to also add a Frame Relay map statement to RouterA so that a return path to RouterB exists.

Connect to RouterA and go into configuration mode. Enter the command **frame−relay map ip 192.1.1.3 200** under interface s 0/0.

```
RouterA#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterA(config)#int s 0/0
RouterA(config−if)#frame−relay map ip 192.1.1.3 200
RouterA(config−if)#exit
RouterA(config)#exit
```

Display the current Frame Relay map table with the **show frame map** command.

```
RouterA#sh frame map
Serial0/0 (up): ip 192.1.1.3 dlci 200(0xC8,0x3080), static,
              CISCO, status defined, active
```

This map tells RouterA that to get to 192.1.1.3 (which is the address of RouterB), it must send traffic out of interface s 0/0 encapsulated in DLCI 200.

Now try to ping RouterB from RouterA. The ping is successful. Both RouterA and RouterB have a defined path to each other.

```
RouterA#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/56/60 ms
```

Lab #16: Full Connectivity witha Partial PVC Mesh and FrameRelay Map Statements

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, three of which must have at least one serial port, and one of which must have three serial ports
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection to the routers
- Three Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

> Note The DCE side of the V.35 crossover cables must be connected to the router FrameSwitch.

Configuration Overview

This configuration will demonstrate a method of achieving full mesh connectivity in a network that does not have a full mesh of PVCs. The configuration for this lab is shown in Figure 4−16. Most network protocols assume transitivity. This means that if RouterB can communicate with RouterA and if RouterC can

communicate with RouterA, then RouterB can communicate with RouterC. This does not apply with Frame Relay.

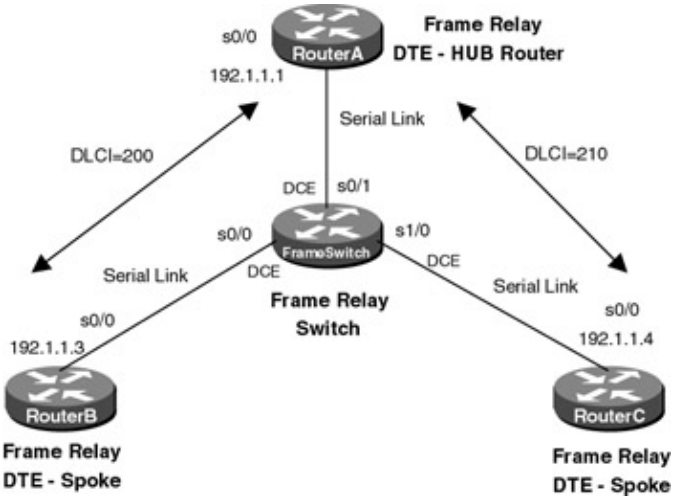


Figure 4−16: Full connectivity with partial PVC mesh

This issue poses a problem in configuring Frame Relay networks. As depicted in Figure 4−16, the configuration has only two PVCs. A company purchasing PVCs from a Frame Relay provider would ideally like to be able to communicate from RouterB to RouterC without having to purchase a third PVC between RouterB and RouterC.

The four routers are connected as shown in Figure 4−16. Three of the routers are configured as Frame Relay DTE devices. The fourth router is configured as a Frame Relay switch. The router configured as a Frame Relay switch is also configured to supply clock to the DTE router. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router's configuration.

> Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Router Configuration

The initial configurations for the four routers in this example are as follows. Key Frame Relay commands are highlighted in bold.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname FrameSwitch
!
!
frame−relay switching ← Configure Frame Relay switching on this interface
!
interface Serial0/0
 no ip address
 encapsulation frame−relay
 clockrate 64000 ← Clock the DTE at 64,000 bps
 frame−relay lmi−type ansi ← Set the LMI type to Annex D
 frame−relay intf−type dce ← Set the interface type to a DCE
 frame−relay route 200 interface Serial0/1  200 ← Define a PVC between
                                                   interface S0/0 and S0/1
!
interface Serial0/1
 no ip address
```

```
 encapsulation frame-relay
 clockrate 64000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/0  200
 frame-relay route 210 interface Serial1/0  210
!
interface Serial1/0
 no ip address
 encapsulation frame-relay
 clockrate 64000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 210 interface Serial0/1  210
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay ← Set the interface encapsulation to Frame Relay
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
no ip classless
!
line con 0
```

```
line aux 0
line vty 0 4
 login
!
end
```

RouterC (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
interface Serial0/0
 ip address 192.1.1.4 255.255.255.0
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's begin by connecting to the router FrameSwitch and verifying that it is working properly. Issue the **show frame pvc** command to display all DLCIs that are passing through the router. The PVC configuration is more complex for this lab than for the previous labs. There are now two PVCs that are configured on this router. Several items indicate that these ports are acting as Frame Relay switch ports. These include the DLCI usage being referenced as switched, the interface being referred to as a Frame Relay DCE, and the indication of Num Pkts Switched. The PVC status should indicate Active.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 16            output pkts 17           in bytes 1590
  out bytes 1620           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:47:35, last time pvc status changed 00:46:33
  Num Pkts Switched 16

PVC Statistics for interface Serial0/1  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 17            output pkts 16           in bytes 1620
  out bytes 1590           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:46:41, last time pvc status changed 00:46:40
  Num Pkts Switched 17
```

```
DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 40            output pkts 36           in bytes 3790
  out bytes 3670           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:46:30, last time pvc status changed 00:15:12
  Num Pkts Switched 40

PVC Statistics for interface Serial1/0  (Frame Relay DCE)

DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

  input pkts 36            output pkts 40           in bytes 3670
  out bytes 3790           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:47:07, last time pvc status changed 00:46:24
  Num Pkts Switched 36
```

Next issue the **show frame route** command. This command will display all active PVCs that are defined on the router. Four DLCIs are configured on this router. RouterA is acting as a hub router. All DLCIs terminate on this router. RouterB and RouterC are acting as spoke routers, each having a PVC terminating on RouterA.

```
FrameSwitch#sh frame route
Input Intf     Input Dlci    Output Intf    Output Dlci    Status
Serial0/0      200           Serial0/1      200            active
Serial0/1      200           Serial0/0      200            active
Serial0/1      210           Serial1/0      210            active
Serial1/0      210           Serial0/1      210            active
```

The **show interface** command will display the status of the serial interfaces on the router. Several important Frame Relay parameters displayed by this command are highlighted in bold, including the interface encapsulation (Frame−Relay), the LMI status (LMI up), the fact that this port is acting as a Frame Relay DCE, the LMI signaling type (ANSI Annex D), and the LMI exchange counters.

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME−RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 297, LMI stat sent  297, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.
```

Let's start by connecting to RouterC. Type the **show frame map** command to display the current Frame Relay map.

```
RouterC#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

We see that RouterC has resolved the IP address of RouterA (192.1.1.1) via inverse ARP. Notice that RouterC has not resolved the address of RouterB. This is because RouterC has a PVC only to RouterA, not to RouterB. In general, a spoke router (such as RouterC) will inverse ARP to the hub router (such as RouterA) but will not inverse ARP to other spoke routers.

Verify that you can ping from RouterC to RouterA:

```
RouterC#ping 192.1.1.1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/56 ms
```

Verify that DLCI 210 is active on interface s0/0 of RouterC:

```
RouterC#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 33            output pkts 31           in bytes 3210
  out bytes 3150           dropped pkts 2           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 1         out bcast bytes 30
  pvc create time 00:10:44, last time pvc status changed 00:09:44
```

Now let's connect to RouterB. The **show frame map** command will verify that RouterB has resolved the IP address of RouterA via inverse ARP. Again, notice that RouterB does not have a mapping to RouterC.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
```

Verify that you can ping RouterA from RouterB:

```
RouterB#ping 192.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms
```

Verify that DLCI 200 is active on interface s0/0 of RouterB:

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 12            output pkts 11           in bytes 1100
  out bytes 1070           dropped pkts 1           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:43:35, last time pvc status changed 00:42:35
```

Now connect to RouterA. The **show frame pvc** command should report two DLCIs coming into RouterA, both assigned to interface s0/0. As we can see from Figure 4-16, one of these DLCIs connects RouterA to RouterB (DLCI 200) and the second DLCI connects RouterA to RouterC (DLCI 210).

```
RouterA#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 16            output pkts 16           in bytes 1590
  out bytes 1590           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:44:41, last time pvc status changed 00:44:41
```

```
DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 36            output pkts 39           in bytes 3670
  out bytes 3760           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:44:32, last time pvc status changed 00:13:12
```

The **show frame map** command will reveal that RouterA has resolved both of its DLCIs to a far−end IP address. DLCI 200 has been resolved to 192.1.1.3 (RouterB) and DLCI 210 has been resolved to 192.1.1.4 (RouterC).

```
RouterA#sh fra map
Serial0/0 (up): ip 192.1.1.3 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.1.1.4 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

Verify that you can reach both RouterB and RouterC with the ping command:

```
RouterA#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/57/60 ms

RouterA#ping 192.1.1.4

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/56/60 ms
```

Now let's reconnect to RouterC. Verify that RouterC still has a Frame Relay map to RouterA with the **show frame map** command.

```
RouterC#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

Enable Frame Relay packet debugging with the **debug frame packet** command.

```
RouterC#debug frame packet
Frame Relay packet debugging is on
```

Verify that our hub router (RouterA) is still reachable at IP address 192.1.1.1.

```
RouterC#ping 192.1.1.1
Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/57/60 ms
```

The ping should be successful. The following screen print shows what the output from the **debug frame packet** command will look like. Notice that the outgoing pings are sent on DLCI 210. The incoming responses also come in on DLCI 210. RouterC knows how to reach RouterA because it has a Frame Relay map entry to RouterA. Notice that there are five outgoing packets and five incoming packets.

```
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
```

```
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
```

Now let's try to ping RouterB at IP address 192.1.1.3.

```
RouterC#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:

Serial0/0:Encaps failed--no map entry link 7(IP).
Serial0/0:Encaps failed--no map entry link 7(IP).
Serial0/0:Encaps failed--no map entry link 7(IP).
Serial0/0:Encaps failed--no map entry link 7(IP).
Serial0/0:Encaps failed--no map entry link 7(IP).
Success rate is 0 percent (0/5)
```

The ping failed. The output of the **debug frame packet** shows that RouterC does not know how to encapsulate the ping that is destined for RouterB. This is caused by RouterC not having a Frame Relay mapping to RouterB.

The solution is to tell RouterC how to get to RouterB with a Frame Relay map statement. Enter configuration mode and enter the command **frame-relay map ip 192.1.1.3 210** under interface s 0/0. This will tell RouterC that if it has traffic for RouterB it should send that traffic out on DLCI 210. This will be enough to get the traffic to RouterA. RouterA then has its own Frame Relay map to RouterB, so the traffic will be able to find its destination.

```
RouterC#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterC(config)#int s 0/0
RouterC(config-if)#frame-relay map ip 192.1.1.3 210
RouterC(config-if)#exit
RouterC(config)#exit
```

Verify that the new Frame Relay map has taken effect by typing the **show frame map** command. Notice how the map to RouterA is dynamic while the map to RouterB is static. This is because the map to RouterA (192.1.1.1) was discovered via inverse ARP while the map to RouterB (192.1.1.3) was manually entered into the RouterC's configuration.

```
RouterC#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.1.1.3 dlci 210(0xD2,0x3420), static,
              CISCO, status defined, active
```

Now try to ping RouterB at IP address 192.1.1.3.

```
RouterC#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:

Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Success rate is 0 percent (0/5)
```

Notice that the ping still fails. The output from the debug command is now different from the first time we tried to ping 192.1.1.3. The first time we tried the ping, we had not added a static Frame Relay map to 192.1.1.3 and the debug output showed encapsulation failures. Now the debug shows that the router knows to encapsulate the ICMP packets in DLCI 210. Why, then, does the ping fail? The ping fails because when the ping traffic gets to RouterB, RouterB does not have a path back to RouterC. We must now go to RouterB and add an equivalent Frame Relay map statement.

Connect to RouterB and display the current Frame Relay map with the **show frame map** command. Verify that only a single dynamic map exists to RouterA at IP address 192.1.1.1.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
```

Enter configuration mode and add the statement **frame−relay map IP 192.1.1.4 200** under interface s 0/0. This will tell RouterB that if it has traffic for 192.1.1.4 (RouterC), it should then send the traffic out on DLCI 200. Encapsulating the traffic on DLCI 200 will send it to RouterA. RouterA then has a Frame Relay map to RouterC and will know how to get the traffic there.

```
RouterB#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterB(config)#int s 0/0
RouterB(config−if)#frame−relay map ip 192.1.1.4 200
RouterB(config−if)#exit
RouterB(config)#exit
```

Verify that there are now two Frame Relay maps for RouterB. The first map, to RouterA (192.1.1.1), is a dynamic map because it was discovered via inverse ARP. The second map, to RouterC, is a static map because it was manually entered in the router's configuration.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.1.1.4 dlci 200(0xC8,0x3080), static,
              CISCO, status defined, active
```

You should now be able to successfully ping RouterC (at IP address 192.1.1.4) from RouterB.

```
RouterB#ping 192.1.1.4
Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 112/113/120 ms
```

Lab #16: Full Connectivity witha Partial PVC Mesh and FrameRelay Map Statements

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, three of which must have at least one serial port, and one of which must have three serial ports
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection to the routers

- Three Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

Note The DCE side of the V.35 crossover cables must be connected to the router FrameSwitch.

Configuration Overview

This configuration will demonstrate a method of achieving full mesh connectivity in a network that does not have a full mesh of PVCs. The configuration for this lab is shown in Figure 4−16. Most network protocols assume transitivity. This means that if RouterB can communicate with RouterA and if RouterC can communicate with RouterA, then RouterB can communicate with RouterC. This does not apply with Frame Relay.

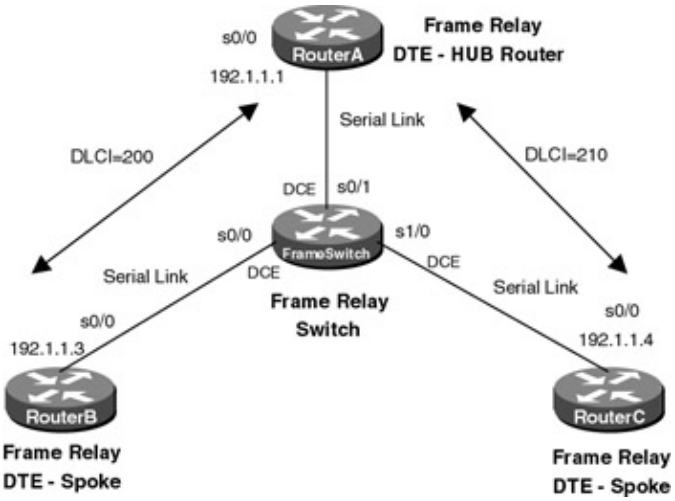


Figure 4−16: Full connectivity with partial PVC mesh

This issue poses a problem in configuring Frame Relay networks. As depicted in Figure 4−16, the configuration has only two PVCs. A company purchasing PVCs from a Frame Relay provider would ideally like to be able to communicate from RouterB to RouterC without having to purchase a third PVC between RouterB and RouterC.

The four routers are connected as shown in Figure 4−16. Three of the routers are configured as Frame Relay DTE devices. The fourth router is configured as a Frame Relay switch. The router configured as a Frame Relay switch is also configured to supply clock to the DTE router. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router's configuration.

Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Router Configuration

The initial configurations for the four routers in this example are as follows. Key Frame Relay commands are highlighted in bold.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
```

```
!
frame-relay switching ← Configure Frame Relay switching on this interface
!
interface Serial0/0
 no ip address
 encapsulation frame-relay
 clockrate 64000 ← Clock the DTE at 64,000 bps
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
 frame-relay intf-type dce ← Set the interface type to a DCE
 frame-relay route 200 interface Serial0/1  200 ← Define a PVC between
                                                   interface S0/0 and S0/1
!
interface Serial0/1
 no ip address
 encapsulation frame-relay
 clockrate 64000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/0  200
 frame-relay route 210 interface Serial1/0  210
!
interface Serial1/0
 no ip address
 encapsulation frame-relay
 clockrate 64000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 210 interface Serial0/1  210
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay ← Set the interface encapsulation to Frame Relay
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
```

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
interface Serial0/0
 ip address 192.1.1.4 255.255.255.0
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's begin by connecting to the router FrameSwitch and verifying that it is working properly. Issue the **show frame pvc** command to display all DLCIs that are passing through the router. The PVC configuration is more complex for this lab than for the previous labs. There are now two PVCs that are configured on this router. Several items indicate that these ports are acting as Frame Relay switch ports. These include the DLCI usage being referenced as switched, the interface being referred to as a Frame Relay DCE, and the indication of Num Pkts Switched. The PVC status should indicate Active.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 16            output pkts 17           in bytes 1590
  out bytes 1620           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
```

```
  pvc create time 00:47:35, last time pvc status changed 00:46:33
  Num Pkts Switched 16

PVC Statistics for interface Serial0/1  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 17            output pkts 16           in bytes 1620
  out bytes 1590           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:46:41, last time pvc status changed 00:46:40
  Num Pkts Switched 17

DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 40            output pkts 36           in bytes 3790
  out bytes 3670           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:46:30, last time pvc status changed 00:15:12
  Num Pkts Switched 40

PVC Statistics for interface Serial1/0  (Frame Relay DCE)

DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

  input pkts 36            output pkts 40           in bytes 3670
  out bytes 3790           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:47:07, last time pvc status changed 00:46:24
  Num Pkts Switched 36
```

Next issue the **show frame route** command. This command will display all active PVCs that are defined on the router. Four DLCIs are configured on this router. RouterA is acting as a hub router. All DLCIs terminate on this router. RouterB and RouterC are acting as spoke routers, each having a PVC terminating on RouterA.

```
FrameSwitch#sh frame route
Input Intf     Input Dlci    Output Intf    Output Dlci    Status
Serial0/0      200           Serial0/1      200            active
Serial0/1      200           Serial0/0      200            active
Serial0/1      210           Serial1/0      210            active
Serial1/0      210           Serial0/1      210            active
```

The **show interface** command will display the status of the serial interfaces on the router. Several important Frame Relay parameters displayed by this command are highlighted in bold, including the interface encapsulation (Frame−Relay), the LMI status (LMI up), the fact that this port is acting as a Frame Relay DCE, the LMI signaling type (ANSI Annex D), and the LMI exchange counters.

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME−RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 297, LMI stat sent  297, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.
```

Let's start by connecting to RouterC. Type the **show frame map** command to display the current Frame Relay map.

```
RouterC#sh frame map
```

```
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

We see that RouterC has resolved the IP address of RouterA (192.1.1.1) via inverse ARP. Notice that RouterC has not resolved the address of RouterB. This is because RouterC has a PVC only to RouterA, not to RouterB. In general, a spoke router (such as RouterC) will inverse ARP to the hub router (such as RouterA) but will not inverse ARP to other spoke routers.

Verify that you can ping from RouterC to RouterA:

```
RouterC#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/56 ms
```

Verify that DLCI 210 is active on interface s0/0 of RouterC:

```
RouterC#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 33            output pkts 31           in bytes 3210
  out bytes 3150           dropped pkts 2           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 1         out bcast bytes 30
  pvc create time 00:10:44, last time pvc status changed 00:09:44
```

Now let's connect to RouterB. The **show frame map** command will verify that RouterB has resolved the IP address of RouterA via inverse ARP. Again, notice that RouterB does not have a mapping to RouterC.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
```

Verify that you can ping RouterA from RouterB:

```
RouterB#ping 192.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms
```

Verify that DLCI 200 is active on interface s0/0 of RouterB:

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 12            output pkts 11           in bytes 1100
  out bytes 1070           dropped pkts 1           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:43:35, last time pvc status changed 00:42:35
```

Now connect to RouterA. The **show frame pvc** command should report two DLCIs coming into RouterA, both assigned to interface s0/0. As we can see from Figure 4-16, one of these DLCIs connects RouterA to

RouterB (DLCI 200) and the second DLCI connects RouterA to RouterC (DLCI 210).

```
RouterA#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 16            output pkts 16           in bytes 1590
  out bytes 1590           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:44:41, last time pvc status changed 00:44:41

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 36            output pkts 39           in bytes 3670
  out bytes 3760           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:44:32, last time pvc status changed 00:13:12
```

The **show frame map** command will reveal that RouterA has resolved both of its DLCIs to a far−end IP address. DLCI 200 has been resolved to 192.1.1.3 (RouterB) and DLCI 210 has been resolved to 192.1.1.4 (RouterC).

```
RouterA#sh fra map
Serial0/0 (up): ip 192.1.1.3 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.1.1.4 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

Verify that you can reach both RouterB and RouterC with the ping command:

```
RouterA#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/57/60 ms

RouterA#ping 192.1.1.4

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/56/60 ms
```

Now let's reconnect to RouterC. Verify that RouterC still has a Frame Relay map to RouterA with the **show frame map** command.

```
RouterC#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

Enable Frame Relay packet debugging with the **debug frame packet** command.

```
RouterC#debug frame packet
Frame Relay packet debugging is on
```

Verify that our hub router (RouterA) is still reachable at IP address 192.1.1.1.

```
RouterC#ping 192.1.1.1
Type escape sequence to abort.
```

```
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/57/60 ms
```

The ping should be successful. The following screen print shows what the output from the **debug frame packet** command will look like. Notice that the outgoing pings are sent on DLCI 210. The incoming responses also come in on DLCI 210. RouterC knows how to reach RouterA because it has a Frame Relay map entry to RouterA. Notice that there are five outgoing packets and five incoming packets.

```
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 210(0x3421), pkt type 0x800, datagramsize 104
```

Now let's try to ping RouterB at IP address 192.1.1.3.

```
RouterC#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:

Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Success rate is 0 percent (0/5)
```

The ping failed. The output of the **debug frame packet** shows that RouterC does not know how to encapsulate the ping that is destined for RouterB. This is caused by RouterC not having a Frame Relay mapping to RouterB.

The solution is to tell RouterC how to get to RouterB with a Frame Relay map statement. Enter configuration mode and enter the command **frame−relay map ip 192.1.1.3 210** under interface s 0/0. This will tell RouterC that if it has traffic for RouterB it should send that traffic out on DLCI 210. This will be enough to get the traffic to RouterA. RouterA then has its own Frame Relay map to RouterB, so the traffic will be able to find its destination.

```
RouterC#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterC(config)#int s 0/0
RouterC(config−if)#frame−relay map ip 192.1.1.3 210
RouterC(config−if)#exit
RouterC(config)#exit
```

Verify that the new Frame Relay map has taken effect by typing the **show frame map** command. Notice how the map to RouterA is dynamic while the map to RouterB is static. This is because the map to RouterA (192.1.1.1) was discovered via inverse ARP while the map to RouterB (192.1.1.3) was manually entered into the RouterC's configuration.

```
RouterC#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.1.1.3 dlci 210(0xD2,0x3420), static,
              CISCO, status defined, active
```

Now try to ping RouterB at IP address 192.1.1.3.

```
RouterC#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:

Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Serial0/0(o): dlci 210(0x3421), pkt type 0x800(IP), datagramsize 104.
Success rate is 0 percent (0/5)
```

Notice that the ping still fails. The output from the debug command is now different from the first time we tried to ping 192.1.1.3. The first time we tried the ping, we had not added a static Frame Relay map to 192.1.1.3 and the debug output showed encapsulation failures. Now the debug shows that the router knows to encapsulate the ICMP packets in DLCI 210. Why, then, does the ping fail? The ping fails because when the ping traffic gets to RouterB, RouterB does not have a path back to RouterC. We must now go to RouterB and add an equivalent Frame Relay map statement.

Connect to RouterB and display the current Frame Relay map with the **show frame map** command. Verify that only a single dynamic map exists to RouterA at IP address 192.1.1.1.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
```

Enter configuration mode and add the statement **frame-relay map IP 192.1.1.4 200** under interface s 0/0. This will tell RouterB that if it has traffic for 192.1.1.4 (RouterC), it should then send the traffic out on DLCI 200. Encapsulating the traffic on DLCI 200 will send it to RouterA. RouterA then has a Frame Relay map to RouterC and will know how to get the traffic there.

```
RouterB#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterB(config)#int s 0/0
RouterB(config-if)#frame-relay map ip 192.1.1.4 200
RouterB(config-if)#exit
RouterB(config)#exit
```

Verify that there are now two Frame Relay maps for RouterB. The first map, to RouterA (192.1.1.1), is a dynamic map because it was discovered via inverse ARP. The second map, to RouterC, is a static map because it was manually entered in the router's configuration.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.1.1.4 dlci 200(0xC8,0x3080), static,
              CISCO, status defined, active
```

You should now be able to successfully ping RouterC (at IP address 192.1.1.4) from RouterB.

```
RouterB#ping 192.1.1.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 112/113/120 ms
```

Lab #17: Full Connectivity with a Partial PVC Mesh and Subinterfaces

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, three of which must have at least one serial port, and one of which must have three serial ports
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection to the routers
- Three Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

> Note The DCE side of the crossover cables must connect to the router FrameSwitch.

Configuration Overview

This configuration will demonstrate a method of achieving full mesh connectivity in a network that does not have a full mesh of PVCs by using subinterfaces. A subinterface is a way to treat a physical interface as multiple virtual interfaces, each having its own network layer address. The configuration for this lab is shown in Figure 4−17. Most network protocols assume transitivity. This means that if RouterB can communicate with RouterA and if RouterC can communicate with RouterA, then RouterB can communicate with RouterC. This does not apply with Frame Relay.

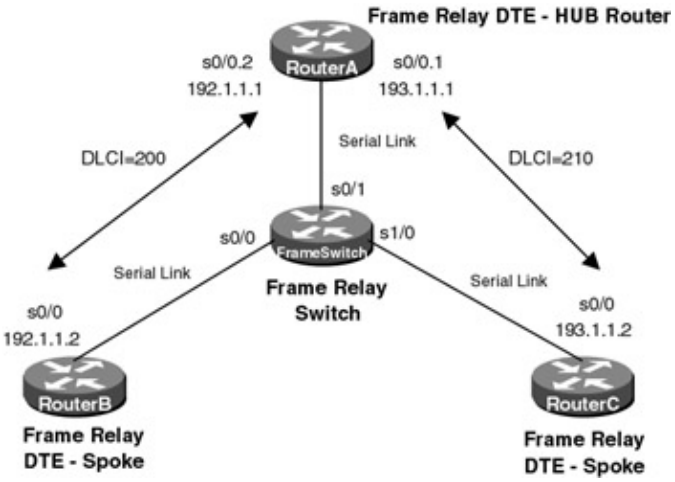


Figure 4−17: Subinterfaces

This issue poses a problem in configuring Frame Relay networks. As depicted in Figure 4−17, the configuration has only two PVCs. A company purchasing PVCs from a Frame Relay provider would ideally like to be able to communicate from RouterB to RouterC without having to purchase a third PVC between RouterB and RouterC.

The four routers are connected as shown in Figure 4−17. Three of the routers are configured as Frame Relay DTE devices. The fourth router is configured as a Frame Relay switch. The router configured as a Frame Relay switch is also configured to supply clock to the DTE router. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router's configuration.

> Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Router Configuration

The configurations for the four routers in this example are as follows. Frame Relay−specific commands are highlighted in bold.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname FrameSwitch
!
!
frame−relay switching ← Define Frame Relay switching on this router
!
interface Serial0/0
 no ip address
 encapsulation frame−relay
 clockrate 64000 ← Clock the DTE at a speed of 64,000 bps
 frame−relay lmi−type ansi ← Set the LMI type to Annex D
 frame−relay intf−type dce ← Set the interface type a DCE
 frame−relay route 200 interface Serial0/1 200 ← Define a PVC between port s0/0
                                                  and s0/1
!
interface Serial0/1
 no ip address
 encapsulation frame−relay
 clockrate 64000
 frame−relay lmi−type ansi
 frame−relay intf−type dce
 frame−relay route 200 interface Serial0/0 200 ← This interface will terminate
                                                  two PVCs
 frame−relay route 210 interface Serial1/0 210
!
interface Serial1/0
 no ip address
 encapsulation frame−relay
 clockrate 64000
 frame−relay lmi−type ansi
 frame−relay intf−type dce
 frame−relay route 210 interface Serial0/1 210
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Serial0/0
```

```
 no ip address
 encapsulation frame-relay ← Set the interface to Frame Relay encapsulation
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
interface Serial0/0.1 point-to-point
 ip address 193.1.1.1 255.255.255.0
 frame-relay interface-dlci 210 ← Assign DLCI 210 to subinterface s0/0.1
!
interface Serial0/0.2 point-to-point
 ip address 192.1.1.1 255.255.255.0
 frame-relay interface-dlci 200 ← Assign DLCI 200 to subinterface s0/0.2
!
router rip
 network 193.1.1.0
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
frame-relay de-list 3 protocol ip gt 512
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation frame-relay ← Set the interface to Frame Relay encapsulation
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
router rip
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
```

```
interface Serial0/0
 ip address 193.1.1.2 255.255.255.0
 encapsulation frame-relay ← Set the interface to Frame Relay encapsulation
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
router rip
 network 193.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's begin by connecting to the router FrameSwitch and verifying that it is working properly. Issue the **show frame pvc** command to display all DLCIs that are passing through the router. The PVC configuration is more complex for this lab than for the previous labs. There are now two PVCs that are configured on this router. Several items indicate that these ports are acting as Frame Relay switch ports. These include the DLCI usage being referenced as switched, the interface being referred to as a Frame Relay DCE, and the indication of Num Pkts Switched. The PVC status should indicate Active.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 165           output pkts 178          in bytes 15902
  out bytes 19810          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 02:56:16, last time pvc status changed 02:55:15
  Num Pkts Switched 165

PVC Statistics for interface Serial0/1  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 178           output pkts 165          in bytes 19810
  out bytes 15902          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 02:55:23, last time pvc status changed 02:55:22
  Num Pkts Switched 178

DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 181           output pkts 161          in bytes 19874
  out bytes 15652          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 02:55:11, last time pvc status changed 02:23:53
  Num Pkts Switched 181

PVC Statistics for interface Serial1/0  (Frame Relay DCE)

DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

  input pkts 161           output pkts 181          in bytes 15652
```

```
  out bytes 19874          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 02:55:48, last time pvc status changed 02:55:05
  Num Pkts Switched 161
```

Next issue the **show frame route** command. This command will display all active PVCs that are defined on the router. Four DLCIs are configured on this router. RouterA is acting as a hub router. All DLCIs terminate on this router. RouterB and RouterC are acting as spoke routers, each having a PVC terminating on RouterA.

```
FrameSwitch#sh frame route
Input Intf        Input Dlci     Output Intf     Output Dlci     Status
Serial0/0         200            Serial0/1       200             active
Serial0/1         200            Serial0/0       200             active
Serial0/1         210            Serial1/0       210             active
Serial1/0         210            Serial0/1       210             active
```

The **show interface** command will display the statuses of the serial interfaces on the router. Several important Frame Relay parameters are displayed by this command and are highlighted in bold including the interface encapsulation (Frame−Relay), the LMI status (LMI up), the fact that this port is acting as a Frame Relay DCE, the LMI signaling type (ANSI Annex D), and the LMI exchange counters.

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME−RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  3, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 1106, LMI stat sent  1103, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.
```

Let's start by connecting to RouterA, display the running configuration.

```
interface Serial0/0
 no ip address
 encapsulation frame−relay
 frame−relay lmi−type ansi
!
interface Serial0/0.1 point−to−point
 ip address 193.1.1.1 255.255.255.0
 frame−relay interface−dlci 210
!
interface Serial0/0.2 point−to−point
 ip address 192.1.1.1 255.255.255.0
 frame−relay interface−dlci 200
```

The preceding screen print contains a portion of the configuration for RouterA. This configuration uses two subinterfaces, highlighted in bold. The physical interface s0/0 is being made to appear as two different physical interfaces, s0/0.1 and s0/0.2. Both s0/0.1 and s0/0.2 are referred to as subinterfaces. Since each subinterface appears as a separate interface, it can have its own IP address. Notice that s0/0.1 and s0/0.2 have IP addresses that are on different networks. Because of this fact, the router can route between these two subinterfaces. This will allow us to achieve full connectivity without having to have a static Frame Relay map statement added on any of the routers as we had to do in our previous configuration. The advantage is that this configuration is totally dynamic and does not depend on any statically mapped parameters.

Under each subinterface there is a **frame−relay interface dlci** command. This command tells each subinterface what DLCI is assigned to it. The subinterface uses this information to know what DLCI to inverse ARP on. Subinterface s0/0.1 will inverse ARP on DLCI 210, and subinterface s0/0.2 will inverse ARP on DLCI 200.

Verify that both DLCI 210 and DLCI 200 are active on RouterA with the **show frame pvc** command.

```
RouterA#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.2

  input pkts 115           output pkts 119          in bytes 10980
  out bytes 13961          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:23:55, last time pvc status changed 00:23:55

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.1

  input pkts 81            output pkts 97           in bytes 7610
  out bytes 11524          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:23:57, last time pvc status changed 00:23:57
```

Display the Frame Relay map for RouterA with the **show frame map** command.

```
RouterA#sh frame map
Serial0/0.1 (up): point-to-point dlci, dlci 210(0xD2,0x3420), broadcast
          status defined, active
Serial0/0.2 (up): point-to-point dlci, dlci 200(0xC8,0x3080), broadcast
          status defined, active
```

Notice how the command output shows us what DLCI is associated with what subinterface. DLCI 200 and 210 are assigned to subinterfaces s0/0.2 and s0/0.1, respectively. Remember that this is defined by the **frame-relay interface-dlci** command in the configuration for RouterA.

Since a subinterface acts as a separate interface and has its own IP address, we need to examine the routing tables of the routers in order to understand how this configuration works. Use the **show ip route** command to display the contents of the routers' routing table. RouterA sees two networks as being directly connected, 192.1.1.0 and 193.1.1.0.

```
RouterA#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

C    192.1.1.0/24 is directly connected, Serial0/0 .2
C    193.1.1.0/24 is directly connected, Serial0/0 .1
```

Verify that both RouterB (192.1.1.2) and RouterC (193.1.1.2) can be reached via the ping command.

```
RouterA#ping 192.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms

RouterA#ping 193.1.1.2

Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 193.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms
```

Now let's connect to RouterC. Verify that DLCI 210 is active by typing the **show frame pvc** command.

```
RouterC#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 60            output pkts 57           in bytes 6469
  out bytes 5686           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 7         out bcast bytes 486
  pvc create time 00:02:45, last time pvc status changed 00:02:45
```

Use the **show frame map** command to check that RouterC has a Frame Relay map to RouterA (193.1.1.1) via DLCI 210.

```
RouterC#sh frame map
Serial0/0 (up): ip 193.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

Now let's take a look at the routing table for RouterC.

```
RouterC#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

R    192.1.1.0/24 [120/1] via 193.1.1.1, 00:00:04, Serial0/0
C    193.1.1.0/24 is directly connected, Serial0/0
```

Notice that RouterC has a directly connected route to the 193.1.1.0 network. This is the network that is connected to the s0/0 interface of RouterC, so it appears as a directly connected network. Also notice that there is a route learned via RIP that has the 192.1.1.0 network as its destination. 192.1.1.0 is the network that RouterB is on. RouterC has learned about a route to RouterB. Keep in mind that this is all automatic. We did not have to enter any static configurations into any of our routers.

Try to ping subinterface s0/0.2 of RouterA at IP address 192.1.1.1. The ping should be successful. Although RouterC only has a Frame Relay map to 193.1.1.1, it has learned a RIP route to 192.1.1.1 via the next hop address of 193.1.1.1. RouterC therefore knows that the ping to 192.1.1.1 should be encapsulated in DLCI 210.

```
RouterC#ping 192.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms
```

Now try to ping the s0/0 interface of RouterB at IP address 192.1.1.2. Again the ping should be successful. RouterC has a RIP route to the 192.1.1.0 network with a next hop address of 193.1.1.1. RouterC knows how to get to 193.1.1.1, since it has a Frame Relay map to that address.

```
RouterC#ping 192.1.1.2
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 108/109/112 ms
```

Now let's connect to RouterB. Verify that DLCI 200 is active by typing the **show frame pvc** command.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 100           output pkts 97           in bytes 11661
  out bytes 9146           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:23:53, last time pvc status changed 00:23:53
```

Display the Frame Relay map with the **show frame map** command. Notice that RouterB only has a Frame Relay map to RouterA's subinterface s0/0.2 at IP address 192.1.1.1.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
```

Display the routing table with the **show ip route** command. Notice that there are two routes in the table. The first route entry is for the directly connected 192.1.1.0 network. The second route is learned via RIP. This route is to the 193.1.1.0 network (RouterC) with a next hop address of 192.1.1.1

```
RouterB#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set

C    192.1.1.0/24 is directly connected, Serial0/0
R    193.1.1.0/24 [120/1] via 192.1.1.1, 00:00:04, Serial0/0
```

Ping the s0/0.1 subinterface (193.1.1.1) on RouterA.

```
RouterB#ping 193.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 193.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms
```

The ping should be successful. RouterB knows how to get to the 193.1.1.0 network because there is a RIP route in the routing table that tells us that the 193.1.1.0 network is reachable via the 192.1.1.0 network. RouterB has a Frame Relay map to 192.1.1.1, so it knows how to send traffic to 193.1.1.1.

Finally, verify that you can ping RouterC at IP address 193.1.1.2.

```
RouterB#ping 193.1.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 193.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 108/109/112 ms
```

We have demonstrated that it is possible to achieve a fully connected Frame Relay network without having to have a full mesh of PVCs using subinterfaces.

Lab #18: Frame Relay Traffic Shaping

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, two of which must have at least one serial port, and one of which must have two serial ports
- Cisco IOS 11.3 or higher
- A PC running a terminal emulation program for console port connection to the routers
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

> Note The DCE side of the crossover cables must be connected to the router FrameSwitch.

Configuration Overview

This configuration will demonstrate the Frame Relay traffic−shaping capabilities of a Cisco router. Frame Relay traffic shaping was introduced with IOS 11.2. Frame Relay traffic shaping allows QOS parameters to be assigned to each Frame Relay circuit on a per PVC basis.

The three routers are connected as shown in Figure 4−18. Two of the routers are configured as Frame Relay DTE devices. The third router is configured as a Frame Relay switch. The router configured as a Frame Relay switch is also configured to supply clock to the DTE router. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router's configuration.

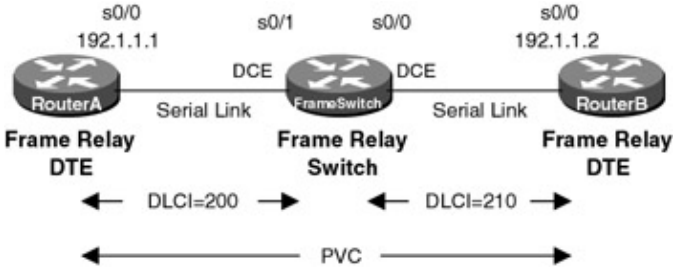


Figure 4−18: Frame Relay traffic shaping

> Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Router Configuration

The configurations for the three routers in this example are as follows. Key Frame Relay commands are shown in bold.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname FrameSwitch
```

```
!
!
frame-relay switching ← Enable Frame Relay switching on this router
!
interface Serial0/0
 no ip address
 encapsulation frame-relay
 clockrate 64000 ← Clock the DTE at a rate of 64,000 bps
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
 frame-relay intf-type dce ← Set the interface type to a DCE
 frame-relay route 210 interface Serial0/1  200 ← Define a PVC between
                                                  interface S0/0 and S0/1
!
interface Serial0/1
 no ip address
 encapsulation frame-relay
 clockrate 64000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/0  210
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay ← Set the interface to Frame Relay encapsulation
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
version 11.3
no service password-encryption
!
hostname RouterB
!
enable password cisco
!
interface Serial0/0
```

```
 ip address 192.1.1.2 255.255.255.0
 encapsulation frame-relay
 no fair-queue
 frame-relay traffic-shaping ← Enable traffic shaping on this interface
 frame-relay class first ← Associate a map class to this interface
 frame-relay lmi-type ansi
!
no ip classless
!
map-class frame-relay first ← Define a map class on this router
 frame-relay adaptive-shaping becn ← Enable BECN flow control
 frame-relay cir 56000 ← Define CIR value for traffic shaping
 frame-relay bc 1100 ← Define a Bc value for traffic shaping
 frame-relay mincir 1000 ← Define a minimum CIR value for traffic shaping
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to the router FrameSwitch and verifying that it is functioning properly. Use the **show frame route** command to display the current active DLCIs on the router. The output should appear as follows. The Frame route tells us that any traffic coming into interface s0/0 with a DLCI value of 210 will be sent out of interface s0/1 with a DLCI value of 200. Any traffic coming into interface s0/1 with a DLCI value of 200 will be sent out of interface s0/0 with a DLCI value of 210. Both DLCIs should show a status of active.

```
FrameSwitch#sh frame route
Input Intf        Input Dlci     Output Intf     Output Dlci     Status
Serial0/0         210            Serial0/1       200             active
Serial0/1         200            Serial0/0       210             active
```

The **show frame pvc** command should indicate that two DLCIs, 210 and 200, are active.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0  (Frame Relay DCE)

DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 5293          output pkts 6931         in bytes 801309
  out bytes 667933         dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 2d00h, last time pvc status changed 2d00h
  Num Pkts Switched 5292

PVC Statistics for interface Serial0/1  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 6931          output pkts 5292         in bytes 667933
  out bytes 801279         dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 2d00h, last time pvc status changed 03:50:05
  Num Pkts Switched 6931
```

We see that this router is acting like a Frame Relay switch because the DLCI usage is indicated to be switched, the serial interface is refered to as a Frame Relay DCE, and there is an indication of the number of packets switched.

Verify that both serial interfaces of the router FrameSwitch are up using the **show interface** command.

```
FrameSwitch#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 34, LMI stat sent  34, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.
FrameSwitch#sh int s 0/1
Serial0/1 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 17, LMI stat sent  17, LMI upd sent  0, DCE LMI up
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DCE
.
.
```

Now connect to RouterA. Verify that DLCI 200 is up and active by typing the **show frame pvc** command. The PVC status should be active.

```
RouterA#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 380           output pkts 871          in bytes 106424
  out bytes 133998         dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 03:49:28, last time pvc status changed 03:49:18
```

Show the current Frame Relay map with the **show frame map** command. Make sure that RouterA sees RouterB (IP address 192.1.1.2) at the far end of the circuit.

```
RouterA#sh frame map
Serial0/0 (up): ip 192.1.1.2 dlci 200(0xC8,0x3080), dynamic,
              broadcast,, status defined, active
```

Verify connectivity end to end by pinging RouterB at IP address 192.1.1.2. The ping should be successful.

```
RouterA#ping 192.1.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/597/768 ms
```

Now let's connect to RouterB. You can see from the router configuration at the beginning of this chapter that RouterB is configured for Frame Relay traffic shaping. Start by typing the **show frame pvc** command and verifying that DLCI 210 is up and active on RouterB.

```
RouterB#sh frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 39            output pkts 30           in bytes 3624
```

```
  out bytes 3120          dropped pkts 0          in FECN pkts 0
  in BECN pkts 0          out FECN pkts 0         out BECN pkts 0
  in DE pkts 0            out DE pkts 0
  out bcast pkts 0        out bcast bytes 0
  Shaping adapts to BECN
  pvc create time 02:41:15, last time pvc status changed 02:41:15
```

Now type the **show frame pvc** command but include DLCI 210 at the end of the command by typing **show frame pvc 210**.

```
RouterB#sh frame pvc 210

PVC Statistics for interface Serial0/0  (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 39           output pkts 30          in bytes 3624
  out bytes 3120          dropped pkts 0          in FECN pkts 0
  in BECN pkts 0          out FECN pkts 0         out BECN pkts 0
  in DE pkts 0            out DE pkts 0
  out bcast pkts 0        out bcast bytes 0
  Shaping adapts to BECN
  pvc create time 02:41:21, last time pvc status changed 02:41:21
  cir 56000     bc 1100      be 0         limit 137    interval 19
  mincir 1000      byte increment 137   BECN response yes
  pkts 30        bytes 3120      pkts delayed 0         bytes delayed 0
  shaping inactive
  Serial0/0  dlci 210 is first come first serve default queueing

  Output queue 0/40, 0 drop, 92 dequeued
```

We see that the command's output is very different. The output now shows the CIR, Bc, Be, MinCIR, and traffic−shaping statistics for this PVC. One of the most important indications in this output is the packets delayed. This shows how many Frame Relay packets have been buffered and delayed due to traffic shaping being activated.

Verify that RouterB has resolved the IP address of RouterA (192.1.1.1) by typing the **show frame map** command.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

Now show the status of the traffic and traffic shaping for DLCI 210 with the **show frame pvc 210** command.

```
RouterB#sh frame pvc 210

PVC Statistics for interface Serial0/0  (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 50           output pkts 40          in bytes 4720
  out bytes 4160          dropped pkts 0          in FECN pkts 0
  in BECN pkts 0          out FECN pkts 0         out BECN pkts 0
  in DE pkts 0            out DE pkts 0
  out bcast pkts 0        out bcast bytes 0
  Shaping adapts to BECN
  pvc create time 02:41:42, last time pvc status changed 02:41:42
  cir 56000     bc 1100      be 0         limit 137    interval 19
  mincir 1000      byte increment 137   BECN response yes
  pkts 40        bytes 4160      pkts delayed 0         bytes delayed 0
  shaping inactive
  Serial0/0  dlci 210 is first come first serve default queueing
```

```
  Output queue 0/40, 0 drop, 92 dequeued
```

Notice that the packet and byte count have incremented since they were last displayed. Packets delayed are still 0 because our CIR is set to 56,000 on a 64,000 bit/sec link. The five pings that were sent did not generate enough traffic to exceed the CIR value of 56,000 bits/sec.

Let's lower the traffic shaping threshold by decreasing the CIR from 56,000 bits/sec to 1100 bits/sec. We can do this by entering configuration mode and entering the command **frame−relay cir 1100**, as shown next.

```
RouterB#config term
Enter configuration commands, one per line. End with CNTL/Z.
RouterB(config)#map−class frame−relay first
RouterB(config−map−class)#frame−relay cir 1100
RouterB(config−map−class)#exit
RouterB(config)#exit
```

The output of the **show frame pvc 210** command will now indicate that the CIR is set to 1100 bits/sec.

```
RouterB#sh frame pvc 210

PVC Statistics for interface Serial0/0  (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 68            output pkts 40           in bytes 5728
  out bytes 4160           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 0         out bcast bytes 0
  Shaping adapts to BECN
  pvc create time 02:50:11, last time pvc status changed 02:50:11
  cir 1100      bc 1100      be 0         limit 15     interval 125
  mincir 1000      byte increment 18    BECN response yes
  pkts 40        bytes 4160      pkts delayed 0         bytes delayed 0
  shaping inactive
  Serial0/0  dlci 210 is first come first serve default queueing

  Output queue 0/40, 0 drop, 0 dequeued
```

Now ping RouterA at IP address 192.1.1.1.

```
RouterB#ping 192.1.1.1
Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 60/600/768 ms
```

Display the traffic shaping statistics for DLCI 210 with the **show frame pvc 210** command. Notice that the packets delayed value is now 4. The CIR is now set low enough that the ping generates enough traffic to exceed the CIR and activate traffic shaping on the router.

```
RouterB#sh frame pvc 210

PVC Statistics for interface Serial0/0  (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 73            output pkts 45           in bytes 6248
  out bytes 4680           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 0         out bcast bytes 0
  Shaping adapts to BECN
  pvc create time 02:50:27, last time pvc status changed 02:50:27
```

```
cir 1100      bc 1100      be 0         limit 15     interval 125
mincir 1000      byte increment 18    BECN response yes
pkts 49        bytes 5096      pkts delayed 4         bytes delayed 416
shaping inactive
Serial0/0  dlci 210 is first come first serve default queueing

Output queue 0/40, 0 drop, 4 dequeued
```

Lab #19: Monitoring and Troubleshooting Frame Relay Connections

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, two of which must have at least one serial port, and one of which must have two serial ports
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for console port connection to the routers
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable

> Note The DCE side of the crossover cables must be connected to the router FrameSwitch.

Configuration Overview

This lab will review the important monitoring and troubleshooting techniques for Frame Relay configurations.

The three routers are connected as shown in Figure 4−19. Two of the routers are configured as Frame Relay DTE devices. The third router is configured as a Frame Relay switch. The router configured as a Frame Relay switch is also configured to supply clock to the DTE router. This is accomplished via the use of a Cisco DCE cable and a clock rate statement in the router's configuration.

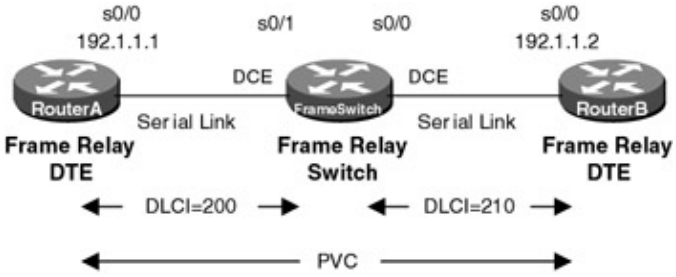


Figure 4−19: Monitoring Frame Relay

> Note Keep in mind that although a Cisco router can act as a Frame Relay switch, this feature is typically only used in test and demonstration situations such as this lab.

Router Configuration

The configurations for the three routers in this example are as follows. Key Frame Relay commands are highlighted in bold.

FrameSwitch (Frame Relay Switch)

```
Current configuration:
!
version 11.2
```

```
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
!
frame-relay switching ← Enable Frame Relay switching on this router
!
interface Serial0/0
 no ip address
 encapsulation frame-relay
 clockrate 64000 ← Generate a 64,000 bps clock to the DTE
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
 frame-relay intf-type dce ← Set the interface type to a DCE
 frame-relay route 210 interface Serial0/1  200 ← Define a PVC between
                                                   S0/0 and S0/1
!
interface Serial0/1
 no ip address
 encapsulation frame-relay
 clockrate 64000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/0 210
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA (Frame Relay DTE)

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
encapsulation frame-relay ← Set the interface to Frame Relay encapsulation
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
router rip
 network 193.1.1.0
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB (Frame Relay DTE)

```
Current configuration:
!
```

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
encapsulation frame-relay ← Set the interface to Frame Relay encapsulation
 frame-relay lmi-type ansi ← Set the LMI type to Annex D
!
router rip
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

There are several important show commands that can be used to verify that a Cisco router acting as a Frame Relay switch is operating properly. The **show frame route** command will display all PVCs that are configured on the router. The following screen print can be explained as follows: Any traffic coming into interface s0/0 with a DLCI value of 210 will be sent out on interface s0/1 on DLCI 200. Any traffic coming into interface s0/1 on DLCI 200 will be sent out on interface s0/0 on DLCI 210. Both of the DLCIs are active.

```
FrameSwitch#sh frame route
Input Intf        Input Dlci     Output Intf     Output Dlci     Status
Serial0/0         210            Serial0/1       200             active
Serial0/1         200            Serial0/0       210             active
```

The **show frame pvc** command will display the status of all DLCIs passing through the router.

```
FrameSwitch#sh frame pvc

PVC Statistics for interface Serial0/0  (Frame Relay DCE)

DLCI = 210, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 35            output pkts 28           in bytes 2650
  out bytes 2945           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:07:07, last time pvc status changed 00:06:32
  Num Pkts Switched 34

PVC Statistics for interface Serial0/1  (Frame Relay DCE)

DLCI = 200, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial0/1

  input pkts 28            output pkts 34           in bytes 2945
  out bytes 2620           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:06:33, last time pvc status changed 00:06:29
  Num Pkts Switched 28
```

We see here that DLCI 210 and DLCI 200 are active. We see several indications here that this router is acting

like a Frame Relay switch. The interface is referred to as a Frame Relay DCE. There is also an indication of number of packets switched.

Now let's examine some troubleshooting and debugging commands for a router acting like a standard Frame Relay DTE. The first thing to check is the status of the interface that is connected to the Frame Relay network. In the case of RouterA, this interface is the s0/0 interface. You can display the status of this interface with the **show int s 0/0** command.

```
RouterA#sh int s 0/0
Serial0/0  is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 192.1.1.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  LMI enq sent  16, LMI stat recvd 16, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 0  LMI type is ANSI Annex D  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 5/0, interface broadcasts 5
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters 00:02:38
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/2  (active/max active)
     Reserved Conversations 0/0  (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     27 packets input, 1095 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     26 packets output, 1024 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

The interface is shown being in the up/up state. This is the ideal condition for the interface. If the interface is in the up/down state, there is a possible problem with the encapsulation settings. Verify that the encapsulation is Frame Relay.

The **show interface** output also contains important information on the status of the Frame Relay link. We see from the following command output that the number of LMI requests sent is 16 and that it matches the number of LMI status messages received. The number of requests should always match the number of messages received. This may sometimes not be the case when the router is in the process of being configured. Use the **clear counters** command to reset the counters after the router is up and running to reset these indications.

The output of the **show interface** command also displays the LMI signaling protocol and the DTE/DCE status of the Frame Relay link. We see that this router is configured for Annex D LMI signaling and is acting like a Frame Relay DTE.

The middle portion of the **show interface** command's output displays information on the packet rates on the interface as well as information on the number of transmission and reception errored packets.

The last line of the commands output displays the status of the V.35 control leads. DCD (carrier detect) must be high in order for the interface to be in an up state.

The **show frame pvc** command appears slightly different on a Frame DTE than it does on a router acting like a Frame Relay switch.

```
RouterB#sh frame pvc
```

```
PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 210, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 34            output pkts 39           in bytes 3521
  out bytes 3166           dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  pvc create time 00:07:25, last time pvc status changed 00:06:55
```

The command's output provides important information on the number of packets sent and received on the router's interface. Congestion notification is reported for BECN, FECN, and DE bits. The last line of the output shows how long the PVC has been active for and also shows the last time that the PVC status changed.

The **show frame lmi** command is used to verify proper communication between the router and the Frame Relay switch. Some of this information is contained in the **show interface** command, but the **show frame lmi** output contains more detailed statistics.

```
RouterB#sh frame lmi

LMI Statistics for interface Serial0/0 (Frame Relay DTE) LMI TYPE = ANSI
  Invalid Unnumbered info 0          Invalid Prot Disc 0
  Invalid dummy Call Ref 0           Invalid Msg Type 0
  Invalid Status Message 0           Invalid Lock Shift 0
  Invalid Information ID 0           Invalid Report IE Len 0
  Invalid Report Request 0           Invalid Keep IE Len 0
  Num Status Enq. Sent 51368         Num Status msgs Rcvd 51120
  Num Update Status Rcvd 0           Num Status Timeouts 248
```

If the router appears to not receive LMI updates, you can use the **debug frame lmi** command to see what exchanges are taking place with the Frame Relay switch. Remember to also use the **term mon** command if you are not connected to the console port of the router.

```
RouterA#debug frame lmi
Frame Relay LMI debugging is on
Displaying all Frame Relay LMI data


Serial0/0 (out): StEnq, myseq 132, yourseen 131, DTE up
datagramstart = 0xD2A9D4, datagramsize = 14
FR encap = 0x00010308
00 75 95 01 01 01 03 02 84 83


Serial0/0 (in): Status, myseq 132
RT IE 1, length 1, type 1
KA IE 3, length 2, yourseq 132, myseq 132


Serial0/0 (out): StEnq, myseq 133, yourseen 132, DTE up
datagramstart = 0xD2A9D4, datagramsize = 14
FR encap = 0x00010308
00 75 95 01 01 00 03 02 85 84


Serial0/0 (in): Status, myseq 133
RT IE 1, length 1, type 0
KA IE 3, length 2, yourseq 133, myseq 133
PVC IE 0x7 , length 0x3 , dlci 200, status 0x2 ← Full Status Response from the
                                                  switch telling the router that
                                                  DLCI 200 is active

Serial0/0 (out): StEnq, myseq 134, yourseen 133, DTE up
datagramstart = 0xD2A9D4, datagramsize = 14
```

```
FR encap = 0x00010308
00 75 95 01 01 01 03 02 86 85

Serial0/0(in): Status, myseq 134
RT IE 1, length 1, type 1
KA IE 3, length 2, yourseq 134, myseq 134
```

The **show frame map** command displays the results of the router's inverse ARP on known DLCIs. When the router requests full status from the Frame Relay switch, the switch responds with a list of all configured DLCIs. The router will inverse ARP on each of these DLCIs and will resolve a far−end IP address from the inverse ARP.

```
RouterB#sh frame map
Serial0/0 (up): ip 192.1.1.1 dlci 210(0xD2,0x3420), dynamic,
              broadcast,, status defined, active
```

The sample Frame Relay map tells us that the far−end router's IP address on the PVC associated with the local DLCI 210 is 192.1.1.1. We also see that the map is dynamic. This means that the mapping was learned dynamically through an inverse ARP. If the map were shown as static, that would have meant that the map was configured manually in the router's configuration.

Once the far−end IP address is known, it should be checked for reachability with the ping command.

```
RouterB#ping 192.1.1.1
Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/57/60 ms
```

If the ping is not successful, you can enable frame packet debugging with the **debug frame packet** command. Remember to use the **term mon** command if you are not connected to the console port. The output of the failed ping that follows shows that the encapsulation has failed on interface s 0/0. This is usually caused by the lack of a Frame Relay map statement to the far end.

```
RouterA#debug frame packet
Frame Relay packet debugging is on

RouterA#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:

Serial0/0:Encaps failed−−no map entry link 7(IP)
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP).
Serial0/0:Encaps failed−−no map entry link 7(IP)
Serial0/0: broadcast search
Serial0/0:Encaps failed−−no map entry link 7(IP).
Success rate is 0 percent (0/5)
```

A successful ping will show the proper encapsulation when monitoring with the **debug frame packet** command. Notice in the successful ping that follows that the traffic to IP address 192.1.1.2 is being encapsulated on DLCI 200.

```
RouterA#ping 192.1.1.2

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 56/56/60 ms
Serial0/0(o): dlci 200(0x3081), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 200(0x3081), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 200(0x3081), pkt type 0x800(IP), datagramsize 104
```

```
Serial0/0(i): dlci 200(0x3081), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 200(0x3081), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 200(0x3081), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 200(0x3081), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 200(0x3081), pkt type 0x800, datagramsize 104
Serial0/0(o): dlci 200(0x3081), pkt type 0x800(IP), datagramsize 104
Serial0/0(i): dlci 200(0x3081), pkt type 0x800, datagramsize 104
```

The **show controllers** command is very useful in doing debugging. It will show you what kind of cable is connected to a particular interface. The command also tells you if the interface is seeing a clock signal.

```
RouterA#sh cont s 0/0
Interface Serial0/0
Hardware is Quicc 68360
DTE V.35 TX and RX clocks detected. ← V.35 DTE cable detected. Router sees an
                                      incoming clock
```

Conclusion

Frame Relay is the most popular wide area networking protocol. We have seen that there are several reasons for the wide acceptance of Frame Relay in the networking world.

The Cisco IOS features extensive support for Frame Relay networking. This chapter contained eight labs that explore Cisco's support for Frame Relay networks.

Chapter 5: Asynchronous Transfer Mode (ATM)

Overview

Topics Covered in This Chapter

- ATM technology overview
- Cisco 4500 ATM configuration
- Cisco ATM loopbacks
- ATM lane
- ATM troubleshooting

Introduction

Asynchronous Transfer Mode (ATM) is an exciting wide area network technology that is the focus of much attention during the last few years. Cisco has extensive support for ATM networking. This chapter will examine ATM technology in detail, and will then explore how the Cisco IOS supports ATM with three hands−on labs.

ATM Overview

Asynchronous Transfer Mode (ATM) is a standard for cell switching where traffic for multiple types of services such as data, voice, and video are transmitted in fixed−size 53−byte cells. As shown in Figure 5−1 and Figure 5−2, ATM differs from other networking protocols in that ATM is the only protocol that keeps the same cell format and protocol end to end.

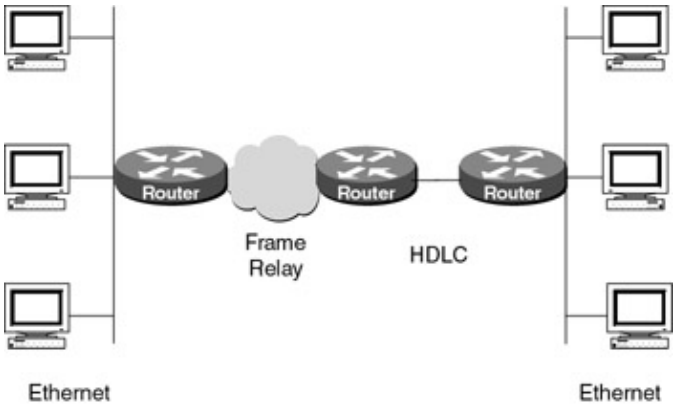


Figure 5−1: Traditional LAN/WAN network

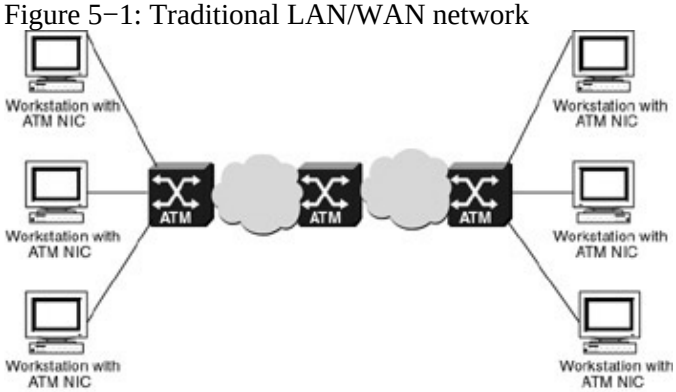


Figure 5−2: ATM network

ATM also differs from other protocols in that it is a switched protocol. All ATM end−user devices are directly connected to a switch.

In Figure 5−1 we see a traditional LAN/WAN architecture. Ethernet−connected workstations reside on shared Ethernet LANs and are connected to a router. Two routers are connected together via a frame relay link and two routers are connected together via an HDLC link. With this architecture, the datalink encapsulation of the traffic traveling between the two Ethernet LANs changes at every router hop. Traffic starts out on an Ethernet LAN with 802.3 encapsulation. It leaves the first router encapsulated in frame relay. Data leaves the second router encapsulated in HDLC. Finally, data leaves the third router encapsulated once again in 802.3 Ethernet. This traditional LAN/WAN architecture has the disadvantage of adding overhead to perform the hop−by−hop encapsulation/deencapsulation/encapsulation as traffic passes from one datalink format to the next.

Figure 5−2 shows an ATM network. In this scenario, data leaves the workstations as ATM traffic and travels across the entire network using the same ATM format. No overhead is lost in any kind of encapsulation/deencapsulation at each hop.

ATM is a cell−switching and multiplexing technology that combines the benefits of circuit switching (guaranteed capacity and constant transmission delay) with those of packet switching (flexibility and efficiency for intermittent traffic).

ATM is referred to as being *asynchronous* because data can be sent in any available time slot, as opposed to TDM where each user is assigned to a specific time slot. If a TDM user has no data to send, that users time slot will remain empty. The source of the traffic is identified by addressing information in the header of each ATM cell.

ATM networks are connection−oriented, meaning that a virtual circuit must be set up across the ATM network prior to any data transfer. These virtual circuits can be either permanent or switched.

ATM Protocol Stack

As shown in Figure 5−3, ATM does not use the standard seven−layer OSI protocol reference model; instead, it uses its own protocol stack.

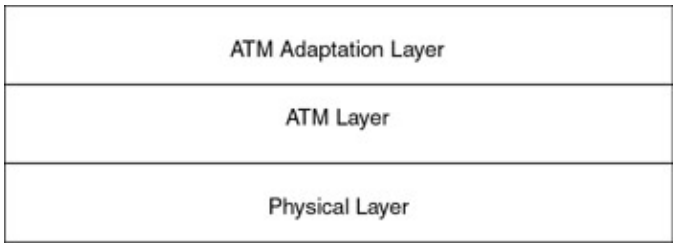


Figure 5−3: ATM protocol stack

The ATM protocol stack is composed of the following components:

- **Physical layer:** The ATM physical layer manages the medium−dependent transmission:
    - ♦ Converting the 53−byte ATM cell into an outgoing bitstream and converting an incoming bitstream to a 53−byte ATM cell
    - ♦ Keeping track of the ATM cell boundaries
    - ♦ Electrical and physical specifications
- **ATM layer:** The ATM layer is responsible for establishing connections and passing cells through the ATM network. To do this, it uses information in the header of each ATM cell. It performs such tasks as:
    - ♦ VPI and VCI switching
    - ♦ Cell multiplexing and demultiplexing
    - ♦ Flow control
    - ♦ Header error control
    - ♦ Cell loss priority (CLP) processing
    - ♦ QOS support

- **ATM adaptation layer (AAL):** The AAL is responsible for isolating higher−layer protocols from the details of the ATM processes. It performs such tasks as:

    - ♦ Segmentation and reassembly of the data
    - ♦ Payload error control
    - ♦ End−to−end timing

ATM Cell Basic Format

ATM transfers information in fixed−size units called cells. As shown in Figure 5−4, each cell consists of 53 bytes. The first 5 bytes contain cell header information, and the remaining 48 bytes contain the user information or payload. Small, fixed−size cells are advantageous in that they prevent larger cells from causing large latency delays. A fixed−size cell can also be switched in hardware, making ATM switching extremely fast. One of ATM's advantages is being able to transmit isochronous traffic, meaning traffic with a well−defined delay between successive cells. Isochronous traffic is important for real−time applications such as video and voice.

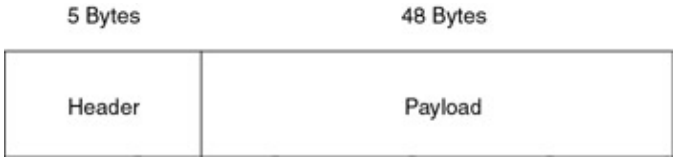


Figure 5−4: ATM cell format

ATM Cell Header

Figure 5−5 shows the structure of an ATM cell header. The 5−byte ATM cell header contains information needed to switch the ATM cell to its next destination. The ATM cell header is examined and updated on a switch−by−switch basis.

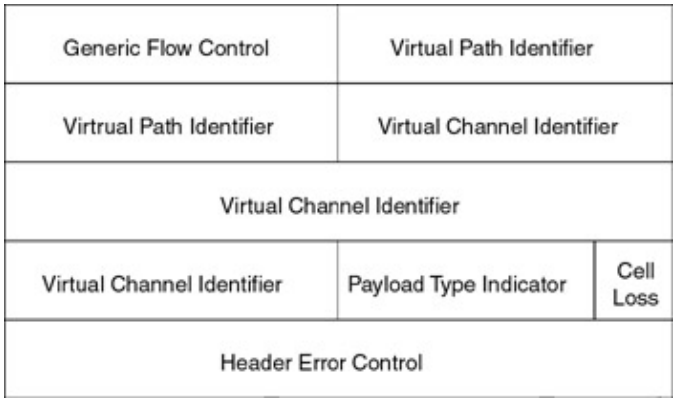


Figure 5−5: ATM UNI header

The fields in the ATM cell header can be explained as follows:

- **Generic Flow Control (GFC):** This is a 4−bit field that is typically not used.
- **Virtual Path Identifier (VPI):** This is a 8−bit field that, along with the VCI, identifies the next destination of a cell as it passes through an ATM network towards its final destination.
- **Virtual Channel Identifier (VCI):** This 16−bit field, along with the VPI, identifies the next destination of a cell as it passes through an ATM network towards its final destination.
- **Payload Type Identifier (PT):** This 3−bit field indicates whether the cell contains user data or control data. It is also used to indicate congestion.
- **Cell Loss Priority (CLP):** This 1−bit field indicates whether the cell should be discarded if it encounters extreme congestion as it moves through the network.
- **Header Error Control (HEC):** The Header Error Control (HEC) field calculates a checksum on the header of the ATM cell only. There is no checksum on the ATM cell payload. The HEC is able to detect and correct single−bit errors. Multiple−bit errors are detected and the ATM cell is discarded.

Both the VPI and VCI value can be changed every time the cell passes through a switch. As with frame relay, there are several preassigned VPI/VCI pairs that are reserved for uses such as signaling, broadcasting, and operations and maintenance use.

There are two types of ATM cell header formats: User Network Interface (UNI) and Network Network Interface (NNI).

The UNI header, shown in Figure 5−5, is used for communication between ATM end user devices and ATM switches.

The NNI header, shown in Figure 5−6, is used for communication between ATM switches. The ATM NNI cell header differs from the ATM UNI cell header in several ways:

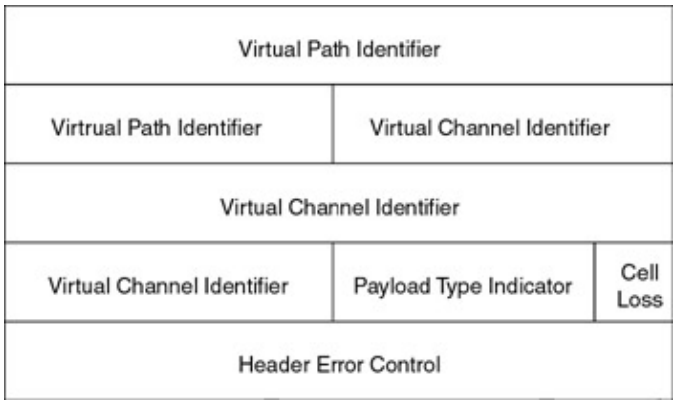


Figure 5−6: ATM NNI header

- The NNI header does not include the Generic Flow Control (GFC) field.
- The NNI header has a Virtual Path Identifier (VPI) field that occupies the first 12 bits, allowing for larger trunks between public ATM switches.

ATM Addressing

An ATM circuit is based on a connection−oriented, end−to−end link. Addressing information is contained in the ATM cell header in the form of a VPI and a VCI.

As shown in Figure 5−7, a virtual channel can be thought of as a transport of cells. A virtual path is a bundle of virtual channels.

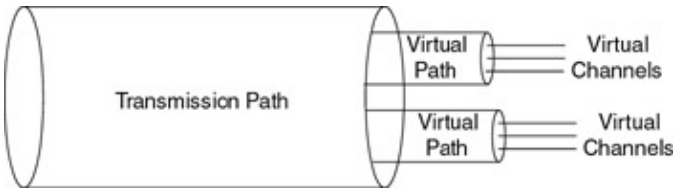


Figure 5−7: VPI, VCI, and transmission paths

Components of an ATM Network

ATM networks are made up of ATM switches and ATM end−user equipment.

An ATM switch is responsible for transmitting ATM cells through an ATM network. The ATM switch takes an incoming cell and reads and updates the cell header information — for example, the VPI and VCI values may change. It then switches the cell towards its final destination.

An ATM end−user device connects to an ATM switch and is responsible for generating and sending ATM traffic into the network.

As shown in Figure 5−8, an ATM network consists of a set of ATM switches interconnected by point−to−point ATM links, often referred to as trunks. ATM switches support two primary types of interfaces: User Network Interface (UNI) and Network−Network Interface (NNI). The UNI connects ATM end systems (such as hosts and routers) to an ATM switch. The NNI connects two ATM switches.

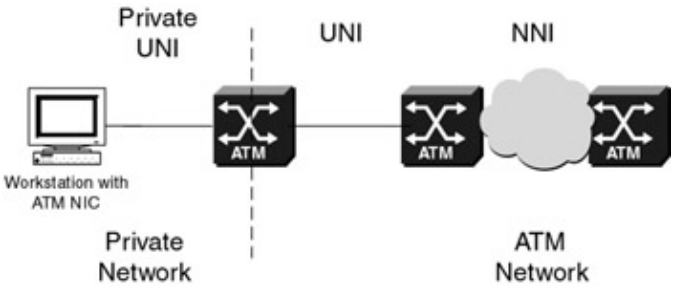


Figure 5−8: ATM UNI and NNI interfaces

ATM Physical Interfaces

ATM is supported on a variety of physical interfaces from T1 to SONET speeds. Although ATM can function over a low−speed circuit such as a T1, it is not very efficient at these speeds. This is due to the high overhead that the combination of the T1 framing and the ATM cell overhead imposes. Some popular ATM interfaces include:

- T1 (1.544Mbps)
- 25Mbps
- DS−3 (44.7Mbps)
- OC−3 (155Mbps)
- OC−12 (622Mbps)
- OC−48 (2448Mbps)

ATM Call Types

ATM supports both permanent virtual circuits (PVCs) and switched virtual circuits (SVCs).

A PVC is similar to a frame relay PVC in that it defines an end−to−end circuit whose path can be rerouted in the event of an intermediate node failure.

An SVC is a switched ATM circuit, which is established, maintained, and released by signaling between the ATM end−user device and the ATM switch. This signaling occurs over a dedicated VPI/VCI pair. UNI signaling requests are carried in a well−known default connection: VPI = 0, VPI = 5. When an ATM device wants to establish a connection with another ATM device, it sends a signaling request packet to its directly connected ATM switch. This request contains the ATM address of the desired ATM endpoint, as well as any QOS parameters required for the connection. The ATM UNI specification is based on the Q.2931 public network signaling protocol. The Q.2931 signaling standard is very similar to the ISDN Q.931 signaling standard and contains commands such as Setup, Call Proceeding, Connect, and Release, which are used to establish and tear down an ATM connection.

ATM Switching Operation

The basic operation of an ATM switch is shown in Figure 5−9. An ATM cell is received across a link on a known VCI/VPI value. The switch looks up the connection value in a local translation table to determine the outgoing port (or ports) of the connection and the new VPI/VCI value of the connection on that link. The switch then retransmits the cell on that outgoing link with the appropriate VPI and VCI. Because all VPIs and VCIs have only local significance across a particular link, these values are remapped as necessary at each switch.

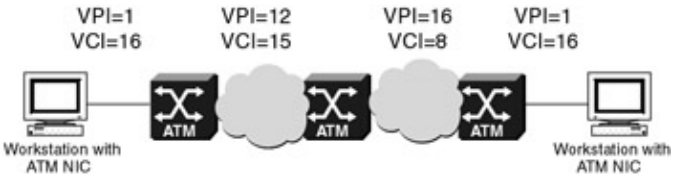


Figure 5−9: ATM switching example

Two types of ATM connections exist: virtual paths, which are identified by virtual path identifiers (VPIs), and virtual channels, which are identified by the combination of a VPI and a virtual channel identifier (VCI). A virtual path is a bundle of virtual channels, all of which are switched transparently across the ATM network on the basis of the common VPI. All VCIs and VPIs, however, have only local significance across a particular link and are remapped, as appropriate, at each switch.

ATM Classes of Service

One of ATM's major advantages is its ability to handle multiple types of traffic, referred to as classes of service, on a single network. The ATM adaptation layer is responsible for mapping these classes of services into actual ATM traffic. Let's look at each of these classes of service and their characteristics:

Constant bit rate (CBR): CBR specifies a connection−oriented synchronous traffic stream. This class of service emulates traditional leased line circuits and appears to the end user as a point−to−point circuit. CBR requires an end−to−end timing relationship.

Variable bit rate real time (VBR−RT): VBR−RT supports real−time connection−oriented synchronous traffic. VBR−RT requires an end−to−end timing relationship.

Available bit rate (ABR): ABR supports variable bit rate asynchronous traffic streams for services such as frame relay and X.25 over ATM. ABR does not require an end−to−end timing relationship.

Unspecified bit rate (UBR): UBR supports connectionless packet data. No guarantees are made as to loss, delay, or available user bandwidth. UBR does not require an end−to−end timing relationship.

ATM Quality of Service (QOS)

ATM has extensive quality of service (QOS) support. A QOS contract can specify values for such items as peak bandwidth, average sustained bandwidth, cell delay variation, and burst size, among others. These QOS parameters are guaranteed through the use of traffic shaping. Traffic shaping uses queues in the ATM switch input and output buffers to limit the data rate and control bursty data so that traffic profile will closely match the agreed−upon QOS parameters. The ATM switch can set the cell loss priority (CLP) bit for those cells that are outside of the traffic contract. This makes the cell discard eligible during periods of congestion.

ATM with a Non−ATM Device

Routers without native ATM interfaces can still take advantage of ATM networking. As specified in RFC 1483, a serial interface on a Cisco router can be configured for ATM−DXI encapsulation.

RFC 1483 describes two methods of transporting multiprotocol traffic over an ATM network. The first method allows multiplexing of multiple protocols over a single PVC. The second method can use a different PVC to carry each protocol. Cisco supports both methods of DXI encapsulation. Cisco can transport the following protocols over a DXI encapsulated interface:

- Apollo Domain
- AppleTalk
- Banyan VINES
- DECnet
- IP
- Novell IPX

- ISO CLNS
- XNS traffic

As shown in Figure 5−10, ATM−DXI works by connecting the DXI encapsulated serial interface of the router to an ATM DSU, referred to as an ADSU. At the ADSU, the DXI header is stripped off and the data is segmented into cells for transport over an ATM network.

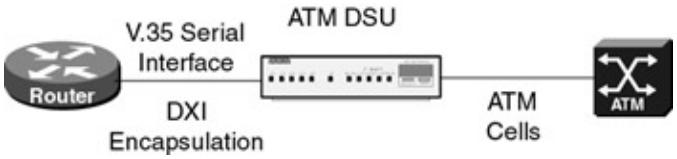


Figure 5−10: ATM ADSU example

ATM LANE

LAN emulation (LANE) was defined by the ATM Forum as a technology to interconnect legacy LANs using ATM attached devices. An emulated LAN (ELAN) appears to be one bridged segment.

When talking about LANE, it is important to understand the terminology. LANE defines a LEC, LECS, and a BUS. The LAN emulation client (LEC) is any system that supports LANE. The client emulates an interface to the LAN to the higher−level protocols. It performs data forwarding, address resolution, and registration of MAC addresses with the LANE server and communicates with other LECs via ATM virtual channel connections (VCCs).

The LAN emulation configuration server (LECS) is responsible for maintaining the database of ELANs and the LES that controls the ELAN. It accepts queries from the LECs and responds with the ATM address of the LES.

The LAN emulation server (LES) is a central control at which all LECS attach. Each LEC maintains a control direct VCC to the LES. This VCC is used to forward registration and control information. The LES maintains a point−to−multipoint VCC, known as the control distribute VCC, to all LECs. The control distribute VCC is used only to forward control information.

The broadcast and unknown server (BUS) acts as a central point for distributing broadcasts and multicasts. Since ATM is a point−to−point technology, it lacks the capability to support "any−to−any" or "broadcast" type traffic. LANE solves this problem by centralizing the broadcast support in the BUS. Each LEC must set up a multicast send VCC to the BUS. The BUS then adds the LEC as a leaf to its point−to−multipoint VCC (known as the multicast forward VCC). The BUS also acts as a multicast server. LANE is defined on ATM adaptation layer 5 (AAL5), which specifies a simple trailer to be appended to a frame before it is broken into ATM cells. The problem is that there is no way to differentiate between ATM cells from different senders when multiplexed on a virtual channel. It is assumed that cells received will be in sequence, and when the end of message (EOM) cell arrives, you should just have to reassemble all of the cells that have already arrived.

The BUS takes the sequence of cells on each multicast send VCC and reassembles them into frames. When a full frame is received, it is queued for sending to all of the LECs on the multicast forward VCC. This way, all the cells from a particular data frame are guaranteed to be sent in order and not interleaved with cells from any other data frames on the point−to−multipoint VCC.

An ELAN provides layer 2 communication between all users on the ELAN. An ELAN can be thought of as a VLAN or a single broadcast domain. When a LEC comes up, it must first find the LECS to discover which ELAN to join. The LEC queries the ATM switch via the Interim Local Management Interface (ILMI). The switch contains the ATM address of the LECS. The LEC then can use UNI signaling to contact the LECS.

Once the LEC has determined the ATM address of the LECs, it creates a signaling packet using that address. It signals the LECs to determine which ELAN it is a member of and to determine the ATM address of the LES that serves that ELAN.

After the LEC has discovered the ATM address of the desired LES, it drops the connection to the LECS and creates a signaling packet with the ATM address of the LES. Once it is connected to the LES, it registers its MAC and ATM address for the ELAN. This information is maintained so that no two LECs can register the same MAC or ATM addresses. The LES adds the LEC as a leaf of its point−to−multipoint control distribute VCC. Finally, the LES issues the LEC a successful LE_JOIN_RESPONSE that contains a LANE client ID (LECID), which is an identifier that is unique to the new client. This ID is used by the LEC to filter its own broadcasts from the BUS.

After the LEC has successfully joined the LES, its first task is to find the ATM address of the BUS and join the broadcast group. To do this, it sends a special LE_ARP packet on the control direct VCC to the LES. The LES recognizes that the LEC is looking for the BUS, responds with the ATM address of the BUS, and forwards that response on the control distribute VCC.

When the LEC has the ATM address of the BUS, its next action is to create a signaling packet with that address and signal a multicast send VCC. Upon receipt of the signaling request, the BUS adds the LEC as a leaf on its point−to−multipoint multicast forward VCC. At this time, the LEC has become a member of the ELAN.

When a LEC has a data packet to send to an unknown destination, it issues an ARP request to the LES using the control direct VCC. The LES forwards the request on the control distribute VCC to all LEC stations. In parallel, the Unicast data packets are sent to the BUS, to be forwarded to all endpoints. Unicast packets continue using the BUS until the ARP request has been resolved.

The ARP response contains the ATM address of the LEC that is associated with the MAC address being sought. The LEC now signals the other LEC directly and sets up a data direct VCC that will be used for Unicast data between the LECs.

Cisco ATM Capabilities

Cisco sells both ATM Stratacom switches and ATM−enabled routers. Cisco routers from the 2600 up to the 12000 can support a variety of ATM interfaces.

Commands Discussed in This Chapter

- **atm pvc** *vcd vpi vci aal−encap*
- *protocol protocol−address* **atm vc** *vcd* **[broadcast]**
- **debug atm packet**
- **lane client {ethernet | tokenring}** [*elan−name*]
- **lane config database** *database−name*
- **lane database** *database−name*
- **lane server−bus** {**ethernet** | **tokenring**} *elan−name* [**elan−id** *id*]
- **loopback diagnostic**
- **loopback line**
- **map−group** *name*
- **map−list** *name*
- **show atm map**
- **show atm traffic**
- **show atm vc [***vcd***]**
- **show interface atm** number
- **show lane** [**interface atm** *slot*/*port*[**.***subinterface−number*] | name *elan−name*] [brief]

Definitions

atm pvc: This interface configuration command creates a permanent virtual circuit (PVC) on the ATM interface of a Cisco router. The *vcd* number describes a unique VPI, VCI pair to the router.

atm vc: This map−list configuration command defines a map statement for an ATM PVC. This command is used with the **map−list** command.

debug atm packet: This debug command will display all ATM cells going into and out of the router.

lane client: This interface configuration command is used to activate a LANE client on the specified subinterface.

lane config database: This interface configuration command is used to associate a named configuration database with the configuration server on the selected ATM interface.

lane database: This global configuration command is used to create a named configuration database that can be associated with a configuration server.

lane server bus: This interface configuration command enables a LANE server and a broadcast−and−unknown server (BUS) on the specified subinterface with the specified ELAN ID.

loopback diagnostic: This interface command places the ATM interface in a loopback mode where all ATM cells leaving an interface are sent back towards the same interface.

loopback line: This interface command places the ATM interface in a loopback mode where all ATM cells coming into an interface are sent back towards the interface that sent the traffic.

map−group: This interface configuration command associates an ATM map list to an ATM interface.

map−list: This global configuration command defines a map statement for a PVC or an SVC.

show atm map: This exec command will cause all configured ATM static maps to be displayed.

show atm traffic: This exec command will display incoming and outgoing traffic information for all ATM interfaces on a router.

show atm vc: This exec command will display information on all active virtual circuits (PVCs and SVCs). When issued with the optional *vcd* argument, this command will display traffic information for individual virtual circuits (PVCs and SVCs).

show interface: This exec command displays information on the status of the ATM interface. Various status information such as total traffic transmitted and received on the interface are displayed.

show lane: This exec command will display detailed information for all the configured LANE components on the specified interfaces.

IOS Requirements

The labs in the chapter were done with IOS 11.2. Most ATM commands discussed in this chapter were introduced in IOS 10.0.

Lab #20: ATM Configuration on a Cisco 4500

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having an ATM OC−3 interface, running IOS 11.2
- Two Multimode fiber cables
- A Cisco rolled cable for console port connection to the routers
- An ATM switch. (This lab can also be done by connecting the two ATM interfaces directly together using a single fiber cable.)

Configuration Overview

This lab will demonstrate a basic ATM configuration on a Cisco 4500 OC−3 multi−mode 155Mbps ATM interface. This lab can be performed in one of two ways. As shown in Figure 5−11, if an ATM switch is available, the two routers can be connected to the ATM switch. This is the preferred method since this is how ATM switching works in the real world. As shown in Figure 5−12, if an ATM switch is not available, the two routers can be directly connected with a multimode fiber.

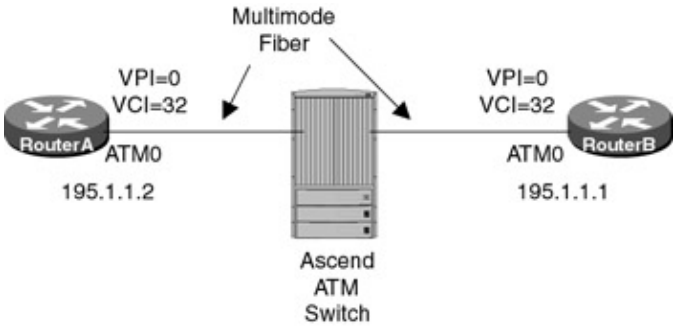


Figure 5−11: Basic ATM configuration with an ATM switch

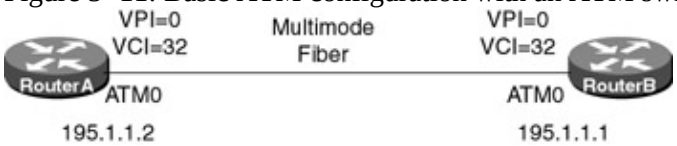


Figure 5−12: Basic ATM configuration without an ATM switch

The IP addressing scheme is as shown in Figures 5−11 and 5−12.

Note The configurations in this lab will not change whether you are directly connected between the two routers or are connected through an ATM switch.

Router Configuration

The configurations for the router in this example are as follows. Key ATM commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
enable password cisco
!
no ip domain−lookup
```

```
!
interface ATM0
 ip address 195.1.1.2 255.255.255.0
 atm pvc 1 0 32 aal5nlpid ← Define an ATM PVC using the VCD, VPI, and VCI
 map-group 1 ← Associate this interface with map-list 1
!
no ip classless
!
map-list 1
 ip 195.1.1.1 atm-vc 1 broadcast ← This statement maps the layer 3 next-hop
                                    address to the layer 2 ATM PVC
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
RouterB#sh run
Building configuration...

Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface ATM0
 ip address 195.1.1.1 255.255.255.0
 atm pvc 1 0 32 aal5nlpid ← Define an ATM PVC using the VCD, VPI, VCI
 map-group 1 ← Associate this interface with map-list 1
!
no ip classless
!
map-list 1
 ip 195.1.1.2 atm-vc 1 broadcast ← This statement maps the layer 3
next-hop address to the layer 2 ATM PVC
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

We see that in both routers' configurations we have to manually define what VPI/VCI combination will be present on the ATM interface. In the case of this lab, we have defined VPI = 0/VCI = 32 on the ATM interface of each router. A static map must also be created. This static map defines the IP address at the far end of the ATM PVC.

Monitoring and Testing the Configuration

Let's start by connecting to RouterA and examining the status of the ATM interface with the **show interface atm 0** command. We see that the interface is an up/up condition.

```
RouterA#show interface atm 0
```

```
ATM0 is up, line protocol is up ← Interface status
  Hardware is ATMizer BX-50
  Internet address is 195.1.1.2/24
  MTU 4470 bytes, sub MTU 4470, BW 156250 Kbit, DLY 100 usec, rely 10/255, load
1/255
  Encapsulation ATM, loopback not set, keepalive set (10 sec)
  Encapsulation(s): AAL5 AAL3/4, PVC mode
  1024 maximum active VCs, 1024 VCs per VP, 1 current VCCs
  VC idle disconnect time: 300 seconds
  Last input 00:09:56, output 00:09:56, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     25 packets input, 2650 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     25 packets output, 2650 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

The **show controller atm 0** command can also be used to verify that the ATM network module is installed and properly functioning.

```
RouterA#show controller atm 0
ATM Unit 0, Slot 2, Type ATMizer BX-50, Hardware Version 1
  ATM Xilinx Code, Version 2, ATMizer Firmware, Version 3.0
  Public SRAM 65536 bytes, Private SRAM 524288 bytes, I/O Base Addr 0x3C200000
  PLIM Type OC-3 Multi-Mode Fiber, Version 3
  Network Transmit Clock
  NIM IS Operational, Configuration OK
  DMA Read 12, DMA Write 12
```

Now connect to RouterB and verify that the ATM interface is in an up/up state.

```
RouterB#show interface atm 0
ATM0 is up, line protocol is up ← Interface status
  Hardware is ATMizer BX-50
  Internet address is 195.1.1.1/24
  MTU 4470 bytes, sub MTU 4470, BW 155520 Kbit, DLY 100 usec, rely 15/255, load
1/255
  Encapsulation ATM, loopback not set, keepalive not supported
  Encapsulation(s): AAL5 AAL3/4, PVC mode
  1024 maximum active VCs, 1024 VCs per VP, 1 current VCCs
  VC idle disconnect time: 300 seconds
  Last input 00:00:43, output 00:00:43, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     30 packets input, 3180 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     30 packets output, 3180 bytes, 0 underruns
     0 output errors, 0 collisions, 5 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Reconnect to RouterA. Use the **show atm vc** command to display the status of all virtual circuits that are configured on the router. We see that our PVC (VPI = 0/VCI = 32) is active on interface ATM0.

```
RouterA#show atm vc
                          AAL /         Peak   Avg.  Burst
Interface     VCD   VPI   VCI Type  Encapsulation  Kbps   Kbps  Cells Status
```

```
ATM0            1     0    32  PVC  AAL5-NLPID    155000 155000    94 ACTIVE
```

Entering the **show atm vc** command with the virtual circuit descriptor for the PVC will show detailed information on the PVC. Type **show atm vc 1** to display detailed information on our PVC.

```
RouterA#sh atm vc 1
ATM0: VCD: 1, VPI: 0, VCI: 32, etype:0x2, AAL5 - NLPID, Flags: 0xC31
PeakRate: 155000, Average Rate: 155000, Burst Cells: 94, VCmode: 0x1
OAM DISABLED, InARP DISABLED
InPkts: 5, OutPkts: 5, InBytes: 530, OutBytes: 530
InPRoc: 5, OutPRoc: 5, Broadcasts: 0
InFast: 0, OutFast: 0, InAS: 0, OutAS: 0
OAM F5 cells sent: 0, OAM cells received: 0
Status: ACTIVE
```

The **show atm map** command displays any ATM maps that have been configured on the router. We see that we have one ATM map. This map is to IP address 195.1.1.1 and is associated with map list 1, which points to IP address 195.1.1.1 at the far end of the ATM PVC.

```
RouterA#show atm map
Map list 1 : PERMANENT
ip 195.1.1.1 maps to VC 1 ← VC #1 is our PVC of VPI = 0/VCI = 32
        , broadcast
```

The **show atm traffic** command will show how many packets have been sent and received on each ATM interface.

```
RouterA#sh atm traffic
25 Input packets
25 Output packets
0 Broadcast packets
0 Packets received on non-existent VC
0 Packets attempted to send on non-existent VC
0 OAM cells received
0 OAM cells sent
```

Verify that you can ping RouterB at IP address 195.1.1.1.

```
RouterA#ping 195.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 195.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

After the ping is completed, type the **show atm traffic** command again. Notice that there are now 30 input and output packets.

```
RouterA#show atm traffic
30 Input packets ← 5 more packets than before the ping
30 Output packets
0 Broadcast packets
0 Packets received on non-existent VC
0 Packets attempted to send on non-existent VC
0 OAM cells received
0 OAM cells sent
```

Now connect to RouterB and type the **show atm vc** command. Verify that our PVC is active on RouterB.

```
RouterB#show atm vc
                                    AAL /          Peak   Avg.  Burst
Interface     VCD   VPI   VCI Type  Encapsulation  Kbps   Kbps  Cells Status
0               1     0    32  PVC  AAL5-NLPID    155000 155000    94 ACTIVE
```

Enable ATM packet debugging with the **debug atm packet** command.

```
RouterA#debug atm packet
ATM packets debugging is on
Displaying all ATM packets
```

Now ping RouterA at IP address 195.1.1.2. Each ping packet that is sent out of RouterB will be marked with an (O) and each ping packet that is received back from RouterA will be marked with an (I).

```
RouterB#ping 195.1.1.2

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 195.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 1/1/1 ms

ATM0(O): ← The (O) indicates that the packet is being sent out of RouterB

This is our vcd number we defined for this PVC
↓
VCD:0x1 DM:0x100 NLPID:0x03CC Length:0x6A
4500 0064 0046 0000 FF01 334D C301 0101 C301 0102 0800 3924 0114 05EC 0000
0000 0096 3D90 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD

ATM0(I): ← The (I) indicates that the packet is being received from RouterA

This is our vcd number we defined for this PVC
↓
VCD:0x1 Type:0x2 NLPID:0x03CC Length:0x6A
4500 0064 0046 0000 FF01 334D C301 0102 C301 0101 0000 4124 0114 05EC 0000
0000 0096 3D90 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
```

The output of the **debug atm packet** command has been truncated here. The actual output would contain five pairs of output and input packet traces.

Lab #21: ATM Loopbacks on a Cisco 4500

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having an ATM OC−3 interface running IOS 11.2
- 2 multimode fiber cables
- A Cisco rolled cable for console port connection to the routers
- An ATM switch. (This lab can also be done by connecting the two ATM interfaces together back to back.)

Configuration Overview

This lab will demonstrate the loopback capabilities of the ATM OC−3 interface on a Cisco 4500.

This lab can be performed in one of two ways. As shown in Figure 5−13, if an ATM switch is available, the two routers can be connected to the ATM switch. This is the preferred method since this is how ATM switching works in the real world. As shown in Figure 5−14, if an ATM switch is not available, the two routers can be directly connected with a multimode fiber.

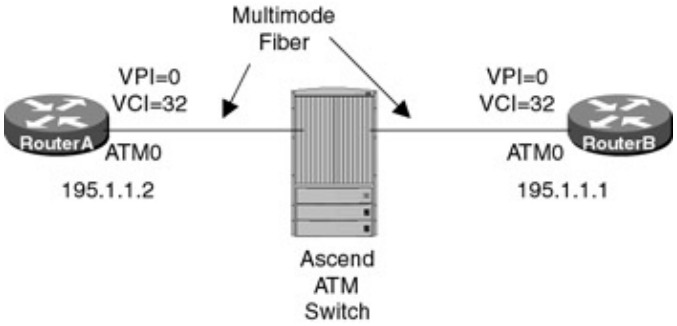


Figure 5−13: ATM loopback configuration with an ATM switch

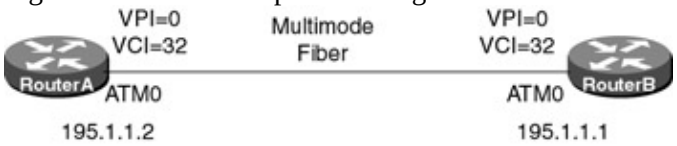


Figure 5−14: ATM loopback configuration without an ATM switch

Loopbacks are an important element of network testability. They provide well−defined test points that can be used when a circuit is down and troubleshooting is required to determine the exact point of failure.

The ATM interface on the Cisco 4500 has two loopback modes. Figure 5−15 shows the loopback diagnostic mode. You enter this loopback mode by entering the **loopback diagnostic** command under the ATM interface of the router that will be put into a loop. This loopback mode will cause any traffic being sent out of the router's ATM interface to be sent back to the router that generated the traffic. The router interface will go from an up/up state to an up/up (looped) state.

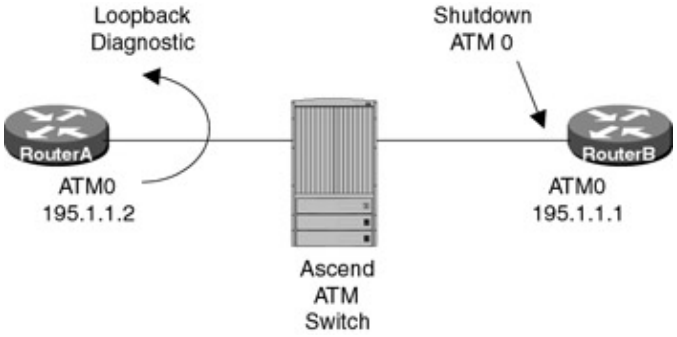


Figure 5−15: Loopback diagnostic

Figure 5−16 shows the loopback line mode. You enter this loopback mode by entering the **loopback line** command under the ATM interface of the router that will be put into a loop. This loopback mode causes any traffic that is coming into the router to be sent back to the network.

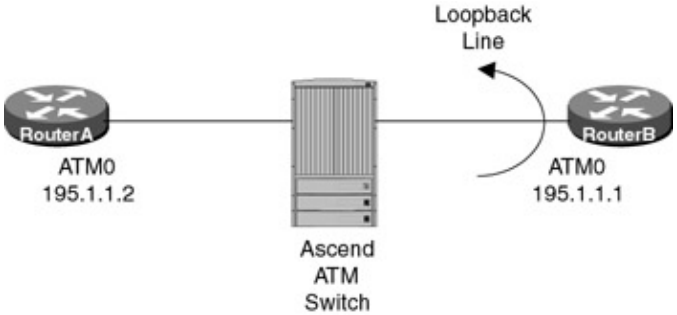


Figure 5−16: Loopback line

Note The configurations in this lab will not change whether you are directly connected between the two routers or are connected through an ATM switch.

Router Configuration

The starting configurations for the routers in this example are as follows. Key ATM commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
no ip domain-lookup
!
interface ATM0
 ip address 195.1.1.2 255.255.255.0
 atm pvc 1 0 32 aal5nlpid ← Define an ATM PVC using the VCD, VPI, and VCI
 map-group 1 ← Associate this interface with map-list 1
!
no ip classless
!
map-list 1
 ip 195.1.1.1 atm-vc 1 broadcast ← This statement maps the layer 3 next-hop
                                    address to the layer 2 ATM PVC
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface ATM0
 ip address 195.1.1.1 255.255.255.0
 atm pvc 1 0 32 aal5nlpid ← Define an ATM PVC using the VCD, VPI, and VCI
 map-group 1 ← Associate this interface with map-list 1
!
no ip classless
!
map-list 1
 ip 195.1.1.2 atm-vc 1 broadcast ← This statement maps the layer 3 next-hop
address to the layer 2 ATM PVC
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

We see that in both configurations we have to manually define what VPI/VCI combination will be present on the ATM interface. In the case of this lab, we have defined VPI = 0/VCI = 32 on the ATM interface of each

router. A static map must also be created. This static map defines the IP address at the far end of the ATM PVC.

Monitoring and Testing the Configuration Loopback Diagnostic

Let's start by connecting to RouterA and verifying that the ATM interface is in an up/up state by typing the **show interface atm 0** command.

```
RouterA#show interface atm 0
ATM0 is up, line protocol is up ← Interface status
  Hardware is ATMizer BX−50
  Internet address is 195.1.1.2/24
  MTU 4470 bytes, sub MTU 4470, BW 156250 Kbit, DLY 100 usec, rely 164/255, load
 1/255
  Encapsulation ATM, loopback not set, keepalive set (10 sec)
  Encapsulation(s): AAL5 AAL3/4, PVC mode
  1024 maximum active VCs, 1024 VCs per VP, 1 current VCCs
  VC idle disconnect time: 300 seconds
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 1 packets/sec
  5 minute output rate 0 bits/sec, 1 packets/sec
     193 packets input, 15058 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     198 packets output, 15588 bytes, 0 underruns
     0 output errors, 0 collisions, 3 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Now connect to RouterB and verify that its interface is in an up/up state by typing the **show interface atm 0** command.

```
RouterB#show interface atm 0
ATM0 is up, line protocol is up ← Interface status
  Hardware is ATMizer BX−50
  Internet address is 195.1.1.1/24
  MTU 4470 bytes, sub MTU 4470, BW 155520 Kbit, DLY 100 usec, rely 15/255, load
1/255
  Encapsulation ATM, loopback not set, keepalive not supported
  Encapsulation(s): AAL5 AAL3/4, PVC mode
  1024 maximum active VCs, 1024 VCs per VP, 1 current VCCs
  VC idle disconnect time: 300 seconds
  Last input 00:00:43, output 00:00:43, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     30 packets input, 3180 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     30 packets output, 3180 bytes, 0 underruns
     0 output errors, 0 collisions, 5 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Let's reconnect to RouterA and ping RouterB at IP address 195.1.1.1. The ping should be 100−percent successful.

```
RouterA#ping 195.1.1.1

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 195.1.1.1, timeout is 2 seconds:
```

```
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

Now connect to RouterB. We are going to disable the ATM interface on RouterB by entering the **shutdown** command under the ATM 0 interface.

```
RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#int atm 0
RouterB(config-if)#shutdown
RouterB(config)#exit

After entering the shutdown command on the ATM interface, the interface will change to an
administratively down state
      ↓
%LINEPROTO-5-UPDOWN: Line protocol on Interface ATM0, changed state to down
%LINK-5-CHANGED: Interface ATM0, changed state to administratively down
```

Verify that the ATM interface on RouterB is in a down state by typing the **show interface atm 0** command.

```
RouterB#show interface atm 0
ATM0 is administratively down, line protocol is down ← The interface has been
                                                       shutdown
  Hardware is ATMizer BX-50
  Internet address is 195.1.1.1/24
  MTU 4470 bytes, sub MTU 4470, BW 155520 Kbit, DLY 100 usec, rely 255/255, load
 1/255
  Encapsulation ATM, loopback not set, keepalive not supported
  Encapsulation(s): AAL5 AAL3/4, PVC mode
  1024 maximum active VCs, 1024 VCs per VP, 0 current VCCs
  VC idle disconnect time: 300 seconds
  Last input 00:00:31, output 00:00:31, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     52784 packets input, 2750708 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     52794 packets output, 2751768 bytes, 0 underruns
     0 output errors, 0 collisions, 8 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Now reconnect to RouterA. Enable ATM packet debugging with the **debug atm packet** command.

```
RouterA#debug atm packet
ATM packets debugging is on
Displaying all ATM packets
```

Now ping RouterB at IP address 195.1.1.1. The ping will fail since the ATM interface on RouterB has been shut down.

```
RouterA#ping 195.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 195.1.1.1, timeout is 2 seconds:

ATM0(O): ← There will be five output (O) ping packets sent out from RouterA.
Since the ping fails, there will not be any returned packets
VCD:0x1 DM:0x100 NLPID:0x03CC Length:0x6A
4500 0064 0064 0000 FF01 332F C301 0102 C301 0101 0800 CEA5 0000 0B0B 0000
0000 0041 A458 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
.
```

```
.
The other three ping packets are not shown here, they are identical to the others.
.
.
ATM0(O): ← Last of 5 output ping packets
VCD:0x1 DM:0x100 NLPID:0x03CC Length:0x6A
4500 0064 0068 0000 FF01 332B C301 0102 C301 0101 0800 AF61 0004 0B0B 0000
0000 0041 C398 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
.
Success rate is 0 percent (0/5) ← The ping will fail because
the ATM interface on RouterB is shut down
```

Enter configuration mode on RouterA by typing the **config term** command. Enter the **loopback diagnostic** command under the ATM interface of RouterA.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#int atm 0
RouterA(config-if)#loopback diagnostic
RouterA(config-if)#exit
RouterA(config)#exit
```

Type the **show run** command to view the router's configuration. Notice that the **loopback diagnostic** command is now under the ATM 0 interface configuration for the router.

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
no ip domain-lookup
!
interface ATM0
 ip address 195.1.1.2 255.255.255.0
 loopback diagnostic ← The loopback diagnostic command will appear under
the interface of the router
 atm pvc 1 0 32 aal5nlpid
 map-group 1
!
no ip classless
!
map-list 1
 ip 195.1.1.1 atm-vc 1 broadcast
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Ping RouterB at IP address 195.1.1.1. We see that the ping fails but the debug output shows us that each packet that is being sent out is being received back. This is because we have a loopback on the ATM interface of RouterA.

```
RouterA#ping 195.1.1.1

Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 195.1.1.1, timeout is 2 seconds:

ATM0(O): ← Output packet from RouterA to RouterB
VCD:0x1 DM:0x100 NLPID:0x03CC Length:0x6A
4500 0064 005F 0000 FF01 3334 C301 0102 C301 0101 0800 5F48 0000 1241 0000
0000 0041 0C80 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD

ATM0(I): ← Loopback response
VCD:0x1 Type:0x2 NLPID:0x03CC Length:0x6A
4500 0064 005F 0000 FF01 3334 C301 0102 C301 0101 0800 5F48 0000 1241 0000
0000 0041 0C80 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD. ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
.
.
There will be a total of 5 output packets and 5 input packets displayed.  Only the
first output and input packet are shown here.
.
.
RouterA# Success rate is 0 percent (0/5)
```

Loopback Line

Now let's disable the loopback we entered on RouterA. Enter router configuration mode and enter the command **no loopback diagnostic** under the ATM 0 interface.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#int atm 0
RouterA(config-if)#no loopback diagnostic ← Disable the loopback diagnostic on
                                              the interface
RouterA(config-if)#exit
RouterA(config)#exit
```

Connect to RouterB and enable the ATM interface. Remember that we had to put the interface into a shutdown state to demonstrate the **loopback diagnostic** command. While in interface configuration mode, type the command **loopback line**. Recall from Figure 5-16 that this command will cause RouterB to loop all traffic that comes into its ATM interface back towards the network.

```
RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#int atm 0
RouterB(config-if)#no shut ← Reactivate the interface
RouterB(config-if)#loopback line ← Enable the line loopback on RouterB. All
                                   traffic that comes into RouterB will be
                                   looped back towards the network
RouterB(config-if)#exit
RouterB(config)#exit

%LINK-3-UPDOWN: Interface ATM0, changed state to up ← The ATM interface will go
                                                      back to an up state as
                                                      soon as the shutdown
                                                      statement is removed
%LINEPROTO-5-UPDOWN: Line protocol on Interface ATM0, changed state to up
```

Type the **show interface atm 0** command on RouterB to display the status of the ATM interface. The interface should be in an up/up (looped) state.

```
RouterB#show int atm 0
ATM0 is up, line protocol is up (looped) ← The interface is looped because we
                                           have enabled a line loopback on the
                                           Router
  Hardware is ATMizer BX-50
  Internet address is 195.1.1.1/24
```

```
  MTU 4470 bytes, sub MTU 4470, BW 155520 Kbit, DLY 100 usec, rely 255/255, load
 1/255

                 The interface is looped
                          ↓
  Encapsulation ATM, loopback set, keepalive not supported
  Encapsulation(s): AAL5 AAL3/4, PVC mode
  1024 maximum active VCs, 1024 VCs per VP, 1 current VCCs
  VC idle disconnect time: 300 seconds
  Last input 00:04:16, output 00:04:16, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     52784 packets input, 2750708 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     52794 packets output, 2751768 bytes, 0 underruns
     0 output errors, 0 collisions, 10 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Connect to RouterA. Enable ATM packet debugging by typing the **debug atm packet** command.

```
RouterA#debug atm packet
ATM packets debugging is on
Displaying all ATM packets
```

Ping RouterB at IP address 195.1.1.1. We see that the ping fails since all traffic coming into RouterB is looped back to the network before entering the Router. Notice that the ping packets are returned from the loopback on RouterB. Five output packets are sent and five input packets are received back.

```
RouterA#ping 195.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 195.1.1.1, timeout is 2 seconds:

ATM0(O): ← Ping packet from RouterA to RouterB
VCD:0x1 DM:0x100 NLPID:0x03CC Length:0x6A
4500 0064 006E 0000 FF01 3325 C301 0102 C301 0101 0800 AB35 0000 076D 0000
0000 0043 CB64 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD

ATM0(I): ← Packet returned from the line loopback on RouterB
VCD:0x1 Type:0x2 NLPID:0x03CC Length:0x6A
4500 0064 006E 0000 FF01 3325 C301 0102 C301 0101 0800 AB35 0000 076D 0000
0000 0043 CB64 ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD
ABCD ABCD ABCD. ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD

.
.
There will be a total of 5 output packets and 5 input packets.  Only the first
output and input packet are shown here
.
.
RouterA# Success rate is 0 percent (0/5) ← The ping fails
because all traffic being sent to
RouterB is looped back to the network before being sent into the Router. All the
ping packets are returned back to
RouterA since RouterB has a line loopback enabled
```

Lab #22: ATM LANE

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet interface and one ATM OC−3 interface
- One Cisco LS1010 Lightstream ATM switch two OC−3 ports
- Two single−mode or multimode fiber cables
- A Cisco rolled cable for console port connection to the routers and ATM switch

Configuration Overview

This lab will demonstrate the LANE capabilities of a Cisco router and LS1010 ATM switch. As shown in Figure 5−17, we will configure RouterA as the LECS, the LES, and the BUS. RouterB will be configured as the LEC. The name of our emulated ELAN will be **mkt**. We will also configure the LECS so that it participates in SSRP.

Figure 5−17: ATM LANE configuration

Note Once ILMI is active between the router and the ATM switch, the LANE configuration server's ATM address must be entered in the 1010 Lightstream. To determine this address, you can display the default ATM addresses with the command show lane default−atm−addresses. In the example shown below, the ATM address of the LECS is 47.00918100000000107BD56901.006070CD51ED.00.

```
RouterA#show lane default−atm−addresses
interface ATM0:
LANE Client:        47.00918100000000107BD56901.006070CD51EA.**
LANE Server:        47.00918100000000107BD56901.006070CD51EB.**
LANE Bus:           47.00918100000000107BD56901.006070CD51EC.**
LANE Config Server: 47.00918100000000107BD56901.006070CD51ED.00
```

NOTE ** is the subinterface number byte in hex.

Note Once ILMI is active between the router and the ATM switch, we must configure the LANE database on RouterA. To do this, we will need the LAN server address. To determine this address, you can display the default ATM addresses on RouterA with the command show lane default−atm−addresses. Below is the output from the command. Note the ATM server address is 47.00918100000000107BD56901.006070CD51EB.

```
RouterA #show lane default−atm−addresses
Interface ATM0:
LANE Client:        47.00918100000000107BD56901.006070CD51EA.**
LANE Server:        47.00918100000000107BD56901.006070CD51EB.**
LANE Bus:           47.00918100000000107BD56901.006070CD51EC.**
LANE Config Server: 47.00918100000000107BD56901.006070CD51ED.00
```

NOTE ** is the subinterface number byte in hex.

Router and Switch Configuration

The configurations for the routers and ATM switch in this example are as follows. Key ATM commands are highlighted in bold.

RouterA

```
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
ip subnet-zero
no ip routing ← LANE will treat our network as a bridged segment
!
!
lane database cisco
  name mkt server-atm-address 47.00918100000000107BD56901.006070CD51EB.01
  default-name mkt ← Set up a name for the ELAN
no lane client flush
!
interface ATM0
 no ip address
 no ip route-cache
 no ip mroute-cache
 atm pvc 1 0 5 qsaal ← Set up ATM to switch signaling
 atm pvc 2 0 16 ilmi
 no atm ilmi-keepalive
 lane config auto-config-atm-address ← Configure the LECS to participate
                                       in SSRP
 lane config database cisco ← Link the configuration server's database name to
                              the ATM interface
!
interface ATM0.1 multipoint
 ip address 172.16.0.1 255.255.0.0
 no ip route-cache
 no ip mroute-cache
 lane server-bus ethernet mkt ← Enable the LANE server and the BUS for the
                                emulated LAN
 lane client ethernet mkt ← Enable a LANE client for the emulated LAN
!
interface Ethernet0
 ip address 1.1.1.1 255.255.255.0
 no ip route-cache
 no ip mroute-cache
!
ip classless
no ip http server
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
ip subnet-zero
no ip routing
!
```

```
lane client flush
cns event-service server
!
interface Ethernet0
 ip address 2.2.2.2 255.255.255.0
 no ip route-cache
 no ip mroute-cache
 media-type auto-select
!
interface ATM0
 no ip address
 no ip route-cache
 no ip mroute-cache
 atm pvc 1 0 5 qsaal
 atm pvc 2 0 16 ilmi
 no atm ilmi-keepalive
!
interface ATM0.2 multipoint
 ip address 172.16.0.3 255.255.0.0
 no ip route-cache
 no ip mroute-cache
 lane client ethernet mkt
!
ip classless
no ip http server
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

LS1010

```
Current configuration:
!
version 11.2
no service pad
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname LS1010
!
boot system flash slot0:ls1010-wp-mz.120-9.bin
!
!
atm lecs-address-default 47.0091.8100.0000.0010.7bd5.6901.0060.70cd.51ed.00 1
atm address 47.0091.8100.0000.0010.7bd5.6901.0010.7bd5.6901.00
atm router pnni
 node 1 level 56 lowest
  redistribute atm-static
!
!
interface ATM0/0/0
 atm maxvp-number 8
 atm maxvc-number 8192
!
interface ATM0/0/1
 mtu 8192
 atm maxvp-number 8
 atm maxvc-number 8192
!
interface ATM0/0/2
!
interface ATM0/0/3
```

```
!
interface ATM2/0/0
 no ip address
 atm maxvp-number 0
!
interface Ethernet2/0/0
 ip address 10.10.3.170 255.255.255.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's begin by connecting to RouterB. Display the ILMI status on RouterB with the command **show atm ilmi-status.** Notice that the ILMI status is up and normal.

```
RouterB#show atm ilmi-status
Interface : ATM0 Interface Type : Private UNI (User-side)
ILMI VCC : (0, 16) ILMI Keepalive : Disabled
ILMI State:       UpAndNormal
Peer IP Addr:     10.10.3.170     Peer IF Name:     ATM0/0/0
Peer MaxVPIbits:  8               Peer MaxVCIbits:  14
Active Prefix(s) :
47.0091.8100.0000.0010.7bd5.6901
```

Display the status of the LECS, the LES, and the BUS with the command **show lane**. Below is the output from the command. Notice that all three are up and operational.

```
RouterA#show lane

LE Config Server ATM0 config table: cisco
Admin: up State: operational
LECS Mastership State: active master
list of global LECS addresses (35 seconds to update):
47.00918100000000107BD56901.006070CD51ED.00  <-------- me
ATM Address of this LECS: 47.00918100000000107BD56901.006070CD51ED.00 (auto)
 vcd  rxCnt  txCnt  callingParty
  30      1      1  47.00918100000000107BD56901.006070CD51EB.01 LES mkt 0 active
cumulative total number of unrecognized packets received so far: 0
cumulative total number of config requests received so far: 8
cumulative total number of config failures so far: 0

LE Server ATM0.1, Elan name: mkt, Admin: up, State: operational
Master/Backup: Master, Type: ethernet, Max Frame Size: 1516
locally set elan-id: not set
elan-id obtained from LECS: not set
ATM address: 47.00918100000000107BD56901.006070CD51EB.01
LECS used: 47.00918100000000107BD56901.006070CD51ED.00 connected, vcd 29
control distribute: vcd 34, 1 members, 7 packets
proxy/ (ST: Init, Conn, Waiting, Adding, Joined, Operational, Reject, Term)
lecid ST vcd    pkts Hardware Addr  ATM Address
   2P O   42       5 0060.70cd.51ea 47.00918100000000107BD56901.006070CD51EA.01


LE BUS ATM0.1,  Elan name: mkt, Admin: up, State: operational
Type: ethernet, Max Frame Size: 1516
ATM address: 47.00918100000000107BD56901.006070CD51EC.01
data forward: vcd 36, 1 members, 77 packets, 3 unicasts
lecid  vcd     pkts   ATM Address
```

```
    2   45       63   47.00918100000000107BD56901.006070CD51EA.01
LE Client ATM0.1  ELAN name: mkt  Admin: up  State: operational
Client ID: 2                LEC up for 56 minutes 49 seconds
ELAN ID: 0
Join Attempt: 1
Known LE Servers: 1
Last Fail Reason: Locally deactivate
HW Address: 0060.70cd.51ea   Type: ethernet            Max Frame Size: 1516
ATM Address: 47.00918100000000107BD56901.006070CD51EA.01
 VCD  rxFrames  txFrames  Type       ATM Address
   0         0         0  configure  47.00918100000000107BD56901.006070CD51ED.00
  41         1         5  direct     47.00918100000000107BD56901.006070CD51EB.01
  43         6         0  distribute 47.00918100000000107BD56901.006070CD51EB.01
  44         0        63  send       47.00918100000000107BD56901.006070CD51EC.01
  46        72         0  forward    47.00918100000000107BD56901.006070CD51EC.0
```

Display the status of the LANE client on RouterB with the command **show lane.** Below is the output from the command. Notice that the LANE client is up and operational.

```
RouterB#show lane

01:59:44: %SYS-5-CONFIG_I: Configured from console by console
LE Client ATM0.2  ELAN name: mkt  Admin: up  State: operational
Client ID: 1                LEC up for 21 seconds
ELAN ID: 0
Join Attempt: 1
Known LE Servers: 1
Last Fail Reason: Locally deactivate
HW Address: 00d0.bbfb.08c8   Type: ethernet            Max Frame Size: 1516
ATM Address: 47.00918100000000107BD56901.00D0BBFB08C8.02
 VCD  rxFrames  txFrames  Type       ATM Address
   0         0         0  configure  47.00918100000000107BD56901.006070CD51ED.00
  70         1         2  direct     47.00918100000000107BD56901.006070CD51EB.01
  71         1         0  distribute 47.00918100000000107BD56901.006070CD51EB.01
  72         0         4  send       47.00918100000000107BD56901.006070CD51EC.01
  73         4         0  forward    47.00918100000000107BD56901.006070CD51EC.01
```

To verify that this is working properly, ping 2.2.2.2 for RouterA. Use the extended ping command to source the packet from 1.1.1.1. This will work because we have disabled IP routing on RouterA and RouterB and LANE treats our network as a bridged segment.

```
RouterA#ping
Protocol [ip]:
Target IP address: 2.2.2.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 1.1.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
.!!!!
```

Now display the LANE information on RouterB. Notice that a new VCD has been created for the connection between RouterA and RouterB.

```
RouterB#show lane
LE Client ATM0.2  ELAN name: mkt  Admin: up  State: operational
Client ID: 1                LEC up for 8 minutes 29 seconds
ELAN ID: 0
```

```
Join Attempt: 1
Known LE Servers: 1
Last Fail Reason: Locally deactivate
HW Address: 00d0.bbfb.08c8   Type: ethernet             Max Frame Size: 1516
ATM Address: 47.00918100000000107BD56901.00D0BBFB08C8.02
 VCD  rxFrames  txFrames  Type       ATM Address
   0         0         0  configure  47.00918100000000107BD56901.006070CD51ED.00
  70         1         4  direct     47.00918100000000107BD56901.006070CD51EB.01
  71         6         0  distribute 47.00918100000000107BD56901.006070CD51EB.01
  72         0        16  send       47.00918100000000107BD56901.006070CD51EC.01
  73        26         0  forward    47.00918100000000107BD56901.006070CD51EC.01
  74         5         3  data       47.00918100000000107BD56901.006070CD51EA.01
```

Troubleshooting ATM

This section will discuss important commands that can be used to monitor and troubleshoot an ATM configuration.

{show interface atm} The **show interface atm** command will display information on the status of the ATM interface on a router.

```
RouterA#show interface atm 0
ATM0 is up, line protocol is up ← Interface status
  Hardware is ATMizer BX-50
  Internet address is 195.1.1.2/24
  MTU 4470 bytes, sub MTU 4470, BW 156250 Kbit, DLY 100 usec, rely 10/255, load
1/255
  Encapsulation ATM, loopback not set, keepalive set (10 sec)
  Encapsulation(s): AAL5 AAL3/4, PVC mode
  1024 maximum active VCs, 1024 VCs per VP, 1 current VCCs
  VC idle disconnect time: 300 seconds
  Last input 00:09:56, output 00:09:56, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     25 packets input, 2650 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     25 packets output, 2650 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

{show controller atm} The **show controller atm 0** command can be used to verify that the ATM network module is installed and properly functioning.

```
RouterA#show controller atm 0
ATM Unit 0, Slot 2, Type ATMizer BX-50, Hardware Version 1
  ATM Xilinx Code, Version 2, ATMizer Firmware, Version 3.0
  Public SRAM 65536 bytes, Private SRAM 524288 bytes, I/O Base Addr 0x3C200000
  PLIM Type OC-3 Multi-Mode Fiber, Version 3
  Network Transmit Clock
  NIM IS Operational, Configuration OK
  DMA Read 12, DMA Write 12
```

{show atm vc} The **show atm vc** command will provide a summary of all active PVCs on a router.

```
RouterA#show atm vc
                                    AAL /         Peak   Avg.  Burst
Interface     VCD   VPI   VCI Type  Encapsulation  Kbps   Kbps  Cells Status
ATM0            1     0    32  PVC  AAL5-NLPID    155000 155000    94 ACTIVE
```

Entering the **show atm vc** command with the virtual circuit descriptor for the PVC will show detailed information on the PVC.

```
RouterA#sh atm vc 1
ATM0: VCD: 1, VPI: 0, VCI: 32, etype:0x2, AAL5 – NLPID, Flags: 0xC31
PeakRate: 155000, Average Rate: 155000, Burst Cells: 94, VCmode: 0x1
OAM DISABLED, InARP DISABLED
InPkts: 5, OutPkts: 5, InBytes: 530, OutBytes: 530
InPRoc: 5, OutPRoc: 5, Broadcasts: 0
InFast: 0, OutFast: 0, InAS: 0, OutAS: 0
OAM F5 cells sent: 0, OAM cells received: 0
Status: ACTIVE
```

{show atm map} The **show atm map** command displays any ATM maps that have been configured on the router.

```
RouterA#show atm map
Map list 1 : PERMANENT
ip 195.1.1.1 maps to VC 1, broadcast
```

{show atm traffic} The **show atm traffic** command will show how many packets have been sent and received on each ATM interface.

```
RouterA#sh atm traffic
25 Input packets
25 Output packets
0 Broadcast packets
0 Packets received on non-existent VC
0 Packets attempted to send on non-existent VC
0 OAM cells received
0 OAM cells sent
```

Conclusion

This chapter has explored the technology of ATM. We have seen that ATM has many advantages over traditional network protocols. The labs in this chapter have demonstrated some of Cisco's ATM capabilities.

Chapter 6: Routing Information Protocol

Overview

Topics Covered in This Chapter

- Detailed technology overview
- Mechanisms to prevent routing loops
- RIP message format
- Basic RIP configuration
- Passive interface configuration
- Configuring RIP timers
- Configuring Unicast RIP updates
- RIP and discontinuous networks
- Detailed troubleshooting

Introduction

Routing Information Protocol (RIP) is a distance vector protocol used to exchange routing information among gateways (routers) and hosts. RIP is based on the Bellham−Ford (distance vector) algorithm, which was originally used in computer routing in 1969 by ARPANET. However, Xerox originally developed the protocol RIP, as we know it today, in the late 1970s as part of their Xerox Networking Services (XNS) protocol suite.

Despite its technical limitations, RIP is one of the most widely used Interior Gateway Protocols (IGPs) designed for medium−size homogeneous networks. RIP owes its widespread installed base to the fact that Berkley distributed routed software along with their

popular 4BSD UNIX. Routed software used RIP to provide consistent routing and reachability information for machines on local networks. TCP/IP sites started using RIP to provide local area routing and eventually began using it in the wide area.

Technology Overview

RIP uses two packet types to convey information: updates and requests. Each RIP−enabled router on the network broadcasts update messages every 30 seconds using UDP port 520 to all directly connected neighbors. Update messages reflect the complete routing database that currently exists in the router. Each entry in the database consists of two elements: the IP address of the network that can be reached and the distance to that network. Request messages are used by the router to discover other RIP−speaking devices on the network.

RIP uses hop count as the metric to measure the distance to a network. Each router adds its internal distance (1) to the route before advertising the route to its neighbors. In Figure 6−1, RouterC is directly connected to NetworkC. When it advertises its route to RouterB, it increments the metric by 1; likewise, RouterB increases the metric to 2 and advertises the route to RouterA. RouterB and RouterA are said to be one and two hops, respectively, from NetworkC.

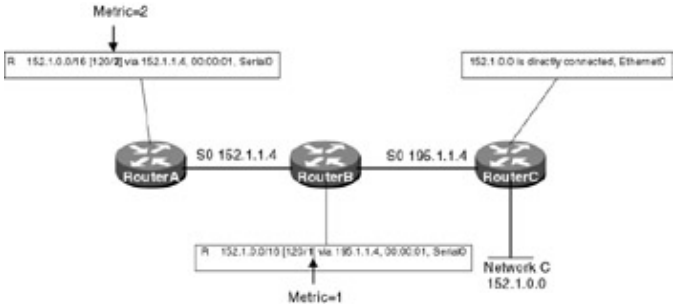


Figure 6−1: RIP metrics

As per Figure 6−1, the number of hops to get to a given destination is the number of routers that a datagram must pass through to get to that destination network. Using hop count for path determination will not always provide the best path. For example, in Figure 6−2, to get from RouterA to NetworkB, RIP will prefer the one 56 Kbps link over the two 1.5 Mbps links. Even though a hop count of 1 across a 56 Kbps serial circuit will be substantially slower than a path with a hop count of 2 that crosses two 1.5 Mbps serial circuits.

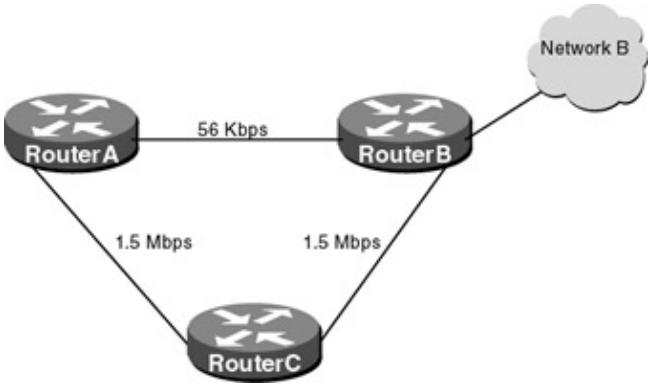


Figure 6−2: Hop count

Routing Loops

The problem with any distance vector routing protocol like RIP is that each router does not have a complete view of the network. Routers must rely on the neighboring routers for network reachablity information. The distance vector routing algorithm creates a slow convergence problem in which inconsistencies arise, because routing update messages propagate slowly across the network. To reduce the likelihood of routing loops, RIP uses the following mechanisms: count−to−infinity, split horizons, poison reverse updates, holddown counters, and triggered updates.

Count−to−Infinity Problem RIP permits a maximum hop count of 15. Any destination that is more then 15 hops away is considered unreachable. This number, while severely limiting the size of a network, prevents a problem called count−to−infinity. See Figure 6−3.

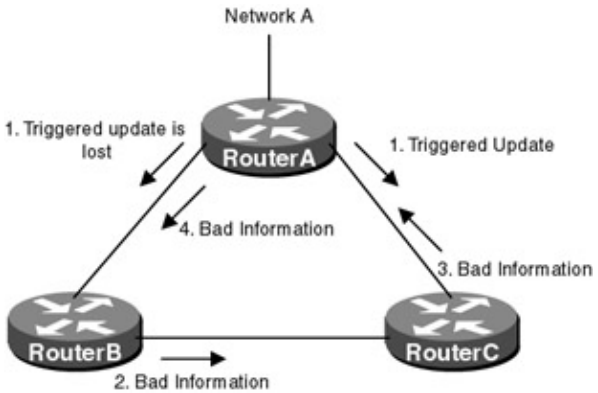


Figure 6−3: Count−to−infinity problem

1. Count−to−infinity works like this. RouterA loses its interface to Network A and generates a triggered update, which is sent to RouterB and RouterC. The triggered update tells RouterB and RouterC that RouterA no longer has a route to NetworkA. The update is delayed during transmission to RouterB

(busy CPU, congested link, and so on) but arrives at RouterC. RouterC removes the route to NetworkA from its routing table.

2. RouterB still has not received the triggered update from RouterA and sends its regular routing update advertising NetworkA as reachable with a hop count of 2. RouterC receives the update and thinks a new route exists to NetworkA.
3. RouterC then advertises to RouterA that it can reach NetworkA with a hop count of 3.
4. RouterA then advertises to RouterB that it can reach NetworkA with a hop count of 4.
5. This loop will continue until the hop count reaches infinity, which is defined by RIP as 16. Once a route reaches infinity (16), it is declared unusable and deleted from the routing table.

With the count−to−infinity problem, the routing information will continue to pass from router to router, incrementing the hop count by 1. This problem, and the routing loop, will continue indefinitely or until some limit is reached. That limit is RIP's maximum hop count. When the hop count of a route exceeds 15, the route is marked unreachable, and over time, eventually removed from the routing table.

Split Horizons The rule of split horizons states that it is never useful for a router to advertise a route back in the direction from which it came. When split horizons are enabled on a router's interface, the router records the interface over which a route was received and does not propagate information about that route back out that interface.

The Cisco router allows you to disable split horizons on a per−interface basis. This is sometimes necessary in NBMA (nonbroadcast multiple access) hub−and−spoke environments. In Figure 6−4, RouterB is connected to RouterC, and RouterA via frame relay, and both PVCs are terminating on one physical interface on RouterB.

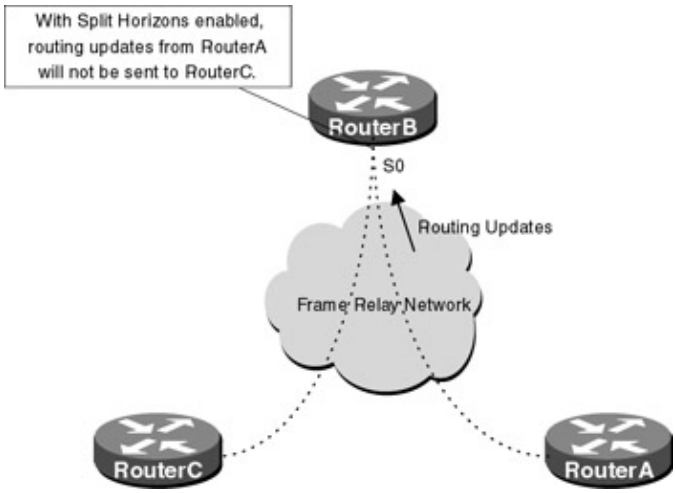


Figure 6−4: Spilt horizon

In Figure 6−4, if split horizons are not disabled on RouterB's serial interface, RouterC will not receive RouterA's routing advertisements and vice versa. Use the **no ip split−horizon** interface subcommand to disable split horizons.

Poison Reverse Split horizon is a scheme used by the router to avoid problems caused by advertising routes back to the router from which they were learned. The split−horizon scheme omits routes learned from one neighbor in updates sent to that neighbor. Split horizon with poisoned reverse includes the routes in updates, but sets their metric to 16 (infinity).

By setting the metric to infinity and advertising the route back to its source, it is possible to immediately break a routing loop. Otherwise, the inaccurate route will stay in the routing table until it times out. The disadvantage of using poison reverse is that it increases the size of the routing table.

Holddowns Holddown timers prevent the router from accepting routing information about a network for a fixed period of time after the route has been removed from the routing table. The idea is to make sure all routers have received the information, and no router sends out an invalid route. For example, in Figure 6−3, RouterB advertised bad information to RouterC because of the delay in the routing update. With holddown

counters, this would not happen because RouterC would not install a new route to NetworkA for 180 seconds. By then, RouterB would have converged with the proper routing information.

Triggered Updates Split horizons with poison reverse breaks any loop between two routers. Loops containing three or more routers can still occur, ending only when infinity (16) is reached. Triggered updates are an attempt to speed up convergence time — whenever the metric of a route changes, the router must send an update message immediately. A triggered update message is sent immediately regardless of when the regular update message is scheduled to be sent.

RIP Message Format

Figure 6−5 shows the format of a RIP message. After the 32−bit header, the message contains a sequence of pairs. The pairs contain the network IP address and an integer distance to that network that can be reached. The various components of a RIP message are described next:

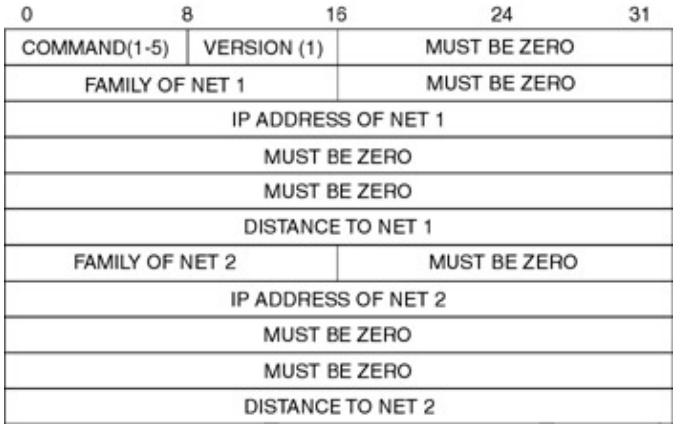


Figure 6−5: RIP message format

Commands: The command is generally either a RIP request (1) or a RIP response (2). Commands 3 and 4 are obsolete and command 5 is reserved for Sun Microsystems internal use.

Version: This field contains the protocol version number. There are two versions of RIP.

Address Family Identifier: RIP was designed to carry routing information for multiple protocols. This field specifies the family of the protocol that is being carried. The address family identifier for IP is 2.

IP Address: This field contains the IP address, which is stored as a four−octet number.

Must Be Zero: RIP can carry network addresses that are up to 12 octets long. Since an IP address only uses 4 of the 32 octets, the remaining 8 octets are padded with zeros.

Distance to Net: This field contains an integer count of the distance to the specified network. It contains a value of 16 if the network is unreachable.

Commands Discussed in This Chapter

- **clear ip route**
- **debug ip rip events**
- **network** {network−number}
- **passive−interface** {type number}
- **router rip**
- **timers basic** {update invalid holddown flush}
- **show ip protocol**
- **show ip route rip**

Definitions

clear ip route: This exec command removes one or more routes from the routing table. The command allows you to enter a specific route or use a * to remove all routes.

debug ip rip events: This exec command displays information on RIP routing transactions. It displays all RIP routing updates that are sent or received by the router.

network: This router configuration command specifies what interfaces will receive and send RIP routing updates. It also specifies what networks will be advertised; if an interfaces network is not specified, it will not be advertised in a RIP update.

passive−interface: This router configuration command disables the sending of routing updates on a given interface. If you disable the sending of routing updates on an interface, the particular subnet will continue to be advertised out other RIP−enabled interfaces and requests will still be sent out the interface. Routes received by a passive interface will still be processed.

router rip: This global configuration command enables the RIP routing process on the router.

timers basic: This router configuration command allows the user to set the update, invalid, holddown, and flush timers for the RIP process. The following is an explanation of what each timer is used for:

- ***update*** The update timer sets the rate in seconds at which routing updates are sent. The default is 30 seconds.
- ***invalid*** The invalid timer sets the interval of time in seconds after which a route is declared invalid. The timer is started if the route is not present in the regular update message. The default is 180 seconds.
- ***holddown*** The holddown timer sets the interval in seconds during which routing information regarding better paths is suppressed. The idea is to make sure all routers have received the information and no router sends out an invalid route. The default is 180 seconds.
- ***flush*** The flush timer sets in seconds the amount of time that must pass before a route is removed from the routing table. The default is 240 seconds.

show ip protocol: This exec command displays the current state of the active routing protocol process.

show ip route rip: This exec command displays the all RIP learned routes.

IOS Requirements

RIP first became available in IOS 10.0.

Lab #23: Basic RIP Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program

- Two Cisco DTE/DCE crossover cables
- One Cisco rolled cable

Configuration Overview

This configuration will demonstrate basic routing using Routing Information Protocol (RIP). As per Figure 6−6, RouterA, RouterB, and RouterC will use RIP to advertise routing information.

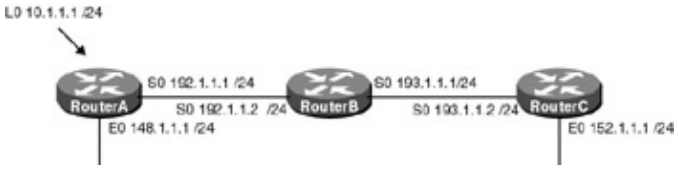


Figure 6−6: Basic RIP

RouterA, RouterB, and RouterC are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 6−6. All routers will be configured for RIP and will advertise all connected networks.

Router Configurations

The configurations for the three routers in this example are as follows (key RIP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point.
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 148.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub.
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
!
router rip ← Enables the RIP routing process on the router
 network 10.0.0.0 ← Specifies what interfaces will receive and send RIP routing
                    updates. It also specifies what networks will be advertised.
 network 148.1.0.0
 network 192.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router rip ← Enables the RIP routing process on the router
 network 192.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised.
 network 193.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub.
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
router rip ← Enables the RIP routing process on the router
network 152.1.0.0 ← Specifies what interfaces will receive and send RIP
routing updates. It also specifies what networks will be advertised
network 193.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
```

```
end
```

Monitoring and Testing the Configuration

RIP is a very simple protocol to configure and troubleshoot. Show the IP routing table on RouterA with the **show ip route** command. The following is the output from this command. Notice that two networks were learned via RIP: 152.1.0.0 and193.1.1.0.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     148.1.0.0/24 is subnetted, 1 subnets
C       148.1.1.0 is directly connected, Ethernet0
R    152.1.0.0/16 [120/2] via 192.1.1.2, 00:00:20, Serial0
C    192.1.1.0/24 is directly connected, Serial0
R    193.1.1.0/24 [120/1] via 192.1.1.2, 00:00:20, Serial0
```

From RouterA, monitor the routing updates being passed using the **debug ip rip** command. The following is the output from this command. Notice that on interface serial 0, the router does not advertise the networks it learned from RouterB (152.1.0.0 and 193.1.1.0), but on all other interfaces, these networks are advertised. This is split horizons at work — remember that when split horizons are enabled, the router will never advertise a route back in the direction from which it came.

```
RouterA#debug ip rip
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (148.1.1.1)
     network 10.0.0.0, metric 1
     network 152.1.0.0, metric 3
     network 192.1.1.0, metric 1
     network 193.1.1.0, metric 2
RIP: sending v1 update to 255.255.255.255 via Loopback0 (10.1.1.1)
     network 148.1.0.0, metric 1
     network 152.1.0.0, metric 3
     network 192.1.1.0, metric 1
     network 193.1.1.0, metric 2
RIP: sending v1 update to 255.255.255.255 via Serial0 (192.1.1.1)
     network 10.0.0.0, metric 1
     network 148.1.0.0, metric 1
```

Now, disable split horizons on RouterA using the interface configuration command **no ip split−horizon**:

```
RouterA(config)#int s0
RouterA(config−if)#no ip split−horizon
```

From RouterA, monitor the routing updates being passed using the **debug ip rip** command. The following is the output from this command. Notice that now all routes are being advertised out serial 0, including the routes learned form RouterB on serial 0.

```
RouterA#debug ip rip
IP: sending v1 update to 255.255.255.255 via Ethernet0 (148.1.1.1)
     network 10.0.0.0, metric 1
     network 152.1.0.0, metric 3
     network 192.1.1.0, metric 1
     network 193.1.1.0, metric 2
RIP: sending v1 update to 255.255.255.255 via Loopback0 (10.1.1.1)
```

```
     network 148.1.0.0, metric 1
     network 152.1.0.0, metric 3
     network 192.1.1.0, metric 1
     network 193.1.1.0, metric 2
RIP: sending v1 update to 255.255.255.255 via Serial0 (192.1.1.1)
     network 10.0.0.0, metric 1
     network 148.1.0.0, metric 1
     network 152.1.0.0, metric 3
     network 192.1.1.0, metric 1
     network 193.1.1.0, metric 2
```

Lab #24: Passive Interface Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Configuration Overview

This configuration will demonstrate the use of the **passive−interface** command, which allows RIP−enabled routers to listen to, but not send, routing updates out a particular interface. The passive−interface router configuration command is typically used when the network router configuration command configures more interfaces than are desirable.

RIP is a classful routing protocol that does not carry subnet information. When enabling RIP on a router, you specify which classful network the protocol will be run on. For example, in Figure 6−7, there are three subnets on RouterA (10.1.1.0/24, 10.1.2.0/24, and 10.1.3.0/24). When enabling RIP, the user specifies which network RIP will run under — in this case, network 10.0.0.0, which encompasses all three subnets.

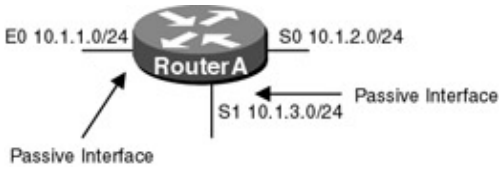


Figure 6−7: Passive interface

The reason RIP changes the network entry from 10.1.1.0 to 10.0.0.0 is because RIP is considered a "classful" protocol. By that we mean that it recognizes the IP address class of the network address that you type and assumes the proper mask. For a class A network like this one, the mask is 255.0.0.0, yielding 10.0.0.0 (no matter what you actually type as the last two octets). The network statement tells the routing protocol to route on the interfaces where the network address matches the one specified in the network statement.

In this scenario, the user only wishes to send RIP updates out network 10.1.2.0, so interface E0 (10.1.1.0) and S1 (10.1.3.0) are made passive interfaces.

RouterA, RouterB, and RouterC are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 6−8. All routers will be configured for RIP, and RouterB and RouterC will advertise all connected networks. RouterA's interface S0 will be passive and will not advertise any routing information; however, it will still receive routing updates.

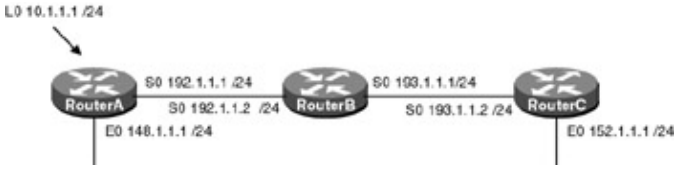


Figure 6−8: RIP passive interface configuration

Router Configurations

The configurations for the three routers in this example are as follows (key RIP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 148.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
!
router rip ← Enables the RIP routing process on the router
passive−interface Serial0 ← Disables the sending of RIP updates on interface
                            Serial 0
 network 10.0.0.0 ← Specifies what interfaces will receive and send RIP routing
                    updates. It also specifies what networks will be advertised
 network 148.1.0.0
 network 192.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.0
service udp−small−servers
service tcp−small−servers
!
hostname RouterB
!
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair−queue
```

```
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router rip ← Enables the RIP routing process on the router
 network 192.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
 network 193.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
!
router rip ← Enables the RIP routing process on the router
network 152.1.0.0 ← Specifies what interfaces will receive and send RIP routing
                    updates. It also specifies what networks will be advertised
network 193.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

The following is the output from the **debug ip rip** command on RouterA. Notice that RIP updates are only being sent out interface Ethernet 0 and Loopback 0. Also note that interface S0 is still receiving RIP updates.

```
RouterA#debug ip rip
RIP: received v1 update from 192.1.1.2 on Serial0
     152.1.0.0 in 2 hops
     193.1.1.0 in 1 hops
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (148.1.1.1)
     network 10.0.0.0, metric 1
     network 152.1.0.0, metric 3
```

```
     network 192.1.1.0, metric 1
     network 193.1.1.0, metric 2
RIP: sending v1 update to 255.255.255.255 via Loopback0 (10.1.1.1)
     network 148.1.0.0, metric 1
     network 152.1.0.0, metric 3
     network 192.1.1.0, metric 1
     network 193.1.1.0, metric 2
```

The following is the output from the **show ip route** command on RouterA and RouterC. Note that RouterA has learned all of the routes from RouterC, but RouterC has no routes from RouterA.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set
     10.0.0.0/24  is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     148.1.0.0/24  is subnetted, 1 subnets
C       148.1.1.0 is directly connected, Ethernet0
R    152.1.0.0/16 [120/2] via 192.1.1.2, 00:00:13, Serial0
C    192.1.1.0/24 is directly connected, Serial0
R    193.1.1.0/24 [120/1] via 192.1.1.2, 00:00:13, Serial0

RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     152.1.0.0/24  is subnetted, 1 subnets
C       152.1.1.0 is directly connected, Ethernet0
R    192.1.1.0/24  [120/1] via 193.1.1.2, 00:00:20, Serial0
C    193.1.1.0/24  is directly connected, Serial0
```

Lab #25: RIP Timer Configurations

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Configuration Overview

This configuration will demonstrate using the **timer basic** command to set the four configurable RIP timers (update, invalid, holddown, and flush timers). Depending on the network topology, it may become necessary

to change the update timers, which control the rate in seconds that routing updates are sent. For example, if the access link is 56 Kbps, generating RIP updates every 30 seconds might not be the most efficient use of bandwidth. However, by increasing the update timer, you also increase the convergence time of the network.

The three other RIP timers are all dependent on the value of the update timer. The invalid timer should be at least three times the value of update timer; the holddown timer should be at least three times the value of update timer; and the flush timer must be at least the sum of invalid and holddown timers. So as you can see, if the update timer is changed, the invalid, holddown, and flush timers must also be changed.

Each time a route is updated, which is dependent on the update interval, the invalid timer is reset. If a route is not seen in an update for 180 seconds, the route is put in holddown, which means that the router will use the route to route packets, but will not announce the route in its updates. It also means that the router will not install any another route to this destination until the holddown counter expires. This happens after 180 seconds, in which case, the route is removed from the routing table.

Note The update interval must be the same value on neighboring routers.

RouterA, RouterB, and RouterC are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 6−9. All routers will be configured for RIP. RouterA, RouterB, and RouterC will advertise all connected networks. The timers on each router will be as follows:

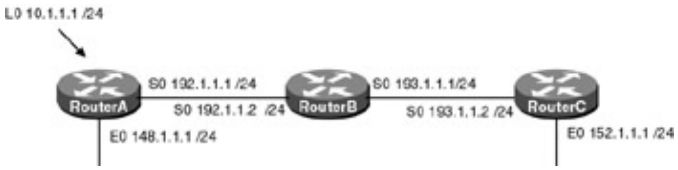


Figure 6−9: Network topology

- Update = 5
- Invalid = 15
- Holddown = 15
- Flush = 30

With these timers set, updates are broadcast every 5 seconds. If a route is not heard from in 15 seconds, the route is declared unusable (invalid). Any information received in routing updates about this particular network is suppressed for an additional 15 seconds (holddown). At the end of the suppression period, the route is flushed from the routing table.

Router Configurations

The configurations for the three routers in this example are as follows (key RIP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
```

```
 ip address 148.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
router rip ← Enables the RIP routing process on the router
timers basic 5 15 15 30 ← Updates are broadcast every 5 seconds. If a router is
                          not heard from in 15 seconds, the route is declared
                          unusable. Further information is suppressed for an
                          additional 15 seconds. At the end of the suppression
                          period, the route is flushed from the routing table
network 10.0.0.0 ← Specifies what interfaces will receive and send RIP routing
                   updates. It also specifies what networks will be advertised
 network 148.1.0.0
 network 192.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.0
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock


!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router rip ← Enables the RIP routing process on the router
timers basic 5 15 15 30 ← Updates are broadcast every 5 seconds. If a router is
                          not heard from in 15 seconds, the route is declared
                          unusable. Further information is suppressed for an
                          additional 15 seconds. At the end of the suppression
                          period, the route is flushed from the routing table
 network 192.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
 network 193.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive  ← Disables the keepalive on the Ethernet interface, allows the
                 interface to stay up when it is not attached to a hub
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
!
router rip ← Enables the RIP routing process on the router
timers basic 5 15 15 30 ← Updates are broadcast every 5 seconds. If a router is
                          not heard from in 15 seconds, the route is declared
                          unusable. Further information is suppressed for an
                          additional 15 seconds. At the end of the suppression
                          period, the route is flushed from the routing table
network 152.1.0.0 ← Specifies what interfaces will receive and send RIP routing
                    updates. It also specifies what networks will be advertised.
network 193.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
```

Monitoring and Testing the Configuration

The following is the output from the **show ip protocols** command. Note that the timers have been changed.

```
RouterA#show ip protocols
Routing Protocol is "rip"
  Sending updates every 5 seconds, next due in 3 seconds
  Invalid after 15 seconds, hold down 15, flushed after 30
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 1, receive any version
    Interface         Send   Recv   Key-chain
    Ethernet0         1      1      2
    Loopback0         1      1      2
    Serial0           1      1      2
  Routing for Networks:
    10.0.0.0
    192.1.1.0
    148.1.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.1.1.2            120      01:13:39
  Distance: (default is 120)
```

Now let's examine how these timers work when a route is lost. Perform the following steps:

1. Add the service timestamps command to RouterA's configuration.

```
RouterA(config)#service timestamps
```

2. On RouterA, monitor the routing table changes with the **debug ip route** command.

```
RouterA#debug ip routing
```

3. Disconnect the serial line between RouterB and RouterC.

The following is the output from the **debug** command. Note that after the route was declared invalid, it was placed in holddown, and approximately 30 seconds later, the route was cleared from the table.

```
07:03:18: RT: delete route to 152.1.0.0 via 192.1.1.2, rip metric
[120/2] ← Route is declared invalid
07:03:18: RT: no routes to 152.1.0.0, entering holddown ← Route is placed in holddown
07:03:18:      193.1.1.0 in 16 hops (inaccessible)
07:03:18: RT: delete route to 193.1.1.0 via 192.1.1.2, rip metric [120/1]
07:03:18: RT: no routes to 193.1.1.0, entering holddown
07:03:45: RT: garbage collecting entry for 152.1.0.0 ← Route is removed from
                                                        the routing table
07:03:45: RT: garbage collecting entry for 193.1.1.0
```

The following is the snapshot of the routing table after the route was declared invalid, but before the route was flushed from the table. At this time, the route is marked down and advertised out to all neighbors with a hop count of 16. After the route is flushed from the table, it is no longer advertised to neighboring routers.

```
RouterA#sho ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     10.0.0.0/24  is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     148.1.0.0/24  is subnetted, 1 subnets
C       148.1.1.0 is directly connected, Ethernet0
R    152.1.0.0/16  is possibly down, routing via 192.1.1.2, Serial0
C    192.1.1.0/24  is directly connected, Serial0
R    193.1.1.0/24  is possibly down, routing via 192.1.1.2, Serial0
```

The following is the output from the **debug ip rip** command taken during the transition from the invalid to holddown to flushed state:

```
07:03:18: RIP: received v1 update from 192.1.1.2 on Serial0
07:03:18:      152.1.0.0 in 16 hops (inaccessible)
07:03:18: RIP: sending v1 update to 255.255.255.255 via Ethernet0 (148.1.1.1)
07:03:18:      network 10.0.0.0, metric 1
07:03:18:      network 152.1.0.0, metric 16
07:03:18:      network 192.1.1.0, metric 1
07:03:45: RIP: sending v1 update to 255.255.255.255 via Ethernet0 (148.1.1.1)
07:03:45:      network 10.0.0.0, metric 1
07:03:45:      network 192.1.1.0, metric 1
```

Let's examine the data in chronological order using the timestamps. At 07:03:18, the route is declared invalid and the holddown counter begins. At this time, the route is advertised to all neighbors during the normal update period with a metric of 16. At 07:03:45, approximately 30 seconds after the route was declared invalid, the route is removed from the routing table and is no longer advertised in the normal routing updates.

Lab #26: Configuring Unicast RIP Updates

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco rolled cable

The RIP neighbor command permits the point−to−point (nonbroadcast) exchange of routing information. This command can be used in combination with the passive−interface router configuration command to exchange information between a subset of routers and access servers all connected to the same LAN.

For example, in Figure 6−10, RouterA wishes to only send routing updates to RouterB on the Ethernet LAN. Since RIP is a broadcast protocol, by default, it will send updates to all devices on the Ethernet LAN. To prevent this from happening, RouterA's Ethernet interface is configured as passive. However, in this case, a neighbor router configuration command is included. This command permits the sending of routing updates to a specific neighbor. One copy of the routing update is generated per defined neighbor.

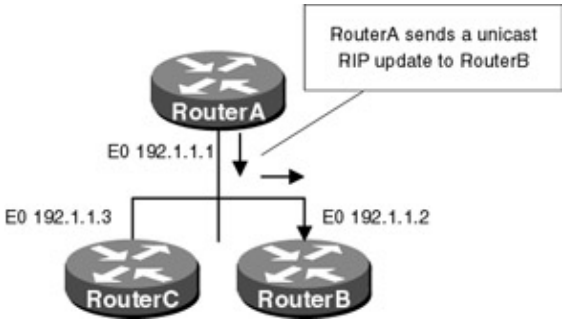


Figure 6−10: RIP Unicast updates

Router Configurations

The configuration for RouterA is as follows (key RIP configurations for RouterA are highlighted in bold).

RouterA

```
Building configuration...

Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 192.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
!
router rip ← Enables the RIP routing process on the router
passive-interface Ethernet0 ← Disables the sending of RIP updates on interface
```

```
                                Ethernet 0
network 192.1.1.0 ← Specifies what interfaces will receive and send RIP routing
                    updates. It also specifies what networks will be advertised
network 1.1.1.1
neighbor 192.1.1.2 ← Permits the point-to-point (nonbroadcast) exchange of
                     routing information
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

The following is the output from the **debug ip rip** command. Note that RIP updates are being sent to the Unicast address 192.1.1.2 on Ethernet 0 and the broadcast address 255.255.255.255 on interface loopback 0.

```
RouterA#debug ip rip
RIP: sending v1 update to 255.255.255.255 via Loopback0 (1.1.1.1)
     network 192.1.1.0, metric 1
RIP: sending v1 update to 192.1.1.2 via Ethernet0 (192.1.1.1)
     network 1.0.0.0, metric 1
```

Lab #27: RIP and Discontiguous Networks

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco rolled cable

A discontiguous network is a network that has subnets of the same major network separated by another major network. For example, in Figure 6−11, network 130.1.1.0/24 on RouterA is separated from network 130.1.2.0/24 on RouterB by the major network 131.1.1.0.

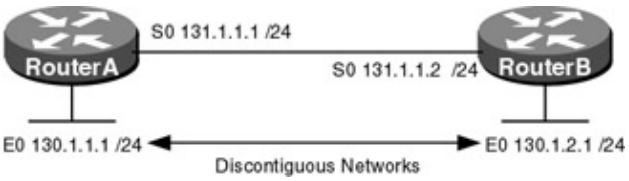


Figure 6−11: Discontiguous networks

Due to the classful nature of RIP and the fact that no mask information is carried in the routing updates, support for discontiguous networks becomes a problem. For example, in Figure 6−11, when the RouterA sends updates for network 130.1.1.0 to RouterB, it summarizes the network at the natural class — in this case, class B (130.1.0.0). When RouterB receives an updated advertising network 130.1.0.0, it drops the update because one of its own interfaces is connected to network 130.1.0.0. The router will not accept an update for a network to which its own interface is connected.

The solution to this problem is to add a secondary address to the interfaces connecting RouterA to RouterB. The secondary address should be in the same major network as the discontiguous network and use the same

subnet mask. As shown in Figure 6−12, with the addition of the secondary address, the networks are no longer discontiguous.

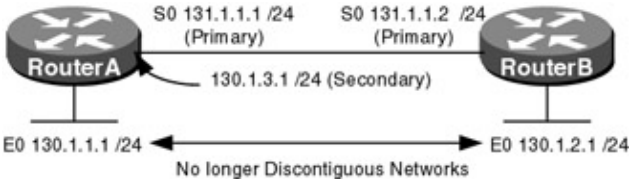


Figure 6−12: Secondary address is used to support discontiguous networks

This configuration will demonstrate the use of secondary addresses to eliminate discontiguous networks across a RIP network. RouterA and RouterB are connected serially via a crossover cable. RouterA will act as the DCE supplying clock to RouterB. The IP addresses are assigned as per Figure 6−11. All routers will be configured for RIP and advertise all connected networks.

Router Configurations

The configuration for RouterA is as follows (key RIP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
ip subnet-zero
!
interface Ethernet0/0
 ip address 130.1.1.1 255.255.255.0
 no ip directed-broadcast
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0/0
 ip address 131.1.1.1 255.255.255.0
 no ip directed-broadcast
 no ip mroute-cache
 clockrate 1000000 ← Acts as DCE providing clock
!
router rip ← Enables the RIP routing process on the router
 network 130.1.0.0
 network 131.1.0.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
```

RouterB

```
Current configuration:
!
```

```
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
ip subnet-zero
!
interface Ethernet0/0
 ip address 130.1.2.1 255.255.255.0
 no ip directed-broadcast
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0/0
 ip address 131.1.1.2 255.255.255.0
no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
!
router rip ← Enables the RIP routing process on the router
 network 130.1.0.0
 network 131.1.0.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
```

Monitoring and Testing the Configuration

Enable RIP update debugging on RouterB with the command **debug ip rip**. The following is the output from the command. Notice that RouterB is receiving an update from RouterA for network 130.1.0.0. As mentioned earlier, when an update is sent across a network boundary (in this case, network 131.1.0.0), it is summarized at the natural class mask.

```
RouterB#
00:43:19: RIP: received v1 update from 131.1.1.1 on Serial0/0
00:43:19:      130.1.0.0 in 1 hops
```

Display the routing table on RouterB with the command **show ip route**. The following is the output. Notice that RouterB does not have an entry for network 130.1.1.0 in its routing table. When RouterB receives the update for network 130.1.0.0, it drops it because it has a direct connection to the same network.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     130.1.0.0/24 is subnetted, 1 subnets
C       130.1.2.0is directly connected, Ethernet0/0
     131.1.0.0/24 is subnetted, 1 subnets
```

```
C       131.1.1.0is directly connected, Serial0/0
```

The solution to this problem is to add a secondary address to the interfaces connecting RouterA to RouterB. The secondary address should be in the same major network as the discontiguous network and use the same subnet mask. As shown in Figure 6−12, add a secondary address to the serial interface of RouterA and RouterB. The following commands add a secondary IP address to the serial interface of RouterA and RouterB:

```
RouterA#conf terminal
RouterA(config)#interface s0/0
RouterA(config−if)#ip address 130.1.3.1 255.255.255.0 secondary

RouterB#conf terminal
RouterB(config)#interface s0/0
RouterB(config−if)#ip address 130.1.3.2 255.255.255.0 secondary
```

Enable RIP update debugging on RouterB with the command **debug ip rip**. The following is the output from the command. Notice that RouterB is now receiving an update from RouterA for network 130.1.1.0. Since the update is no longer being sent across a network boundary, it is not summarized at the natural class mask.

```
00:59:03: RIP: received v1 update from 130.1.3.1 on Serial0/0
00:59:03:      130.1.1.0 in 1 hops
```

Display the routing table on RouterB with the command **show ip route**. The following is the output. Notice that there is now an entry for network 130.1.1.0 in its routing table.

```
RouterB#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, ia − IS−IS inter area
       * − candidate default, U − per−user static route, o − ODR
       P − periodic downloaded static route

Gateway of last resort is not set

     130.1.0.0/24  is subnetted, 4 subnets
C       130.1.3.0 is directly connected, Serial0/0
C       130.1.2.0 is directly connected, Ethernet0/0
R       130.1.1.0 [120/1] via 130.1.3.1, 00:00:25, Serial0/0
R       130.1.0.0 [120/1] via 131.1.1.1, 00:00:25, Serial0/0
     131.1.0.0/24 is subnetted, 2 subnets
R       131.1.0.0 [120/1] via 130.1.3.1, 00:00:25, Serial0/0
C       131.1.1.0 is directly connected, Serial0/0
```

Troubleshooting RIP

The Cisco IOS provides many tools for troubleshooting routing protocols. The following is a list of key commands along with a sample output from each that will aid in troubleshooting RIP.

{debug ip rip} This exec command displays information on RIP routing transactions. The output shows whether the router is sending or receiving an update, the networks contained in the update, and the metric or hop count for each.

```
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (148.1.1.1)
 network 10.0.0.0, metric 1
 network 192.1.1.0, metric 1
network 148.1.0.0, metric 1
RIP: received v1 update from 192.1.1.2 on Serial0
193.1.1.0 in 1 hops
```

{debug ip routing} This exec command displays information on routing table updates. The output shows what routes have been added or deleted, and for the distance vector routing protocols, what routes are in holddown.

```
RT: delete route to 152.1.0.0 via 192.1.1.2, rip metric [120/2]
 RT: no routes to 152.1.0.0, entering holddown
 RT: delete route to 193.1.1.0 via 192.1.1.2, rip metric [120/1]
 RT: no routes to 193.1.1.0, entering holddown
RT: add 193.1.1.0/24  via 192.1.1.2, rip metric [120/1]
```

{show ip protocol} This exec command displays the parameters and current state of the active routing protocol process. The output shows the routing protocol used, timer information, inbound and outbound filter information, protocols being redistributed, and the networks that the protocol is routing for. This command is very useful for troubleshooting a router that is sending bad router updates.

```
RouterA#show ip protocols
Routing Protocol is "rip"
  Sending updates every 5 seconds, next due in 0 seconds
  Invalid after 15 seconds, hold down 15, flushed after 30
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 1, receive any version
    Interface        Send     Recv     Key-chain
    Ethernet0        1        1        2
    Loopback0        1        1        2
    Serial0          1        1        2
  Routing for Networks:
    10.0.0.0
    192.1.1.0
    148.1.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.1.1.2            120      00:00:01
  Distance: (default is 120)
```

{show ip route rip} This exec command quickly displays all of the routes learned via RIP. This is a quick way to verify that a router is receiving RIP updates.

```
RouterA#show ip route rip
R    152.1.0.0/16 [120/2] via 192.1.1.2, 00:00:00, Serial0
R    193.1.1.0/24 [120/1] via 192.1.1.2, 00:00:00, Serial0
```

Conclusion

RIP is the most widely used Interior Gateway Routing Protocol (IGRP) in large organizations today, especially in organizations that have a large UNIX−based routing environment. However, it is worth noting the limitations one faces when deploying a large RIP network:

- RIP uses a 4−bit metric to count router hops to destinations. This limits the size of a RIP network, which cannot contain more than 15 hops to a destination. This is a severe limitation when trying to implement a typical modern large−scale network.
- RIP uses hop count as a routing metric, which does provide the most optimal path selection. More advanced protocols like IGRP use complex metrics to determine the optimal path.
- RIP was deployed prior to subnetting and has no direct subnet support. RIP assumes that all interfaces on the network have the same mask.
- RIP broadcasts a complete list of networks that it can reach every 30 seconds by default. This can amount to a significant amount of traffic, especially on low−speed links.

- RIP has no security features built in. A RIP−enabled device will accept RIP updates from any other device on the network. More modern routing protocols, such as OSPF, enable the router to authenticate updates.

Chapter 7: Interior Gateway Routing Protocol

Overview

Topics Covered in This Chapter

- Detailed technology overview
- Mechanisms to prevent routing loops
- IGRP route types
- Basic IGRP configuration
- Passive Interfaces
- IGRP unequal−cost load balancing
- IGRP Unicast updates
- IGRP timer configurations
- Detailed troubleshooting examples

Introduction

Interior Gateway Routing Protocol (IGRP) is a Cisco proprietary distance vector routing protocol developed in 1986 to address the limitations of RIP. Although RIP works quite well in small homogenous internetworks, its small hop count (16) severely limits the size of the network and its single metric (hop count) does not provide the routing flexibility needed in complex networks. IGRP addresses the shortcomings of RIP by allowing the network to grow up to 255 hops and by providing a wide range of metrics (link reliability, bandwidth, internetwork delay, and load) to provide routing flexibility in today's complex networks.

Technology Overview

Routing Loops

The problem with a first− or second−generation distance vector routing protocol like IGRP is that each router does not have a complete view of the network. Routers must rely on the neighboring routers for network reachability information, thus creating a slow convergence problem in which inconsistencies arise because routing update messages propagate slowly across the network. To reduce the likelihood of routing loops caused by inconsistencies across the network, IGRP uses the following mechanisms: split horizons, poison reverse updates, holddown counters, and flash updates.

Split Horizon

The rule of split horizon states that it is never useful for a router to advertise a route back in the direction from which it came. When split horizons is enabled on a router's interface, the router records the interface over which a route was received and does not propagate information about that route back out that interface.

The Cisco router allows you to disable split horizons on a per−interface basis. This is sometimes necessary in NBMA (Non Broadcast Multiple Access) hub and spoke environments. In Figure 7−1, RouterB is connected to RouterC and RouterA via Frame Relay, both PVCs terminating on one physical interface on RouterB.

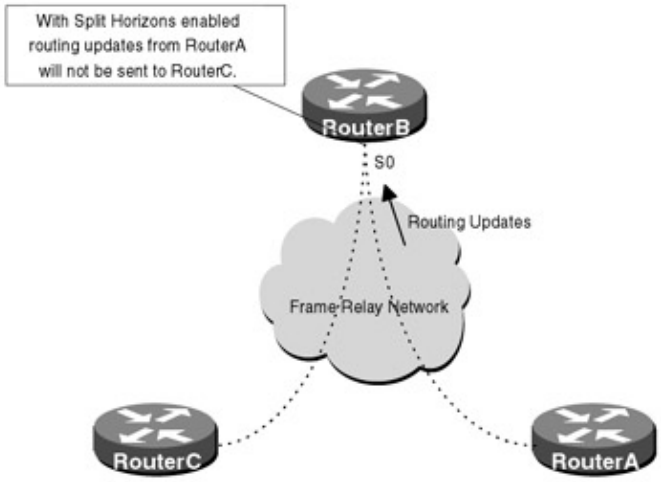


Figure 7−1: Split horizons

In Figure 7−1, if split horizon is not disabled on RouterB's Serial interface, then RouterC will not receive RouterA's routing advertisements and vice versa. Use the **no ip split−horizon** interface subcommand to disable split horizons.

Poison Reverse

Split horizon is a scheme used by the router to avoid problems caused by advertising routes back to the router from which they were learned. The split horizon scheme omits routes learned from one neighbor in updates sent to that neighbor. Split horizon with poison reverse includes the routes in updates but sets the metric to 4294967295.

When a router sees increases in routing metrics, it generally indicates a routing loop. The router then sends poison reverse updates to remove the route and place it in holddown. In Cisco's implementation of IGRP, poison reverse updates are sent if a route metric has increased by a factor of 1.1 or greater.

By setting the hop count to max and advertising the route back to its source, it is possible to immediately break a routing loop. Otherwise, the inaccurate route will stay in the routing table until it times out. The disadvantage to poison reverse is that it increases the size of the routing table.

Holddown

Holddown timers prevent the router from accepting routing information about a network for a fixed period of time after the route has been removed from the routing table. The idea is to make sure all routers have received the information, and no router sends out an invalid route. For example, in Figure 7−2 RouterB advertises bad information to RouterC because of the delay in the routing update. Holddown counters would prevent this from happening because RouterC would not install a new route to NetworkA for 280 seconds. By then RouterB would have converged with the proper routing information.

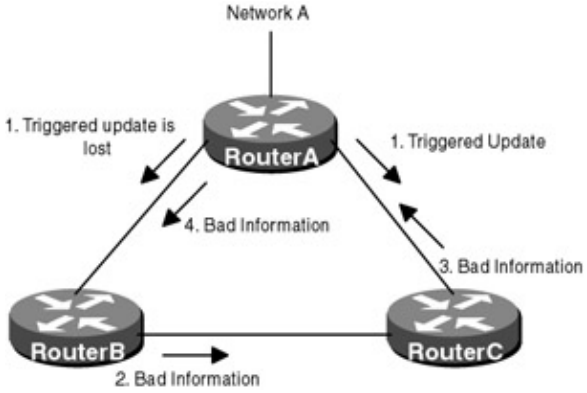


Figure 7−2: Routing loop

Flash Updates

Flash updates are an attempt to speed up convergence time; whenever the metric of a route changes, the router must send an update message immediately. A flash update message is sent immediately, regardless of when the regular update message is scheduled to be sent.

IGRP Routes

IGRP advertises three types of routes: interior, system, and exterior (see Figure 7−3). Interior routes are routes between subnets that are attached to the same router interface. System routes are routes to networks that are in the same autonomous system, and exterior routes are routes to networks outside the autonomous system.

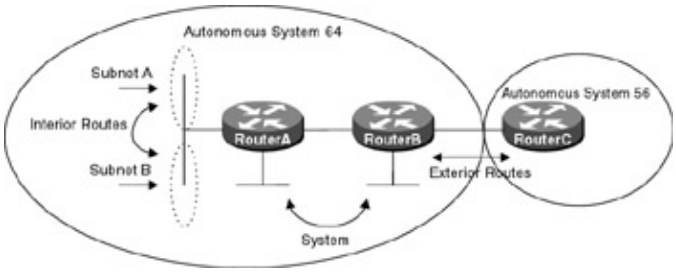


Figure 7−3: IGRP route types

Commands Discussed in This Chapter

- **clear ip route**
- **debug ip igrp events**
- **debug ip igrp transaction**
- **neighbor (ip−address)**
- **network (network−number)**
- **router igrp (autonomous−system number)**
- **show ip route igrp**
- **show ip protocol**
- **timers basic**
- **traffic−share {balanced | min}**
- **variance (multiplier)**

Definitions

clear ip route: This exec command removes one or more routes from the routing table. The command allows you to enter a specific route or use a * to remove all routes.

debug ip igrp events: This exec command displays information on IGRP routing transactions. It displays all IGRP routing updates that are sent or received by the router.

debug ip igrp transaction: This exec command displays transaction information on Interior Gateway Routing Protocol (IGRP) routing transactions.

neighbor: This router configuration command permits the point−to−point (nonbroadcast) exchange of routing information. By default, IGRP routing advertisements are sent as broadcast traffic. The neighbor command allows advertisements to be sent to define neighbors as unicast traffic.

network: This router configuration command specifies a list of networks on which the IGRP routing process will run. This command sends IGRP updates to the interfaces that are specified. If an interface's network is not specified, it will not be advertised in any IGRP update.

router igrp: This global command enables the Interior Gateway Routing Protocol (IGRP) routing process on the router. The autonomous system number used is a routing domain identifier, not a true ASN as defined in

RFC 1930.

show ip route igrp: This exec command displays the all IGRP learned routes.

show ip protocol: This exec command displays the current state of the active routing protocol process.

timers basic: This router configuration command allows the user to tune the IGRP timers.

***update*:** The update timer sets the rate in seconds at which routing updates are sent. The default is 90 seconds.

***invalid*:** The invalid timer sets the interval of time in seconds after which a route is declared invalid. The timer is started if the route is not present in the regular update message. The default is 270 seconds.

***holddown*:** The holddown timer sets the interval in seconds during which routing information regarding better paths is suppressed. The idea is to make sure all routers have received the information, and no router sends out an invalid route. The default is 280 seconds.

***flush*:** The flush timer sets in seconds the amount of time that must pass before a route is removed from the routing table. The default is 630 seconds.

traffic−share: This router configuration command controls how traffic is distributed among routes when there are multiple routes to the same destination network that have different costs. The traffic can be distributed proportionately to the ratios of the metrics or can be set to only use routes that have minimum costs.

variance: This router configuration command controls the load balancing over multiple IGRP paths. This command allows the administrator to load−balance across multiple paths even if the metrics of the paths are different. By default, the amount of variance is set to 1 (equal−cost load balancing). The variance command allows you to define how much worse an alternate path can be before that path is allowed to be used. For example, if the variance is set to 4, the router will load−balance across paths that are up to four times as bad as the best route.

IOS Requirements

IGRP first became available in IOS 10.0.

Lab #28: Basic IGRP Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables
- One Cisco Rolled cable for access the console port of the router

Configuration Overview

This configuration will demonstrate basic routing using Interior Gateway Routing Protocol (IGRP). As per Figure 7−4, RouterA, RouterB, and RouterC will use IGRP to advertise routing information.

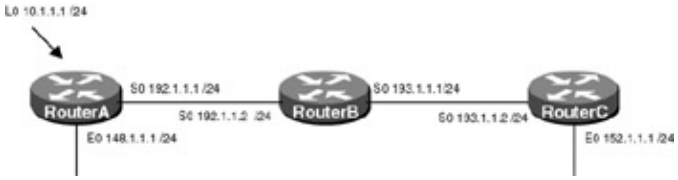


Figure 7−4: Basic IGRP

RouterA, RouterB, and RouterC are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 7−4. All routers will be configured for IGRP and will advertise all connected networks.

Router Configurations

The configurations for the three routers in this example are as follows (key IGRP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a
                      test point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 148.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows
                the interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
router IGRP 64 ← Enables the IGRP routing process on the router
 network 10.0.0.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 148.1.0.0
 network 192.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.0
```

```
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router igrp 64 ← Enables the IGRP routing process on the router
network 192.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 193.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
!
router igrp 64 ← Enables the IGRP routing process on the router
network 152.1.0.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
network 193.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Like RIP, IGRP is a very simple protocol to configure and troubleshoot. Show the IP routing table on RouterA with the **show ip route** command; what follows is the output from this command. Notice that two networks were learned via IGRP, 152.1.0.0 and 193.1.1.0.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
      i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     148.1.0.0/24 is subnetted, 1 subnets
C       148.1.1.0 is directly connected, Ethernet0
I    152.1.0.0/16 [100/10576] via 192.1.1.2, 00:00:40, Serial0
C    192.1.1.0/24 is directly connected, Serial0
I    193.1.1.0/24 [100/10476] via 192.1.1.2, 00:00:40, Serial0
```

From RouterA, monitor the routing updates being passed using the **debug ip igrp transactions** command; what follows is the output from this command. Notice that on interface serial 0 the router does not advertise the networks it learns from RouterB (152.1.0.0 and 193.1.1.0), but on all other interfaces these networks are advertised. This is split horizon at work; remember when split horizon is enabled, the router will never advertise a route back in the direction from which it came.

```
RouterA#debug ip igrp transactions
IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
 network 10.0.0.0, metric=501
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
 network 148.1.0.0, metric=1100
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Serial0 (192.1.1.1)
 network 10.0.0.0, metric5501
 network 148.1.0.0, metric51100
```

Now disable split horizons on RouterA using the interface configuration command **no ip split horizons**.

```
RouterA(config)#int s0
RouterA(config−if)#no ip split−horizon
```

From RouterA, monitor the routing updates being passed using the **debug ip igrp transactions** command; what follows is the output from this command. Notice that now all routes are being advertised out serial 0, including the routes learned from RouterB on serial 0.

```
RouterA# debug ip igrp transactions
IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
 network 10.0.0.0, metric=501
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
 network 148.1.0.0, metric=1100
 network 152.1.0.0, metric=10576
```

```
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Serial0 (192.1.1.1)
 network 10.0.0.0, metric=501
 network 148.1.0.0, metric=1100
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
```

From RouterA, delete the IGRP process and add a new process using autonomous system 56, with the following commands.

```
RouterA#configure terminal
RouterA(config)#no router igrp 64
RouterA(config)#router igrp 56
RouterA(config−router)#network 10.0.0.0
RouterA(config−router)# network 148.1.0.0
RouterA(config−router)# network 192.1.1.0
```

Show the IP routing table on RouterA with the **show ip route** command; what follows is the output from this command. Notice that no networks are being learned via IGRP; this is because the autonomous system numbers are different. The autonomous system number must match or the routers will not exchange routing information.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
 D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
 N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
 E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
 i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
 U − per−user static route, o − ODR

Gateway of last resort is not set

 10.0.0.0/24 is subnetted, 1 subnets
C 10.1.1.0 is directly connected, Loopback0
 148.1.0.0/24 is subnetted, 1 subnets
C 148.1.1.0 is directly connected, Ethernet0
C 192.1.1.0/24 is directly connected, Serial0
```

Lab #28: Basic IGRP Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables
- One Cisco Rolled cable for access the console port of the router

Configuration Overview

This configuration will demonstrate basic routing using Interior Gateway Routing Protocol (IGRP). As per Figure 7−4, RouterA, RouterB, and RouterC will use IGRP to advertise routing information.

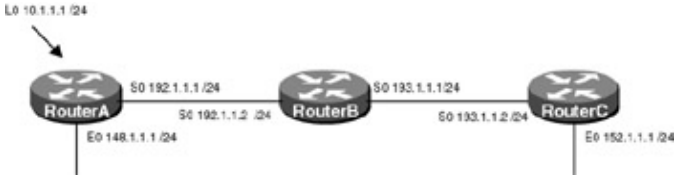


Figure 7−4: Basic IGRP

RouterA, RouterB, and RouterC are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 7−4. All routers will be configured for IGRP and will advertise all connected networks.

Router Configurations

The configurations for the three routers in this example are as follows (key IGRP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a
                      test point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 148.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows
                the interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
router IGRP 64 ← Enables the IGRP routing process on the router
 network 10.0.0.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 148.1.0.0
 network 192.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.0
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!
```

```
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router igrp 64 ← Enables the IGRP routing process on the router
network 192.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 193.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
!
router igrp 64 ← Enables the IGRP routing process on the router
network 152.1.0.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
network 193.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Like RIP, IGRP is a very simple protocol to configure and troubleshoot. Show the IP routing table on RouterA with the **show ip route** command; what follows is the output from this command. Notice that two networks were learned via IGRP, 152.1.0.0 and 193.1.1.0.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
```

```
        D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
        N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
        E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
        U − per−user static route, o − ODR

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     148.1.0.0/24 is subnetted, 1 subnets
C       148.1.1.0 is directly connected, Ethernet0
I    152.1.0.0/16 [100/10576] via 192.1.1.2, 00:00:40, Serial0
C    192.1.1.0/24 is directly connected, Serial0
I    193.1.1.0/24 [100/10476] via 192.1.1.2, 00:00:40, Serial0
```

From RouterA, monitor the routing updates being passed using the **debug ip igrp transactions** command; what follows is the output from this command. Notice that on interface serial 0 the router does not advertise the networks it learns from RouterB (152.1.0.0 and 193.1.1.0), but on all other interfaces these networks are advertised. This is split horizon at work; remember when split horizon is enabled, the router will never advertise a route back in the direction from which it came.

```
RouterA#debug ip igrp transactions
IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
 network 10.0.0.0, metric=501
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
 network 148.1.0.0, metric=1100
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Serial0 (192.1.1.1)
 network 10.0.0.0, metric5501
 network 148.1.0.0, metric51100
```

Now disable split horizons on RouterA using the interface configuration command **no ip split horizons**.

```
RouterA(config)#int s0
RouterA(config−if)#no ip split−horizon
```

From RouterA, monitor the routing updates being passed using the **debug ip igrp transactions** command; what follows is the output from this command. Notice that now all routes are being advertised out serial 0, including the routes learned from RouterB on serial 0.

```
RouterA# debug ip igrp transactions
IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
 network 10.0.0.0, metric=501
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
 network 148.1.0.0, metric=1100
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Serial0 (192.1.1.1)
 network 10.0.0.0, metric=501
 network 148.1.0.0, metric=1100
 network 152.1.0.0, metric=10576
 network 192.1.1.0, metric=8476
 network 193.1.1.0, metric=10476
```

From RouterA, delete the IGRP process and add a new process using autonomous system 56, with the following commands.

```
RouterA#configure terminal
RouterA(config)#no router igrp 64
RouterA(config)#router igrp 56
RouterA(config−router)#network 10.0.0.0
RouterA(config−router)# network 148.1.0.0
RouterA(config−router)# network 192.1.1.0
```

Show the IP routing table on RouterA with the **show ip route** command; what follows is the output from this command. Notice that no networks are being learned via IGRP; this is because the autonomous system numbers are different. The autonomous system number must match or the routers will not exchange routing information.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
 D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
 N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
 E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
 i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
 U − per−user static route, o − ODR

Gateway of last resort is not set

 10.0.0.0/24 is subnetted, 1 subnets
C 10.1.1.0 is directly connected, Loopback0
 148.1.0.0/24 is subnetted, 1 subnets
C 148.1.1.0 is directly connected, Ethernet0
C 192.1.1.0/24 is directly connected, Serial0
```

Lab #29: Passive Interface Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables
- One Cisco Rolled cable for accessing the console port of the router

Configuration Overview

This configuration will demonstrate the use of the passive−interface command, which allows IGRP−enabled routers to listen to, but not send, routing updates out a particular interface. The passive−interface router configuration command is typically used when the network router configuration command configures more interfaces than is desirable. In Figure 7−5, for example, RouterA has three subnets (10.1.1.0/24, 10.1.2.0/24, and 10.1.3.0/24) defined. Since IGRP is a classful protocol, when it is enabled it is turned on for the classful network of 10.0.0.0. This encompasses all three subnets; the passive interface command allows the user to turn off IGRP advertisements on a particular interface (subnet).

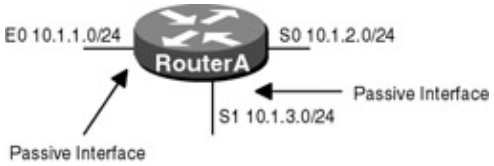


Figure 7−5: Passive interface

The reason IGRP changes the network entry from 10.1.1.0 to 10.0.0.0 is that IGRP is considered a "classful" protocol. By that we mean that it recognizes the IP address class of the network address that you type and assumes the proper mask. For a Class A network like this one, the mask is 255.0.0.0, yielding 10.0.0.0 (no matter what you actually type as the last two octets). The network statement tells the routing protocol to route on the interfaces where the network address matches the one specified in the network statement.

In this lab scenario, the user only wishes to send IGRP updates out network 10.1.2.0, so interface E0 (10.1.1.0) and S1 (10.1.3.0) are made passive interfaces.

RouterA, RouterB, and RouterC are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 7−6. All routers will be configured for IGRP; RouterB and RouterC will advertise all connected networks. RouterA's interface S0 will be passive and will not advertise any routing information; however, it will still receive routing updates.

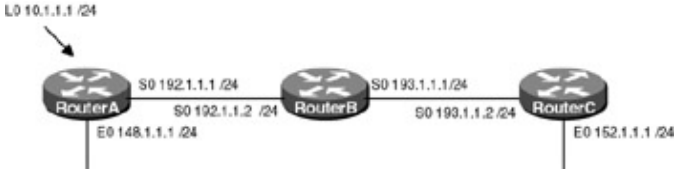


Figure 7−6: IGRP Passive Interface Configuration

Router Configurations

The configurations for the three routers in this example are as follows (key IGRP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a
                      test point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 148.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
!
router IGRP 64 ← Enables the IGRP routing process on the router
passive-interface Serial0 ← Disables the sending of IGRP updates on interface
                            Serial 0
network 10.0.0.0 ← Specifies what interfaces will receive and send IGRP routing
                   updates. It also specifies what networks will be advertised
 network 148.1.0.0
 network 192.1.1.0
!
```

```
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.0
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router igrp 64 ← Enables the IGRP routing process on the router
network 192.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 193.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
!
router igrp 64 ← Enables the IGRP routing process on the router
network 152.1.0.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
network 193.1.1.0
!
```

```
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display the information about IGRP with the command **show ip protocols**, noticing that RouterA's serial interface is passive.

```
RouterA#show ip protocols
Routing Protocol is "igrp 64"
  Sending updates every 90 seconds, next due in 31 seconds
  Invalid after 270 seconds, hold down 280, flushed after 630
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  IGRP maximum hopcount 100
  IGRP maximum metric variance 1
  Redistributing: igrp 64
  Routing for Networks:
    10.0.0.0
    148.1.0.0
    192.1.1.0
  Passive Interface(s):
    Serial0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.1.1.2            100      00:00:48
  Distance: (default is 100)
```

What follows is the output from the **debug ip igrp transactions** command on RouterA. Notice that IGRP updates are only being sent out interface Ethernet 0 and Loopback 0; also note that interface S0 is still receiving IGRP updates.

```
RouterA#debug ip igrp transactions

IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
      network 10.0.0.0, metric=501
      network 152.1.0.0, metric=10576
      network 192.1.1.0, metric=8476
      network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
      network 148.1.0.0, metric=1100
      network 152.1.0.0, metric=10576
      network 192.1.1.0, metric=8476
      network 193.1.1.0, metric=10476
IGRP: received update from 192.1.1.2 on Serial0
      network 152.1.0.0, metric 10576 (neighbor 8576)
      network 193.1.1.0, metric 10476 (neighbor 8476)
```

What follows is the output from the **show ip route** command on RouterA and RouterC. Note that RouterA has learned all of the routes from RouterC, but RouterC has no routes from RouterA.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     148.1.0.0/24 is subnetted, 1 subnets
C       148.1.1.0 is directly connected, Ethernet0
I    152.1.0.0/16 [100/10576] via 192.1.1.2, 00:00:29, Serial0
C    192.1.1.0/24 is directly connected, Serial0
I    193.1.1.0/24 [100/10476] via 192.1.1.2, 00:00:29, Serial0

RouterC#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     152.1.0.0/24  is subnetted, 1 subnets
C       152.1.1.0 is directly connected, Ethernet0
I    192.1.1.0/24 [100/10476] via 193.1.1.2, 00:00:13, Serial0 ← Route from RouterB
C    193.1.1.0/24 is directly connected, Serial0
```

Lab #30: IGRP Unequal−Cost Load Balancing

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports and one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Ethernet cables
- One Ethernet hub
- Two Cisco DTE/DCE crossover cables
- One Cisco Rolled cable for accessing the console port of the router

Overview

IGRP can be configured to load−balance on up to four unequal−cost paths to a given destination. This feature is known as unequal−cost load balancing and is set using the variance command. By default, the router will load balance across up to four equal−cost paths, and the variance command lets you set how much worse an alternate path can be (in terms of metrics) and still be used to load−balance across.

For example, if RouterA has two routes to network 1.1.1.1, one with a cost of 4 and one with a cost of 8, by default the route will only use the path with a cost of 4 when sending packets to 1.1.1.1. However, if a variance of 2 is set, the router will load−balance across both paths. This is because the route with the cost of 8 is within the variance, which in this case can be up to two times as bad as the preferred route (4 (preferred route) * 2 = 8).

Configuration Overview

This configuration will demonstrate the use of the variance command, which allows IGRP−enabled routers to load−balance across unequal−cost paths. The variance command will be set on RouterA so that both paths to network 3.3.3.3 are used.

RouterA, RouterB, and RouterC are connected serially via a crossover cable, and RouterA and RouterB are also connected via an Ethernet hub. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 7−7. All routers will be configured for IGRP; RouterA will be configured to load−balance traffic that is destined for 3.3.3.3 over two unequal−cost paths.

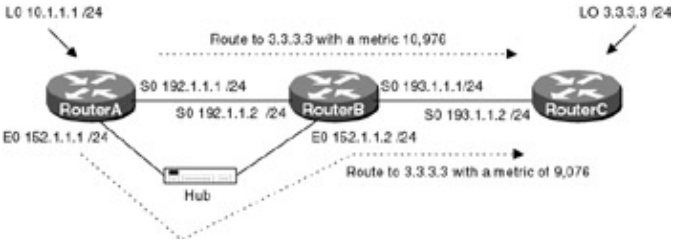


Figure 7−7: IGRP unequal−cost load balancing

Router Configurations

The configurations for the three routers in this example are as follows (key IGRP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a
                      test point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 keepalive
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
!
router IGRP 64 ← Enables the IGRP routing process on the router
variance 2
 network 10.0.0.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 152.1.0.0
 network 192.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
```

```
end
```

RouterB

```
!
version 11.0
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0
 no ip address
ip address 152.1.1.2 255.255.255.0
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router igrp 64 ← Enables the IGRP routing process on the router
network 192.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
network 193.1.1.0
network 152.1.0.0

!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
!
router igrp 64 ← Enables the IGRP routing process on the router

network 193.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised

network 3.0.0.0
!
no ip classless
!
```

```
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display the routing table on RouterA with the command **show ip route**, noticing that there are two routes to network 3.3.3.3, one via the Ethernet interface and one via the serial interface. The cost to reach the network over each path is different; however, since the variance is set to two, as long as the cost of the second path is not greater than two times the preferred path, that route will be used.

Let's take a closer look at this. The best route to network 3.0.0.0 is via the Ethernet interface with a cost of 9,076. Since the variance is set to 2, as long as the cost on any other route to network 3.0.0.0 is below 18,152 (9,076 * 2), then it will be used. Since the cost of the route via the Serial interface is 10,976, which is lower than 18,152, it is used.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

I    3.0.0.0/8 [100/9076] via 152.1.1.1, 00:00:04, Ethernet0
               [100/10976] via 192.1.1.2, 00:00:04, Serial0
10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     152.1.0.0/24 is subnetted, 1 subnets
C       152.1.1.0 is directly connected, Ethernet0
C    192.1.1.0/24 is directly connected, Serial0
I    193.1.1.0/24 [100/10476] via 192.1.1.2, 00:00:27, Serial0
                  [100/8576] via 152.1.1.1, 00:00:27, Ethernet0
```

From RouterA, display the route to host 3.3.3.3 with the command **show ip route 3.3.3.3,** noticing that both routes are shown, although there is an asterisk next to the first route. The asterisk indicates that the next packet leaving RouterA destined for host 3.3.3.3 will use this route.

```
RouterA#show ip route 3.3.3.3
Routing entry for 3.0.0.0/8
  Known via "igrp 64", distance 100, metric 9076
  Redistributing via igrp 64
  Advertised by igrp 64 (self originated)
  Last update from 192.1.1.2 on Serial0, 00:00:18 ago
  Routing Descriptor Blocks:
  * 152.1.1.1, from 152.1.1.1, 00:00:18 ago, via Ethernet0
      Route metric is 9076, traffic share count is 1
      Total delay is 26000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
    192.1.1.2, from 192.1.1.2, 00:00:18 ago, via Serial0
      Route metric is 10976, traffic share count is 1
      Total delay is 45000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
```

From RouterA, ping host 3.3.3.3.

```
RouterA#ping 3.3.3.3

Type the escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
!!!!!
```

Now from RouterA, display the route to host 3.3.3.3 with the command **show ip route 3.3.3.3,** noticing that the asterisk is now by the second route. This is because the router is load−balancing the traffic destined for network 3.0.0.0 over both links.

```
RouterA#show ip route 3.3.3.3
Routing entry for 3.0.0.0/8
 Known via "igrp 64", distance 100, metric 9076
 Redistributing via igrp 64
 Advertised by igrp 64 (self originated)
 Last update from 192.1.1.2 on Serial0, 00:00:06 ago
 Routing Descriptor Blocks:
   152.1.1.1, from 152.1.1.1, 00:00:06 ago, via Ethernet0
     Route metric is 9076, traffic share count is 1
     Total delay is 26000 microseconds, minimum bandwidth is 1544 Kbit
     Reliability 255/255, minimum MTU 1500 bytes
     Loading 1/255, Hops 1
 * 192.1.1.2, from 192.1.1.2, 00:00:07 ago, via Serial0
     Route metric is 10976, traffic share count is 1
     Total delay is 45000 microseconds, minimum bandwidth is 1544 Kbit
     Reliability 255/255, minimum MTU 1500 bytes
     Loading 1/255, Hops 1
```

Remove the variance command on RouterA with the router configuration command **no variance.**

```
outerA#configure terminal
RouterA(config)#router igrp 64
RouterA(config−router)#no variance
```

From RouterA, display the route to host 3.3.3.3 with the command **show ip route 3.3.3.3**, noticing that only one route is being used. This is the route with the lowest metric; no load balancing is being performed.

Lab #31: IGRP Timer Configurations

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables
- One Cisco Rolled cable for console port access

Configuration Overview

This configuration will demonstrate using the timer basic command to set the four configurable IGRP timers (update, invalid, holddown, and flush timers). Depending on the network topology, it may become necessary to change the update timers, which control the rate in seconds at which routing updates are sent. For example, if the access link is 56 Kbps, generating IGRP updates every 90 seconds might not be the most efficient use of bandwidth. However, by increasing the update timer, you also increase the convergence time of the network.

The three other IGRP timers are all dependent on the value of the update timer. The invalid timer should be at least three times the value of the update timer; the holddown timer should be at least three times the value of the update timer; and the flush timer must be at least the sum of the invalid and holddown timers.

Each time a route is updated, which is dependent on the update interval, the invalid timer is reset. By default, if a route is not seen in an update for 270 seconds, the route is put in holddown, which means that the router will use the route to route packets but will not announce the route in its updates. It also means that the router will not install any other route to this destination until the holddown counter expires. This happens after 630 seconds, at which time the route is flushed from the routing table.

Note The update interval must be the same value on neighboring routers.

RouterA, RouterB, and RouterC are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 7−8. All routers will be configured for IGRP. RouterA, RouterB, and RouterC will advertise all connected networks. The timers on each router will be as follows:

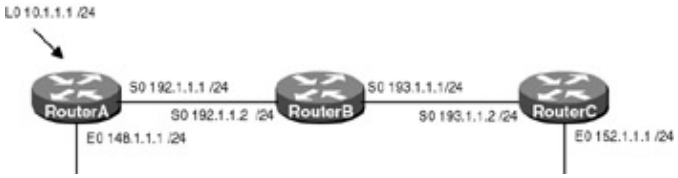


Figure 7−8: IGRP timer configuration

- Update 5
- Invalid=15
- Holddown=15
- Flush=30

With these timers set, updates are broadcast every 5 seconds. If a route is not heard from in 15 seconds, the route is declared unusable (invalid). Any information received in routing updates about this particular network is suppressed for an additional 15 seconds (holddown). At the end of the suppression period, the route is flushed from the routing table.

Router Configurations

The configurations for the three routers in this example are as follows (key IGRP configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a
                      test point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 148.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows
                the interface to stay up when it is not attached to a hub
!
interface Serial0
```

```
 ip address 192.1.1.1 255.255.255.0
!
!
router igrp 64 ← Enables the IGRP routing process on the router
 timers basic 5 15 15 30 ← Updates are broadcast every 5 seconds. If a router
                            is not heard from in 15 seconds, the route is
                            declared unusable. Further information is suppressed
                            for an additional 15 seconds. At the end of the
                            suppression period, the route is flushed from the
                            routing table

network 10.0.0.0 ← Specifies what interfaces will receive and send IGRP routing
                   updates. It also specifies what networks will be advertised
 network 148.1.0.0
 network 192.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.0
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router igrp 64 ← Enables the IGRP routing process on the router
timers basic 5 15 15 30 ← Updates are broadcast every 5 seconds. If a router is
                           not heard from in 15 seconds, the route is declared
                           unusable. Further information is suppressed for an
                           additional 15 seconds. At the end of the suppression
                           period, the route is flushed from the routing table

network 192.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 193.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows
                the interface to stay up when it is not attached to a hub
!
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
!
router igrp 64 ← Enables the IGRP routing process on the router
timers basic 5 15 15 30 ← Updates are broadcast every 5 seconds. If a router is
                          not heard from in 15 seconds, the route is declared
                          unusable. Further information is suppressed for an
                          additional 15 seconds. At the end of the suppression
                          period, the route is flushed from the routing table

network 152.1.0.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
network 193.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

What follows is the output from the **show ip protocols** command on RouterA. Note that the timers have been changed.

```
RouterA#show ip protocols
Routing Protocol is "igrp 64"
  Sending updates every 5 seconds, next due in 1 seconds
  Invalid after 15 seconds, hold down 15, flushed after 30
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  IGRP maximum hopcount 100
  IGRP maximum metric variance 1
  Redistributing: igrp 64
  Routing for Networks:
    10.0.0.0
    148.1.0.0
    192.1.1.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.1.1.2            100      00:01:14
  Distance: (default is 100)
```

Lab #32: Configuring Unicast IGRP Updates

Equipment Needed

The following equipment is need to perform this lab exercise:

- One Cisco router with one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco Rolled cable for console port access

The IGRP neighbor command permits the point−to−point (nonbroadcast) exchange of routing information. This command can be used in combination with the passive−interface router configuration command to exchange information between a subset of routers and access servers all connected to the same LAN.

For example, in Figure 7−9 RouterA wishes to only send routing updates to RouterB on the Ethernet LAN. Since IGRP is a broadcast protocol, by default it will send updates to all devices on the Ethernet LAN. To prevent this from happening, RouterA's Ethernet interface is configured as passive. In this case, however, a neighbor router configuration command is included. This command permits the sending of routing updates to a specific neighbor. One copy of the routing update is generated per defined neighbor.

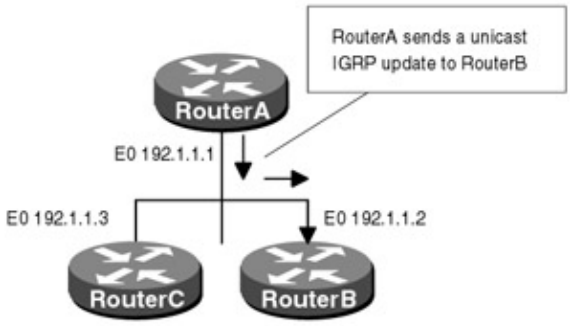


Figure 7−9: IGRP unicast updates

Router Configurations

The configuration for RouterA is as follows (key IGRP configurations for RouterA are highlighted in bold).

RouterA

```
Building configuration...

Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 192.1.1.1 255.255.255.0
 no keepalive ← Disables the keep-alive on the Ethernet interface, allows
                the interface to stay up when it is not attached to a hub
!
!
```

```
router igrp 64 ← Enables the IGRP routing process on the router
passive-interface Ethernet0 ← Disables the sending of IGRP updates on interface
                              Ethernet 0

network 192.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
network 1.1.1.1
neighbor 192.1.1.2 ← Permits the point-to-point (nonbroadcast) exchange of
                     routing information
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

What follows is the output from the **debug ip igrp events** command. Note that IGRP updates are being sent to the unicast address 192.1.1.2 on Ethernet 0 and the broadcast address 255.255.255.255 on interface loopback 0.

```
RouterA#debug ip igrp events
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
IGRP: Update contains 0 interior, 1 system, and 0 exterior routes.
IGRP: Total routes in update: 1
IGRP: sending update to 192.1.1.2 via Ethernet0 (192.1.1.1)
IGRP: Update contains 0 interior, 1 system, and 0 exterior routes.
```

Troubleshooting IGRP

The Cisco IOS provides many tools for troubleshooting routing protocols. What follows is a list of key commands along with a sample output from each that will aid in troubleshooting IGRP.

{debug ip igrp events} This exec command displays summary information on Interior Gateway Routing Protocol (IGRP) routing messages. The information contains the source and destination of each routing update, the type of routes contained in the update (system, exterior, and interior), and the number of routes in each update.

```
RouterA# debug ip igrp events
IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
IGRP: Update contains 0 interior, 2 system, and 0 exterior routes.
IGRP: Total routes in update: 2
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
IGRP: Update contains 0 interior, 2 system, and 0 exterior routes.
IGRP: Total routes in update: 2
IGRP: sending update to 255.255.255.255 via Serial0 (192.1.1.1)
IGRP: Update contains 0 interior, 2 system, and 0 exterior routes.
IGRP: Total routes in update: 2
```

{debug ip igrp transactions} This exec command displays transaction information on Interior Gateway Routing Protocol (IGRP) routing transactions. The information contains the source and destination of each update, the routes that are received or being advertised, and the metric of each route.

```
RouterA#debug ip igrp transactions
IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
```

```
      network 10.0.0.0, metric=501
      network 152.1.0.0, metric=10576
      network 192.1.1.0, metric=8476
      network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Loopback0 (10.1.1.1)
      network 148.1.0.0, metric=1100
      network 152.1.0.0, metric=10576
      network 192.1.1.0, metric=8476
      network 193.1.1.0, metric=10476
IGRP: sending update to 255.255.255.255 via Serial0 (192.1.1.1)
      network 10.0.0.0, metric=501
      network 148.1.0.0, metric=1100
IGRP: received update from 192.1.1.2 on Serial0
      network 152.1.0.0, metric 10576 (neighbor 8576)
      network 193.1.1.0, metric 10476 (neighbor 8476)
```

{debug ip routing} This exec command displays information on routing table updates. The output shows what routes have been added or deleted, and for the distance vector routing protocols, what routes are in holddown.

```
RouterA#debug ip routing
RT: add 148.1.1.0/24 via 0.0.0.0, connected metric [0/0]
RT: add 10.1.1.0/24 via 0.0.0.0, connected metric [0/0]
RT: add 192.1.1.0/24 via 0.0.0.0, connected metric [0/0]
```

{show ip protocol} This exec command displays the parameters and current state of the active routing protocol process. The output shows the routing protocol used, timer information, inbound and outbound filter information, protocols being redistributed, and the networks that the protocol is routing for. This command is very useful for troubleshooting a router that is sending bad router updates.

```
RouterA#show ip protocols
Routing Protocol is "igrp 64"
  Sending updates every 5 seconds, next due in 1 seconds
  Invalid after 15 seconds, hold down 15, flushed after 30
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  IGRP maximum hopcount 100
  IGRP maximum metric variance 1
  Redistributing: igrp 64
  Routing for Networks:
    10.0.0.0
    148.1.0.0
    192.1.1.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.1.1.2            100      00:16:28
  Distance: (default is 100)
```

{show ip route igrp} This exec command quickly displays all of the routes learned via IGRP. This is a quick way to verify that a router is receiving IGRP updates.

```
RouterA#show ip route igrp
I    152.1.0.0/16 [100/10576] via 192.1.1.2, 00:00:00, Serial0
I    193.1.1.0/24 [100/10476] via 192.1.1.2, 00:00:00, Serial0
```

Conclusion

Since IGRP is a distance vector protocol, it suffers from some of the same limitations as RIP, namely slow convergence. However, unlike RIP, IGRP can scale across large networks. IGRP's maximum hop count of 255 allows the protocol to be run on even the largest networks, and since IGRP uses four metrics (Internetwork delay, bandwidth, reliability, and load) instead of one (hop count) to calculate route feasibility, this intuitive route selection provides optimal performance in even the most complex networks.

Chapter 8: OSPF

Overview

Topics Covered in This Chapter

- Detailed technology overview
- OSPF terminology
- OSPF protocol packets
- Basic OSPF configuration
- OSPF route summarization
- Configuring OSPF priority "DR Election"
- Configuring OSPF virtual links
- OSPF stub areas
- Configuring OSPF neighbor authentication
- Configuring OSPF on NBMA network "Non−Broadcast Model"
- Configuring OSPF on NBMA network "Broadcast Model"
- Configuring OSPF on NBMA network "Point−to−Multipoint"
- Configure OSPF interface parameters
- Detailed troubleshooting examples

Introduction

Open Shortest Path First (OSPF) is a link state routing protocol developed for IP networks. It was developed to be used within a single autonomous system to distribute routing information. The following chapter will discuss terminology, key concepts, configuration issues, and troubleshooting techniques for OSPF−enabled networks.

OSPF Terminology

When dealing with OSPF, it is important to understand the terminology being used.

Autonomous System: Abbreviated AS, it is a group of routers that are under the control of a single administrative entity, for example, all of the routers belonging to a particular corporation.

LSA: Link state advertisement is used to describe the local state of a router. The LSAs contain information about the state of the router's interfaces and the state of any adjacencies that are formed. The LSAs are flooded through the network. The information contained in the LSA sent by each router in the domain is used to form the router's topological database. From this database, a shortest path is calculated to each destination.

Area: An area is a collection of routers that have an identical topological database. OSPF uses areas to break an AS into multiple link−state domains. Since the topology of an area is invisible to another area, no flooding leaves an area; this greatly reduces the amount of routing traffic within an AS. Areas are used to contain link state updates and allow administrators to build hierarchical networks.

Cost: The metric that the router uses to compare routes to the same destination. The lower the cost, the more preferred the route is. OSPF calculates the cost of using a link based on bandwidth; the higher the bandwidth, the lower the cost and the more preferable the route is.

Router ID: The router ID is a 32−bit number assigned to each OSPF−enabled router, which is used to uniquely identify the router within an autonomous system. The router ID calculated at boot time is the highest loopback address on the router; if no loopback interfaces are configured, the highest IP address on the router is used.

Adjacency: OSPF forms adjacencies between neighboring routers in order to exchange routing information. On a multiaccess network, each router forms an adjacency with the designated router.

Designated Router: Abbreviated DR, it is used to reduce the number of adjacencies that need to be formed on a multiaccess network such as Ethernet, TokenRing, or Frame Relay. The reduction in the amount of adjacencies formed greatly reduces the size of the topological database. The DR becomes adjacent with all other routers on the multiaccess network. The routers send their LSAs to the DR, and the DR is responsible for forwarding them throughout the network. The idea behind a DR is that routers have a central point to send information to, versus every router exchanging information with every other router on the network.

Backup Designated Router: Abbreviated BDR, it is formed on a multiaccess network and is responsible for taking over for the DR if it should fail.

Inter−Area route: A route that is generated in an area other than the local one, inside the current OSPF routing domain.

Intra−Area route: A route that is within one area.

Neighbor: Neighbors are routers that share a common network. For example, two routers on an Ethernet interface are said to be neighbors.

Flooding: A technique used to distribute LSAs between routers.

Hello: A hello packet is used to establish and maintain neighbor relationships. The hello packet is also used to elect a DR for the network.

Technology Overview

Let's start with a brief introduction to OSPF before going into more detail. OSPF uses a link state algorithm to calculate the shortest path to all destinations in each area. When a router is first enabled, or if any routing changes occur, the router configured for OSPF floods link state advertisements (LSAs) to all routers in the same hierarchical area. The LSAs contain information on the state of the router's links and the router's relationship to its neighboring routers. From the collection of LSAs, the router forms what is called a link state database. All routers in the area have an identical database describing the area's topology.

The router then runs the Dijkstra algorithm using the link state database to form a shortest path tree to all destinations inside the area. From this shortest path tree, the IP routing table is formed. Any changes that occur on the network are flooded via link state packets and will cause the router to recalculate the shortest path tree using the new information.

Link State Routing Protocol

OSPF uses a link state algorithm to calculate the shortest path to all known destinations. Link state refers to the state of a router's interface (up, down, IP address, type of network, and so on) and the router's relationship to its neighbors (how the routers are connected on the network). The link state advertisements are flooded to each router and are used to create a topological database.

The Dijkstra algorithm is run on each router using the topological database, created by all LSAs received from all of the routers in the area. The algorithm places each router at the root of the tree and calculates the shortest

path to each destination based on the cost to reach that network.

Flooding

Flooding is the process of distributing link state advertisements between adjacent routers. The flooding procedure carries the LSA one hop further from its point oforigin.

Since all routers in an OSPF domain are interconnected via adjacencies, the information disseminates throughout the network. To make this process reliable, each link state advertisement must be acknowledged.

Dijkstra Algorithm

The Dijkstra algorithm is the heart of OSPF. Once the router receives all link state advertisements, it then uses the Dijkstra algorithm to calculate the shortest path to each destination inside the area based on the cumulative cost to reach that destination. Each router will have a complete view of the network topology inside the area. It builds a tree with itself as the route and has the entire path to any destination network or host.

However, the view of the topology from one router will be different from that of another because each router uses itself as the root of the tree. The Dijkstra algorithm is run any time a router receives a new link state advertisement.

Areas

OSPF uses areas to segment the AS and contain link state updates (see Figure 8−1). LSAs are only flooded within an area, so separating the areas reduces the amount of routing traffic on a network.

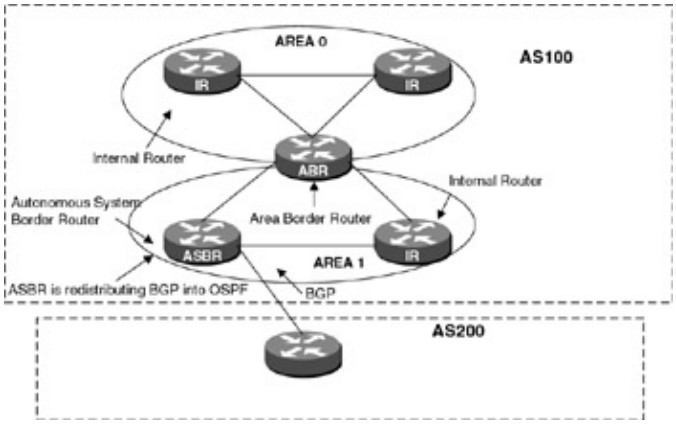


Figure 8−1: OSPF areas

Each router within an area has an identical topological database as all other routers in the same area. A router in multiple areas has a separate topological database for each area it is connected to.

Routers that have all of their interfaces within the same area are called internal routers (IRs). Routers that connect areas within the same AS are called area border routers (ABRs), and routers that act as gateways redistributing routing information from one AS to another AS are called autonomous system border routers (ASBRs).

Backbone Area 0

OSPF has a concept of a backbone area referred to as area 0. If multiple areas are configured, one of these areas must be configured as area 0. The backbone (area 0) is the center for all areas; that is, all areas must have a connection to the backbone. In cases where an area does not have direct physical connectivity to the backbone, a virtual link must be configured. Virtual links will be discussed later in the chapter.

All areas inject routing information into the backbone area (area 0), and the backbone propagates routing information back to each area.

Designated Router (DR)

All multiaccess networks with two or more attached routers elect a designated router (DR). The DR concept enables a reduction in the number of adjacencies that need to be formed on a network. In order for OSPF−enabled routers to exchange routing information, they must form an adjacency with one another. If a DR was not used, then each router on a multiaccess network would need to form an adjacency with every other router (since link state databases are synchronized across adjacencies). This would result in N−1 adjacencies.

Instead, all routers on a multiaccess network form adjacencies only with the DR and BDR. Each router sends the DR and BDR routing information, and the DR is responsible for flooding this information to all adjacent routers and originating a network link advertisement on behalf of the network. The backup designated router is used in case the DR fails.

The reduction in adjacencies reduces the volume of routing protocol traffic as well as the size of the topological database.

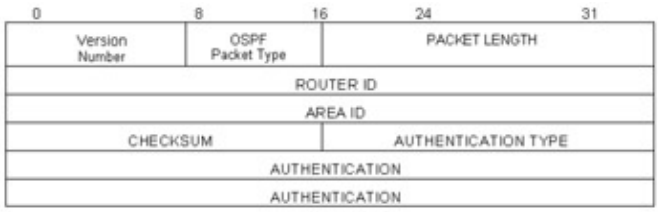
The DR is elected using the hello protocol, which was described earlier in this chapter. The election of the DR is determined by the router priority, which is carried in the hello packet. The router with the highest priority will be elected the DR; if a tie occurs, the router with the highest router ID is selected.

The router ID is the IP address of the loopback interface. If no loopback is configured, the router ID is the highest IP address on the router. The router priority can be configured on the router interface with the **ip ospf priority** command.

When a router first becomes active on a multiaccess network, the router checks to see if there is currently a DR for the network. If a DR is present, the router accepts the DR regardless of what its priority is. Once a DR is elected, no other router can become the DR unless the DR fails. If there is no DR present on the network, then the routers negotiate the DR based on router priority.

OSPF Protocol Packets

The OSPF protocol runs directly over IP protocol 89 and begins with the same 24−byte header, as shown here:



There are five OSPF packet types:

Type	Packet Name	Protocol Function
1	Hello	Discover and maintain neighbors
2	Database Description	Summarize database contents
3	Link State Request	Request for database information
4	Link State Update	Database update
5	Link State Acknowledgment	Acknowledgment

Hello packets: The hello protocol is responsible for discovering neighbors and maintaining the neighbor relationship. Hello packets are sent periodically out the router's interface, depending on the network type. The hello protocol is also responsible for electing a DR on multiaccess networks. The role of the DR is discussed later in the chapter.

Database Description packets: Database description packets are OSPF type 2 packets. These packets are responsible for describing the contents of the link state database of the router and are one of the first steps to forming an adjacency. Database descriptor packets are sent in a poll response manner. One router is designated the master, and the other the slave. The master sends database polls, which are acknowledged by the database descriptor packets, sent by the slave.

Link State Request packets: Link state request packets are OSPF type 3 packets. Once the complete databases are exchanged between routers using the database description packets, the routers compare the database of their neighbor with their own database. At this point, the router may find that parts of the neighbor's database may be more up−to−date than its own. If so, the router requests these pieces using the link state request packet.

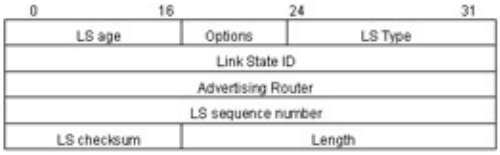
Link State Update packet: Link state update packets are OSPF packet type 4. The router uses a flooding technique to pass LSA. There are multiple LSA types (Router, Network, Summary, and External), which are described in detail later in this chapter.

Link State Acknowledgment packet: Link state acknowledgments are OSPF type 4 packets, which are used to acknowledge the receipt of LSAs. This acknowledgment makes the OSPF flooding procedure reliable.

Link State Advertisements

Each of the router types mentioned in Figure 8−1 generates a different type of link state advertisements. Although there are more LSA types, we will only be discussing the four major LSAs.

All link state advertisements begin with the same 20−byte header, as seen here:



LS age: The time in seconds since the link state advertisement was originated.

Options: The optional capabilities supported by the router.

LS type: Type of link state advertisement.

Link state ID: This field identifies the portion of the Internet environment that is being described by the advertisement.

Advertising router: The router ID of the router that originated the packet.

LS sequence number: Used to detect old or duplicate link state advertisements.

LS checksum: Checksum of the complete contents of the link state advertisement.

Length: The length in bytes of the link state advertisement, including the 20−byte header.

Router Link

Each router in the area generates a router LSA (type 1 LSA). This advertisement describes the state and cost of the router's interfaces to that area. All of the router's links to the area must be described in a single router LSA. The router LSAs are only flooded throughout a single area.

Network Link

Network link advertisements are type 2 LSAs. The DR for each multiaccess network that has more than one attached router originates a network advertisement. The advertisement describes all of the routers attached to the network as well as the DR itself.

Summary Link

Summary LSAs are type 3 and 4 LSAs. The ABR generates Summary LSAs, which describe a route to a single destination. The summary LSA is advertised within the single area, and the destination described is external to the area yet part of the same autonomous system. Only intra−area routes are advertised in the backbone.

External Link

The autonomous system border router (ASBR) generates an external type 5 LSA, which advertises each destination known to the router that is external to the AS. AS external type 5 LSAs are used to advertise default routes into the AS.

There are two types of external routes, external type 1 and external type 2. The difference between the two is the way the cost or metric of the route is calculated. External type 1 routes use the external cost plus the internal cost of reaching a route. External type 2 only uses the external cost of reaching the route. Type 2 routes are always preferred over type 1 routes and are the default type for any route that is redistributed into OSPF.

How It Works

When an OSPF−enabled router first comes online, it sends out a hello packet to the multicast address 224.0.0.5. This packet is sent out to all OSPF−enabled interfaces periodically, depending on the interface type. For broadcast media, such as Ethernet or Token Ring or point−to−point interfaces, the hello packet is sent every 10 seconds. On an NBMA (non−broadcast multiple access medium), such as Frame Relay or ATM, the hello packet is sent out every 30 seconds.

The hello packets are not only used to build neighbor relationships and discover what neighbors are on the same wire, they are also used to describe any optional capabilities of the router, such as whether the router is in a regular or stub area. The hello packet is also used to elect the DR designated router on multiaccess networks.

After the neighbor is discovered, bidirectional communication is assured, and a designated router is elected (on a multiaccess medium); the router attempts to form an adjacency with the neighboring router.

To form an adjacency, the routers must synchronize their databases; to do this, each router describes its databases to the other by sending a sequence of database description packets. This process is called the database exchange process and will be covered in more detail later in the chapter.

During the database exchange process, the two routers form a master/slave relationship. Each database description packet sent by the master contains a sequence number. The slave acknowledges receipt of the packet by echoing back the sequence number.

During the database exchange process, each router checks its own database to see if any of the link state advertisements received by its neighbor are more recent than its own database copy. If any are, the router makes note of this, and after the database exchange process is over, the router requests updated LSAs using a link state request packet. Each router responds to the link state request using a link state update. When the requesting router receives the updated LSA, it acknowledges the packet. When the database description process is complete and all link state requests have been updated, the databases are synchronized.

How an Adjacency Is Formed

In order for a router to exchange link state database information with another router, an adjacency must be formed. This is a key part of OSPF and therefore needs to be completely understood.

On a Cisco router, you can check the status of the adjacency using the **show ip ospf neighbor** command. What follows is the output from the command; notice that the state of the adjacency is full, which means that RouterB's database is synchronized with neighbor 1.1.1.1, which is RouterA.

```
RouterB#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   FULL/BDR        0:00:37     10.10.3.1       Ethernet0
```

There are 5 states that neighbor routers go through before fully forming an adjacency or having a full neighbor state. Figure 8−2 shows an example of how an adjacency is formed between two neighboring routers on a broadcast medium. RouterA and RouterB both connect to an Ethernet network, and RouterB is configured with a higher DR priority.

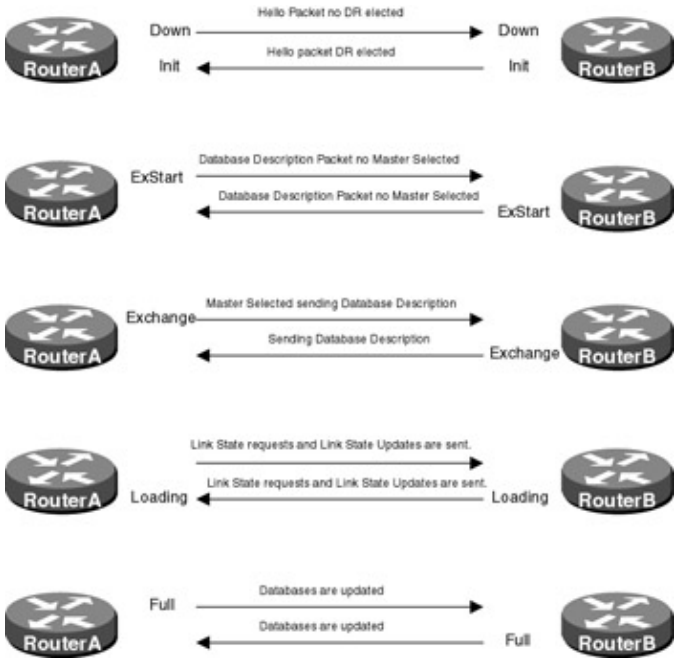


Figure 8−2: How a router forms adjacencies

When RouterA and RouterB first come online, they both initialize and begin sending hello packets. At this point in time, neither router knows of the presence of the other router on the network and no DR is selected. RouterB hears the hello from RouterA changing the state of the adjacency from down to Initializing (Init). This can be seen from the **show ip ospf neighbor** command on RouterB.

```
RouterB#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   INIT/DROTHER    0:00:39     10.10.3.1       Ethernet0
```

At this point, the routers have seen themselves in the hello packet from their neighbors, and bidirectional communication is established. The adjacency changes from Initializing to 2WAY. This can be seen from the **show ip ospf neighbor** command on RouterB.

At the end of this stage, the DR and BDR are elected for the network and the router then decides whether or not to form an adjacency with its neighbor. On a multiaccess network, routers will only form adjacencies with the DR and BDR on the network.

```
Routerb#show ip ospf neighbor
```

```
Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   2WAY/DROTHER    0:00:36     10.10.3.1       Ethernet0
```

RouterB in the next hello packet indicates to RouterA that it is the designated router (DR) for the link. At this point, the state of the adjacency changes from Initializing to Exchange (Exstart). This can be seen from the **show ip ospf neighbor** command on RouterB. During the Exstart state, a master−slave relationship is formed between the two routers and the slave router adopts the master's database description (DD) sequence number.

```
routerb#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   EXSTART/BDR     0:00:32     10.10.3.1       Ethernet0
                                á RouterA is the Backup DR
```

After the master−slave relationship is formed and the routers agree on a common DD sequence number, the routers begin to exchange database description packets. At this point, the state of the adjacency changes from Exstart to Exchange. This can be seen from the **show ip ospf neighbor** command on RouterB.

```
routerb#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   EXCHANGE/DR     0:00:38     10.10.3.1       Ethernet0
```

After the complete databases are exchanged between routers using the database description (DD) packets, the routers compare the database of their neighbor with their own database. At this point, the router may find that parts of the neighbor's database may be more up−to−date than its own. If so, the router requests these pieces using a link state request packet. At this point, the state of the adjacency is loading. This can be seen from the **show ip ospf neighbor** command on RouterB.

```
routerb#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   LOADING/DR      0:00:38     10.10.3.1       Ethernet0
```

After the link state update requests have all been satisfied, RouterA and RouterB databases are deemed synchronized and the routers are fully adjacent. This can be seen from the **show ip ospf neighbor** command on RouterB. See Figure 8−2.

```
RouterB#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   FULL/BDR        0:00:37     10.10.3.1       Ethernet0
```

Sniffer Trace of Database Synchronization

Using a sniffer on an Ethernet LAN, let's examine how the database synchronization process between two routers works. Figure 8−3 depicts the setup.

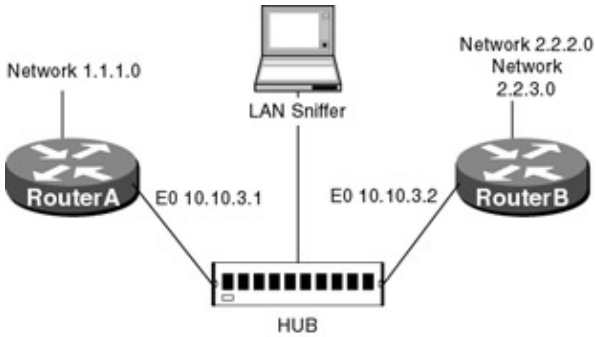


Figure 8−3: Database synchronization process

Step1: RouterA and RouterB send out hello packets; notice in the first two packets, no DR (designated router) is elected. In the third hello packet, RouterB is elected the DR.

Hello from RouterA

```
Open Shortest Path First Protocol
        Version = 2, Type = Hello (1), Message len = 44
        Source gateway IP address = 1.1.1.1 ← Router ID of RouterA
        Area ID = 0.0.0.0 ← AREA of advertising Interface
        Checksum = 0xFA9C
        Authentication type = 0 (None), Value = 0000000000000000
        Network mask = 255.255.255.0
        Interval = 10 second(s) ← Hello Interval
        Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
        Router priority = 1 ← Priority used to select DR
        Router dead interval = 40 second(s) ← If the Router does not receive a
                                              Hello in 40 seconds the neighbor
                                              is delcared dead
        Designated router = 0.0.0.0 ← No DR is elected
```

Hello from RouterB

```
Open Shortest Path First Protocol
        Version = 2, Type = Hello (1), Message len = 44
        Source gateway IP address = 2.2.3.2 ← Router ID of RouterB
        Area ID = 0.0.0.0 ← AREA of advertising Interface
        Checksum = 0xF79A
        Authentication type = 0 (None), Value = 0000000000000000
        Network mask = 255.255.255.0
        Interval = 10 second(s) ← Hello Interval
        Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
        Router priority = 1 ← Priority used to select DR
        Router dead interval = 40 second(s) ← If the Router does not receive
a Hello in 40 seconds the neighbor is delcared dead
        Designated router = 0.0.0.0 ← No DR is elected
```

Hello from RouterB

```
Open Shortest Path First Protocol
        Version = 2, Type = Hello (1), Message len = 48
        Source gateway IP address = 2.2.3.2
        Area ID = 0.0.0.0
        Checksum = 0xDB7D
        Authentication type = 0 (None), Value = 0000000000000000
        Network mask = 255.255.255.0
        Interval = 10 second(s)
        Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
        Router priority = 1
        Router dead interval = 40 second(s)
        Designated router = 2.2.3.2 ← RouterB is elected DR
```

Step2: RouterB was selected the designated router and now the routers exchange database description packets. The first packet is just an initialization packet and contains no database information. RouterB becomes the master because it has the higher router ID.

Database Description Packet from RouterB

```
Open Shortest Path First Protocol
        Version = 2, Type = Database Desp. (2), Message len = 32
        Source gateway IP address = 2.2.3.2
        Area ID = 0.0.0.0
        Checksum = 0xD519
```

```
        Authentication type = 0 (None), Value = 0000000000000000
          Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
        Init = 1, More = 1, Master ← RouterB is the Master
        á
        This is an Initialization packet
      DD Sequence number = 8633 ← The sequence number is 8633
```

Step3: RouterA and RouterB continue to send database description packets with polls coming from the master and responses coming from the slave. Both the polls from the master and responses from the slave contain summaries of the link state database. This exchange is done when both the poll from the master and the response from the slave have the more bit (M−bit) off or set to zero.

Database Description Packet from RouterA

```
Open Shortest Path First Protocol
        Version = 2, Type = Database Desp. (2), Message len = 52
        Source gateway IP address = 1.1.1.1
        Area ID = 0.0.0.0
        Checksum = 0x5199
        Authentication type = 0 (None), Value = 0000000000000000
          Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
        Init = 0, More = 1, Slave
        DD Sequence number = 8633
        Link state advertisement #1:
          LS age = 40 seconds
        Optional capabilities = 0X22
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
          LS type = Router links
          Link state ID = 1.1.1.1
          Advertising router = 1.1.1.1
          LS Sequence number = 2147483650,   LS checksum = 0xE013,
          Length = 48
Database Description Packet from RouterB

Open Shortest Path First Protocol
        Version = 2, Type = Database Desp. (2), Message len = 52
        Source gateway IP address = 2.2.3.2
        Area ID = 0.0.0.0
        Checksum = 0x7A7B
        Authentication type = 0 (None), Value = 0000000000000000
          Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
        Init = 0, More = 1, Master
        DD Sequence number = 8634
        Link state advertisement #1:
          LS age = 40 seconds ← This is the age of the LSA
        Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
          LS type = Router links
          Link state ID = 2.2.3.2
          Advertising router = 2.2.3.2
          LS Sequence number = 2147483651,   LS checksum = 0xCE1C,
          Length = 60
Database Description Packet from RouterA

Open Shortest Path First Protocol
        Version = 2, Type = Database Desp. (2), Message len = 32
        Source gateway IP address = 1.1.1.1
        Area ID = 0.0.0.0
```

```
        Checksum = 0xD821
        Authentication type = 0 (None), Value = 0000000000000000
          Optional capabilities = 0X02
             . . . . ..1. = external routing capability
             . . . .  . . . 0 = no Type of Service routing capability
       Init = 0, More = 0, Slave ← The M-bit is set to zero this is the end of
                                   the database exchange
        DD Sequence number = 8634
```

Step4: Now that RouterA and RouterB have exchanged database information, each router looks at its own database and compares it with the information that it just received from its neighbor. If the information in its database is not as current as the information received from the neighbor, the router will request this information be sent. This is done using a link state request packet.

Link State Request from RouterB

```
Open Shortest Path First Protocol
        Version = 2, Type = LS Req. (3), Message len = 36
        Source gateway IP address = 2.2.3.2
        Area ID = 0.0.0.0
        Checksum = 0xF4CF
        Authentication type = 0 (None), Value = 0000000000000000
        Link state advertisement #1:
          LS type = Router links
          LS ID = 1.1.1.1
          Advertising router = 1.1.1.1
```

Link State Request from RouterA

```
Open Shortest Path First Protocol
        Version = 2, Type = LS Req. (3), Message len = 36
        Source gateway IP address = 1.1.1.1
        Area ID = 0.0.0.0
        Checksum = 0xF1CD
        Authentication type = 0 (None), Value = 0000000000000000
        Link state advertisement #1:
          LS type = Router links
          LS ID = 2.2.3.2
          Advertising router = 2.2.3.2
```

Step5: After the router receives a link state request packet from its neighbor, it sends the piece of the database that is requested; when the router receives the update, it sends an acknowledgment packet back to the sender acknowledging receipt of the packet.

Link State Update from RouterA

```
Open Shortest Path First Protocol
        Version = 2, Type = LS Upd. (4), Message len = 76
        Source gateway IP address = 1.1.1.1
        Area ID = 0.0.0.0
        Checksum = 0x611E
        Authentication type = 0 (None), Value = 0000000000000000
        Number of advertisements = 1
        Link state advertisement #1:
          LS age = 41 seconds
        Optional capabilities = 0X22
             .... ..1. = external routing capability
             .... ...0 = no Type of Service routing capability
          LS type = Router links
          Link state ID = 1.1.1.1
          Advertising router = 1.1.1.1
          LS Sequence number = 2147483650,   LS checksum = 0xE013,
          Length = 48
          Router type flag = Unknown (0x00)
          Number of router links = 2
          Router link #1:
```

```
            Link ID = 10.10.3.0
            Link data = 255.255.255.0
            Link type = Connection to a stub network
            Number of TOS = 0, TOS 0 metric = 10
          Router link #2:
            Link ID = 1.1.1.1
            Link data = BROADCAST
            Link type = Connection to a stub network
            Number of TOS = 0, TOS 0 metric = 1
        Link state advertisement #2:
```

Link State Update from RouterB

```
Open Shortest Path First Protocol
        Version = 2, Type = LS Upd. (4), Message len = 88
        Source gateway IP address = 2.2.3.2
        Area ID = 0.0.0.0
        Checksum = 0x7FEE
        Authentication type = 0 (None), Value = 0000000000000000
        Number of advertisements = 1
       Link state advertisement #1:
          LS age = 41 seconds
        Optional capabilities = 0X02
             .... ..1. = external routing capability
             .... ...0 = no Type of Service routing capability
          LS type = Router links
          Link state ID = 2.2.3.2
          Advertising router = 2.2.3.2
          LS Sequence number = 2147483651,   LS checksum = 0xCE1C,
          Length = 60
          Router type flag = Unknown (0x00)
          Number of router links = 3
          Router link #1:
            Link ID = 2.2.2.2
            Link data = BROADCAST
            Link type = Connection to a stub network
            Number of TOS = 0, TOS 0 metric = 1
          Router link #2:
            Link ID = 2.2.3.2
            Link data = BROADCAST
            Link type = Connection to a stub network
            Number of TOS = 0, TOS 0 metric = 1
          Router link #3:
            Link ID = 10.10.3.0
            Link data = 255.255.255.0
            Link type = Connection to a stub network
            Number of TOS = 0, TOS 0 metric = 10
        Link state advertisement #2:
```

Link State Acknowledgment from RouterA

```
Open Shortest Path First Protocol
        Version = 2, Type = LS Ack. (5), Message len = 64
        Source gateway IP address = 1.1.1.1
        Area ID = 0.0.0.0
        Checksum = 0x7F58
        Authentication type = 0 (None), Value = 0000000000000000
        Link state advertisement #1:
          LS age = 41 seconds
        Optional capabilities = 0X02
             .... ..1. = external routing capability
             ..... ..0 = no Type of Service routing capability
          LS type = Router links
          Link state ID = 2.2.3.2
          Advertising router = 2.2.3.2
          LS Sequence number = 2147483651,   LS checksum = 0xCE1C,
          Length = 60
        Link state advertisement #2:
          LS age = 1 seconds
        Optional capabilities = 0X02
```

```
              .... ..1. = external routing capability
              .... ...0 = no Type of Service routing capability
            LS type = Network links
            Link state ID = 10.10.3.2
            Advertising router = 2.2.3.2
            LS Sequence number = 2147483649,   LS checksum = 0x8D9D,
            Length = 32
Link State Acknowledgment from RouterB

Open Shortest Path First Protocol
        Version = 2, Type = LS Ack. (5), Message len = 44
        Source gateway IP address = 2.2.3.2
        Area ID = 0.0.0.0
        Checksum = 0x52B2
        Authentication type = 0 (None), Value = 0000000000000000
        Link state advertisement #1:
          LS age = 6 seconds
        Optional capabilities = 0X22
             .... ..1. = external routing capability
             .... ...0 = no Type of Service routing capability
          LS type = Router links
          Link state ID = 1.1.1.1
          Advertising router = 1.1.1.1
          LS Sequence number = 2147483651,   LS checksum = 0xFFD9,
          Length = 48
```

Step6: All link state requests have been fulfilled; the databases are synchronized, and the routers are fully adjacent.

OSPF Network Types

OSPF has four network types or models (broadcast, non−broadcast, point−to−point, and point−to−multipoint). Depending on the network type, OSPF works differently; understanding how OSPF works on each network model is essential in designing a stable and robust OSPF network.

Broadcast

The broadcast network type is the default type on LANs (Token Ring, Ethernet, and FDDI).

However, any interface can be configured as broadcast using the **ip ospf network** interface command.

- On a broadcast model, a DR and BDR are elected and all routers form adjacencies with the DR and BDR. This achieves optimal flooding because all LSAs are sent to the DR and the DR floods them to each individual router on the network.
- Neighbors do not need to be defined.
- All routers are on the same subnet.
- Care must be taken if the broadcast model is used on NBMA networks such as Frame Relay or ATM. Since a DR is elected, all routers must have physical connectivity to the DR. A full meshed environment should be used or the DR should be statically configured using the priority command to assure physical connectivity.
- The hello timer is 10 seconds, the dead interval is 40 seconds, and the wait interval is 40 seconds.

In Figure 8−4, RouterA and RouterC are connected via Frame Relay to RouterB. The network is a hub and spoke environment configured as an OSPF network type broadcast.

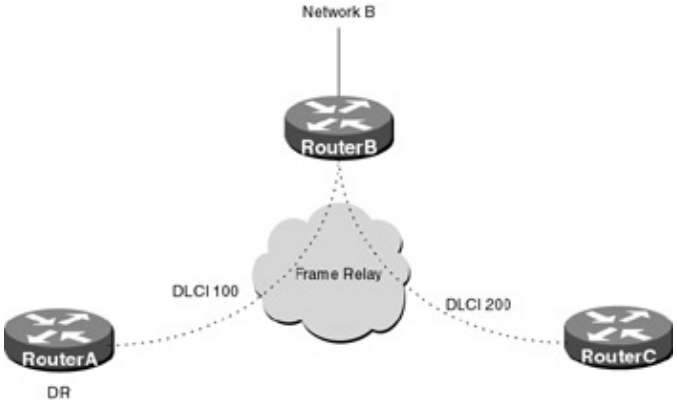


Figure 8−4: NBMA using network type broadcast
Since RouterB is the only router that has logical connectivity to each router on the network, it must be elected the DR.

If a broadcast model is used on an NBMA network, all routers should be fully meshed or care should be taken to which router is elected DR. In a hub and spoke environment, the hub should be configured as the DR.

What follows is the output from the command **show ip ospf interface ethernet 0;** note that the command shows the network type along with other key OSPF parameters.

```
Routerb#show ip ospf interface e0
Ethernet0 is up, line protocol is up
  Internet Address 10.10.3.2 255.255.255.0, Area 0
  Process ID 64, Router ID 2.2.3.2, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 9.9.21.9, Interface address 10.10.3.1
  Backup Designated router (ID) 2.2.3.2, Interface address 10.10.3.2
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 0:00:07
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 9.9.21.9  (Designated Router)
```

Non−Broadcast

The non−broadcast network type is the default type on serial interfaces configured for frame relay encapsulation. However, any interface can be configured as non−broadcast using the **ip ospf network** interface command.

- With the non−broadcast model, a DR and BDR are elected and all routers form adjacencies with the DR and BDR. This achieves optimized flooding because all LSAs are sent to the DR and the DR floods them to each individual router on the network.
- Due to the lack of broadcast capabilities, neighbors must be defined using the neighbor command.
- All routers are on the same subnet.
- As in the broadcast model, a DR is elected. Care must be taken to assure that the DR has logical connectivity to all routers on the network.
- The hello timer is 30 seconds, the dead interval is 120 seconds, and the wait interval is 120 seconds.

What follows is the output from the command **show ip ospf interface serial 0.2**, which is configured for Frame Relay encapsulation; note that the command shows the network type along with other key OSPF parameters.

```
Serial0.2 is up, line protocol is up
  Internet Address 193.1.1.1 255.255.255.0, Area 0
  Process ID 64, Router ID 2.2.3.2, Network Type NON_BROADCAST, Cost: 64
  Transmit Delay is 1 sec, State DOWN, Priority 1
  No designated router on this network
  No backup designated router on this network
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
```

Point−to−Point

The network type point−to−point is the default type on serial interfaces that are not using Frame Relay encapsulation or can be selected as a subinterface type point−to−point. A subinterface is a logical way of defining an interface. The same physical interface can be split into multiple logical interfaces; this was originally created to deal with issues caused by split horizons on NBMA networks.

The point−to−point model can be configured on any interface using the **ip ospf network point−to−point** interface command.

- With a point−to−point model, no DR and BDR are elected and directly connected routers form adjacencies.
- Each point−to−point link requires a separate subnet.
- The hello timer is 10 seconds, the dead interval is 40 seconds, and the wait interval is 40 seconds.

What follows is the output from the command **show ip ospf interface serial 0**, which is not configured for Frame Relay encapsulation; note that the command shows the network type along with other key OSPF parameters.

```
Routerb#show ip ospf interface s0
Serial0 is up, line protocol is up
  Internet Address 193.1.1.1 255.255.255.0, Area 0
  Process ID 64, Router ID 2.2.3.2, Network Type POINT_TO_POINT, Cost: 64
  Transmit Delay is 1 sec, State DOWN,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
```

Point−to−Multipoint

The network type point−to−multipoint can be configured on any interface using the **ip ospf network point−to−multipoint** interface command.

- No DR is elected.
- Neighbors do not need to be defined, because additional LSAs are used to convey neighbor router connectivity.
- One subnet is used for the whole network.
- The hello timer is 30 seconds, the dead interval is 120 seconds, and the wait interval is 120 seconds.

What follows is the output from the command **show ip ospf interface serial 0;** notice the command shows the network type along with other key OSPF parameters.

```
routerb#show ip ospf interface s0
Serial0 is up, line protocol is up
  Internet Address 193.1.1.1 255.255.255.0, Area 0
  Process ID 64, Router ID 2.2.3.2, Network Type POINT_TO_MULTIPOINT, Cost: 64
  Transmit Delay is 1 sec, State DOWN,
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
```

Commands Discussed in This Chapter

- **area** *area−id* **authentication** [*message−digest*]
- **area** *area−id* **range** *address mask*
- **debug ip ospf events**
- **debug ip ospf packet**
- **ip ospf authentication−key** *password*
- **ip ospf cost** *cost*

- **ip ospf dead−interval** *seconds*
- **ip ospf hello−interval** *seconds*
- **ip ospf message−digest−key** *keyid* **md5** *key*
- **ip ospf network** {*broadcast* | *non−broadcast* | *point−to−multipoint*}
- **ip ospf priority** number
- **neighbor** *ip−address* [*priority number*] [*poll−interval seconds*]
- **network** *address wildcard−mask* **area** *area−id*
- **passive−interface** *type number*
- **router ospf** *process−id*
- **show ip ospf** [*process−id*]
- **show ip ospf** [*process−id area−id] database*
- **show ip ospf interface** [*type number*]
- **show ip ospf neighbor** [*type number*] [*neighbor−id*] *detail*
- **show ip ospf virtual−links**

Definitions

area authentication: This router configuration command enables authentication for an OSPF area. Authentication assures that the routing information that the router is receiving is from a trusted source.

area range: This router configuration command consolidates and summarizes routes at an area boundary.

debug ip ospf events: The output for this exec command displays OSPF information such as adjacencies, flooding information, designated router selection, and shortest path first (SPF) calculation.

debug ip ospf packet: The output from this exec command displays detailed OSPF information for all OSPF packets received by the router.

ip ospf authentication−key: This interface configuration command is used to assign a password that will be used by neighboring routers if simple password authentication is configured on the router.

ip ospf cost: This interface configuration command sets the OSPF cost for sending a packet out that particular interface. Cisco uses bandwidth as a metric for best path selection; that is, an Ethernet interface is preferred over a T1 interface. This command would be used if the administrator wished to change the default cost of an interface or to prefer one identical interface to another. The path cost is calculated using the following formula: cost=100,000,000/bandwidth in bits per second. The cost is inversely proportional to the bandwidth of the link. The higher the bandwidth, the lower the cost.

ip ospf dead−interval: This interface configuration command sets the amount of time in seconds that a router will wait before declaring a neighbor router dead after not receiving a hello packet.

ip ospf hello−interval: This interface configuration command sets the interval in seconds that OSPF hello packets are sent out for the router interface.

ip ospf message−digest−key: This interface configuration command enables OSPF MD5 authentication.

ip ospf network: This interface configuration command will change the OSPF network type to a type other than the default for a given medium.

ip ospf priority: This interface configuration command is used to set the priority of the router interface that is used for DR election. The router with the highest priority will be elected the DR for the multiaccess network. If two routers have the same OSPF priority, the router with the highest router ID will be elected the DR. A router with a priority of zero is ineligible to become the DR or BDR. In some cases, particularly on non−broadcast multiple access networks (NBMA), such as Frame Relay or ATM, it is necessary to specify which router becomes the designated router for the network. In Figure 8−5, since RouterB is the only router that has full logical connectivity (meaning RouterB is the only router with a PVC to every router on the

network) to all the routers on the network, it is essential that it be elected the DR. If RouterA was elected DR, RouterC and RouterD would not be able to form an adjacency with the RouterA because they don't have logical connectivity.

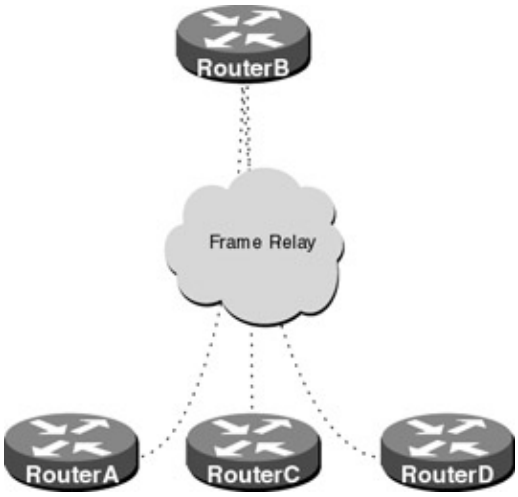


Figure 8−5: Designated router on NBMA networ

neighbor: This router configuration command is used on non−broadcast networks (due to the lack of broadcast capabilities) to define OSPF neighbors. On broadcast networks, this is not needed because neighbors are found using the hello protocol; on NBMA networks, the neighbor commands must be defined on any router that has the potential of becoming the DR or BDR. The neighbor command has two optional parameters associated with it: OSPF neighbor priority and the poll interval. The OPSF neighbor priority is an 8−bit number indicating the priority value of the non−broadcast neighbor associated with the IP address specified. The OSPF priority is used in DR and BDR election; the router with the highest priority will be elected the DR. A router with an OSPF priority of zero is not eligible for DR election. The poll interval is used if a neighboring router has become inactive (hello packets have not been seen for a period of time, which exceeds the routers dead interval); it may still be necessary to send hello packets to the dead neighbor. In this case, the hello packets will be sent at a reduced rate, which is the defined poll interval. It is recommend that the poll interval be larger than the hello interval; the default is 120 seconds.

network: This router configuration command defines on what interface OSPF will run and what OSPF area the interface will be in. A wildcard mask is used in conjunction with the IP address, which allows the user to specify one or more interfaces in an area using only a single command. When using a wildcard mask, a 0 means must match, a 1 means does not matter. In the following example, OSPF process 64 is defined; the first line only enables OSPF on one interface 192.1.1.1. Line 2 enables OSPF on any interface with an IP address 132.10.x.x; line 3 enables OSPF on all interfaces on the router.

```
router ospf 64
network 192.1.1.1  0.0.0.0 area 1
                á Wildcard Mask of all zero specifies one particular address
network 132.10.0.0  0.0.255.255 area 2
                 á This Wildcard Mask specifies that only the first two Octets
                    must match
network 0.0.0.0  255.255.255.255 area 0
                  á A wildcard mask of all ones specifies any interface on the
                     router
```

passive−interface: This router configuration command disables the sending and receiving of OSPF router information. The specified interface address appears as a stub network in the OSPF domain.

router ospf: This global configuration command enables an OSPF process on the router. The OSPF process number is an internally used identification parameter; multiple OSPF processes can be defined on the same router.

show ip ospf interface: This exec command displays information about all OSPF configured interfaces.

show ip ospf: This exec command displays information about the OSPF process.

show ip ospf neighbor: This exec command displays information about all OPSF neighbors.

show ip ospf virtual−links: This exec command displays information about the current state of any configured virtual links.

IOS Requirements

OSPF first appeared in IOS 10.0; however, some of the commands described in this chapter require later IOS versions.

Lab #33: Basic OSPF Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE cross over cable
- Cisco rolled cable for console port access

Configuration Overview

This configuration will demonstrate basic OSPF configuration on a Cisco router. As per Figure 8−6, RouterA and RouterB will be running OSPF to advertise routing information.

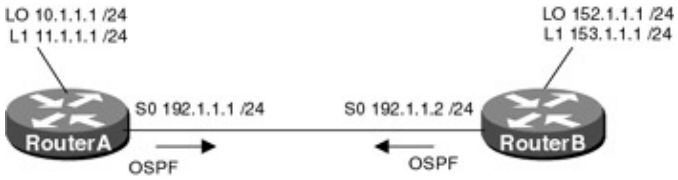


Figure 8−6: Basic OSPF configuration

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 8−6. RouterA and RouterB have loopback interfaces defined to provide a test point.

Enabling OSPF

There are two steps to enabling OSPF on a router: First an OSPF process is defined and then an interface is added to the process. The command to start an OSPF process is **Router OSPF [Process #]**, the process number is used internally by the router. Multiple OSPF routing process can be configured on one router.

The following command enables OSPF process 64 on the router and assigns all interfaces on the router to area 0.

```
router ospf 64
network 0.0.0.0 255.255.255.255 area 0
```

Router Configurations

The configurations for the two routers in this example are as follows (key OSPF configurations are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 10.1.1.1 255.255.255.0
!

interface Loopback1 ← Defines a virtual interface that will be used as a test
                      point
 ip address 11.1.1.1 255.255.255.0

!
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 no fair-queue
!

!
router ospf 64 ← Enables OSPF process 64 on the router

                    ↓ Wildcard Mask can be used to specify
multiple interfaces with one command
 network 192.1.1.1 0.0.0.0 area 0 ← Specifies what interface OSPF will be run
                                    and what area the interface will be in
            á The IP address of the interface that
OSPF will run on in this case interface S0

network 10.1.1.1 0.0.0.0 area 0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2

no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 152.1.1.1 255.255.255.0
```

```
!

interface Loopback1 ← Defines a virtual interface that will be used as a test
                      point
 ip address 153.1.1.1 255.255.255.0

!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock
!
interface Serial1
 no ip address
 shutdown
!
router ospf 64 ← Enables OSPF process 64 on the router

          ↓ An IP address of all zeros and a Wildcard Mask of all ones enables
            OSPF on all interfaces
network 0.0.0.0 255.255.255.255 area 0 ← Specifies what interface OSPF will be
                                         run on and what area the interface
                                         will be in

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Show the IP routing table on RouterA with the command **show ip route;** what follows is the output from this command. Notice that two host-specific routes were learned via OSPF, 152.1.1.1, and 153.1.1.1; this is because loopback interfaces are treated as stub hosts.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Loopback0
     11.0.0.0/24 is subnetted, 1 subnets
C       11.1.1.0 is directly connected, Loopback1
     153.1.0.0/32 is subnetted, 1 subnets
O       153.1.1.1 [110/65] via 192.1.1.2, 00:00:41, Serial0/0
     152.1.0.0/32 is subnetted, 1 subnets
O       152.1.1.1 [110/65] via 192.1.1.2, 00:00:41, Serial0/0
C    192.1.1.0/24 is directly connected, Serial0/0
```

Show the IP routing table on RouterB with the command **show ip route;** what follows is the output from the command. Notice that only one host-specific route was learned via OSPF 10.1.1.1.

```
RouterB#show ip route
```

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     10.0.0.0/32 is subnetted, 1 subnets
O       10.1.1.1 [110/65] via 192.1.1.1, 00:17:16, Serial0/0
     153.1.0.0/24 is subnetted, 1 subnets
C       153.1.1.0 is directly connected, Loopback1
     152.1.0.0/24 is subnetted, 1 subnets
C       152.1.1.0 is directly connected, Loopback0
     C    192.1.1.0/24 is directly connected, Serial0/0
```

The numbers after the destination address [110/65] are the administrative distance and the routing metric respectively. The administrative distance is used by the router to compare routes learned from multiple routing protocols. For example, if the router learned about network 10.1.1.0 from OSPF and RIP, it would prefer the route learned via OPSF because OSPF has a lower administrative distance than RIP (110 vs. 120).

The second number is the metric or cost of using the route, which is inversely proportional to the bandwidth of a link. The router uses the metric to compare routes, which are learned via the same routing protocol. For example, if the router learned about network 10.1.1.0 from two separate routers running OSPF, it would prefer the route with the lower metric.

From RouterA, use the command **show ip ospf interface** to display what interfaces OSPF is configured on; what follows is the output from this command. Notice that on RouterA interface s0/0 and interface Loopback 0 are configured for OSPF and are in area 0. The command also shows the network type, timer intervals, and adjacent neighbors.

```
RouterA#show ip ospf interface
Ethernet0/0 is administratively down, line protocol is down
   OSPF not enabled on this interface
Serial0/0 is up, line protocol is up
  Internet Address 192.1.1.1/24, Area 0
  Process ID 64, Router ID 11.1.1.1, Network Type POINT_TO_POINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:03
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 152.1.1.1
  Suppress hello for 0 neighbor(s)
Serial0/1 is administratively down, line protocol is down
   OSPF not enabled on this interface
Loopback0 is up, line protocol is up
  Internet Address 10.1.1.1/24, Area 0
  Process ID 64, Router ID 11.1.1.1, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
Loopback1 is up, line protocol is up
   OSPF not enabled on this interface
```

From RouterB, use the command **show ip ospf neighbors** to display the status of the routers neighbors; what follows is the output from the command. Notice that the neighbor ID is 11.1.1.1; this is the IP address of the loopback interface. The loopback interface is always used as the router ID. If no loopback interface is configured, then the highest IP address on the router is used.

```
RouterB#show ip ospf neighbor

Neighbor ID     Pri    State        Dead Time     Address         Interface
11.1.1.1        1      FULL/-       00:00:37      192.1.1.1       Serial0/0
```

The state of the router is [Full/−]; this means RouterA's and RouterB's databases are synchronized and the routers are fully adjacent. Full is the last state of forming an adjacency; the other states are described in detail earlier in the chapter. The − appears on point−to−point interfaces where there is no concept of DR or BDR.

Lab #34: Configuring OSPF Priority "DR Election"

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, each having one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Four Ethernet cables and one Ethernet hub
- A Cisco rolled cable for console port access

Configuration Overview

This configuration will demonstrate configuring OSPF priority on an interface to influence the selection of the DR and BDR. As per Figure 8−7, four routers are connected to an Ethernet hub, all running OSPF to advertise routing information.

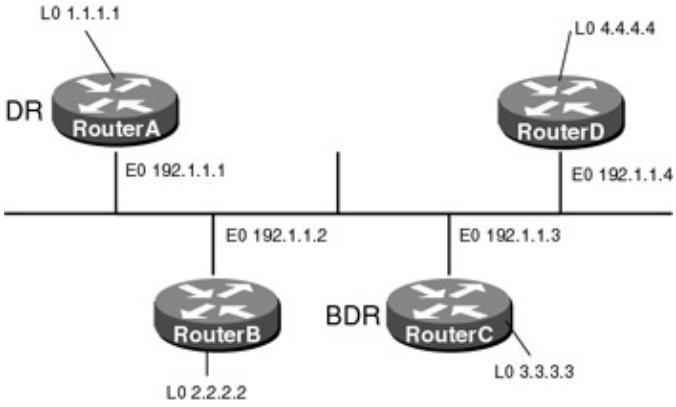


Figure 8−7: DR election process
RouterA will be set with an OSPF priority of 100, which will be the highest priority on the network, so RouterA will be selected the DR. RouterC will have the second highest priority (2) and will be elected the BDR. RouterB will be set for a priority of zero, which means that it is ineligible as becoming the DR. RouterD has no priority set and defaults to a priority of 1.

The IP addresses are assigned as per Figure 8−7; all routers have loopback interfaces defined for test purposes.

Router Configurations

The configurations for the two routers in this example are as follows (key OSPF configurations are highlighted in bold).

RouterA

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
```

```
hostname RouterA
!
!
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
 ip address 192.1.1.1 255.255.255.0
 ip ospf priority 100 ← Sets the priority used by the router in DR election for
                        a particular interface
!
interface Serial0/0
 no ip address
 shutdown
!

!
router ospf 64 ← Enables OSPF process 64 on the router
 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      and what area the I interface will be in

  network 1.1.1.1 0.0.0.0 area 0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0/0
 ip address 192.1.1.2 255.255.255.0
 ip ospf priority 0 ← Sets the priority used by the router in DR election for a
                      particular interface. An OSPF priority of zero means that
                      the router is ineligible in the DR election process
!
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      and what area the I interface will be in

 network 2.2.2.2 0.0.0.0 area 0
!
no ip classless
!
!
line con 0
line aux 0
```

```
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 3.3.3.3 255.255.255.0
!
interface Ethernet0/0
 ip address 192.1.1.3 255.255.255.0
 ip ospf priority 2 ← Sets the priority used by the router in DR election for a
                      particular interface

!
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 3.3.3.3 0.0.0.0 area 0 ← Specifies what interface OSPF will be run on
                                  and what area the interface will be in
network 192.1.1.0 0.0.0.255 area 0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 4.4.4.4 255.255.255.0
!
interface Ethernet0/0
 ip address 192.1.1.4 255.255.255.0
!
interface Serial0/0
 no ip address
 shutdown
 no fair-queue
!
!
router ospf 64 ← Enables OSPF process 64 on the router
```

```
 network 192.1.1.0 0.0.255 area 0 ← Specifies what interface OSPF will be run
                                    and what area the I interface will be in
 network 4.4.4.4 0.0.0.0 area 0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterA, display the OSPF neighbors with the **show ip ospf neighbor** command. Notice that RouterA has three OSPF neighbors, 4.4.4.4, 3.3.3.3, and 2.2.2.2. These neighbor IDs are the Router IDs of each router. OSPF uses the highest loopback interfaces as its router ID; if no loopback interface is configured on the router, the highest IP address is used.

```
RouterA#show ip ospf neighbor

Neighbor ID   Pri    State          Dead Time    Address      Interface
4.4.4.4     1       FULL/DROTHER   00:00:35     192.1.1.4    Ethernet0/0
3.3.3.3     2       FULL/BDR       00:00:39     192.1.1.3    Ethernet0/0
2.2.2.2     0       FULL/DROTHER   00:00:34      92.1.1.2    Ethernet0/0
```

The state of each neighbor is full meaning that RouterA has formed an adjacency with each one of its neighbors. Only the DR and BDR form adjacencies with all of the routers on the network. RouterA is the DR for the network and neighbor 3.3.3.3 is the BDR for the network and all other neighbors are DROTHER, which means that they are neither the BDR or DR for the network.

From RouterB, display the OSPF neighbors with the **show ip ospf neighbor** command. Notice that the state of neighbor 4.4.4.4 is 2WAY and not full. This is because 4.4.4.4 is not the DR or the BDR for the network, so an adjacency is not formed between the two routers.

RouterA (1.1.1.1) is the DR for the network, and RotuerC (3.3.3.3) is the BDR for the network.

```
RouterB#show ip ospf neighbor

Neighbor ID   Pri    State          Dead Time     Address       Interface
4.4.4.4       1      2WAY/DROTHER   00:00:32      192.1.1.4     Ethernet0/0
3.3.3.3       2      FULL/BDR       00:00:36      192.1.1.3     Ethernet0/0
1.1.1.1       100    FULL/DR        00:00:35      192.1.1.1     Ethernet0/0
```

From RouterC, display the OSPF neighbors with the **show ip ospf neighbor** command. Notice that the state of the neighbors is full; this is because RouterC is the BDR for the link.

```
RouterC#sho ip ospf neighbor

Neighbor ID   Pri    State          Dead Time     Address       Interface
4.4.4.4       1      FULL/DROTHER   00:00:36      192.1.1.4     Ethernet0/0
1.1.1.1       100    FULL/DR        00:00:29      192.1.1.1     Ethernet
2.2.2.2       0      FULL/DROTHER   00:00:35      192.1.1.2     Ethernet0/0
```

From RouterD, display the OSPF neighbors with the **show ip ospf neighbor** command. Notice that the state of the neighbor 2.2.2.2 is 2WAY and not full; adjacencies are only formed with the DR or BDR. RouterB (2.2.2.2) is neither the DR nor the BDR for the link.

```
RouterD#show ip ospf neighbor
```

```
Neighbor ID   Pri    State          Dead Time     Address       Interface
2.2.2.2       0      2WAY/DROTHER   00:00:36      192.1.1.2     Ethernet0/0
3.3.3.3       2      FULL/BDR       00:00:32      192.1.1.3     Ethernet0/0
1.1.1.1       100    FULL/DR        00:00:30      192.1.1.1     Ethernet0/0
```

The DR and BDR are selected for the link depending on Router priority; however, once a DR is selected for a network, it remains the DR until the router goes down. On RouterB's Ethernet interface, change the OSPF priority to 200.

```
RouterB(config)#INT E0/0
RouterB(config-if)#IP OSPF priority 200
```

From RouterA, display the OSPF neighbors with the **show ip ospf neighbor** command. Notice that neighbor 2.2.2.2 (RouterB) priority has change to 200; however, it has not become the DR.

```
RouterA#show ip ospf neighbor

Neighbor ID   Pri    State          Dead Time     Address       Interface
4.4.4.4       1      FULL/DROTHER   00:00:31      192.1.1.4     Ethernet0/0
3.3.3.3       2      FULL/BDR       00:00:35      192.1.1.3     Ethernet0/0
2.2.2.2       200    FULL/DROTHER   00:00:30      192.1.1.2     Ethernet0/0
```

The DR will only change if the present DR is no longer on the network. Power down RouterA and display the OSPF neighbors on RouterD with the **show ip ospf neighbor** command. Notice that the BDR RouterC has become the DR and RouterB is now the DR.

```
RouterD#show ip ospf neighbor

Neighbor ID   Pri    State          Dead Time     Address       Interface
2.2.2.2       200    FULL/BDR       00:00:31      192.1.1.2     Ethernet0/0
3.3.3.3       2      FULL/DR        00:00:36      192.1.1.3     Ethernet0/0
```

If all routers were removed from the network and then added again, RouterB would then be elected the DR (OSPF priority 200) and the BDR would be RouterA (OSPF priority 100).

Power all of the Router down and restart them; this will force a new DR/BDR election. Display the OSPF neighbors on RouterD with the **show ip ospf neighbor** command. Notice that RouterB is the DR and RouterA is the BDR.

```
RouterD#show ip ospf neighbor

Neighbor ID   Pri    State          Dead Time     Address       Interface
1.1.1.1       100    FULL/BDR       00:00:33      192.1.1.1     Ethernet0/0
3.3.3.3       2      2WAY/DROTHER   00:00:33      192.1.1.3     Ethernet0/0
2.2.2.2       200    FULL/DR        00:00:30      192.1.1.2     Ethernet0/0
```

Lab #35: Configuring OSPF Virtual Links

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having one Ethernet port and one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Ethernet cables and one Ethernet hub
- One Cisco DTE/DCE crossover cable

- Cisco rolled cable for console port access

Configuration Overview

This configuration will demonstrate configuring an OSPF virtual circuit. In Figure 8−8, area 4 does not have a direct connection to area 0. Area 1 is used as a transport area to connect area 4 to area 0. A virtual link is configured between RouterB and RouterC. The IP addresses are assigned as per Figure 8−8, and loopback interfaces are configured on each router.

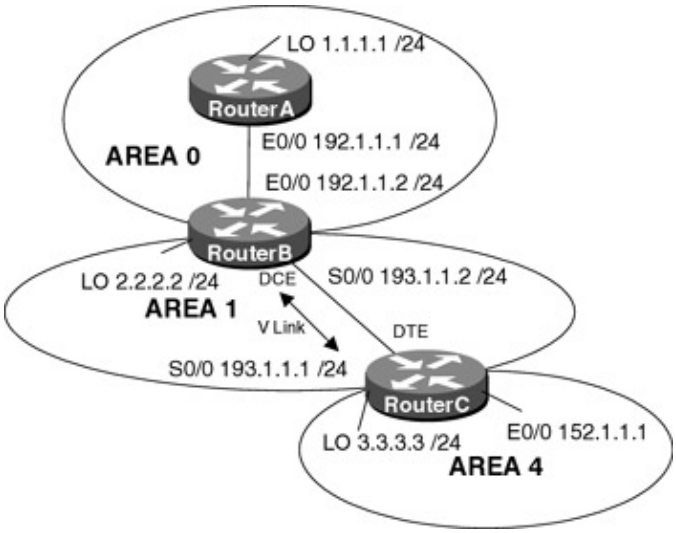


Figure 8−8: OSPF virtual links

The virtual link is configured under the router process using the command:

```
area [area-id] virtual-link [RID]
```

The area−id is the transit area; for this example, it is area 1. The RID is the router ID of the router at the other end of the virtual link. The router ID is highest loopback address on the router; if no loopback is configured, then the router ID is the highest IP address.

Router Configurations

The configurations for the routers in this example are as follows (key OSPF configurations are highlighted in bold).

RouterA

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
 ip address 192.1.1.1 255.255.255.0
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      and what area that interface will be in
!
no ip classless
```

```
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0/0
 ip address 192.1.1.2 255.255.255.0
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000
!
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      and what area that interface will be in
!

 network 193.1.1.0 0.0.0.255 area 1

area 1 virtual-link 3.3.3.3  ← Defines the virtual link across area 1 to
                               3.3.3.3 which is the router ID of RouterC
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 3.3.3.3 255.255.255.0
!
interface Ethernet0/0
 ip address 152.1.1.1 255.255.255.0
```

```
 no keepalive ← Disables the keepalive on the ethernet interface, allows the
                interface to stay up when it is not attached to a hub

!
interface Serial0/0
 ip address 193.1.1.1 255.255.255.0
!
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 193.1.1.0 0.0.0.255 area 1 ← Specifies what interface OSPF will be run
                                      and what area that interface will be in
 network 152.1.1.0 0.0.0.255 area 4
 area 1 virtual-link 2.2.2.2 ← Defines the virtual link across area 1 to
                               2.2.2.2 which is the router ID of RouterB

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
```

Monitoring and Testing the Configuration

From RouterC and RouterB, verify that the virtual link is up with the command **show ip ospf virtual-links**. Here is the output from the command:

```
RouterC#show ip ospf virtual-links
Virtual Link OSPF_VL0 to router 2.2.2.2 is up
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 1, via interface Serial0/0, Cost of using 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:07
    Adjacency State FULL (Hello suppressed)

RouterB#show ip ospf virtual-links
Virtual Link OSPF_VL0 to router 3.3.3.3 is up
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 1, via interface Serial0/0, Cost of using 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:07
    Adjacency State FULL (Hello suppressed)
```

Display the routing table on RouterA with the command **show ip route**. Notice that RouterA has a route to the 152.1.1.0 network, which is attached to RouterC.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
  152.1.0.0/24 is subnetted, 1 subnets
```

```
O IA    152.1.1.0 [110/84] via 192.1.1.2, 00:00:26, Ethernet0/0
C    192.1.1.0/24 is directly connected, Ethernet0/0
O IA 193.1.1.0/24 [110/74] via 192.1.1.2, 00:00:26, Ethernet0/0
 Remove the virtual link statement from RouterB and RouterC.
RouterB#configure terminal
RouterB(config)#router ospf 64
RouterB(config-router)#no area 1 virtual-link 3.3.3.3

RouterC#configure terminal
RouterC(config)#router ospf 64
RouterC(config-router)#no area 1 virtual-link 2.2.2.2
```

Display the routing table on RouterA with the command **show ip route**. Notice that RouterA no longer has a route to the 152.1.1.0 network, which is attached to RouterC.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
C    192.1.1.0/24 is directly connected, Ethernet0/0
O IA 193.1.1.0/24 [110/74] via 192.1.1.2, 00:36:05, Ethernet0/0
```

Add the virtual link statement back to RouterB and RouterC.

```
RouterB# configure terminal
RouterB(config)#router ospf 64
RouterB(config-router)#area 1 virtual-link 3.3.3.3

RouterC# configure terminal
RouterC(config)#router ospf 64
RouterC(config-router)#area 1 virtual-link 2.2.2.2
```

Verify that the virtual link is up with the command **show ip ospf virtual-links**. Add another loopback interface to RouterB using IP address 5.5.5.5 with a 24-bit mask.

```
RouterB#configure terminal
RouterB(config)#interface loopback 1
RouterB(config-if)#ip address 5.5.5.5 255.255.255.0
```

Verify once again that the virtual link is up. Now reload RouterB, making sure that the configuration is written to memory.

Log into RouterC; you will see the following error message.

```
RouterC#
%OSPF-4-ERRRCV: Received invalid packet: mismatch area ID, from backbone area must
be virtual-link but not found from 193.1.1.2, Serial0/0
```

The virtual link is not found because the Router ID of RouterB has changed. Remember the Router ID is the highest loopback interface on the router, or if no loopback interface is defined, the Router ID is the highest IP address. The router ID is only calculated at boot time or any time the OSPF process is restarted. This is why the virtual link was up after we added the new loopback, but went down after the router was reloaded. When virtual links are configured, care must be taken when adding IP address or loopback interfaces to the router because the Router ID may change.

Display the OSPF neighbors on RouterC with the command **show ip ospf neighbor**. Notice that the Router ID of RouterB has changed from 2.2.2.2 to 5.5.5.5. This is why the error message is seen on RouterC.

```
RouterC#show ip ospf neighbor

Neighbor ID   Pri    State         Dead Time     Address       Interface
5.5.5.5       1      FULL/-        00:00:30      193.1.1.2     Serial0/0
```

Lab #36: Configuring OSPF Neighbor Authentication

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Ethernet cables and one Ethernet hub
- Cisco rolled cable for console port access

Overview

Neighbor authentication allows the router to authenticate the source of each routing update that is received. An authentication key or password is exchanged between routers; if the keys do not match, the routing update is rejected.

Cisco uses two types of neighbor authentication: plain text and Message Digest Algorithm Version 5 (MD5). Plain text authentication sends a key over the wire. Since the key is passed in plain text, it can be read during transit and therefore is not recommended.

Message Digest Algorithm Version 5 (MD5) authentication sends a message digest instead of the key. The MD5 algorithm is used to produce a "hash" of the key and this is what is sent. For additional information on the MD5 algorithm, refer to RFC 1321.

Configuration Overview

This configuration will demonstrate neighbor authentication using both plain text authentication and MD5 authentication. RouterA will be configured to use plain text authentication when exchanging routing updates with RouterB and MD5 authentication when exchanging routing updates with RouterC.

RouterA's Ethernet interface and RouterB's Ethernet interface are in OSPF area 0; both Routers have plain text neighbor authentication configured for area 0. RouterA's serial interface and RouterB's serial interface are in area 1; both routers have MD5 authentication configured for area 1.

Router A is connected to RouterB via an Ethenet network, and connected to RouterC via a V.35 crossover cable. RouterA will act as the DCE and supply clock. The IP addresses are assigned as per Figure 8−9.

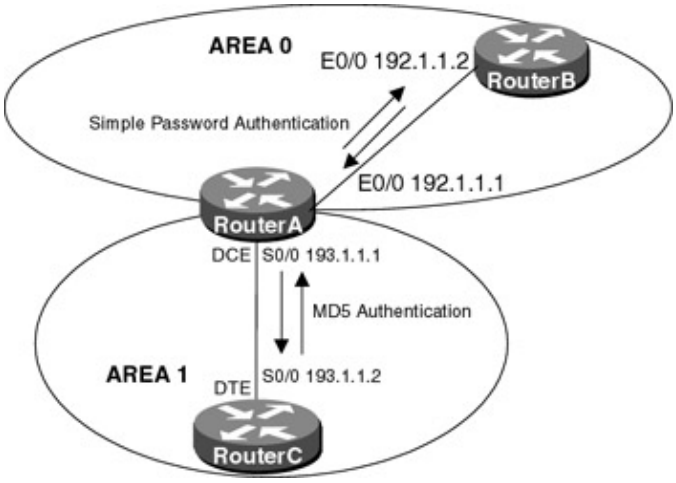


Figure 8−9: OSPF neighbor authentication

There are two steps in enabling neighbor authentication on a router. First, authentication is enabled for a particular area under the routing process.

Plain Text Neighbor Authentication

```
RouterA#configure terminal
RouterA(config)#router ospf 64
RouterA(config−router)#area 0 authentication
```

MD5 Neighbor Authentication

```
RouterA#configure terminal
RouterA(config)#router ospf 64
RouterA(config−router)#area 0 authentication
RouterA(config−router)#area 0 authentication message−digest
```

The next step is to define the authentication key under the interface.

Plain Text Neighbor Authentication

```
RouterA#configure terminal
RouterA(config)#interface e0/0
RouterA(config−if)#ip ospf authentication−key cisco
```

Note The authentication password for all OSPF routers on a network must be the same if they are to communicate with each other via OSPF.

MD5 Neighbor Authentication

```
RouterA#configure terminal
RouterA(config)#interface e0/0
RouterA(config−if)#ip ospf message−digest−key 1 md5 cisco
```

Note When using MD5 authentication, the key identifier must be the same on neighbor routers.

Router Configurations

The configurations for the routers in this example are as follows (key OSPF neighbor authentication commands are highlighted in bold).

RouterA

```
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
!
interface Ethernet0/0
 ip address 192.1.1.1 255.255.255.0
 ip ospf authentication−key cisco ← Assigns a password to be used by
```

```
                                    neighboring routers that are using OSPF's
                                    simple password authentication

!
interface Serial0/0
 ip address 193.1.1.1 255.255.255.0
 ip ospf message-digest-key 1 md5 cisco ← Assigns a password and MD5 key to be
                                          used by neighboring routers that are
                                          using  OSPF's MD5 authentication

 clockrate 500000
!
!
router ospf 64
 network 192.1.1.0 0.0.0.255 area 0
 network 193.1.1.0 0.0.0.255 area 1
 area 0 authentication ← Enables plain text neighbor authentication for OSPF
                         area 0
 area 1 authentication message-digest ← Enables MD5 authentication for OSPF
                                        area 1

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
!
!
interface Ethernet0/0
 ip address 192.1.1.2 255.255.255.0
 ip ospf authentication-key cisco ← Assigns a password to be used by
                                    neighboring routers that are using OSPF's
                                    simple password authentication
!
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      and what area that interface will be in

 area 0 authentication ← Enables plain text neighbor authentication for OSPF
                         area 0
!
no ip classless
ip ospf name-lookup
!
!
line con 0
line aux 0
line vty 0 4
 login
!
```

```
end
```

RouterC

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.0
 ip ospf message-digest-key 1 md5 cisco ← Assigns a password and MD5 key to be
                                          used by neighboring routers that are
                                          using  OSPF's MD5 authentication

!
interface Serial0/1
 no ip address
 shutdown
!
router ospf 64
 network 193.1.1.0 0.0.0.255 area 1
 area 1 authentication message-digest ← Enables MD5 authentication for  OSPF
                                        area 1

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterC, display the OSPF parameters with the command **show ip ospf interface s0/0.** Notice that MD5 is enabled on the interface and the key is 1. This is very important because keys must match in each router.

```
RouterC#show ip ospf interface s0/0
Serial0/0 is up, line protocol is up
  Internet Address 193.1.1.2/24, Area 1
  Process ID 64, Router ID 193.1.1.2, Network Type POINT_TO_POINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:01
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 193.1.1.1
  Suppress hello for 0 neighbor(s)
  Message digest authentication enabled
    Youngest key id is 1
```

From RouterC, display the state of the OSPF neighbors with the command **show ip ospf neighbor**. Notice that RouterC has formed an adjacency with neighbor 193.1.1.1 (RouterA).

```
RouterC#show ip ospf neighbor

Neighbor ID   Pri    State         Dead Time     Address       Interface
193.1.1.1     1      FULL/ -       00:00:33      193.1.1.1     Serial0/0
```

Change the message digest key on RouterC from 1 to 5.

```
RouterC#configure term
RouterC(config)#interface s0/0
RouterC(config-if)#no ip ospf message-digest-key 1 md5 cisco
RouterC(config-if)#ip ospf message-digest-key 5 md5 cisco
```

Monitor the OSPF events on RouterC with the command **debug ip ospf events;** what follows is the output from the command. Notice that RouterC is receiving an MD key of 1 and sending an MD key of 5, causing an authentication key mismatch. Since the neighbor authentication does not match, RouterC simply disregards the OSPF packets from RouterA.

The authentication key and the md5 password must match or the routers will not exchange OSPF information.

```
OSPF: Send with youngest Key 5
OSPF: Rcv pkt from 193.1.1.1, Serial0/0  : Mismatch Authentication Key - No message digest
key 1 on interface
```

Lab #37: Configuring OSPF on NBMA Network "Non−Broadcast Model"

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having one serial port
- One Cisco router with three serial ports acting as a frame relay switch
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Three Cisco V.35 DCE/DTE crossover cables
- Cisco rolled cable for console port access

Overview

When configuring OSPF on NBMA networks, such as Frame Relay or ATM, care must be taken to which router becomes the DR and BDR for the network. The DR and BDR require full logical connectivity with all routers on the network.

Also depending on which one of the four network types is used (broadcast, non−broadcast, point−to−point, or point−to−multipoint), additional configuration may be necessary.

This lab will address OSPF over an NBMA Frame Relay network using the non−broadcast network type. This is the default network type for physical interfaces configured for Frame Relay. On the non−broadcast network type, a DR/BDR is elected for the network; however, due to the lack of broadcast capabilities, the DR and BDR must have a static list of all routers attached to the frame relay cloud. This is achieved using the neighbor command under the OSPF process.

As shown in Figure 8−10, RouterB must be elected the DR because it is the only router that has full physical connectivity with all other routers on the network.

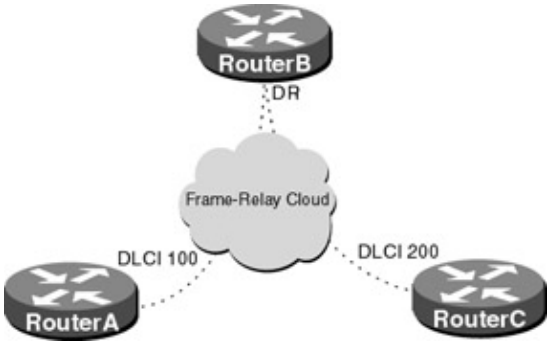


Figure 8−10: Logical connectivity

Configuration Overview

This lab will demonstrate configuring OSPF over Frame Relay using the non−broadcast network type. RouterA, RouterB, and RouterC will connect serially via a crossover cable to a Cisco router (FrameSwitch), which will act as a frame relay switch.

The FrameSwitch will act as the DCE supplying clock for all attached routers; detailed documentation on configuring a Cisco router as a frame relay switch can be found in Chapter 4.

The IP addresses are as per Figure 8−11; RouterA's and RouterC's serial interfaces will be configured with an OSPF priority of 0. This will ensure that RouterB becomes the DR for the network. Neighbor statements for each router will be configured on RouterB.

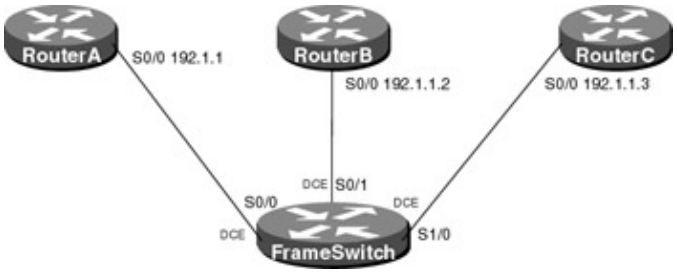


Figure 8−11: Physical connectivity

Router Configurations

The configurations for the routers in this example are as follows (key OSPF commands are highlighted in bold).

FrameSwitch

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
!
frame-relay switching
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 no ip address
 encapsulation frame-relay IETF
 no fair-queue
 clockrate 500000
 frame-relay lmi-type ansi
```

```
 frame-relay intf-type dce
 frame-relay route 100 interface Serial0/1  100
!
interface Serial0/1
 no ip address
 encapsulation frame-relay IETF
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 100 interface Serial0/0  100
 frame-relay route 200 interface Serial1/0  200
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Serial1/0
 no ip address
 encapsulation frame-relay IETF
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/1  200
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA

```
!
version 11.2

no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay IETF
ip ospf priority 0 ← Sets the priority used by the router in DR election for a
                     particular interface. An OSPF priority of zero means that
                     the router is ineligible in the DR election process
 frame-relay map ip 192.1.1.2 100 broadcast
 frame-relay map ip 192.1.1.3 100 broadcast
 frame-relay lmi-type ansi
!
!
router ospf 64 ← Enables OSPF process 64 on the router

 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
```

```
                                        on and what area the interface will be in
network 1.1.1.1 0.0.0.0 area 0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
ip address 2.2.2.2 255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation frame-relay IETF
 frame-relay map ip 192.1.1.1 100 broadcast
 frame-relay map ip 192.1.1.3 200 broadcast
 frame-relay lmi-type ansi
!
!
router ospf 64 ← Enables OSPF process 64 on the router

network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                     on and what area the interface will be in

network 2.2.2.2 0.0.0.0 area 0
neighbor 192.1.1.1 ← Used to define the neighbor on a non-broadcast network
neighbor 192.1.1.3
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
```

```
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 3.3.3.3  255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay IETF

 ip ospf priority 0 ← Sets the priority used by the router in DR election for a
                      particular interface. An OSPF priority of zero means that
                      the router is ineligible in the DR election process

 frame-relay map ip 192.1.1.1 200 broadcast
 frame-relay map ip 192.1.1.2 200 broadcast
 frame-relay lmi-type ansi
!
interface Serial0/1
 no ip address
 shutdown
!
!
router ospf 64 ← Enables OSPF process 64 on the router

 network 192.1.1.0 0.0.0.255 area 0
network 3.3.3.3 0.0.0.0 area 0

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterA, show the OSPF interface statistics with the command **show ip ospf interface s0/0**. Notice that the interface is network type non−broadcast, which means that a DR is elected and the router and OSPF neighbors must be configured manually.

The priority of the interface is 0, which means that it is ineligible to be elected the DR or BDR for the network.

```
RouterA#show ip ospf interface s0/0
Serial0/0 is up, line protocol is up
  Internet Address 192.1.1.1/24, Area 0
  Process ID 64, Router ID 1.1.1.1, Network Type NON_BROADCAST, Cost: 64
  Transmit Delay is 1 sec, State DROTHER, Priority 0
  Designated Router (ID) 192.1.1.2, Interface address 192.1.1.2
  No backup designated router on this network
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
    Hello due in 00:00:13
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 192.1.1.2  (Designated Router)
  Suppress hello for 0 neighbor(s)
```

Show the status of the OSPF neighbors on RouterA with the command **show ip ospf neighbor**. Notice that RouterA is fully adjacent with RouterB (2.2.2.2), which is the DR for the network.

```
RouterA#show ip ospf neighbor

Neighbor ID   Pri    State         Dead Time     Address       Interface
2.2.2.2       1      FULL/DR       00:01:42      192.1.1.2     Serial0/0
```

Show the IP routing table on RouterA with the command **show ip route;** note that RouterA has a route to the loopback addresses on RouterB and RouterC.

After the destination address are two numbers 110/65. The 110 is the administrative distance of OSPF, which is used to compare multiple routes to the same destination. The lower the administrative distance, the more trustworthy the route is. For example, RIP has an administrative distance of 120, so if RouterA learned the same route from RIP and OSPF, the OSPF route is preferred because it has the lower administrative distance.

The second number, 65, is the metric or cost of using the route. This is used to compare routes that are learned via the same routing protocol; the route with the lowest cost is preferred.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     1.0.0.0/24  is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/32  is subnetted, 1 subnets
O       2.2.2.2 [110/65] via 192.1.1.2, 00:02:22, Serial0/0
     3.0.0.0/32  is subnetted, 1 subnets
O       3.3.3.3 [110/65] via 192.1.1.3, 00:02:22, Serial0/0
C    192.1.1.0/24  is directly connected, Serial0/0
```

Display the status of the OSPF neighbors on RouterB with the command **show ip ospf neighbor**. Notice that RouterB has two neighbors, RouterA (1.1.1.1) and RouterC (3.3.3.3), and has formed a full adjacency with each. RouterB was elected DR for the network because it has the highest priority. The DR and BDR form full adjacencies with every other router. On a non−broadcast network, the DR and BDR are the only routers that need to define neighbors.

```
RouterB#show ip ospf neighbor

Neighbor ID   Pri    State         Dead Time     Address       Interface
3.3.3.3       0      FULL/DROTHER  00:01:55      192.1.1.3     Serial0/0
1.1.1.1       0      FULL/DROTHER  00:01:43      192.1.1.1     Serial0/0
```

Display the status of the OSPF neighbors on RouterC with the command **show ip ospf neighbor**. Notice that RouterC has only one neighbor, RouterB, which is the DR for the network. No BDR was elected for the network because RouterB was the only router that had a nonzero OSPF priority. If the OSPF priority of a router is zero, then it cannot be elected the DR or BDR.

```
RouterC#show ip ospf neighbor

Neighbor ID   Pri    State         Dead Time     Address       Interface
192.1.1.2     3      FULL/DR       00:01:37      192.1.1.2     Serial0/0
```

In an NBMA network, care must be taken to assure a full mesh topology or static selection of the DR using interface priority. For this lab topology, it is imperative that RouterB's serial interface becomes the DR

because it is the only router that has full connectivity to all other routers.

Let's see what would happen when RouterA becomes the DR for the network. Change the OSPF priority on RouterB and remove the neighbor statements with the following commands.

```
RouterB#configure terminal
RouterB(config)#interface s0/0
RouterB(config−if)#ip ospf priority 0
RouterB(config−if)#router ospf 64
RouterB(config−router)#no neighbor 192.1.1.1
RouterB(config−router)#no neighbor 192.1.1.3
```

Change the OSPF priority of RouterA's serial interface to 10 and add a neighbor statement for RouterB and RouterC with the following command.

```
RouterA#configure terminal
RouterA(config)#interface s0
RouterA(config−if)#ip ospf priority 10
RouterA(config−if)#router ospf 64
RouterA(config−router)#neighbor 192.1.1.2
RouterA(config−router)#neighbor 192.1.1.3
```

Make sure the configurations are written to NVRAM and reload the routers. The reason the routers need to be reloaded is that, once a DR is elected for the network, no other router can become the DR until the DR is dead.

From RouterA, show the OSPF interface statistics with the command **show ip ospf interface s0/0.** Notice that RouterA (1.1.1.1) is now the DR for the network.

```
RouterA#show ip  ospf interface s0
Serial0 is up, line protocol is up
  Internet Address 192.1.1.1/24, Area 0
  Process ID 64, Router ID 1.1.1.1, Network Type NON_BROADCAST, Cost: 64
  Transmit Delay is 1 sec, State DR, Priority 10
  Designated Router (ID) 1.1.1.1, Interface address 192.1.1.1
  No backup designated router on this network
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
    Hello due in 00:00:01
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 2.2.2.2
```

Display the status of the OSPF neighbors on RouterC with the command **show ip ospf neighbor**. Notice that RouterA has only formed an adjacency with RouterB (2.2.2.2) but not RouterC. This is because it only has physical connectivity to RouterB; the DR must have physical connectivity to all routers on the network.

```
RouterA#show ip ospf neighbor

Neighbor ID   Pri    State              Dead Time     Address       Interface
N/A           0      ATTEMPT/DROTHER    −             192.1.1.3     Serial0
2.2.2.2       0      FULL/DROTHER       00:01:55      192.1.1.2     Serial0
```

Display the routing table on RouterC; notice there are no routes. This is because RouterC was unable to form an adjacency with the DR.

```
RouterC#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR
```

```
Gateway of last resort is not set

     3.0.0.0/24 is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
C    192.1.1.0/24 is directly connected, Serial0
```

Lab #38: Configuring OSPF on NBMA Network "Broadcast Model"

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having one serial port
- One Cisco router with three serial ports acting as a frame relay switch
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Three Cisco V.35 DCE/DTE crossover cables
- Cisco rolled cable for console port access

Overview

The broadcast network type is a workaround for having to define all neighbors statically. When an interface is configured for broadcast, it will behave as if it were connected to a LAN. A DR and BDR will still be elected for the network, so care must be taken to assure that the router elected DR/BDR has physical connectivity to all routers on the network.

Configuration Overview

This lab will demonstrate configuring OSPF over Frame Relay using the broadcast network type. RouterA, RouterB, and RouterC will connect serially via a crossover cable to a Cisco router (FrameSwitch), which will act as a frame relay switch.

The FrameSwitch will act as the DCE supplying clock for all attached routers; detailed documentation on configuring a Cisco router as a frame relay switch can be found in Chapter 4.

The network type will be set on all routers using the interface command **ip ospf network broadcast**. The IP addresses are as per Figure 8−12; RouterA and RouterC serial interfaces will be configured with an OSPF priority of 0. This will ensure that RouterB becomes the DR for the network.

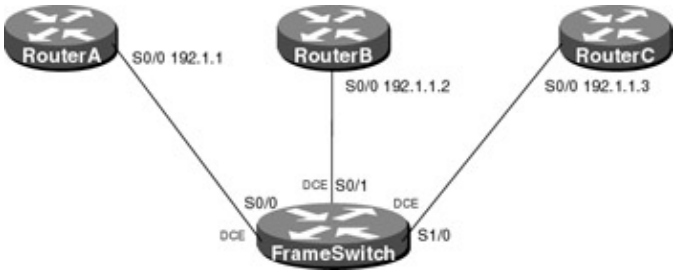


Figure 8−12: Configuring OSPF on NBMA network

Router Configurations

The configurations for the routers in this example are as follows (key OSPF commands are highlighted in bold).

FrameSwitch

```
:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
!
frame-relay switching
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 no ip address
 encapsulation frame-relay IETF
 no fair-queue
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 100 interface Serial0/1  100
!
interface Serial0/1
 no ip address
 encapsulation frame-relay IETF
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 100 interface Serial0/0  100
 frame-relay route 200 interface Serial1/0  200
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Serial1/0
 no ip address
 encapsulation frame-relay IETF
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/1  200
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
```

```
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay IETF
ip ospf network broadcast ← Defines the network type as broadcast
ip ospf priority 0 ← Sets the priority used by the router in DR election for a
                     particular interface. An OSPF priority of zero means that
                     the router is ineligible in the DR election process
 frame-relay map ip 192.1.1.2 100 broadcast
 frame-relay map ip 192.1.1.3 100 broadcast
 frame-relay lmi-type ansi
!
interface Serial1/0
 no ip address
 shutdown
!
!
router ospf 64 ← Enables OSPF process 64 on the router

 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      on and what area the interface will be in
network 1.1.1.1 0.0.0.0 area 0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
ip address 2.2.2.2 255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation frame-relay IETF
ip ospf network broadcast ← Defines the network type as broadcast

 frame-relay map ip 192.1.1.1 100 broadcast
 frame-relay map ip 192.1.1.3 200 broadcast
 frame-relay lmi-type ansi
!
interface Serial0/1
 no ip address
```

```
 shutdown
!
router ospf 64 ← Enables OSPF process 64 on the router

network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                     on and what area the interface will be in
network 2.2.2.2 0.0.0.0 area 0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
ip address 3.3.3.3  255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay IETF
ip ospf network broadcast ← Defines the network type as broadcast
 ip ospf priority 0 ← Sets the priority used by the router in DR election for a
                      particular interface. An OSPF priority of zero means that
                      the router is ineligible in the DR election process

 frame-relay map ip 192.1.1.1 200 broadcast
 frame-relay map ip 192.1.1.2 200 broadcast
 frame-relay lmi-type ansi
!
interface Serial0/1
 no ip address
 shutdown
!
!
router ospf 64 ← Enables OSPF process 64 on the router
network 192.1.1.0 0.0.0.255 area 0
network 3.3.3.3 0.0.0.0 area 0

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
```

```
end
```

Monitoring and Testing the Configuration

From RouterA, show the OSPF interface statistics with the command **show ip ospf interface s0/0.** Notice that the interface is network type broadcast, which means that a DR is elected. Since the network is broadcast, no neighbors need to be defined.

The priority of the interface is 0, which means that it is ineligible to be elected the DR or BDR for the network.

```
RouterA#show ip ospf interface s0/0
Serial0 is up, line protocol is up
  Internet Addr\ess 192.1.1.1/24, Area 0
  Process ID 64, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 64
  Transmit Delay is 1 sec, State DROTHER, Priority 0
  Designated Router (ID) 2.2.2.2, Interface address 192.1.1.2
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:01
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 2.2.2.2  (Designated Router)
```

Show the status of the OSPF neighbors on RouterA with the command **show ip ospf neighbor**. Notice that RouterA is fully adjacent with RouterB (2.2.2.2), which is the DR for the network.

```
RouterA#show ip ospf neighbor

Neighbor ID   Pri    State           Dead Time     Address       Interface
2.2.2.2       1      FULL/DR         00:00:34      192.1.1.2     Serial0
```

Show the status of the OSPF neighbors on RouterB with the command **show ip ospf neighbor**. Notice that RouterB is fully adjacent with RouterA (1.1.1.1) and RouterC (3.3.3.3). Both routers are DROTHER, which means that they are neither the DR nor the BDR for the network.

```
RouterB#show ip ospf neighbor

Neighbor ID   Pri    State           Dead Time     Address       Interface
3.3.3.3       0      FULL/DROTHER    00:00:34      192.1.1.3     Serial0
1.1.1.1       0      FULL/DROTHER    00:00:35      192.1.1.1     Serial0
```

Let's see what would happen when RouterA becomes the DR and RouterB becomes the BDR for the network. Change the OSPF priority on RouterB with the following commands.

```
RouterB#configure terminal
RouterB(config)#interface s0/0
RouterB(config-if)#ip ospf priority 1
```

Change the OSPF priority of RouterA's serial interface to 10 with the following command.

```
RouterA#configure terminal
RouterA(config)#interface s0
RouterA(config-if)#ip ospf priority 10
```

Make sure the configurations are written to NVRAM and reload the routers. The reason the routers need to be reloaded is that, once a DR is elected for the network, no other router can become the DR until the DR is dead.

From RouterA, show the OSPF interface statistics with the command **show ip ospf interface s0/0.** Notice that RouterA is now the DR and RouterB is the BDR for the network.

```
RouterA#show ip ospf interface s0
Serial0 is up, line protocol is up
  Internet Address 192.1.1.1/24, Area 0
  Process ID 64, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 64
  Transmit Delay is 1 sec, State DR, Priority 10
  Designated Router (ID) 1.1.1.1, Interface address 192.1.1.1
  Backup Designated router (ID) 2.2.2.2, Interface address 192.1.1.2
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:05
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 2.2.2.2  (Backup Designated Router)
```

Show the status of the OSPF neighbors on RouterB with the command **show ip ospf neighbor**. Notice that RouterB is fully adjacent with RouterA (1.1.1.1) and RouterC (3.3.3.3). RouterA is the DR for the network and RouterC is DROTHER.

```
RouterB#show ip ospf neighbor

Neighbor ID   Pri    State           Dead Time     Address       Interface
3.3.3.3       0      FULL/DROTHER    00:00:34      192.1.1.3     Serial0
1.1.1.1       10     FULL/DR         00:00:36      192.1.1.1     Serial0
```

Show the status of the OSPF neighbors on RouterC with the command **show ip ospf neighbor**. Notice that RouterC thinks that RouterB is the DR for the network; this is because RouterC does not have a physical connection to the RouterA.

```
RouterC#show ip ospf neighbor
Neighbor ID   Pri    State           Dead Time     Address       Interface
2.2.2.2              FULL/DR         00:00:31      192.1.1.2     Serial0
```

Let's take a look at the OSPF database on RouterC with the command **show ip ospf database**. This command shows the table of link state advertisements the router uses as input to the Dijkstra algorithm to determine the routing table. Notice that RouterC has the links 1.1.1.1 and 2.2.2.2, which are the loopback interfaces on RouterA and RouterB, respectively.

```
RouterC#show ip ospf database

       OSPF Router with ID (3.3.3.3) (Process ID 64)

                Router Link States (Area 0)
Link ID      ADV Router      Age      Seq#           Checksum      Link count
1.1.1.1      1.1.1.1         663      0x80000003     0xB4B8        2
2.2.2.2      2.2.2.2         662      0x80000003     0xC27D        2
3.3.3.3      3.3.3.3         665      0x80000003     0xC869        2
Net Link States (Area 0)
Link ID      ADVRouter       Age      Seq#           Checksum
192.1.1.1    1.1.1.1         663      0x80000001     0xFE87
```

Show the routing table on RouterC with the command **show ip route**. Notice that there is not a route to 1.1.1.1 or 2.2.2.2.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     3.0.0.0/24  is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
C    192.1.1.0/24  is directly connected, Serial0
```

This is because the advertising router the DR for the broadcast network (RouterA) is not reachable. This can be seen from the command **show ip ospf database router** on RouterC.

```
RouterC#show ip ospf database router

       OSPF Router with ID (3.3.3.3) (Process ID 64)


                Router Link States (Area 0)

  Adv Router is not-reachable
  LS age: 63
  Options: (No TOS-capability, No DC)
  LS Type: Router Links
  Link State ID: 1.1.1.1
  Advertising Router: 1.1.1.1
  LS Seq Number: 80000004
  Checksum: 0xB2B9
  Length: 48
   Number of Links: 2

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 1.1.1.1
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metrics: 1

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.1.1.1
     (Link Data) Router Interface address: 192.1.1.1
      Number of TOS metrics: 0
       TOS 0 Metrics: 64
```

Lab #39: Configuring OSPF on NBMA Network "Point-to-Multipoint Model"

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having one serial port
- One Cisco router with three serial ports acting as a frame relay switch
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Three Cisco V.35 DCE/DTE crossover cables
- Cisco rolled cable for console port access

Overview

A point-to-multipoint network type is treated as a numbered point-to-point interface having one or more neighbors. When an interface is configured for point-to-multipoint, no DR/BDR are elected and neighbors do not need to be defined, greatly simplifying configuring OSPF over an NBMA network.

Configuration Overview

This lab will demonstrate configuring OSPF over Frame Relay using the point-to-multipoint network type. RouterA, RouterB, and RouterC will connect serially via a crossover cable to a Cisco router (FrameSwitch), which will act as a frame relay switch.

The FrameSwitch will act as the DCE supplying clock for all attached routers; detailed documentation on configuring a Cisco router as a frame relay switch can be found in Chapter 4.

The network type will be set on all routers using the interface command **ip ospf network point−to−multipoint**. The IP addresses are as per Figure 8−13; no **neighbors** need to be defined and OSPF priorities need not be set since DR/BDR is not elected for the network.

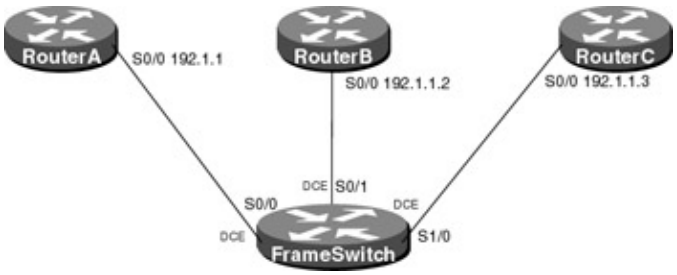


Figure 8−13: Physical connectivity

Router Configurations

The configurations for the routers in this example are as follows (key OSPF commands are highlighted in bold).

FrameSwitch

```
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname FrameSwitch
!
!
frame−relay switching
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 no ip address
 encapsulation frame−relay IETF
 no fair−queue
 clockrate 500000
 frame−relay lmi−type ansi
 frame−relay intf−type dce
 frame−relay route 100 interface Serial0/1  100
!
interface Serial0/1
 no ip address
 encapsulation frame−relay IETF
 clockrate 500000
 frame−relay lmi−type ansi
 frame−relay intf−type dce
 frame−relay route 100 interface Serial0/0  100
 frame−relay route 200 interface Serial1/0  200
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Serial1/0
 no ip address
 encapsulation frame−relay IETF
 clockrate 500000
 frame−relay lmi−type ansi
```

```
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/1  200
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay IETF
ip ospf network point-to-multipoint ← Defines the network type as point-to-
                                      multipoint
frame-relay map ip 192.1.1.2 100 broadcast
 frame-relay map ip 192.1.1.3 100 broadcast
 frame-relay lmi-type ansi
!
interface Serial1/0
 no ip address
 shutdown
!
!
router ospf 64 ← Enables OSPF process 64 on the router

 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      on and what area the interface will be in
network 1.1.1.1 0.0.0.0 area 0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
```

```
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 2.2.2.2 255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation frame-relay IETF
ip ospf network point-to-multipoint ← Defines the network type as point-to-
                                      multipoint
 frame-relay map ip 192.1.1.1 100 broadcast
 frame-relay map ip 192.1.1.3 200 broadcast
 frame-relay lmi-type ansi
!
interface Serial0/1
 no ip address
 shutdown
!
router ospf 64 ← Enables OSPF process 64 on the router

network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                     on and what area the interface will be in
network 2.2.2.2 0.0.0.0 area 0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 3.3.3.3  255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay IETF
ip ospf network point-to-multipoint ← Defines the network type as point-to-
                                      multipoint
 frame-relay map ip 192.1.1.1 200 broadcast
 frame-relay map ip 192.1.1.2 200 broadcast
```

```
 frame-relay lmi-type ansi
!
interface Serial0/1
 no ip address
 shutdown
!
!
router ospf 64 ← Enables OSPF process 64 on the router
network 192.1.1.0 0.0.0.255 area 0
network 3.3.3.3 0.0.0.0 area 0

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterA, show the OSPF interface statistics with the command **show ip ospf interface s0/0.** Notice that the interface is network type POINT_TO_MULTIPOINT, which means that a DR/BDR is elected and no neighbors need to be defined.

```
RouterA#show ip ospf interface s0/0
Serial0 is up, line protocol is up
  Internet Address 192.1.1.1/24, Area 0
  Process ID 64, Router ID 1.1.1.1, Network Type POINT_TO_MULTIPOINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_MULTIPOINT,
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
    Hello due in 00:00:01
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 2.2.2.2
```

Display the neighbor state on RouterA with the command **show ip ospf neighbor**. Notice that RouterA is fully adjacent with RouterB (2.2.2.2), but there is no concept of DR; the link is treated as a point-to-point link; however, the difference is that all routers can be on the same subnet.

```
RouterA#show ip ospf neighbor
Neighbor ID      Pri      State       Dead Time      Address      Interface
2.2.2.2          1        FULL/  -    00:01:59      192.1.1.2     Serial0/0
```

Lab #40: Configure OSPF Interface Parameters

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having one serial port and one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco V.35 DCE/DTE crossover cables
- One Ethernet hub and two Ethernet cables
- Cisco rolled cable for console port access

Overview

The Cisco IOS allows the administrator to alter certain interface−specific OSPF parameters. This lab will deal with three of the more commonly used ones (cost, hello interval, and dead interval).

The cost parameter sets the cost of OSPF sending a packet over particular interface. By default, the cost is calculated using the formula (100,000,000/bandwidth of the link). So by default, the cost of using an Ethernet interface is 100 million divided by 10 million, which equals 10. The OSPF cost parameter is very useful in manipulating the flow of traffic, especially if you wish to prefer a slower link to a faster link.

The OSPF hello interval is the length of time in seconds between sending hello packets on a particular interface. The hello interval must be consistent across all routers in an attached network. The hello interval will vary based on the interface network type (broadcast = 10, non−broadcast = 30, point−to−point = 10, and point−to−multipoint = 30).

The OSPF dead interval is the length of time in seconds that a hello packet must not be seen from a neighboring router before the neighbor is declared down. The dead interval must be consistent across all routers in an attached network. According to the interface network type, the dead interval will vary (broadcast = 40, non−broadcast = 120, point−to−point = 40, and point−to−multipoint = 120).

Configuration Overview

This lab will demonstrate configuring OSPF interface specific parameters (cost, hello interval, and dead interval). The serial interface between RouterA and RouterB will be configured for a cost of 66, which is two higher than the default of the serial link connecting RouterB and RouterC. The OSPF hello interval will be set to 20, and the dead interval will be set to 120 on the serial interface connecting RouterA and RouterB.

RouterA and RouterC will attach to RouterB via a V.35 crossover cable. RouterB will act as the DCE supplying clock. RouterA and RouterC will also be attached via an Ethenet hub; all IP addresses are as per Figure 8−14.

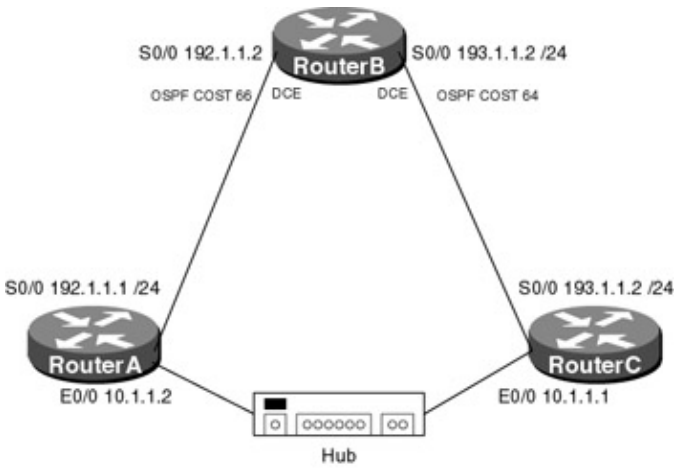


Figure 8−14: Configure OSPF interface parameters

Router Configurations

The configurations for the routers in this example are as follows (key OSPF commands are highlighted in bold).

RouterA

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
```

```
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
  ip address 10.1.1.2 255.255.255.0
 !
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 ip ospf cost 66
 ip ospf hello-interval 20
ip ospf dead-interval 120
!
interface Serial1/0
 no ip address
 shutdown
!
!
router ospf 64 ← Enables OSPF process 64 on the router

 network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      on and what area the interface will be in
network 1.1.1.1 0.0.0.0 area 0
network 10.1.1.0 0.0.0.0.255 area 0

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 2.2.2.2 255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 ip ospf cost 66
 ip ospf hello-interval 20
 ip ospf dead-interval 120
 clockrate 500000
 !
interface Serial0/1
ip address 193.1.1.2 255.255.255.0
  clockrate 500000
!
```

```
router ospf 64 ← Enables OSPF process 64 on the router

network 192.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                     on and what area the interface will be in
network 2.2.2.2 0.0.0.0 area 0
network 193.1.1.0 0.0.0.255 area 0

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
interface Loopback0 ← Defines a virtual interface the IP address is used as the
                      router ID
 ip address 3.3.3.3  255.255.255.0

!
interface Ethernet0/0
 ip address 10.1.1.2 255.255.255.0
 !
interface Serial0/0
 ip address 193.1.1.1 255.255.255.0
 !
interface Serial0/1
 no ip address
 shutdown
!
!
router ospf 64 ← Enables OSPF process 64 on the router
network 192.1.1.0 0.0.0.255 area 0
network 3.3.3.3 0.0.0.0 area 0
network 10.1.1.0 0.0.0.255 area 0

!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterA, show the OSPF interface statistics with the command **show ip ospf interface s0/0.** Notice that the hello and dead intervals have been changed and the OSPF cost of sending a packet out the interface has changed to 66.

```
RouterA#show ip ospf int s0/0
Serial0 is up, line protocol is up
  Internet Address 192.1.1.1/24, Area 0
  Process ID 64, Router ID 192.1.1.1, Network Type POINT_TO_POINT, Cost: 66
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 20, Dead 120, Wait 120, Retransmit 5
    Hello due in 00:00:08
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 193.1.1.2
  Suppress hello for 0 neighbor(s)
```

Display the routing table on RouterA with the command **show ip route**. Notice the cost of reaching the loopback interface on RouterC (3.3.3.3) is 11. The reason that it is 11 is the cost of the Ethernet is 10 and the cost of a loopback interface is 1.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0is directly connected, Loopback0
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2[110/67] via 192.1.1.2, 00:15:08, Serial0
     3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/11] via 10.1.1.2, 00:15:08, Ethernet0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Ethernet0
C    192.1.1.0/24 is directly connected, Serial0
193.1.1.0/24 [110/74] via 10.1.1.2, 00:15:08, Ethernet0
```

Notice the cost of reaching the loopback interface on RouterC (3.3.3.3) is 11. The reason that it is 11 is that the OSPF cost of sending a packet out the Ethernet on RouterA is 10 and the cost of sending a packet out the loopback interface on RouterC is 1. This can been seen by displaying the OSPF statisics on RouterA's Ethernet interface and RouterC's loopback interface.

```
RouterA#show ip ospf interface e0/0
Ethernet0 is up, line protocol is up
  Internet Address 10.1.1.1/24, Area 0
  Process ID 64, Router ID 192.1.1.1, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 192.1.1.1, Interface address 10.1.1.1
  Backup Designated router (ID) 3.3.3.3, Interface address 10.1.1.2
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:09
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 3.3.3.3  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)

RouterC#show ip ospf int loopback 0
Loopback0 is up, line protocol is up
  Internet Address 3.3.3.3/24, Area 0
  Process ID 64, Router ID 3.3.3.3, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
```

Change the OSPF cost of RouterA's Ethernet interface to 200.

```
RouterA#configure terminal
RouterA(config)#interface e0/0
RouterA(config−if)#ip ospf cost 200
```

Display the routing table on RouterA with the command **show ip route**. Notice that the route to 3.3.3.3 has now changed; RouterA now uses the path over the serial interface, which has a cost of 131, which is now lower than the new cost of using the Ethernet interface.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/67] via 192.1.1.2, 00:03:20, Serial0
     3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/131] via 192.1.1.2, 00:03:20, Serial0
     10.0.0.0/24 is subnetted, 1 subnets
```

Now let's take a look at what happens when the hello intervals do not match on routers connected to the same network. On RouterA's serial interface, change the hello interval to 30 seconds.

```
RouterA#configure terminal
RouterA(config)#interface s0/0
RouterA(config−if)#ip ospf hello−interval 30
```

Display the status of the OSPF neighbors on RouterA with the **command show ip ospf neighbor**. The neighbor relationship with RouterB is gone.

```
RouterA#show ip ospf neighbor

Neighbor ID      Pri         State       Dead Time      Address      Interface
3.3.3.3          1       FULL/BDR    00:00:37       10.1.1.2     Ethernet0
```

Monitor the OSPF events on RouterA with command **debug ip ospf events**. Notice that RouterA is receiving an OSPF packet from RouterB and the hello intervals do not match. If either the hello interval or the dead interval do not match, then the router will not form an adjacency with its neighbor.

```
RouterA#
OSPF: Mismatched hello parameters from 192.1.1.2
```

Lab #41: Inter−Area and External Route Summarization

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having one serial port and one Ethernet port
- Two Cisco routers, each having one serial port and two Ethernet ports
- One Cisco router with an Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco V.35 DCE/DTE crossover cables
- Two Ethernet hub and four Ethernet cables (or two Ethernet Crossover cables)
- Cisco rolled cable for console port access

Overview

Cisco allows you to summarize addresses in order to conserve resources by limiting the number of routes that need to be advertised between areas. Two types of address summarization are supported on a Cisco router: inter−area summarization and external route summarization. Inter−area summarization is used to summarize addresses between areas, while external summarization is used to summarize a set of external routes that have been injected into the domain.

Configuration Overview

This lab will demonstrate the various summarization techniques used in OSPF. Since area 0 contains all of the addresses for subnetwork 152.1.1.0, they can all be summarized by RouterB and RouterC in one update, 152.1.1.0/24. In addition, RouterD is acting as a ASBR redistributing the RIP learned routes from RouterE into OSPF. The seven networks can be summarized into one network entry (130.1.0.0/21).

To do this, the area range command is used, which specifies the area that the summary belongs to, the summary address, and the mask.

RouterA will attach to RouterB via a V.35 crossover cable. RouterB will act as the DCE supplying clock to RouterA. RouterB and RouterC will be attached via an Ethenet hub. RouterC will attach to RouterD via a V.35 crossover cable. RouterC will act as the DCE supplying clock to RouterD. RouterD will attach to RouterE via an Ethenet hub. The second Ethernet interfaces on RouterB and RouterC will not attach to anything, so keep−alives will need to be disabled. The reason that Ethernet interfaces were used instead of loopback interfaces is that loopback interfaces are advertised as /32 networks across area boundaries.

RIP is run between RouterD and RouterE; RouterE will advertise all subnetworks that are attached. RouterD will redistribute the RIP learned routes into OSPF; mutual redistribution is not used, since it is not needed to illustrate summarization. However, if you want RouterE to see the OSPF networks, it will need to be added. All IP addresses are as per Figure 8−15.

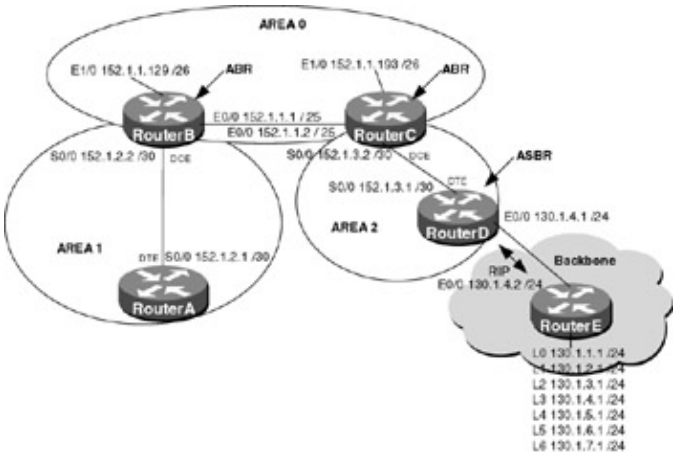


Figure 8−15: OSPF route summarization

Router Configurations

The configurations for the routers in this example are as follows (key OSPF commands are highlighted in bold).

RouterA

```
Building configuration...

Current configuration:
!
version 11.2
no service udp−small−servers
```

```
no service tcp-small-servers
!
hostname RouterA
!
!
interface Ethernet0
 ip address 192.1.1.2 255.255.255.0
!
interface Serial0
 ip address 152.1.2.1 255.255.255.252
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 152.1.2.0 0.0.0.255 area 1 ← Specifies what interface OSPF will be run
                                      on and what area the interface will be in

!
line con 0
line 1 16
 no exec
 transport input all
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
!
hostname RouterB
!
!
interface Ethernet 1/0
 ip address 152.1.1.129 255.255.255.192
no keepalive
!
interface Ethernet0/0
 ip address 152.1.1.1 255.255.255.128
!
interface Serial0/0
 ip address 152.1.2.2 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 1000000
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 152.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      on and what area the interface will be in
 network 152.1.2.0 0.0.0.255 area 1
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
!
```

RouterC

```
Current configuration:
!
hostname RouterC
!
interface Ethernet 1/0
 ip address 152.1.1.193 255.255.255.192
 no ip directed-broadcast
no keepalive
!
interface Ethernet0/0
 ip address 152.1.1.2 255.255.255.128
 no ip directed-broadcast
!
interface Serial0/0
ip address 152.1.3.2 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 1000000
!
!
router ospf 64 ← Enables OSPF process 64 on the router

 network 152.1.1.0 0.0.0.255 area 0
 network 152.1.3.0 0.0.0.255 area 2
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterD

```
Current configuration:
!
version 11.2
!
hostname RouterD
!
!
interface Ethernet0
 ip address 130.1.4.1 255.255.255.0
 no ip directed-broadcast
!
interface Serial0
 ip address 152.1.3.1 255.255.255.252
 no ip directed-broadcast
 ip ospf interface-retry 0
!
router ospf 64 ← Enables OSPF process 64 on the router
 redistribute rip metric 10 subnets ← Redistributes RIP into OSPF (For this lab
                                      exercise only one way redistribution is
                                      needed)
  network 152.1.3.0 0.0.0.255 area 2
network 152.1.3.0 0.0.0.255 area 2
!
router rip ← Enables RIP routing
 network 130.1.0.0
!
```

```
no ip classless
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterE

```
version 11.2
!
hostname RouterE
!
interface Loopback0
 ip address 130.1.1.1 255.255.255.0
 no ip directed-broadcast
!
interface Loopback1
 ip address 130.1.2.1 255.255.255.0
 no ip directed-broadcast
!
interface Loopback2
 ip address 130.1.3.1 255.255.255.0
 no ip directed-broadcast
!
interface Loopback3
 ip address 130.1.5.1 255.255.255.0
 no ip directed-broadcast
!
interface Loopback4
 ip address 130.1.6.1 255.255.255.0
 no ip directed-broadcast
no ip directed-broadcast
!
interface Loopback5
 ip address 130.1.7.1 255.255.255.0
 no ip directed-broadcast
!
interface Ethernet0
 ip address 130.1.4.2 255.255.255.0
 no ip directed-broadcast
!
router rip ← Enables RIP routing
 network 130.1.0.0
!
ip classless
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

Display the routing table on RouterA with command **show ip route**. What follows is the output from the command; notice that RouterA has an entry for networks 152.1.1.128/26, 152.1.1.192/26, and 152.1.1.0/25.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
```

```
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     130.1.0.0/24  is subnetted, 7 subnets
O E2    130.1.3.0 [110/10] via 152.1.2.2, 00:19:45, Serial0
O E2    130.1.2.0 [110/10] via 152.1.2.2, 00:19:45, Serial0
O E2    130.1.1.0 [110/10] via 152.1.2.2, 00:19:45, Serial0
O E2    130.1.7.0 [110/10] via 152.1.2.2, 00:19:45, Serial0
O E2    130.1.6.0 [110/10] via 152.1.2.2, 00:19:45, Serial0
O E2    130.1.5.0 [110/10] via 152.1.2.2, 00:19:45, Serial0
O E2    130.1.4.0 [110/10] via 152.1.2.2, 00:19:45, Serial0
     152.1.0.0/16  is variably subnetted, 5 subnets, 4 masks
O IA    152.1.1.128/26 [110/65] via 152.1.2.2, 00:19:49, Serial0
O IA    152.1.1.192/26 [110/84] via 152.1.2.2, 00:00:22, Serial0
O IA    152.1.1.0/25 [110/74] via 152.1.2.2, 00:19:49, Serial0
O IA    152.1.3.0/30 [110/122] via 152.1.2.2, 00:19:49, Serial0
C       152.1.2.0/30 is directly connected, Serial0
C    192.1.1.0/24 is directly connected, Ethernet0
```

Since all of these networks are part of Area 0, the area border routers (ABRs) RouterB and RouterC can summarize the networks into one entry, 152.1.1.0 /24. To do this, add the following commands to RouterB and RouterC under the OSPF process.

```
RouterB(config)#router ospf 64
RouterB(config−router)#area 0 range 152.1.1.0 255.255.255.0

RouterC(config)#router ospf 64
RouterC(config−router)#area 0  range 152.1.1.0 255.255.255.0
```

Display the routing table on RouterA with the command **show ip route**. What follows is the output from the command; notice that RouterA now only has one entry, 152.1.1.0 /24, instead of three.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     130.1.0.0/24  is subnetted, 7 subnets
O E2    130.1.3.0 [110/10] via 152.1.2.2, 00:27:23, Serial0
O E2    130.1.2.0 [110/10] via 152.1.2.2, 00:27:23, Serial0
O E2    130.1.1.0 [110/10] via 152.1.2.2, 00:27:23, Serial0
O E2    130.1.7.0 [110/10] via 152.1.2.2, 00:27:23, Serial0
O E2    130.1.6.0 [110/10] via 152.1.2.2, 00:27:23, Serial0
O E2    130.1.5.0 [110/10] via 152.1.2.2, 00:27:23, Serial0
O E2    130.1.4.0 [110/10] via 152.1.2.2, 00:27:23, Serial0
     152.1.0.0/16  is variably subnetted, 3 subnets, 2 masks
O IA    152.1.1.0/24 [110/84] via 152.1.2.2, 00:01:42, Serial0
O IA    152.1.3.0/30 [110/122] via 152.1.2.2, 00:27:27, Serial0
C       152.1.2.0/30 is directly connected, Serial0
C    192.1.1.0/24 is directly connected, Ethernet0
```

RouterD is acting as an ASBR, redistributing the RIP learned routes from RouterE into OSPF. Display the routing table on RouterA with the command **show ip route**. What follows is the output from the command; notice that RouterA has entries for all seven networks.

```
RouterA# sho ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     130.1.0.0/24 is subnetted, 7 subnets
O E2    130.1.3.0 [110/10] via 152.1.2.2, 00:36:40, Serial0
O E2    130.1.2.0 [110/10] via 152.1.2.2, 00:36:40, Serial0
O E2    130.1.1.0 [110/10] via 152.1.2.2, 00:36:40, Serial0
O E2    130.1.7.0 [110/10] via 152.1.2.2, 00:36:40, Serial0
O E2    130.1.6.0 [110/10] via 152.1.2.2, 00:36:40, Serial0
O E2    130.1.5.0 [110/10] via 152.1.2.2, 00:36:40, Serial0
O E2    130.1.4.0 [110/10] via 152.1.2.2, 00:36:40, Serial0
     152.1.0.0/16 is variably subnetted, 3 subnets, 2 masks
O IA    152.1.1.0/24 [110/84] via 152.1.2.2, 00:10:59, Serial0
O IA    152.1.3.0/30 [110/122] via 152.1.2.2, 00:36:44, Serial0
C       152.1.2.0/30 is directly connected, Serial0
C    192.1.1.0/24 is directly connected, Ethernet0
```

The networks range from 130.1.1.0 to 130.1.7.0; since these are contiguous, they can be summarized into one entry by the ASBR. To accomplish this, add the following command to RouterD under the OSPF process.

```
RouterD(config)#router ospf  64
RouterD(config−router)#summary−address 130.1.0.0 255.255.248.0
```

Without summarization, a router advertising these seven networks would need to send a separate route update for each of these networks. Summarization allows a router to advertise more than one network with a single advertisement. In the case of our seven networks, they can be advertised as 130.1.0.0 with a 21−bit mask.

In the table that follows, we see that all seven networks have an exact match for their first 21 bits. Notice that the .1, .2, .3, .4, .5, .6, and .7 networks use seven combinations of the three remaining bits (.0 is not used). Thus, a 21−bit mask can be used to summarize the networks.

First 21 Bits Match ↓	↓
130.1.1.0	**10000010.00000001.00000**001.00000000
130.1.2.0	**10000010.00000001.00000**010.00000000
130.1.3.0	**10000010.00000001.00000**011.00000000
130.1.4.0	**10000010.00000001.00000**100.00000000
130.1.5.0	**10000010.00000001.00000**101.00000000
130.1.6.0	**10000010.00000001.00000**110.00000000
130.1.7.0	**10000010.00000001.00000**111.00000000
255.255.248.0	
Summary mask	**11111111.11111111.11111**000.00000000

Display the routing table on RouterA; what follows is the output. Notice that the seven entries are gone and only one entry exits (network 130.1.0.0 /21).

```
RouterA#SHO IP ROute
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     130.1.0.0/21  is subnetted, 1 subnets
```

```
O E2    130.1.0.0 [110/10] via 152.1.2.2, 00:03:00, Serial0
     152.1.0.0/16  is variably subnetted, 3 subnets, 2 masks
O IA    152.1.1.0/24 [110/84] via 152.1.2.2, 00:03:00, Serial0
O IA    152.1.3.0/30 [110/122] via 152.1.2.2, 00:03:00, Serial0
C       152.1.2.0/30 is directly connected, Serial0
C    192.1.1.0/24 is directly connected, Ethernet0
```

Lab #42: Regular, Stub, Totally Stub, and NSSA Areas

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having one serial port and one Ethernet port
- Two Cisco routers, each having one serial port and two Ethernet ports
- One Cisco router with an Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco V.35 DCE/DTE crossover cables
- Two Ethernet hub and four Ethernet cables (or two Ethernet Crossover cables)
- Cisco rolled cable for console port access

Overview

Cisco routers support multiple area types (Regular, Stub, NSSA, and Totally Stub); the difference in area types lies in the kind of LSAs that are permitted in the area.

In a *regular area*, all types of LSAs are permitted. The benefit is that all routers have all routing information and therefore, have the best path to the destination. The drawback is that any flap caused by a link failure outside the area will cause a partial SPF calculation.

In a *stub area*, no external LSAs are permitted; therefore, none are injected by the ABR. External LSAs are used to describe destinations outside the OSPF domain. For example, a route received from another routing protocol, such as RIP, and redistributed into OSPF is considered external and would be advertised in an external LSA.

While stub areas prevent flapping outside of the domain from affecting the area, they do not prevent flapping that occurs within the domain from affecting the area. Since summary LSAs are still permitted, flaps that occur in other areas will still affect the stub area.

Totally stubby areas, like stub areas, prevent external LSAs. Unlike a stub area, however, totally stubby areas do not permit summary LSAs. So flaps that occur within other areas will not affect the totally stubby area.

An *NSSA* area is similar to a stub area; however, it can import external routes into the area. The routes are carried across the area as type 7 LSAs and converted to type 5 LSAs by the ABR. A NSSA area would be used if, for example, you wanted to prevent external LSAs from entering the area, but you still needed to send external LSAs out of the area, for example, if one of the routers in the area was a ASBR.

Configuration Overview

This lab will demonstrate the various area types used in OSPF. The connectivity and addressing are the same as in Lab #41.

RouterA will attach to RouterB via a V.35 crossover cable. RouterB will act as the DCE supplying clock to RouterA. RouterB and RouterC will be attached via an Ethenet hub. RouterC will attach to RouterD via a V.35 crossover cable. RouterC will act as the DCE supplying clock to RouterD. RouterD will attach to RouterE via an Ethenet hub. The second Ethernet interfaces on RouterB and RouterC will not attach to anything, so keep−alives will need to be disabled. The reason that Ethernet interfaces were used instead of loopback interfaces is that loopback interfaces are advertised as /32 networks across area boundaries.

RIP is run between RouterD and RouterE; RouterE will advertise all subnetworks that are attached. RouterD will redistribute the RIP learned routes into OSPF; mutual redistribution is not used. However, if you want RouterE to see the OSPF networks, it will need to be added. All IP addresses are as per Figure 8−16.

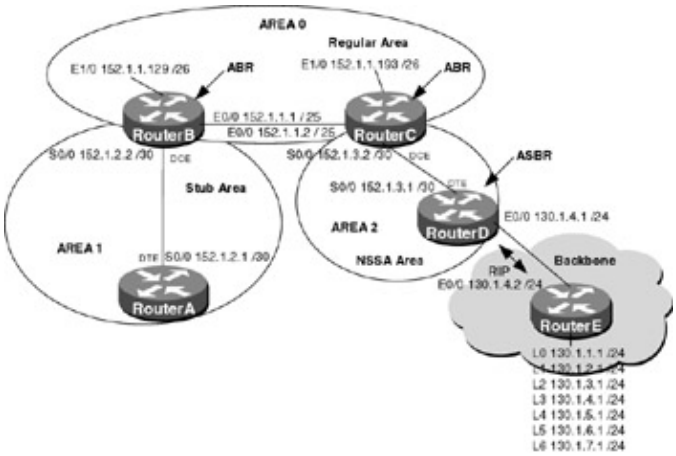


Figure 8−16: OSPF stub areas

Router Configurations

The configurations for the routers in this example are as follows (key OSPF commands are highlighted in bold).

RouterA

```
Building configuration...

Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
!
interface Ethernet0
 ip address 192.1.1.2 255.255.255.0
!
interface Serial0
 ip address 152.1.2.1 255.255.255.252
!
router ospf 64 ← Enables OSPF process 64 on the router
 network 152.1.2.0 0.0.0.255 area 1 ← Specifies what interface OSPF will be run
                                      on and what area the interface will be in

!
line con 0
line 1 16
 no exec
 transport input all
line aux 0
line vty 0 4
 password cisco
```

```
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
!
hostname RouterB
!
!
interface Ethernet 1/0
 ip address 152.1.1.129 255.255.255.192
no keepalive
!
interface Ethernet0/0
 ip address 152.1.1.1 255.255.255.128
!
interface Serial0/0
 ip address 152.1.2.2 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 1000000
!
router ospf 64 ← Enables OSPF process 64 on the router
area 0 range 152.1.1.0 255.255.255.0
 network 152.1.1.0 0.0.0.255 area 0 ← Specifies what interface OSPF will be run
                                      on and what area the interface will be in
 network 152.1.2.0 0.0.0.255 area 1
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
!
```

RouterC

```
Current configuration:
!
hostname RouterC
!
interface Ethernet 1/0
 ip address 152.1.1.193 255.255.255.192
 no ip directed-broadcast
no keepalive
!
interface Ethernet0/0
 ip address 152.1.1.2 255.255.255.128
 no ip directed-broadcast
!
interface Serial0/0
ip address 152.1.3.2 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 1000000
```

```
!
!
router ospf 64 ← Enables OSPF process 64 on the router
area 0 range 152.1.1.0 255.255.255.0
 network 152.1.1.0 0.0.0.255 area 0
 network 152.1.3.0 0.0.0.255 area 2
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterD

```
Current configuration:
!
version 11.2
!
hostname RouterD
!
!
interface Ethernet0
 ip address 130.1.4.1 255.255.255.0
 no ip directed-broadcast
!
interface Serial0
 ip address 152.1.3.1 255.255.255.252
 no ip directed-broadcast
 ip ospf interface-retry 0
!
router ospf 64 ← Enables OSPF process 64 on the router
summary-address 130.1.0.0 255.255.248.0
redistribute rip metric 10 subnets ← Redistributes RIP into OSPF (For this lab
                                     exercise only one way redistribution is
                                     needed)
  network 152.1.3.0 0.0.0.255 area 2
network 152.1.3.0 0.0.0.255 area 2
!
router rip ← Enables RIP routing
 network 130.1.0.0
!
no ip classless
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterE

```
version 11.2
!
hostname RouterE
!
interface Loopback0
 ip address 130.1.1.1 255.255.255.0
 no ip directed-broadcast
```

```
!
interface Loopback1
 ip address 130.1.2.1 255.255.255.0
 no ip directed-broadcast
!
interface Loopback2
 ip address 130.1.3.1 255.255.255.0
 no ip directed-broadcast
!
interface Loopback3
 ip address 130.1.5.1 255.255.255.0
 no ip directed-broadcast
!
interface Loopback4
 ip address 130.1.6.1 255.255.255.0
 no ip directed-broadcast
no ip directed-broadcast
!
interface Loopback5
 ip address 130.1.7.1 255.255.255.0
 no ip directed-broadcast
!
interface Ethernet0
 ip address 130.1.4.2 255.255.255.0
 no ip directed-broadcast
!
router rip ← Enables RIP routing
 network 130.1.0.0
!
ip classless
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

Display the routing table on RouterA; what follows is the output. Notice that RouterA has a OSPF external route to network 130.1.0.0 and two OSPF internal routes.

```
RouterA#sho ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     130.1.0.0/21 is subnetted, 1 subnets
O E2    130.1.0.0 [110/10] via 152.1.2.2, 00:00:01, Serial0
     152.1.0.0/16 is variably subnetted, 3 subnets, 2 masks
O IA    152.1.1.0/24 [110/84] via 152.1.2.2, 00:00:01, Serial0
O IA    152.1.3.0/30 [110/122] via 152.1.2.2, 00:00:01, Serial0
C       152.1.2.0/30 is directly connected, Serial0
C    192.1.1.0/24 is directly connected, Ethernet0
```

Enable OSPF SPF debugging on RouterA with the command **debug ip ospf spf**. Now on RouterD, shut down the Ethernet interface attached to RouterE. What follows is the output from the debug command on RouterA; notice that RouterA received an LSA type 5 packet and is running partial SPF.

```
RouterA#debug ip ospf spf

OSPF: Detect change in LSA type 5, LSID 130.1.7.255, from 152.1.3.1 area 1
OSPF: Schedule partial SPF − type 5 id 130.1.7.255 adv rtr 152.1.3.1
OSPF: Service partial SPF 0/1 /0
OSPF: Start partial processing Type 5 External LSA 130.1.7.255, mask 255.255.248
.0, adv 152.1.3.1, age 3600, seq 0x8000000D, metric 16777215, metric−type 1
OSPF: delete lsa id 130.1.7.255, type 5, adv rtr 152.1.3.1 from delete list
```

Configure Area1 to be a stub area. To do this, add the following commands under the OSPF process. All routers in a stub area must be configured as stubs for that area.

```
RouterA(config)#router ospf 64
RouterA(config−router)#area 1 stub

RouterB(config)#router ospf 64
RouterB(config−router)#area 1 stub
```

Display the routing table on RouterA; what follows is the output. Notice that RouterA no longer has an OSPF external route to network 130.1.0.0; instead, a default route has been added. The two internal routes remain, however, because stub areas do not prevent inter−area updates.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is 152.1.2.2 to network 0.0.0.0

     152.1.0.0/16 is variably subnetted, 3 subnets, 2 masks
O IA    152.1.1.0/24 [110/84] via 152.1.2.2, 00:02:12, Serial0
O IA    152.1.3.0/30 [110/122] via 152.1.2.2, 00:02:12, Serial0
C       152.1.2.0/30 is directly connected, Serial0
C    192.1.1.0/24 is directly connected, Ethernet0
O*IA 0.0.0.0/0 [110/65] via 152.1.2.2, 00:02:12, Serial0
```

Enable OSPF SPF debugging on RouterA with the command **debug ip ospf spf**. Now on RouterD, shut down the Ethernet interface attached to RouterE. What follows is the output from the debug command on RouterA; notice that RouterA has not received any type 5 packet and no partial SPF has been triggered.

```
RouterA#debug ip ospf spf
```

While debugging is still enabled on RouterA, shut down the serial interface on RouterD. What follows is the output from the debug; notice that RouterA received a type 3 (summary LSA) and is performing a partial SPF. Remember that in a stub area, inter−area traffic is still injected, so flaps from other areas still affect the local area.

```
OSPF: Detect change in LSA type 3, LSID 152.1.3.0, from 152.1.1.129 area 1
OSPF: Schedule partial SPF − type 3 id 152.1.3.0 adv rtr 152.1.1.129
OSPF: Service partial SPF 1/0/0
OSPF: Start partial processing Summary LSA 152.1.3.0, mask 255.255.255.252, adv
152.1.1.129, age 3600, seq 0x80000003 (Area 1)
OSPF: delete lsa id 152.1.3.0, type 3, adv rtr 152.1.1.129 from delete list
```

It is important to remember that all routers in an area must be consistent. That is, if one is configured as a stub, they must all be configured as stubs. This is signaled through the E bit in the optional field; if this does not match, the routers will not form an adjacency.

Make area1 on RouterA a regular area, with the following command.

```
RouterA(config)#router ospf 64
RouterA(config-router)#no area 1 stub
```

Now display the OSPF neighbor table on RouterA with the command **show ip ospf neighbors**. What follows is the output form the command; notice that the adjacency with RouterB is now down. The reason is that area 1 on RouterB is still configured as a stub area, and so there is an E bit mismatch.

```
RouterA#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
152.1.1.129       1   DOWN/  -           -        152.1.2.2       Serial0
```

This can also be verified through debugging on RouterA. Enable debugging of OSPF adjacencies on RouterA with the command **debug ip ospf adj**. What follows is the output from the debug; notice that there is a mismatch in the stub area option bit.

```
RouterA#debug ip ospf adj

OSPF: Hello from 152.1.2.2 with mismatched Stub/Transit area option bit
```

The last example showed how to configure an area to prevent external LSAs from being flooded in. In order to prevent summary LSAs from other areas from affecting the local area, the area must be configured as a totally stubby area. To do this, add the following command under the OSPF process.

```
RouterA(config)#router ospf 64
RouterA(config-router)#area 1 stub no-summary

RouterB(config)#router ospf 64
RouterB(config-router)#area 1 stub no-summary
```

Display the routing table on RouterA; what follows is the output. Notice that RouterA no longer has any OSPF external or inter-area routes; instead, it has a single default route pointing out of the area.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is 152.1.2.2 to network 0.0.0.0

     152.1.0.0/30 is subnetted, 1 subnets
C       152.1.2.0 is directly connected, Serial0
C    192.1.1.0/24 is directly connected, Ethernet0
O*IA 0.0.0.0/0 [110/65] via 152.1.2.2, 00:00:10, Serial0
```

Enable OSPF SPF debugging on RouterA with the command **debug ip ospf spf**. Now on RouterD, shut down the serial interface attached to RouterB. What follows is the output from the debug command on RouterA; notice that RouterA has not received any type 3 packet and no partial SPF has been triggered.

```
RouterA#debug ip ospf spf
```

Since RouterD in Area 3 is an ASBR, we cannot make area 2 a stub area. If you tried to configure area 2 as a stub area, you will receive the following error. Since RouterD is a ASBR for area 2, by definition it cannot be in a stub area.

```
RouterD(config)#router ospf 64
RouterD(config-router)# area 2 stub

OSPF: Stub command is invalid when it is ASBR
```

To get around this limitation, OSPF uses a special area type called an NSSA area. An NSSA area is similar to a OSPF stub area, but it has the capability to import AS external routes in a limited capacity within the NSSA area. To configure area 2 as a NSSA, use the following commands.

```
RouterD(config)#router ospf 64
RouterD(config−router)#area 2 nssa

RouterC(config)#router ospf 64
RouterC(config−router)#area 2 nssa
```

Display the routing table on RouterC; what follows is the output. Notice that network 130.1.0.0 is being learned as a N2 − OSPF NSSA external type 2 route.

```
RouterC#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, ia − IS−IS inter area
       * − candidate default, U − per−user static route, o − ODR
       P − periodic downloaded static route

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 5 subnets, 4 masks
O       152.1.1.129/32 [110/11] via 152.1.1.1, 00:00:39, Ethernet0/0
C       152.1.1.192/26 is directly connected, Ethernet1/0
C       152.1.1.0/25 is directly connected, Ethernet0/0
C       152.1.3.0/30 is directly connected, Serial0/0
O IA    152.1.2.0/30 [110/58] via 152.1.1.1, 00:00:09, Ethernet0/0
     130.1.0.0/21 is subnetted, 1 subnets
O N2    130.1.0.0 [110/10] via 152.1.3.1, 00:00:09, Serial0/0
```

Troubleshooting OSPF

The Cisco IOS provides many tools for troubleshooting OSPF. Here is a list of key commands along with sample output from each.

{show ip ospf } This exec command displays general information about OSPF routing processes. This command can display information on a specific routing process when you enter the process number after the command. If no process number is entered, the command will display all OSPF processes on the router.

```
RouterA#show ip ospf
 Routing Process "ospf 64" with ID 192.1.1.1
 Supports only single TOS(TOS0) routes
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Number of DCbitless external LSA 0
 Number of DoNotAge external LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
    Area BACKBONE(0)
        Number of interfaces in this area is 3
        Area has no authentication
        SPF algorithm executed 16 times
        Area ranges are
        Link State Update Interval is 00:30:00 and due in 00:05:11
        Link State Age Interval is 00:20:00 and due in 00:15:10
        Number of DCbitless LSA 1
        Number of indication LSA 0
        Number of DoNotAge LSA 0
```

{show ip ospf interface} This exec command displays information on all OSPF configured interfaces. The command can be used to display information on only a particular interface, by specifying the interface after the command. If no interface is specified, the command will display OSPF information for all interfaces on the router.

This one command will tell you the status of the interface, the IP address, the area that it is attached to, the router ID, the interface network type, the interval timers, and more. When troubleshooting an OSPF network, this should be one of the first commands you use.

```
RouterA#show ip ospf interface s0
Serial0 is up, line protocol is up
 Internet Address 192.1.1.1/24, Area 0
 Process ID 64, Router ID 192.1.1.1, Network Type POINT_TO_POINT, Cost: 66
 Transmit Delay is 1 sec, State POINT_TO_POINT,
 Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
  Hello due in 00:00:22
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
```

{show ip ospf neighbor} This exec command displays OSPF neighbor information for the router. The command can be used to display information about a particular neighbor, specifying the neighbor's router ID after the command. If no router ID is specified, the command will display information on all OSPF neighbors.

The most important information this command gives is the state of the adjacency with the neighbor. When troubleshooting an OSPF network, this should be the second command you use.

```
RouterA#show ip ospf neighbor

Neighbor ID     Pri     State        Dead Time     Address      Interface
3.3.3.3         1       FULL/BDR     00:00:36      10.1.1.2     Ethernet0
```

{show ip ospf database} This exec command displays information related to the OSPF database for a specific router.

```
RouterA#show ip ospf database

       OSPF Router with ID (192.1.1.1) (Process ID 64)

          Router Link States (Area 0)
Link ID      ADV Router     Age       Seq#           Checksum Link count
3.3.3.3      3.3.3.3        1594      0x80000008     0x7174      4
192.1.1.1    192.1.1.1      1751      0x8000000C     0x573D      3
193.1.1.2    193.1.1.2      140       0x8000000A     0x610B      4
          Net Link States (Area 0)
Link ID      ADV Router     Age       Seq#           Checksum
10.1.1.1     192.1.1.1      1751      0x80000004     0x1185
```

{show ip ospf virtual−links} This exec command displays information about the virtual links configured on the router.

```
RouterC#show ip ospf virtual−links
Virtual Link OSPF_VL0 to router 2.2.2.2 is up
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 1, via interface Serial0/0, Cost of using 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:07
    Adjacency State FULL (Hello suppressed)
```

{debug ip ospf events} This exec command displays information on OSPF−related events, such as the forming of adjacencies, flooding information, designated router selection, and shortest path first (SPF)

calculation. The output debug command that follows shows the steps that the router goes through while attempting to form adjacencies.

```
RouterC#debug ip ospf events
OSPF: 2 Way Communication to 193.1.1.2 on Serial0, state 2WAY
OSPF: Send DBD to 193.1.1.2 on Serial0 seq 0x1 opt 0x2 flag 0x7 len 32
OSPF: Rcv DBD from 193.1.1.2 on Serial0 seq 0x1CCD opt 0x2 flag 0x7 len 32 state
 EXSTART
OSPF: NBR Negotiation Done. We are the SLAVE
OSPF: Send DBD to 193.1.1.2 on Serial0 seq 0x1CCD opt 0x2 flag 0x2 len 72
OSPF: Rcv DBD from 193.1.1.2 on Serial0 seq 0x1CCE opt 0x2 flag 0x3 len 72 state
 EXCHANGE
OSPF: Send DBD to 193.1.1.2 on Serial0 seq 0x1CCE opt 0x2 flag 0x0 len 32
OSPF: Rcv DBD from 193.1.1.2 on Serial0 seq 0x1CCF opt 0x2 flag 0x1 len 32 stat
```

{debug ip ospf packet} This exec command displays information on every OSPF packet that is received by the router, such as OSPF version, OSPF packet type, packet length, router ID, area ID, authentication type, and authentication key.

```
RouterC#debug ip ospf packets
OSPF: rcv. v:2 t:1 l:48 rid:193.1.1.2
      aid:0.0.0.0 chk:3437 aut:0 auk: from
Serial0
```

Using Table 8−1, we can tell that RouterC received an OSPF version 2 hello packet that is 48 bytes long. The Router ID of the sending router is 193.1.1.2, which is attached to area 0; there is no authentication used on this packet.

Table 8−1: Debug IP OSPF Packet

Field	Description
v:	OSPF version.
t:	OSPF packet type. Possible packet types follow: 1— Hello 2 —Data description 3 — Link state request 4 —Link state update 5 — Link state acknowledgment
l:	OSPF packet length in bytes.
rid:	OSPF router ID.
aid:	OSPF area ID.
chk:	OSPF checksum.
aut:	OSPF authentication type. Possible authentication types follow: 0 —No authentication 1 —Simple password 2 —MD5
auk:	OSPF authentication key.

keyid:	MD5 key ID.
seq:	Sequence number.

Conclusion

As you can see from this chapter, a network that is based on a link state protocol, such as OSPF, is considerably more complex to configure, design, and troubleshoot than a network based on a distance vector protocol, such as RIP. However, OSPF provides the following advantages over RIP version 1:

- No limitation on hop count.
- Faster convergence than RIP, because routing changes are flooded throughout the network instantly.
- Security, in that OSPF supports router authentication and RIP does not.
- OSPF has a concept of route tagging of external routes that are injected into the AS. This allows the protocol to keep track of external routes that are injected by other protocols such as BGP.
- OSPF is classless; RIP is classful.
- OSPF uses the available bandwidth more effectively, by only sending routing updates when there is a change.
- OSPF uses multicast packets verses broadcast packets to send LSAs. This assures that routers that are not configured for OSPF do not have to process the packet.

Chapter 9: Enhanced Interior Gateway Routing Protocol

Overview

Topics Covered in This Chapter

- Detailed technology overview
- EIGRP terminology
- EIGRP metrics explained
- EIGRP passive interface
- Basic EIGRP configuration
- EIGRP unequal−cost load balancing
- EIGRP timer configuration
- Configuring EIGRP on an NBMA network
- Detailed troubleshooting examples

Introduction

Enhanced Interior Gateway Routing Protocol (EIGRP) is a Cisco proprietary advanced distance vector routing protocol, which was first released in 1994 (IOS 9.21) to address the limitations of traditional distance vector and link state protocols.

Traditional distance vector protocols, such as RIP, forward routing updates to all attached neighbors, which in turn forward the updates to their neighbors. This hop−by−hop propagation of routing information creates large convergence times and looped topology problems.

Link state protocols such as OSPF have been offered as an alternative to the tradition distance vector protocols. The problem with link state protocols is that they solve the convergence problems of traditional distance vector protocols by replicating the topology

information across the entire domain. This replication becomes undesirable in large networks and greatly affects CPU utilization due to the number of SPF calculations that need to be run.

EIGRP Terminology

When dealing with EIGRP, it is important to understand the terminology being used.

Successor: The successor is the directly connected neighboring router that has the best route to reach a particular destination. This is the route that is used by the router to forward packets to a given destination. In order for a neighbor to become the successor for a particular destination, it must first meet the feasibility condition.

The feasibility condition states that the route must be advertised from a neighbor that is downstream with respect to the destination, and the cost to reach the destination must be less than or equal to the cost of the route that is currently being used by the routing table.

For example, in Figure 9−1, RouterB's successor to reach NetworkA is RouterA, because the cost to reach NetworkA is 2, which is lower than going through RouterC, which is 3. However, if the metric of the link between RouterA and RouterB changed from 1 to 20, RouterC would meet the feasibility condition and become the successor.

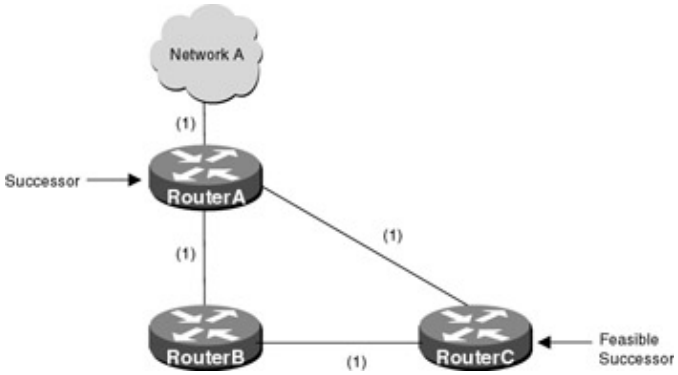


Figure 9−1: EIGRP terminology

Feasible successor: The feasible successor is a neighboring router that the destination can be reached through, but is not used because the cost to reach the destination is higher than via a different router. The feasible successor can be thought of as having the next best route to a destination.

Feasible successors are kept in the topology table and are used as backup routes. For example, in Figure 9−1, RouterB's feasible successor to reach NetworkA is RouterC. RouterC has a route to NetworkA — however, it is not the least cost path and therefore is not used to forward data.

Feasibility condition: The feasibility condition is used to prevent routing loops. In order for the feasibility condition to be met, the route must be advertised from a neighbor that is downstream with respect to the destination. The cost to reach the destination must be less than or equal to the cost of the route that is currently being used in the routing table. If the feasibility condition is met, the neighbor becomes the successor. For example, in Figure 9−1, if the link between RouterB and RouterA were to fail, RouterA would no longer be the successor. RouterC would move from being the feasible successor to the successor. If the link between RouterA and RouterB became active again, RouterA would take over as successor because it meets the feasibility condition. It is downstream from NetworkA and its cost to reach NetworkA is less than RouterC cost to reach NetworkA.

Active state: When the router loses its route to a destination and no feasible successor is available, the router goes into active state. While in active state, the router sends out queries to all neighbors in order to find a route to the destination. At this time, the router must run the routing algorithm to recompute a new route to the destination.

Passive state: When the router loses its successor but has a feasible successor, it goes into passive state.

Hello: Hello packets are exchanged between neighboring routers. As long as hello packets are received, the router can determine that the neighbor is alive and functioning.

ACKs: Acknowledgementpackets are sent by the router to acknowledge the receipt of update packets.

Update: Update packets are used by the router to send routing information between neighbors. Update messages are sent if the metric of a route changes or when a router first comes up.

Query: When the router loses its route to a destination and no feasible successor is available, the router goes into active state. While in this active state, the router sends out query packets to all neighbors for a particular destination. The router waits for a response back from all neighbors before starting the computation for a new successor.

Replies: Replies are sent in response to queries. The reply contains information on how to reach a destination. If the queried neighbor does not have the information requested, it sends queries to all its neighbors.

Technology Overview

When an EIGRP−enabled router first comes online, it sends hello packets out all EIGRP−enabled interfaces, using multicast address 224.0.0.10. The hello packets are used for two things: discovering neighboring routers and, after the neighbor are discovered, determining if a neighbor has become unreachable or inoperative.

Once a new neighbor is discovered via the hello packet, the router records the IP address and interface that the neighbor was discovered on. The router then sends an update containing all of the routes that it knows about to the neighbor, and the neighbor does the same. This information is stored in the EIGRP topology table.

Subsequently, hello packets are sent out every 5 seconds, or every 60 seconds on low−speed NBMA networks. The hello packets allow the router to discover loss of its neighbor dynamically and quickly. If a hello packet is not received from the neighbor router before the expiration of the HoldTimer, the neighbor is declared down. At this point, the neighbor adjacency is deleted and all routes associated with that neighbor are removed.

The topology table includes the router's metric to reach the destination as well as the neighbors metric to reach the destination. The DUAL algorithm uses the topology table to find the lowest metric loop free path to each destination. The next−hop router for the lowest−cost path is referred to as the successor and is the next−hop IP address that is loaded in the routing table. The DUAL algorithm also tries to find a feasible successor, or the next−best route, which is kept in the topology database.

If the router loses its successor and a feasible successor is available, no route recomputation is necessary. The router simply makes the feasible successor the successor and adds the new route to the routing table, remaining in a passive state. However, if no feasible successor is available, the router goes into active state for the destination network and recomputation for the route is necessary.

While the router is in active state, it sends a query packet out to all EIGRP−enabled interfaces, except the interface the successor is on, inquiring if the neighbor has a route to the given destination. The neighbors respond and notify the sender if it has a route to the destination or not. Once all replies are received, the router can than calculate a new successor. If the neighbor receiving the query packet was using the sender to reach the destination network (as its successor), it will query all of its neighbors for a route to the destination. The queried neighbors go through the same process. This creates a cascading of queries through the network, searching the network for a path to the destination.

As long as EIGRP has a feasible successor, no recomputation is necessary. This prevents the router from having to use CPU cycles and also speeds up convergence. Routers that are not affected by topology changes are not involved in recomputations.

EIGRP Metrics

The EIGRP metric is a 32−bit number, which is calculated using bandwidth, delay, reliability, loading, and MTU. Calculating the metric for a route is a two−step process using the five different characteristics of the link and the K values. The K values are configurable, but this is not recommended. The default K values are K1 = 1, K2 = 0, K3 = 1, K4 = 0, and K5 = 0.

1. **Metric** = K1*Bandwidth + (K2 * Bandwidth)/(256 − load) + K3*Delay
2. If K5 is not equal to zero, take the metric from step 1 and multiply it by[K5/(reliability + K4)]. If K5 is zero, ignore step 2.

   ```
   Metric = Metric * [K5/(reliability + K4)]
   ```

As shown above, Cisco sets K2, K4, and K5 to zero. This leaves only two variables to compute the EIGRP metric (bandwidth and delay). Since three of the K values are zero, the formula reduces to:

```
Metric = Bandwidth + Delay
```

The bandwidth is derived by finding the smallest of all bandwidths in the path to the destination and dividing 10,000,000 by that number.

Delay is found by adding all of the delays along the paths and dividing that number by 10. The sum of the two numbers is then multiplied by 256. This equation can be written as:

```
Metric = [(10,000,000/min bandwidth) + (SUM(interface delay)/10)] * 256
```

Let's look at Figure 9−2 and determine what the metric is to reach network 1.0.0.0 from RouterB.

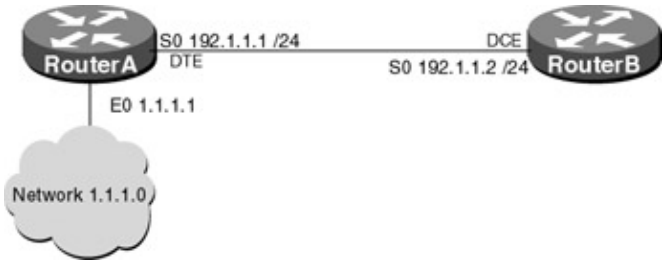


Figure 9−2: EIGRP metric

Use the **show interface** command on each router to determine what the bandwidth and delay is for each interface.

```
RouterB#show interfaces S0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 192.1.1.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/3 (active/max active)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 0 bits/sec, 1 packets/sec
  5 minute output rate 0 bits/sec, 1 packets/sec
     155 packets input, 10368 bytes, 0 no buffer
     Received 80 broadcasts, 0 runts, 1 giants, 0 throttles
     5 input errors, 1 CRC, 2 frame, 0 overrun, 0 ignored, 1 abort
     246 packets output, 13455 bytes, 0 underruns
     0 output errors, 0 collisions, 910 interface resets
     0 output buffer failures, 0 output buffers swapped out
     154 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

RouterA#show interfaces e0/0
Ethernet0/0 is up, line protocol is up
  Hardware is AmdP2, address is 00e0.1e5b.25a1 (bia 00e0.1e5b.25a1)
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 243/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive not set
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:08, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     6 packets output, 1071 bytes, 0 underruns
     6 output errors, 0 collisions, 2 interface resets
```

```
     0 babbles, 0 late collision, 0 deferred
     6 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

To reach network 1.1.1.0 from RouterB, a packet will cross the serial interface between RouterA and RouterB and the Ethernet interface on RouterA. Since the lowest bandwidth is used for the calculation, the bandwidth of the serial interface is used.

```
Metric = [(10,000,000/BW Serial link) + ((delay on serial link + delay on
 the Ethernet link)/10)] * 256

Metric = [(10,000,000/1544) + ((20000 + 1000)/10)] * 256

Metric = 2195456
```

Lets take a look at the routing table on RouterB and see if our calculations are correct.

```
RouterB#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

D    1.0.0.0/8 [90/2195456] via 192.1.1.1, 00:21:50, Serial0/0
C    192.1.1.0/24 is directly connected, Serial0/0
```

IOS Requirements

EIGRP first became available in IOS 9.21; however, EIGRP was significantly enhanced in releases 10.3.(11), 11.0(8), and 11.1(3). All Labs were performed using IOS 11.2.

Commands Discussed in This Chapter

- **debug eigrp fsm**
- **debug eigrp packet**
- **debug ip eigrp**
- **ip hello−interval eigrp** autonomous−system−number seconds
- **ip hold−time eigrp** autonomous−system−number seconds
- **network** network−number
- **no ip split−horizon eigrp** autonomous−system−number
- **passive−interface** type number
- **router eigrp** autonomous−system number
- **show ip eigrp interfaces** [interface] [as−number]
- **show ip eigrp neighbors** [type number]
- **show ip eigrp topology** autonomous−system−number
- **show ip eigrp traffic** autonomous−system−number
- **show ip protocols**
- **traffic−share** [balanced | min]
- **variance** multiplier

Definitions

debug eigrp fsm: This exec command displays information on EIGRP feasible successor metrics (FSM).

debug eigrp packet: This exec command displays information on any EIGRP messages traveling between the routers.

debug ip eigrp: This exec command displays information on any EIGRP packets that are sent or received by the router.

ip hello−interval eigrp: This interface configuration command sets the hello interval in seconds for the EIGRP routing process. The default hello timer is 60 seconds for low−speed (any network T1 or slower) nonbroadcast multiple access (NBMA) networks, and for all other networks the default is 5 seconds.

ip hold−time eigrp: This interface configuration command sets the holdtime in seconds for an EIGRP process. The default holdtime is 180 seconds for low−speed (any network T1 or slower) NBMA networks, and for all other networks the default is 15 seconds.

network: This router configuration command specifies a list of networks on which the EIGRP routing process will run. This command sends EIGRP updates to the interfaces that are specified. If an interface's network is not specified, it will not be advertised in any EIGRP updates.

no ip split−horizon eigrp: This interface configuration command disables split horizons on a particular interface. Split horizons block the sending of routing information out the same interface from which it was received. This behavior is used to prevent routing loops; however, in the case of NBMAs such as frame relay or ATM, this behavior can prevent routing information from being passed to spoke routers.

passive−interface: This router configuration command disables the sending of routing updates on a given interface. If you disable the sending of routing updates on an interface, the particular network will continue to be advertised out other EIGRP−enabled interfaces. Any routing updates received by the router on a passive interface will still be processed.

router eigrp: This global command enables the Enhanced Interior Gateway Routing Protocol (EIGRP) routing process on the router.

show ip eigrp interfaces: This exec command displays information on all interfaces configured for EIGRP.

show ip eigrp neighbors: This exec command displays information on all neighbors that are discovered by EIGRP. This command is helpful in determining when neighbors become active or inactive.

show ip eigrp topology: This exec command displays the EIGRP topology table and is very useful in debugging problems with the DUAL algorithm.

show ip eigrp traffic: This exec command displays the number of EIGRP packets sent and received by the router.

show ip protocols: This exec command displays the current state of all active routing protocol processes.

traffic−share: This router configuration command controls how traffic is distributed across routes when there are multiple routes to the same destination network that have different costs. The traffic can be distributed proportionately to the ratios of the metrics or can be set to only use routes that have minimum costs.

variance: This router configuration command controls the load balancing over multiple EIGRP paths. This command allows the administrator to load balance across multiple paths even if the metrics of the paths are different. By default, the amount of variance is set to 1 (equal−cost load balancing). The variance command

allows the user to define how much worse the metric of an alternate path can be and still be used to route packets to a given destination. For example, if the variance is set to 2, the router will load balance across up to four paths as long as the metric is lower than twice the metric of the best route. This will be explained in more detail in Lab #45.

Lab #43: Basic EIGRP Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program for connecting to the console port of the router
- One Ethernet hub and 2 Ethernet cables
- Two Cisco DTE/DCE crossover cables
- One Cisco rolled cable

Configuration Overview

This configuration will demonstrate basic routing using EIGRP. All routers will be configured for EIGRP.

RouterA and RouterB are connected via an Ethernet hub, and RouterC is connected to RouterA and RouterB serially via a crossover cable. RouterC will act as the DCE, supplying clock to RouterA and RouterB. The IP addresses are assigned as per Figure 9−3. All routers will be configured for EIGRP and will advertise all connected networks.

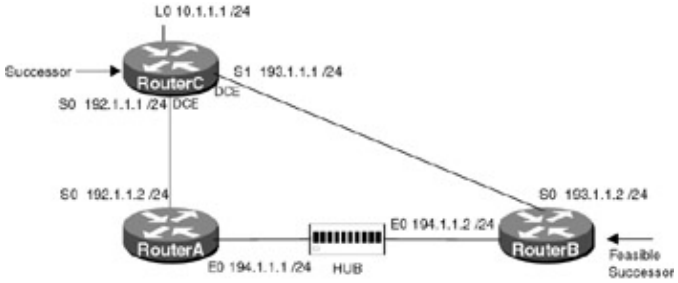


Figure 9−3: Basic EIGRP configuration

Router Configurations

The configurations for the three routers in this example are as follows (key EIGRP configurations are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Ethernet0
 ip address 194.1.1.1 255.255.255.0
!
```

```
interface Serial0
 ip address 192.1.1.2 255.255.255.0
!
interface Serial1
 no ip address
 shutdown
!
!
router eigrp 64 ← Enables the EIGRP routing process on the router
network 192.1.1.0 ← Specifies what interfaces will receive and send EIGRP
                    routing updates. It also specifies what networks will be
                    advertised

 network 194.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.0
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0
 ip address 194.1.1.2 255.255.255.0
!
interface Serial0
 ip address 193.1.1.2 255.255.255.0
 no fair-queue
 !
!
router eigrp 64← Enables the EIGRP routing process on the router
network 193.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertise

 network 194.1.1.0
!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!

version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.0
```

```
!
interface Ethernet0
 no ip address
shutdown
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
clockrate 500000 ← Acts as DCE providing clock
!
interface Serial1
 ip address 193.1.1.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
router eigrp 64 ← Enables the EIGRP routing process on the router
network 10.0.0.0 ← Specifies what interfaces will receive and send IGRP routing
                    updates. It also specifies what networks will be advertised
network 193.1.1.0
network 192.1.1.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Like IGRP, EIGRP is a very simple protocol to configure and troubleshoot. Show the IP routing table on RouterA with the **show ip route** command. Below is the output from this command. Notice that two networks were learned via EIGRP, 10.0.0.0 and 193.1.1.0. EIGRP routing table entries are identified by the letters "D" and "EX". A "D" is a route that is within the same AS, and an "EX" is a route that has been received from a different AS. EIGRP internal routes have an administrative distance of 90 and EIGRP external routes have a administrative distance of 170.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

D    10.0.0.0/8 [90/2297856] via 192.1.1.1, 00:22:19, Serial0
C    192.1.1.0/24 is directly connected, Serial0
D    193.1.1.0/24 [90/2195456] via 194.1.1.2, 00:22:20, Ethernet0
C    194.1.1.0/24 is directly connected, Ethernet0
```

The network 10.0.0.0 appears as a classful network in RouterA's routing table instead of 10.1.1.0, which is the subnet on RouterC. The reason for this is, by default, automatic summarization is performed when there are two or more network router configuration commands configured for the IP EIGRP process. To disable automatic summarization, use the command **no auto−summary.**

Display the information about EIGRP with the command **show ip protocols**. Notice that on RouterC, automatic summarization is on and the router is summarizing network 10.0.0.0.

```
RouterC#show ip protocols
Routing Protocol is "eigrp 64"
```

```
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 64
      Automatic network summarization is in effect
      10.0.0.0/8  for Serial 0
      Summarizing with metric 2297856
  Routing for Networks:
    10.0.0.0
      Passive Interface(s):
    Routing Information Sources:
    Gateway         Distance      Last Update
    (this router)          5      00:00:03
    Gateway         Distance      Last Update
    10.1.3.2              90      00:41:18
    10.1.2.2              90      00:00:14
  Distance: internal 90 external 170
```

Let's see what happens when auto summary is disabled on RouterC. From RouterC, enter the command **no auto summary** under the EIGRP routing process.

```
RouterC#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
RouterC(config)#router eigrp 64
RouterC(config-router)#no auto-summary
```

Display the information about EIGRP with the command **show ip protocols**. Notice that on RouterC, automatic network summarization is now off.

```
RouterC#show ip protocols
Routing Protocol is "eigrp 64"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 64
  Automatic network summarization is not in effect
  Routing for Networks:
    10.0.0.0
```

Display the contents of the routing table on RouterA with the command **show ip route.** Notice that the subnet 10.1.1.0 is now in the routing table.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

10.0.0.0/24 is subnetted, 1 subnets
D       10.1.1.0 [90/2297856] via 192.1.1.1, 00:13:00, Serial0
C    192.1.1.0/24 is directly connected, Serial0
D    193.1.1.0/24 [90/2195456] via 194.1.1.2, 00:13:00, Ethernet0
C    194.1.1.0/24 is directly connected, Ethernet0
```

From RouterA, monitor the hello exchanges being passed between neighbors using the **debug eigrp packets** command. Below is the output from this command. Notice the hello packets being sent between neighbors. Hello packets are sent periodically between neighboring routers, allowing the router to quickly and dynamically discover the loss of a neighbor.

```
RouterA#debug eigrp packets
EIGRP Packets debugging is on
    (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK)
RouterA#
EIGRP: Received HELLO on Serial0 nbr 192.1.1.1
  AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
EIGRP: Sending HELLO on Ethernet0
  AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Sending HELLO on Serial0
  AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Received HELLO on Ethernet0 nbr 194.1.1.2
  AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
```

Display RouterA's EIGRP neighbors with the command **show ip eigrp neighbors**. Below is the output from the command. Notice that RouterA has two neighbors, 192.1.1.1 (RouterC) and 194.1.1.2 (RouterB). The command displays information on the autonomous system number, how long the neighbor has been up, and what interface the neighbor is on.

```
RouterA#show ip eigrp neighbors
IP-EIGRP neighbors for process 64
H   Address      Interface    Hold      Uptime     SRTT    RTO     Q      Seq
                              (sec)                (ms)            Cnt    Num
0   192.1.1.1    Se0          13        00:13:14   15      200     0      31
1   194.1.1.2    Et0          10        00:57:17   35      210     0      38
```

From RouterA, display the EIGRP topology database with the command **show ip eigrp topology.** Notice the letter (P) preceding the destination address. This indicates the router is in passive state for the particular destination. When a router is in the passive state, no EIGRP recomputations are being performed for this destination. The router will only perform a recomputation if it has lost its successor and no feasible successor is available.

For destination network 10.0.0.0, RouterA can reach the network two ways: via 192.1.1.1 or via 194.1.1.2. The successor is the route via 192.1.1.1. This is because the cost to reach network 10.0.0.0 is less via 192.1.1.1 than via 194.1.1.2. However, since network 10.0.0.0 can be reached via 194.1.1.2, this route becomes the feasible successor.

```
RouterA#show ip eigrp topology
IP-EIGRP Topology Table for process 64

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status

P 10.0.0.0/8, 1 successors, FD is 2297856
         via 192.1.1.1 (2297856/128256), Serial0
P 192.1.1.0/24, 1 successors, FD is 2169856
         via Connected, Serial0
P 193.1.1.0/24, 1 successors, FD is 2195456
         via 194.1.1.2 (2195456/2169856), Ethernet0
         via 192.1.1.1 (2681856/2169856), Serial0
P 194.1.1.0/24, 1 successors, FD is 281600
         via Connected, Ethernet0
```

Let's see what happens when RouterA loses its primary route (successor) to destination network 10.0.0.0. From RouterA, shut down the serial interface 0.

```
RouterA#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
RouterA(config)#interface s0
RouterA(config-if)#shutdown
```

Display the EIGRP topology on RouterA with the command **show ip eigrp topology**. Notice the successor for network 10.0.0.0 is now 194.1.1.2, which is RouterB's Ethernet interface. Also note that the router's state for this destination is still (P) passive, the router will only go active if it loses its successor and no feasible successor is available.

```
RouterA#show ip eigrp topology
IP-EIGRP Topology Table for process 64

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status

P 10.0.0.0/8, 1 successors, FD is 2323456
         via 194.1.1.2 (2323456/2297856), Ethernet0
P 193.1.1.0/24, 1 successors, FD is 2195456
         via 194.1.1.2 (2195456/2169856), Ethernet0
P 194.1.1.0/24, 1 successors, FD is 281600
         via Connected, Ethernet0
```

From RouterB, delete the IGRP process and add a new process using autonomous system 56, with the following commands:

```
RouterB#configure terminal
RouterB(config)#no router eigrp 64
RouterB(config)#router eigrp 56
RouterB(config-router)#network 193.1.1.0
RouterB(config-router)# network 194.1.1.0
```

Show the EIGRP neighbors on RouterA with the command **show ip eigrp neighbors**.

```
RouterA#show ip eigrp neighbors
IP-EIGRP neighbors for process 64
H   Address     Interface    Hold    Uptime     SRTT    RTO     Q     Seq
                             (sec)              (ms)            nt    Num
1   192.1.1.1   Se0          11      00:06:31   43      258     0     55
```

Notice that RouterB is no longer a neighbor and no networks are being learned via EIGRP for RouterB. This is because the autonomous system numbers are different. The autonomous system number must match, or the routers will not exchange routing information.

Lab #44: Passive Interface Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables
- One Cisco rolled cable for accessing the console port of the router

Configuration Overview

This configuration will demonstrate the use of the **passive−interface** command, which allows EIGRP−enabled routers to disable the sending of EIGRP packets on a particular interface.

The **passive−interface** router configuration command is typically used when the **network** router configuration command configures more interfaces than is desirable. For example, in Figure 9−4, RouterA has three subnets (10.1.1.0/24, 10.1.2.0/24, and 10.1.3.0/24) defined. When EIGRP is enabled, it is turned on for the classful network 10.0.0.0. This encompasses all three subnets; the **passive−interface** command allows the user to turn off EIGRP advertisements on a particular interface (subnet).

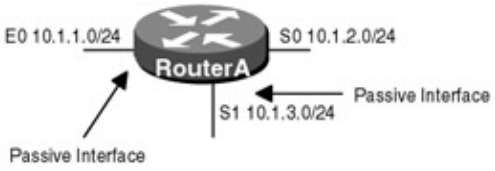


Figure 9−4: Passive interface

In this lab scenario, the user only wishes to send EIGRP updates out network 10.1.2.0, so interface E0 (10.1.1.0) and S1 (10.1.3.0) are made passive.

RouterA is connected to RouterB and RouterC via a serial crossover cable. RouterA will act as the DCE supplying clock to RouterB and RouterC. The IP addresses are assigned as per Figure 9−5. All routers will be configured for EIGRP, and RouterB and RouterC will advertise all connected networks. RouterA's interface S1 will be passive and will not advertise any routing information.

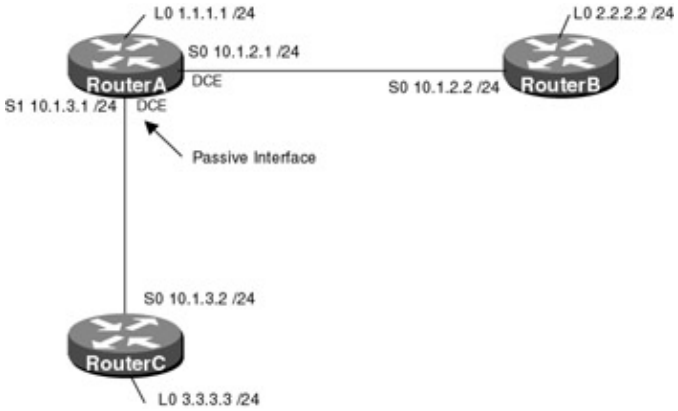


Figure 9−5: EIGRP passive interface configuration

Router Configurations

The configurations for the three routers in this example are as follows (key EIGRP configurations are highlighted in bold).

RouterA

```
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0
 no ip address
shutdown
!
interface Serial0
```

```
 ip address 10.1.2.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
interface Serial1
 ip address 10.1.3.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router eigrp 64 ← Enables the EIGRP routing process on the router
 network 10.0.0.0 ← Specifies what interfaces will receive and send EIGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 1.0.0.0
 no auto-summary ← Turns off automatic summarization of subnet routes into
                   network-level routes
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 10.1.2.2 255.255.255.0
!
router eigrp 64 ← Enables the EIGRP routing process on the router
 network 10.0.0.0 ← Specifies what interfaces will receive and send EIGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 2.0.0.0
 no auto-summary ← Turns off automatic summarization of subnet routes into
                   network-level routes
!
no ip classless
!
line con 0
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
```

```
hostname RouterC
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 10.1.3.2 255.255.255.0
!
router eigrp 64 ← Enables the EIGRP routing process on the router
 network 3.0.0.0
 network 10.0.0.0 ← Specifies what interfaces will receive and send EIGRP
                    routing updates. It also specifies what networks will be
                    advertised

 no auto-summary ← Turns off automatic summarization of subnet routes into
                   network-level routes

!
no ip classless
!
line con 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

When EIGRP is enabled on the router, the classful network number is used — just like with RIP or IGRP. If you enter a subnet with the **network** command, the router converts it back to the classful network. For example, if you try to enable EIGRP on subnetwork 10.1.1.0, the router converts it to its natural network number, which is 10.0.0.0, therefore enabling EIGRP on any subnetwork of 10.0.0.0. EIGRP updates can be disabled on an interface basis through the use of the **passive−interface** command.

Lets look at RouterA and see what interfaces EIGRP is configured on with the command **show ip eigrp interfaces.** Notice that EIGRP is configured on S0 and S1, which are both subnets of 10.0.0.0.

```
RouterA#show ip eigrp interfaces
IP-EIGRP interfaces for process 64
                      Xmit Queue     Mean    Pacing Time    Multicast    Pending
Interface    Peers    Un/Reliable    SRTT    Un/Reliable    Flow Timer   Routes
Se0          1        0/0            94      0/15           479          0
Lo0          0        0/0            0       0/10           0            0
Se1          1        0/0            12      0/15           50           0
```

Show the routing table on RouterA with the command **show ip route.** Note that RouterA has a route to network 3.3.3.0, which is on RouterC.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

     1.0.0.0/24  is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/24  is subnetted, 1 subnets
```

```
D       2.2.2.0 [90/2297856] via 10.1.2.2, 00:18:07, Serial0
     3.0.0.0/24 is subnetted, 1 subnets
D       3.3.3.0 [90/2297856] via 10.1.3.2, 00:18:08, Serial1
     10.0.0.0/24 is subnetted, 2 subnets
C       10.1.3.0 is directly connected, Serial1
C       10.1.2.0 is directly connected, Serial0
```

Now let's add the **passive−interface** command for serial interface S1 on RouterA and see what happens.

```
RouterA#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config−if)#router eigrp 64
RouterA(config−router)#passive−interface s1
```

Display the information about EIGRP with the command **show ip protocols**. Notice that RouterA's interface S1 is now passive.

```
RouterA#show ip protocols
Routing Protocol is "eigrp 64"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 64
  Automatic network summarization is not in effect
  Routing for Networks:
    10.0.0.0
    1.0.0.0
  Passive Interface(s):
    Serial1
  Routing Information Sources:
    Gateway         Distance      Last Update
    (this router)          5      01:20:53
    10.1.3.2              90      00:38:50
    10.1.2.2              90      00:18:36
  Distance: internal 90 external 170
```

Display the EIGRP neighbor table on RouterA with the command **show ip eigrp neighbors**. Notice that RouterC (10.1.3.2) is no longer a neighbor.

```
RouterA#show ip eigrp neighbors
IP−EIGRP neighbors for process 64
H     Address     Interface    Hold    Uptime     SRTT    RTO    Q     Seq
                               (sec)              (ms)           Cnt   Num
0     10.1.2.2    Se0          14      01:01:10   64      384    0     19
```

When an EIGRP−enabled interface is made passive, no EIGRP packets are sent out to that interface, and since EIGRP uses hello packets to form adjacencies with neighbors, no adjacencies will be formed.

Show the routing table on RouterA with the command **show ip route**. Notice that the route to 3.3.3.0 has been removed. Unlike RIP or IGRP (where updates are received but not sent), when the **passive−interface** command is used with EIGRP, routing updates are neither received nor sent because no neighbor relationship is formed.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route
```

```
Gateway of last resort is not set

     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/24 is subnetted, 1 subnets
D       2.2.2.0 [90/2297856] via 10.1.2.2, 00:29:40, Serial0
     10.0.0.0/24 is subnetted, 2 subnets
C       10.1.3.0 is directly connected, Serial1
C       10.1.2.0 is directly connected, Serial0
```

Lab #45: EIGRP Unequal−Cost Load Balancing

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one Ethernet port and one serial port
- Two Cisco routers with one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Ethernet cables
- One Ethernet hub
- Two Cisco DTE/DCE crossover cables
- One Cisco rolled cable for accessing the console port of the router

Overview

EIGRP can be configured to load balance on up to four unequal cost paths to a given destination. This feature is known as unequal−cost load balancing and is set using the **variance** command. By default, the router will load balance across up to four equal cost paths. The **variance** command lets you set how much worse an alternate path can be (in terms of metrics) and still be used to load balance across.

For example, if RouterA has two routes to network 3.3.3.3, one with a cost of 4 and one with a cost of 8, by default, the route will only use the path with a cost of 4 when sending packets to 1.1.1.1. However, if a variance of 2 is set, the router will load balance across both paths. This is because the route with the cost of 8 is within the variance, which in this case can be up to two times as bad as the preferred route.

Configuration Overview

This configuration will demonstrate the use of the **variance** command, which allows EIGRP−enabled routers to load balance across unequal−cost paths. The **variance** command will be set on RouterA so that both paths to network 3.3.3.3 are used.

RouterA, RouterB, and RouterC are connected serially via a crossover cable, and RouterA and RouterB are also connected via an Ethernet hub. RouterB will act as the DCE supplying clock to RouterA and RouterC. The IP addresses are assigned as per Figure 9−6. All routers will be configured for EIGRP; RouterA will be configured to load balance traffic that is destined for 3.3.3.3 over two unequal−cost paths.

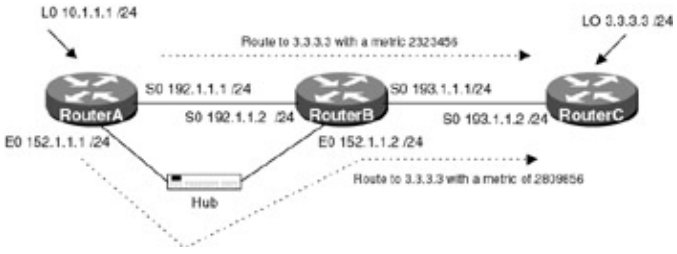


Figure 9−6: EIGRP unequal−cost load balancing

Router Configurations

The configurations for the three routers in this example are as follows (key EIGRP configurations are highlighted in bold).

RouterA

```
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 10.1.1.1 255.255.255.0
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 keepalive
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
!router eigrp 64 ← Enables the EIGRP routing process on the router
variance 2
 network 10.0.0.0 ← Specifies what interfaces will receive and send EGRP
                    routing updates. It also specifies what networks will be
                    advertised
network 152.1.0.0
 network 192.1.1.0
!
no ip classless
!
!line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
service udp−small−servers
service tcp−small−servers
!
hostname RouterB
!
interface Ethernet0
 ip address 152.1.1.2 255.255.255.0
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair−queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router eigrp 64 ← Enables the IGRP routing process on the router
```

```
network 192.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised
 network 193.1.1.0
network 152.1.0.0

!
!
line con 0
line aux 0
line vty 0 4
 login
```

RouterC

```
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
!
interface Ethernet0
 no ip address
shutdown
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 64 ← Enables the IGRP routing process on the router

network 193.1.1.0 ← Specifies what interfaces will receive and send IGRP
                    routing updates. It also specifies what networks will be
                    advertised

network 3.0.0.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display the routing table on RouterA with the command **show ip route**. Notice that there are two routes to network 3.3.3.0: one via the Ethernet interface and one via the serial interface. The cost to reach the network over each path is different; however, since the variance is set to 2, as long as the cost of the second path is not greater than two times the preferred path, the route will be used.

Let's take a closer look at this. The best route to network 3.0.0.0 is via the Ethernet interface with a cost of 2,323,456. Since the variance is set to 2, as long as the cost of any other route to network 3.0.0.0 is below 4,646,912 (2,323,456*2), it will be used. Since the cost of the route via the serial interface is 2,809,856, which

is lower than 4,646,912, it is used.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

D    3.0.0.0/8 [90/2323456] via 152.1.1.2, 00:00:09, Ethernet0
               [90/2809856] via 192.1.1.2, 00:00:10, Serial0
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
D       10.0.0.0/8 is a summary, 00:00:10, Null0
C       10.1.1.0/24 is directly connected, Loopback0
     152.1.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       152.1.1.0/24 is directly connected, Ethernet0
D       152.1.0.0/16 is a summary, 00:00:10, Null0
C    192.1.1.0/24 is directly connected, Serial0
D    193.1.1.0/24 [90/2195456] via 152.1.1.2, 00:00:10, Ethernet0
                  [90/2681856] via 192.1.1.2, 00:00:10, Serial0
```

From RouterA, display the route to host 3.3.3.3 with the command **show ip route 3.3.3.3**. Notice that both routes are shown; however, there is an asterisk next to the first route. The asterisk indicates that the next packet leaving RouterA destined for host 3.3.3.3 will use this route.

```
RouterA#show ip route 3.3.3.3
Routing entry for 3.0.0.0/8
  Known via "eigrp 64", distance 90, metric 2323456, type internal
  Redistributing via eigrp 64
  Last update from 192.1.1.2 on Serial0, 00:09:05 ago
  Routing Descriptor Blocks:
  * 152.1.1.2, from 152.1.1.2, 00:09:05 ago, via Ethernet0
      Route metric is 2323456, traffic share count is 1
      Total delay is 26000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
    192.1.1.2, from 192.1.1.2, 00:09:06 ago, via Serial0
      Route metric is 2809856, traffic share count is 1
      Total delay is 45000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 252/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

From RouterA ping host 3.3.3.3.

```
RouterA#ping 3.3.3.3

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
!!!!!
```

Now, from RouterA, display the route to host 3.3.3.3 with the command **show ip route 3.3.3.3.** Notice that the asterisk is now by the second route. This is because the router is load balancing the traffic destined for network 3.0.0.0 over both links.

```
RouterA#show ip route 3.3.3.3
Routing entry for 3.0.0.0/8
  Known via "eigrp 64", distance 90, metric 2323456, type internal
  Redistributing via eigrp 64
  Last update from 192.1.1.2 on Serial0, 00:10:09 ago
  Routing Descriptor Blocks:
    152.1.1.2, from 152.1.1.2, 00:10:09 ago, via Ethernet0
      Route metric is 2323456, traffic share count is 1
```

```
      Total delay is 26000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
  * 192.1.1.2, from 192.1.1.2, 00:10:10 ago, via Serial0
      Route metric is 2809856, traffic share count is 1
      Total delay is 45000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 252/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

Remove the variance command on RouterA with the router configuration command no variance.

```
RouterA#configure terminal
RouterA(config)#router eigrp 64
RouterA(config-router)#no variance
```

From RouterA, display the route to host 3.3.3.3 with the command show ip route 3.3.3.3. Notice that only one route is being used. This is the route with the lowest metric, and no load balancing is being performed.

```
RouterA#show ip route 3.3.3.3
Routing entry for 3.0.0.0/8
  Known via "eigrp 64", distance 90, metric 2323456, type internal
  Redistributing via eigrp 64
  Last update from 152.1.1.2 on Ethernet0, 00:00:01 ago
  Routing Descriptor Blocks:
  * 152.1.1.2, from 152.1.1.2, 00:00:01 ago, via Ethernet0
      Route metric is 2323456, traffic share count is 1
      Total delay is 26000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

Lab #46: EIGRP Timer Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables
- One Cisco rolled cable for accessing the console port of the router

Overview

EIGRP uses hello packets to discover neighbors and to learn when their neighbors become unreachable or inoperative. By default, hello packets are sent out every 5 seconds, or every 60 seconds on low−speed (T1 or less) NBMA media. The hello interval is configurable and may require tweaking, depending on the network topology.

Hello packets carry a holdtime, which is the amount of time in which a router receiving the hello will consider the sender of the hello packet reachable. The default holdtime is three times the hello interval, or 15 seconds. For slow−speed NBMA networks, the default holdtime is 180 seconds. If the router does not receive another hello packet within the holdtime, the neighbor is considered down.

Configuration Overview

This configuration will demonstrate changing EIGRP hello and holdtime intervals. The hello interval will be changed from the default of 5 seconds to 10 seconds. The holdtime will be set to three times the hello interval, or 30 seconds.

RouterA will be connected to RouterB via a serial crossover cable, and RouterB will act as the DCE supplying clock. The IP addresses are as in Figure 9−7.

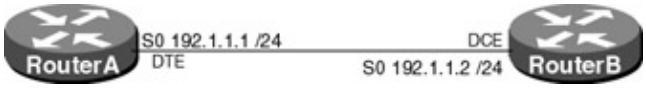


Figure 9−7: EIGRP timers

Router Configurations

The configurations for the two routers in this example are as follows (key EIGRP configurations are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
 ip hello-interval eigrp 64 10 ← EIGRP uses hello packets to discover neighbors
                                 and to learn when their neighbors become
                                 unreachable or inoperative. This command sets
                                 the interval at which hello packets are sent
                                 every 10 seconds
 ip hold-time eigrp 64 30 ← The Holdtime is the amount of time in which a
                            router receiving the hello packet will consider the
                            sender of the hello packet reachable. If the router
                            does not receive another hello pack within the
                            HoldTime, the neighbor is considered down. This
                            command sets the holdtime to 30 seconds, which is
                            three times the hello interval

!
router eigrp 64 ← Enables the EIGRP routing process on the router
network 192.1.1.0 ← Specifies what interfaces will receive and send EIGRP
                    routing updates. It also specifies what networks will be
                    advertised


!
no ip classless
!
line con 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 ip hello-interval eigrp 64 10 ← EIGRP uses hello packets to discover neighbors
                                 and to learn when their neighbors become
                                 unreachable or inoperative. This command sets
                                 the interval at which hello packets are sent
                                 to every 10 seconds

 ip hold-time eigrp 64 30 ← The Holdtime is the amount of time in which a
                            router receiving the hello packet will consider the
                            sender of the hello packet reachable. If the router
                            does not receive another hello pack within the
                            HoldTime, the neighbor is considered down. This
                            command sets the holdtime to 30 seconds, which is
                            three times the hello interval

 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 64 ← Enables the EIGRP routing process on the router
 network 192.1.1.0 ← Specifies what interfaces will receive and send EIGRP
                     routing updates. It also specifies what networks will be
                     advertised

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

On RouterA, enable timestamping on debug output. Timestamping marks each packet generated from the debug command with the time at which it occurred. This is very useful when trying to determine the order of events. By default, debug messages are not timestamped. You can enable timestamping by performing the following task in global configuration mode:

```
RouterA(config)#service timestamps debug
```

On RouterA, enable debugging of EIGRP packets, with the command **debug eigrp packets.** Below is the output from this command. Notice that the hello packets are now being sent every 10 seconds.

```
→ 02:54:54: EIGRP: Sending HELLO on Serial0
      02:54:54:   AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
      02:54:55: EIGRP: Received HELLO on Serial0 nbr 192.1.1.2
      02:54:55:   AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
→ 02:55:04: EIGRP: Sending HELLO on Serial0
      02:55:03:   AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
```

Display the EIGRP neighbor table on RouterB with the command **show ip eigrp neighbors.** Notice that RouterB has one neighbor, RouterA (192.1.1.1).

```
RouterB#show ip eigrp neighbors
IP-EIGRP neighbors for process 64
H     Address      Interface     Hold     Uptime     SRTT     RTO     Q     Seq
                                 (sec)    (ms)                        Cnt   Num
0   192.1.1.1      Se0           27 00:   29:00      0        3000    0     28
```

Now change the hello interval on RouterA to 60 seconds.

```
RouterA#configure terminal
RouterA(config)#interface s0
RouterA(config-if)#ip hello-interval eigrp 64 60
```

Display the neighbor table on RouterB using the **show ip eigrp neighbors** command. Note that RouterB no longer has any neighbors. The reason is that RouterA is only sending hello packets every 60 seconds, but it is still telling RouterB that its holdtimer is 30 seconds. After RouterB receives the first hello packet, it declares the neighbor 192.1.1.1 (RouterA) down because it does not receive another hello packet within 30 seconds. Remember the holdtimer is sent within the hello packet, so if you change the hello interval you must also change the holdtimer. The holdtimer should be three times greater than the hello interval.

```
RouterB#show ip eigrp neighbors
IP-EIGRP neighbors for process 64
```

Now change the HoldTimer on RouterA to 180 seconds.

```
RouterA#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#interface s0
RouterA(config-if)#ip hold-time eigrp 64 180
```

Display the neighbor table on RouterB. Note the neighbor is back and the holdtimer is now counting down from 180 seconds — not 30.

```
RouterB#show ip eigrp neighbors
IP-EIGRP neighbors for process 64
H   Address       Interface     Hold     Uptime     SRTT     RTO     Q     Seq
                                (sec)    (ms)                        Cnt   Num
0   192.1.1.1     Se0           179      00:00:00   0        2000    1     0
```

Lab #47: Configuring EIGRP on an NBMA Network

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers with one serial port
- One Cisco router with three serial ports
- Cisco IOS 10.0 or higher

- A PC running a terminal emulation program
- Three Cisco DTE/DCE crossover cables
- One Cisco rolled cable for accessing the console port of the router

Overview

This lab will address the issue of split horizons when enabling EIGRP on an NMBA network such as frame relay. The network is a typical hub−and−spoke environment, shown in Figure 9−8, where RouterB acts as the hub and has a PVC defined to each spoke. All routers are on the same subnet. The problem is that the rule of split horizon prevents RouterB from advertising routing information out the same physical interface that it was received on. Since the PVCs from RouterA and RouterC terminate on the same physical interface of RouterB, split horizons prevent RouterA from receiving the routing updates from RouterC and vice versa.

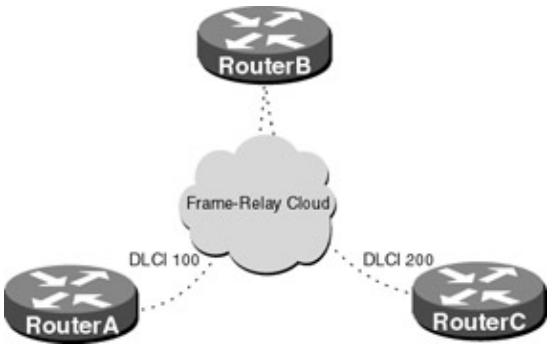


Figure 9−8: Logical connectivity

Configuration Overview

This lab will demonstrate configuring EIGRP over frame relay. RouterA, RouterB, and RouterC will connect serially via a crossover cable to a Cisco router (frame switch), which will act as a frame−relay switch.

The frame switch will act as the DCE supplying clock for all attached routers. Detailed documentation on configuring a Cisco router as a frame−relay switch can be found in Chapter 4. The IP addresses are as per Figure 9−9.

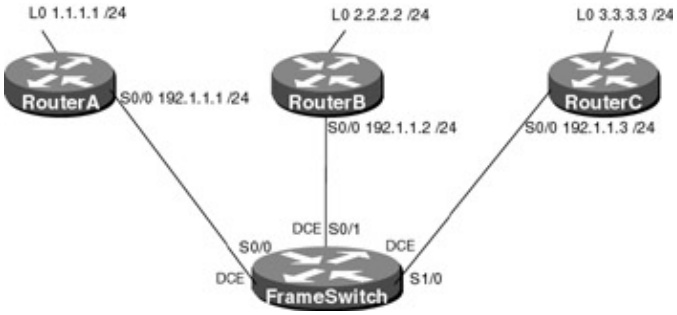


Figure 9−9: Physical connectivity

Router Configurations

The configurations for the routers in this example are as follows (key EIGRP commands are highlighted in bold).

FrameSwitch

```
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
```

```
!
frame-relay switching
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 no ip address
 encapsulation frame-relay IETF
 no fair-queue
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 100 interface Serial0/1 100
!
interface Serial0/1
 no ip address
 encapsulation frame-relay IETF
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 100 interface Serial0/0 100
 frame-relay route 200 interface Serial1/0 200
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Serial1/0
 no ip address
 encapsulation frame-relay IETF
 clockrate 500000
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 200 interface Serial0/1 200
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Loopback0 ← Defines a virtual interface
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation frame-relay IETF
frame-relay map ip 192.1.1.2 100 broadcast
```

```
 frame-relay map ip 192.1.1.3 100 broadcast
 frame-relay lmi-type ansi
!
interface Serial1/0
 no ip address
 shutdown
!
!
router eigrp 64 ← Enables EIGRP routing process
network 192.1.1.0
network 1.0.0.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Loopback0 ← Defines a virtual interface
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation frame-relay IETF
 frame-relay map ip 192.1.1.1 100 broadcast
 frame-relay map ip 192.1.1.3 200 broadcast
 frame-relay lmi-type ansi
!
interface Serial0/1
 no ip address
 shutdown
!
router eigrp 64 ← Enables EIGRP routing process

network 192.1.1.0
network 2.0.0.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 11.2
no service password-encryption
```

```
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
interface Loopback0 ← Defines a virtual interface
 ip address 3.3.3.3  255.255.255.0

!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 ip address 192.1.1.3 255.255.255.0
 encapsulation frame-relay IETF

frame-relay map ip 192.1.1.1 200 broadcast
 frame-relay map ip 192.1.1.2 200 broadcast
 frame-relay lmi-type ansi
!
!
router eigrp 64

network 192.1.1.0
network 3.0.0.0

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display RouterA's routing table with the command **show ip route**. Note that RouterA has not received any information on network 3.0.0.0. This is due to split horizons. RouterB will not advertise a route out the same interface it was received on.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     1.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       1.1.1.0/24 is directly connected, Loopback0
D    2.0.0.0/8 [90/2297856] via 192.1.1.2, 00:11:06, Serial0/0
C    192.1.1.0/24 is directly connected, Serial0/0
```

Disable split horizons on RouterB with the command **no ip split-horizon eigrp 64**.

```
RouterB#configure terminal
RouterB(config)#int
RouterB(config)#interface s0/0
RouterB(config-if)#no ip split-horizon eigrp 64
```

Now display the routing table on RouterA. Note RouterA now has a route to network 3.0.0.0.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     1.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       1.1.1.0/24 is directly connected, Loopback0
D    2.0.0.0/8 [90/2297856] via 192.1.1.2, 00:00:57, Serial0/0
D    3.0.0.0/8 [90/2809856] via 192.1.1.2, 00:00:57, Serial0/0
C    192.1.1.0/24 is directly connected, Serial0/0
```

Troubleshooting EIGRP

The Cisco IOS provides many tools for troubleshooting routing protocols. Below is a list of key commands, along with a sample output from each, that will aid in troubleshooting EIGRP.

{show ip eigrp neighbor} This exec command displays information on all neighbors that are discovered by EIGRP. This command is helpful in determining if the router has any neighbors. The command also shows the amount of time the neighbor has been active and the amount of time remaining on the holdtimer.

```
RouterB#show ip eigrp neighbors
IP−EIGRP neighbors for process 64
H   Address       Interface     Hold     Uptime      SRTT     RTO     Q     Seq
                                (sec)    (ms)                         Cnt   Num
1   192.1.1.3     Se0/0         163      00:08:30    12       200     0     21
0   192.1.1.1     Se0/0         163      00:08:39    8        200     0     21
```

{show ip protocol} This exec command displays the parameters and current state of the active routing protocol process. The output shows the routing protocol used, timer information, inbound and outbound filter information, protocols being redistributed, and the networks that the protocol is routing.

```
RouterB#show ip protocols
Routing Protocol is "eigrp 64"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 64
  Automatic network summarization is in effect
  Automatic address summarization:
    2.0.0.0/8 for Serial0/0
      Summarizing with metric 128256
    192.1.1.0/24 for Loopback0
  Routing for Networks:
    2.0.0.0
    192.1.1.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    (this router)          5      1d01h
    192.1.1.1             90      00:06:30
    192.1.1.3             90      00:06:30
  Distance: internal 90 external 170
```

{show ip eigrp topology} This exec command displays the EIGRP topology table, which shows the state of the Diffusing Update Algorithm (DUAL), which is helpful in identifying possible DUAL problems.

```
RouterB#show ip eigrp topology
IP−EIGRP Topology Table for process 64

Codes: P − Passive, A − Active, U − Update, Q − Query, R − Reply,
       r − Reply status

P 1.0.0.0/8, 1 successors, FD is 2297856
         via 192.1.1.1 (2297856/128256), Serial0/0
P 2.0.0.0/8, 1 successors, FD is 128256
         via Summary (128256/0), Null0
P 2.2.2.0/24, 1 successors, FD is 128256
         via Connected, Loopback0
P 3.0.0.0/8, 1 successors, FD is 2297856
         via 192.1.1.3 (2297856/128256), Serial0/0
P 192.1.1.0/24, 1 successors, FD is 2169856
         via Connected, Serial0/0
```

{show ip eigrp interfaces} This exec command displays information on all EIGRP−enabled interfaces. The command can be used as a quick reference to verify that EIGRP is configured on a particular interface in a particular AS.

```
RouterB#show ip eigrp interfaces
IP−EIGRP interfaces for process 64

            Xmit      Queue          Mean    Pacing Time   Multicast     Pending
Interface   Peers     Un/Reliable    SRTT    Un/Reliable   Flow Timer    Routes
Lo0         0         0/0            0       0/10          0             0
Se0/0       2         0/0            10      0/15          50            0
```

{debug eigrp packet} This exec command displays information on any EIGRP packet that comes into or leaves the router. This command is useful for analyzing the messages traveling between neighbor routers.

```
RouterB#
EIGRP: Sending HELLO on Loopback0
  AS 64, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Received HELLO on Loopback0 nbr 2.2.2.2
  AS 64, Flags 0x0, Seq 0/0 idbQ 0/0
```

{show ip eigrp traffic} This exec command displays the type and number of EIGRP packets sent and received by the router.

```
RouterB#show ip eigrp traffic
IP−EIGRP Traffic Statistics for process 64
  Hellos sent/received: 12139/12128
  Updates sent/received: 4/1
  Queries sent/received: 0/0
  Replies sent/received: 0/0
  Acks sent/received: 0/2
  Input queue high water mark 1, 0 drops
```

Conclusion

EIGRP is an enhanced version of IGRP, which uses the same distance vector algorithm and distance information as IGRP. EIGRP has been enhanced, making it converge faster and operate more efficiently than IGRP.

EIGRP also provides the following benefits:

- EIGRP provides fast convergence through use of the Diffusing Update Algorithm (DUAL).
- EIGRP only sends partial updates for routes that have changed, instead of sending the entire routing table.
- EIGRP supports variable−length subnet masking.
- The EIGRP metric is large enough to support thousands of hops.

Chapter 10: Border Gateway Protocol (BGP)

Overview

Topics Covered in This Chapter

- Detailed technology overview
- BGP terminology
- Synchronization within an AS
- BGP message format
- BGP path attributes
- BGP route summarization
- BGP route aggregation
- BGP route reflectors
- Manipulating BGP path selection
- BGP confederations
- AS path manipulation
- Route filtering based on network number
- BGP communities
- BGP soft configuration
- Regular expressions
- Filtering based on AS path
- BGP backdoor links
- Detailed troubleshooting examples

Introduction

Border Gateway Protocol (BGP) is a path vector interautonomous system routing protocol that is based on distance vector algorithms. "Inter" means routing between entities and "intra" means routing within an entity. An autonomous system (AS) is a collection of routers or end stations that are under the same administrative control and viewed as a single entity. The reason BGP is called a path vector protocol is that the BGP routing information carries a sequence of autonomous system numbers, which indicate the path the route has traversed. This information is used to construct a graph of AS connectivity from which routing loops can be pruned.

In the previous chapters we looked at interior gateway routing protocols (RIP, OSPF, IGRP, EIGRP) that are designed to operate in a single autonomous system under a single administrative control. BGP was introduced to facilitate a loop−free exchange of routing information between autonomous systems while controlling the expansion of routing tables through classless interdomain routing (CIDR) and providing a structured view of the Internet through the use of autonomous systems.

In a sense, the Internet could have been a large OSPF network — if this was the case, all organizations that participated in it would have to adhere to the same administrative policies. By segregating the Internet into multiple autonomous systems we are able to create one large network that consists of smaller, more manageable networks. Within these smaller networks, called ASs, an organization's unique rules and administrative policies can be applied. Each AS is identified by a unique number that is assigned by an Internet registry.

In Figure 10−1 we have two Internet service providers — Xnet and Ynet — each of which consists of multiple networks running multiple Interior Gateway Protocols (IGPs). Each service provider is assigned an AS number by an Internet registry. This number represents its entire network. When companies Xnet and Ynet wish to exchange routing information, they do so using BGP.

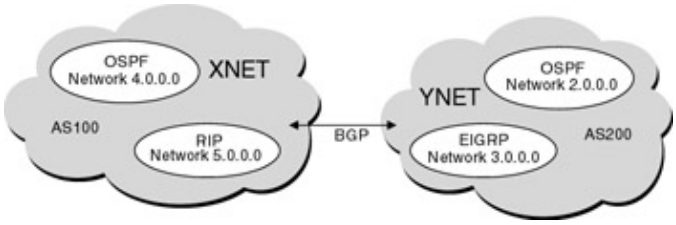


Figure 10−1: Autonomous systems
Company Ynet advertises networks 2.0.0.0 and 3.0.0.0 to company Xnet, and the routes are marked as originating from AS 200. Company Xnet does not need to have a full topological view of company Ynet, nor does it need to understand it internal routing or policies. It simply knows that networks 2.0.0.0 and 3.0.0.0 are in AS 200.

BGP Terminology

Before diving into the intricate details of BGP, it is important to have a clear understanding of key terms and concepts, some of which are used interchangeably.

EBGP vs. IBGP: Although BGP was designed to be used between autonomous systems (EBGP), it is often used within an AS (IBGP) to carry information between border routers running EBGP to other ASs. This allows all BGP path attributes to be maintained across the AS, as shown in Figure 10−2.

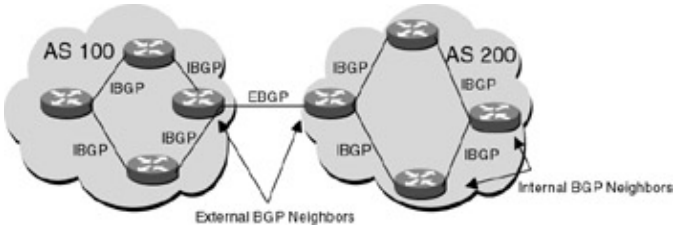


Figure 10−2: EBGP vs. IBGP
By definition, an external BGP neighbor is a router whose administrative and policy control is outside of your autonomous system. An internal BGP neighbor is a router that is under the same administrative control.

CIDR: Classless interdomain routing (CIDR) was developed to address the explosive growth of IP addresses present in IP routing tables on Internet routers and the exhaustion of IP address space. CIDR is an address allocation scheme that eliminates the concept of network class within BGP. In CIDR, an IP network is represented by a prefix, which is the IP address and a number that indicates the leftmost contiguous significant bits in the address. For example, in Figure 10−3 there are a number of class C networks that are present on service provider A's network. Without CIDR, the service provider must advertise each network individually. With CIDR, service provider A can advertise all of these networks with one classless advertisement (200.10.0.0/16).

Figure 10−3: Network advertisement without CIDR
Supernet: A supernet is a network advertisement whose prefix boundary contains fewer bits than the network's natural mask. For example, in Figure 10−4 the natural network mask for the class C network 200.10.1.0 is 255.255.255.0. However when we represent it as 200.10.0.0/16 the mask is 16, which is less than 24, hence a supernet.

Figure 10−4: Network advertisement with CIDR

IP prefix: An IP prefix is an IP network address along with an indication of the number of bits that make up the network number. The IP prefix is what is present in the routing table — 10.0.0.0/8 is an IP prefix.

NLRI: Network layer reachability information is how BGP supports classless routing (CIDR). The NLRI is part of the BGP update message and is used to list a set of destinations that are reachable. The NLRI field in the BGP update message contains 2−tuples <length, prefix>, the length is the number of bits in the mask and the prefix is the IP address. The two combined represent the network number. For example, the network 10.0.0.0/8 would be advertised in the NLRI field of a BGP update message as <8,10.0.0.0>.

Autonomous system: An autonomous system (AS) is a group of routers or hosts that are under the same administrative control and policies. Autonomous system numbers are assigned by an Internet registry.

Synchronization: Before BGP can announce a route, the route must be present in the IP routing table. In other words, BGP and IGP must be in sync before the networks can be advertised. Cisco allows BGP to override the synchronization requirement with the command **no synchronization**. This allows BGP to announce routes that are known via BGP but are not in the routing table. The reason that this rule exists is it is important that the AS be consistent with the routes it advertises.

For example in Figure 10−5, RouterA and RouterB are the only routers running BGP. If synchronization is disabled on RouterB, it will advertise network 1.0.0.0/8 to AS 200. When RouterD wishes to send traffic to network 1.0.0.0, it sends the packet to RouterB, which does a recursive lookup in its IP routing table and forwards the packet to RouterC. Since RouterC is not running BGP, it has no visibility to network 1.0.0.0 and therefore drops the packet. This is why BGP requires synchronization between BGP and IGP. Care must be taken when disabling synchronization. If an AS is a transit AS, all routes should be running fully meshed IBGP before synchronization is disabled.

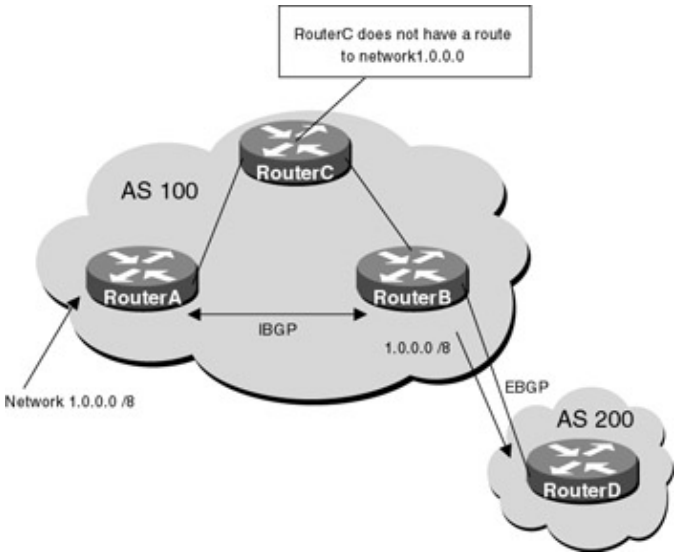


Figure 10−5: Synchronization within an AS

Technology Overview

Border Gateway Protocol (BGP) is an interautonomous system routing protocol whose primary function is to exchange network reachability information with other BGP speakers. (A BGP speaker is any device that is configured for BGP.) BGP uses TCP as its transport protocol (port 179), which provides reliable data transfer.

Two BGP routers form a transport protocol connection between one another. The two routers are called "neighbors" or "peers." Once the transport connection is formed, the peer routers exchange messages to open and confirm connection parameters. It is in this stage that the routers exchange information on BGP version number, AS number, hold time, BGP identifier, and other optional parameters. If the peers disagree on any of the parameters, a notification error is sent and the peer connection does not get established.

If the parameters are agreed upon by the peer routers, the entire BGP routing table is exchanged using

UPDATE messages. The UPDATE messages contain a list of destinations reachable via each system (network layer reachability information) along with path attributes for each route. The path attributes contain information such as ORIGIN of the route and degree of preference. Path attributes will be covered in great detail later in this chapter.

The BGP table is valid for each peer for the duration of the BGP connection. If any routing information changes, the neighbor router uses incremental updates to convey this information. BGP does not require that routing information be refreshed. If no routing changes occur, BGP peers only exchange keep−alive packets. The KeepAlive messages are sent periodically to ensure that the connection is kept active.

BGP Neighbor Negotiation

Before BGP speakers can exchange network layer reachablity information (networks being advertised), a BGP session must be established. Figure 10−6 illustrates the states that BGP neighbor negotiation goes through before a connection becomes fully established.

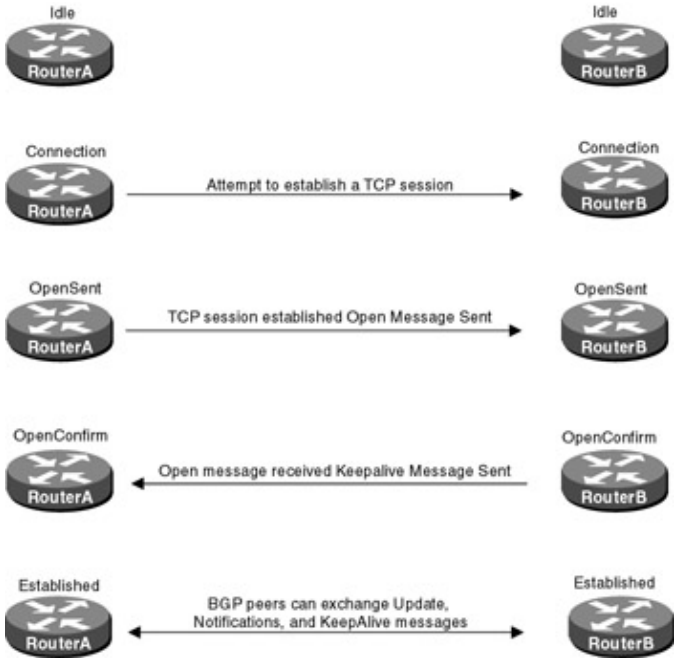


Figure 10−6: BGP neighbor negotiation

Idle: Initially, BGP is in an idle state until an operator initiates a start event, which is usually caused by establishing or restarting a BGP session.

Connect: In this state, BGP is waiting for the transport protocol connection to be completed. If the transport protocol connection succeeds an OPEN message is sent to the peer router and the BGP state changes to OpenSent. If the connection fails, the local system changes to active state and continues to listen for connections.

Active state: In this state, BGP is trying to acquire a peer by initiating a transport protocol connection. If the connection is successful, an OPEN message is sent to the peer router. If the connection retry timer expires, the BGP state changes to connect and continues to listen for connections that may be initiated by the remote BGP peer.

OpenSent state: In this state, BGP is waiting for an OPEN message from its peer. When an OPEN message is received, all fields are checked for correctness. If an error is detected, the local system sends a NOTIFICATION message and changes its state to idle. If there are no errors, BGP starts sending KeepAlive messages to its peer.

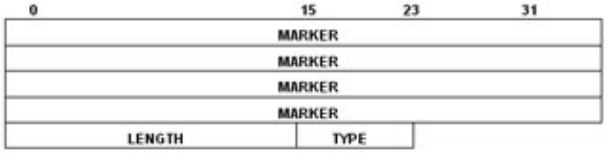
OpenConfirm: In this state, BGP waits for a KeepAlive or a notification message. If the local system receives a KeepAlive message, it changes its state to established. If the hold timer expires before a KeepAlive

message is received, the local system sends a notification message and changes its state to idle.

Established: This is the final stage of the neighbor negotiation. In the established state, BGP peers can exchange Update, Notifications, and KeepAlive messages.

BGP Message Format

All BGP messages use the same fixed size header, which is shown next.



Marker: The Marker field is a 16−byte field that is used to either authenticate incoming BGP messages or to detect loss of synchronization between BGP peers. If the type of message is OPEN, or if the OPEN message carries no authentication information, the marker is all 1s. Otherwise, the Marker field is computed based on the authentication being used.

Length: The Length field is a 2−byte field that indicates the total length of the message. The smallest permitted length is 19 bytes and the largest is 4,096.

Type: The Type field is a 1−byte field that indicates the type of BGP message. There are four BGP message types:

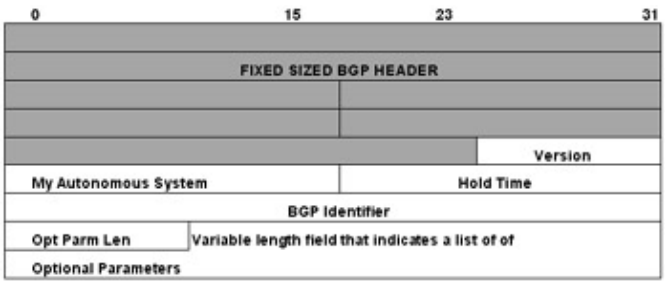
- OPEN
- UPDATE
- KeepAlive
- Notification

Open Message Format

BGP cannot exchange routing information until neighbor negotiation is complete. After the transport connection is established, the first message sent is an OPEN message. This message contains information on BGP version number, AS number, hold time, BGP identifier, and other optional parameters. If the peers disagree on any of the parameters, a notification error is sent and the peer connection does not get established.

If the OPEN message is acceptable, meaning the peer router agrees on the parameters, then a KeepAlive message is sent back to confirm the OPEN message.

In addition to the fixed−size BGP header, the OPEN message contains the following fields:



Version: The Version field is a 1−byte field that indicates the version of the BGP protocol. During the neighbor negotiation, peer routers agree on the BGP version number to be used. The highest version that both routers support is usually used.

My Autonomous System: This field is 2 bytes long and indicates the AS number of the sending router.

Hold Time: This field is 2 bytes long and indicates the maximum time in seconds that may elapse between the receipt of a successive KeepAlive and/or UPDATE message. If the hold time is exceeded, the neighbor is considered dead.

The hold time is negotiated between neighbors and is set to the lowest value. The router that receives the OPEN message must calculate the hold time by using the smaller of its configured hold time and the hold time received in the OPEN message.

BGP Identifier: This field is 4 bytes long and indicates the BGP identifier of the sending router. This is the router ID, which is the highest loopback address or the highest IP address on the router at BGP session startup.

Optional Parameter Length: This field is 1 byte long and indicates the total length in bytes of the Optional Parameter field. If no optional parameters are present, the field is set to 0.

Optional Parameters: This is a variable−length field that indicates the list of optional parameters used in BGP neighbor session negotiation.

Update Message Format

The UPDATE message is the primary message used to communicate information between BGP peers. When a BGP speaker advertises or withdraws a route from a peer router, an UPDATE message is used. The UPDATE message always includes the fixed−length BGP header, and can optionally include the following:

Field	Length
Unfeasible Routes Length	2 bytes
Withdrawn Routes	Variable
Total Path Attribute Length	2 bytes
Path Attributes	Variable
Network Layer Reachability Information	Variable

Unfeasible Routes Length: This 2−byte field indicates the total length in octets of the Withdrawn Routes field. If the value is 0, it indicates that no routes are being withdrawn from service.

Withdrawn Routes: This variable−length field contains a list of IP address prefixes of the routes that are being withdrawn from service.

Total Path Attribute Length: This 2−byte field indicates the total length in octets of the Path Attributes field.

Path Attributes: This variable−length field contains a list of BGP attributes associated with the prefixes in the Network Layer Reachability field. The Path Attributes field gives information on the prefixes that are being advertised such as degree of preference or origin of the prefix. This information is used for filtering and in the route decision process. The path attributes fall into four categories:

1. **Well−known mandatory:** An attribute that has to exist in the BGP update and must be recognized by all BGP vendor implementations. ORIGIN, AS_PATH and Next_Hop are three examples of well−know mandatory attributes.

   ORIGIN: The ORIGIN attribute is an example of a well−known mandatory attribute that indicates the origin of the routing update with respect to the AS that originated it. This attribute tells how the original route was put into the BGP table. A route could be learned via an IGP such as OSPF, which was redistributed into BGP, learned through an external routing protocol (EGP), or learned through something other than an IGP or EGP such as a static route.

There are three possible origins (IGP, EGP, INCOMPLETE). The router uses this information in its decision process when choosing between multiple routes. The router prefers the path with the lowest ORIGIN type — IGP is lower than EGP, and EGP is lower than INCOMPLETE.

AS_PATH: AS_PATH is a well−known mandatory attribute that indicates the autonomous systems through which routing information contained in this UPDATE message have passed. In Figure 10−7, prefix 1.0.0.0/8 is advertised to AS 300 and AS 200. When AS 300 passes the prefix to AS 200, it appends its AS number to the AS_PATH. When AS 200 receives the UPDATE from AS300, it knows that 1.0.0.0 originated in AS100 and then passed through AS300.

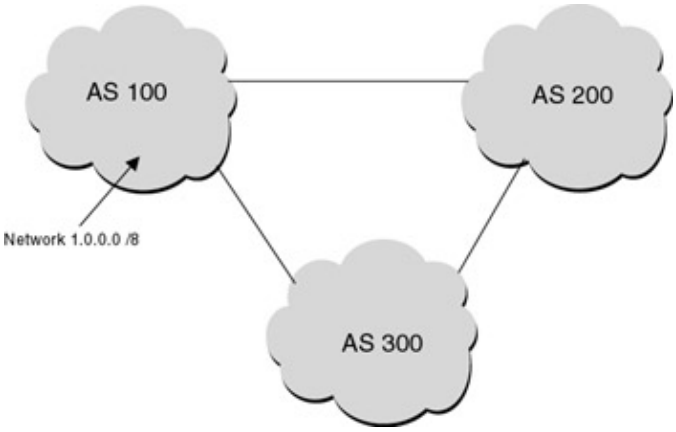


Figure 10−7: AS_PATH attribute

NEXT_HOP: This attribute defines the IP address of the border router that should be used as the next hop to the destinations listed in the UPDATE message. For example, in Figure 10−8, RouterB learns route 1.0.0.0/8 via 192.1.1.1, which is the IP address of its EBGP peer. When RouterB advertises this route to RouterC, it includes the next−hop information unaltered. RouterC will receive an IBGP update for network 1.0.0.0/8 with a next hop of 192.1.1.1.

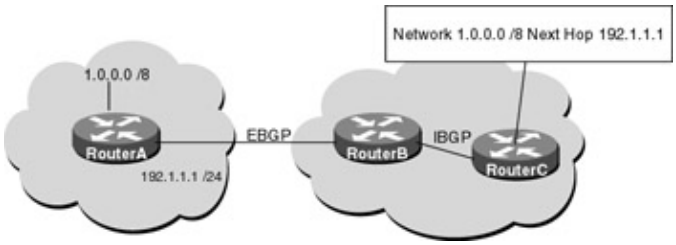


Figure 10−8: BGP NEXT_HOP attribute

2. **Well−known discretionary:** An attribute that must be recognized by all BGP implementations, but may or may not be sent in a BGP update.

LOCAL_PREF: The local preference attribute is a degree of preference given to a route to compare it with other routes to the same destination. The higher local preference is, the more preferred the route. Local preference is not included in UPDATE messages that are sent to BGP neighbors outside of the AS. If the attribute is contained in an update received from a BGP neighbor in a different AS, it is ignored.

In Figure 10−9, we see where the local preference attribute would be used. RouterB and RouterC are both advertising network 1.0.0.0/8 into AS 400; however, since the link between RouterB and RouterD is a high−speed link, we would like traffic destined for network 1.0.0.0/8 to use this route. The LOCAL_PREF attribute will be used to manipulate the flow of traffic within AS 400. RouterD gives routes coming from RouterB a local preference of 150 and RouterE gives routes coming from RouterC a local preference of 100. Since RouterD and RouterE are exchanging routes via IBGP, they both will use the route with the highest local preference. In this case, all traffic destined for network 1.0.0.0/8 will be routed over the high−speed link between RouterB and RouterD.

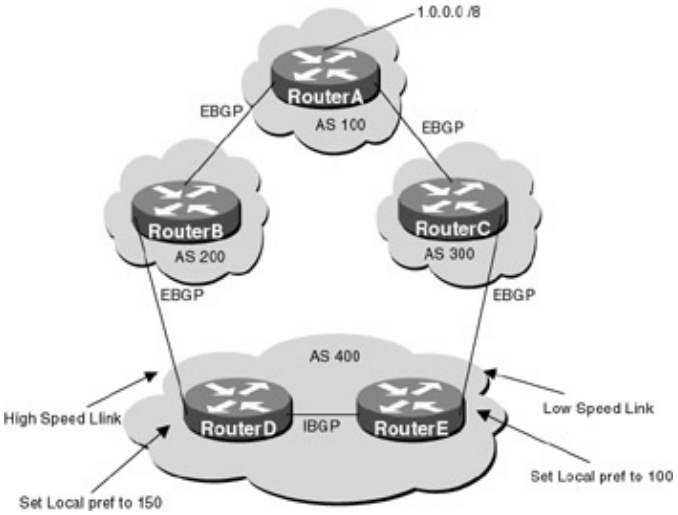


Figure 10−9: BGP LOCAL_PREF attribute

ATOMIC_AGGREGATE: The atomic aggregator attribute indicates that information has been lost. When routes are aggregated, this causes a loss of information because the aggregate is coming from different sources that have different attributes. If a router sends an aggregate that causes loss of information, it is required to attach the ATOMIC_AGGREGATE attribute to the route.

3. ***Optional Transitive:*** An optional transitive attribute is not required to be supported by all BGP implementations. However, if it is not recognized by the BGP process, it will look at the transitive flag. If the flag is set, the BGP implementation should accept the attribute and pass it along to other BGP speakers.

   AGGREGATOR: This attribute identifies the BGP speaker (IP address) and AS number that performed the route aggregation.
4. ***Optional nontransitive:*** An optional nontransitive attribute is not required to be supported by all BGP implementations. However if the attribute is not recognized by the BGP process, it will look at the transitive flag. If the flag is not set, the attribute should be quietly ignored and not passed along to other BGP peers.

   MULTI_EXIT_DISC (MED): MED is used by the BGP speakers to discriminate between multiple exit points to a neighboring AS. A lower MED is preferred over a higher one. The MED attribute is exchanged between ASs, but a MED attribute that comes into an AS does not leave the AS (nontransitive). This differs from the local preference attribute. External routers can influence the path selection of another AS, whereas with local preference you can only influence the route selection within your own AS.

   In Figure 10−10, the MED attribute is used to influence the path that RouterA uses to reach network 1.0.0.0/8. Both RouterB and RouterC are advertising network 1.0.0.0/8 to RouterA; however, since RouterC is closer to this network, we would like all traffic destined to network 1.0.0.0/8 from RouterA to be routed through RouterC. To achieve this, we set the MED on route 1.0.0.0/8 advertised from RouterC to 100, and the MED on route 1.0.0.0/8 advertised from RouterB to 200. Since the MED coming from RouterC is lower, RouterA will prefer this route.

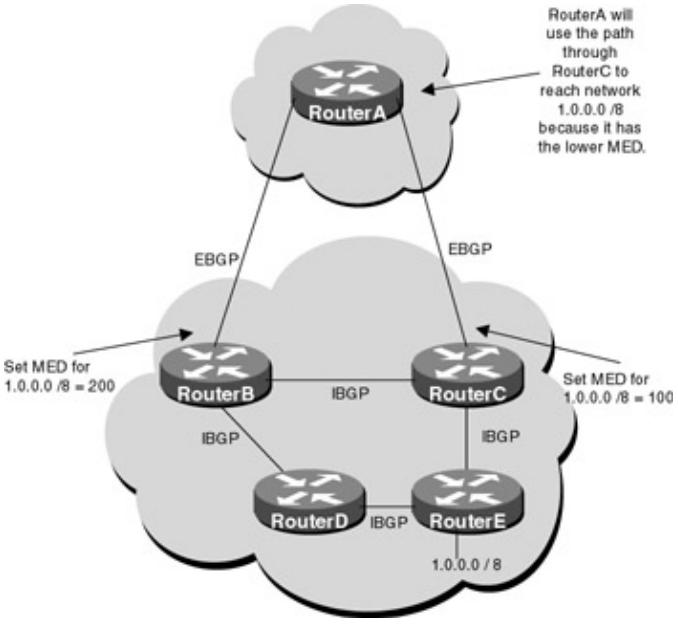


Figure 10−10: BGP Multi−Exit Discriminator (MED)

5. **Network Layer Reachability:** This variable−length field contains a list of IP address prefixes that are reachable via the sender. All path attributes contained in a given UPDATE message apply to the destinations carried in the Network Layer Reachablity information field of the UPDATE message.

KeepAlive Message Format

KeepAlive messages are exchanged periodically between peers to determine whether peers are reachable. KeepAlive messages are exchanged between peers often enough to prevent the hold timer from expiring.

Notification Message Format

A Notification message is sent whenever an error condition is detected. The BGP connection is closed immediately after sending it. In addition to the fixed−size BGP header, the Notification message contains the following fields:

Field	Length
Error code	1 byte
Error Subcode	1 byte
Data	Variable

Error Code: This 1−byte field indicates the type of notification. The following are the possible types:

Error Code	Symbolic Name
1	Message header error
2	OPEN message error
3	UPDATE message error
4	Hold timer expired
5	Finite state machine error
6	Cease

Error Subcode: This 1−byte field provides more specific information about the nature of the reported error. Each error code may have one or more error subcodes associated with it. The following is a list of error subcodes:

Error Code	Error Subcode
1 — Message header error	— Connection not synchronized

	— Bad message length — Bad message type
2 — OPEN message error subcodes	— Unsupported version number — Bad peer AS — Bad BGP identifier — Unsupported optional parameter — Authentication failure — Unacceptable hold time
3 — UPDATE message error subcodes	— Malformed attribute list — Unrecognized well−known attribute — Missing well−known attribute — Attribute flags error — Attribute length error — Invalid ORIGIN attribute — AS routing loop — Invalid NEXT_HOP attribute — Optional attribute error — Invalid network field — Malformed AS_PATH
4 — Hold timer expired	Not applicable
5 — Finite state machine error	Not applicable
6 — Cease	Not applicable

Data: This variable−length field is used to diagnose the reason for the notification. The contents of the Data field depend upon the error code and error subcode.

Commands Discussed in This Chapter

- **aggregate−address** address mask
- **bgp confederation identifier**
- **bgp confederation peers**
- **clear ip bgp**
- **debug ip bgp**
- **ip as−path access−list**
- **neighbor**{ip−address|peer−group−name}**distribute−list**
- **neighbor** ip−address **filter−list**
- **neighbor** {ip−address | peer−group−name} **next−hop−self**

- **neighbor** {ip−address} **remote−as number**
- **neighbor** {ip−address | peer−group−name} **route−map**
- **neighbor** ip−address **route−reflector−client**
- **network** (network−number)
- **no auto−summary**
- **no synchronization**
- **route−map** map−tag [permit | deny] [sequence−number]
- **router bgp** autonomous−system
- **show ip bgp**
- **show ip bgp community**
- **show ip bgp filter−list**
- **show ip bgp neighbor**
- **show ip bgp summary**
- **set community**

Definitions

aggregate−address: This router configuration command creates an aggregate entry in the BGP routing table if there are any more specific BGP routes available that fall in the aggregate range. For example, if a router had routes to networks 192.1.24.0/24, 192.1.25.0/24, 192.1.26.0/24 and 192.1.27.0/24, all of these networks could be advertised in one aggregate route: 192.1.24.0/22.

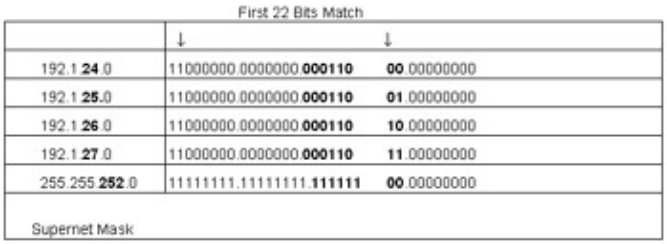


Figure 10−11: Supernetting

Figure 10−11 shows the concept of route aggregation, also referred to as supernetting. Four class C networks, each using a standard 24−bit mask, are shown in the diagram:

- 192.1.24.0/24
- 192.1.25.0/24
- 192.1.26.0/24
- 192.1.27.0/24

Without supernetting, a router advertising these four networks would need to send a separate route update for each of these networks. Supernetting allows a router to advertise more than one network with a single advertisement. In the case of our four networks, a router can advertise the 192.1.24.0 network with a 22−bit mask. Notice that the default 24−bit mask has been shortened to 22 bits. Supernetting works by reducing the number of subnet bits in a routing advertisement. This has the effect of matching several networks with a single routing update.

In Figure 10−11 we see that all four networks have an exact match for their first 22 bits. The remaining two bits of the original 24−bit mask are now part of the supernet. Notice that the .24, .25, .26, and .27 networks use all four combinations of the two remaining bits. Thus, a 22−bit subnet mask can be used to advertise four class C networks with a single advertisement of 192.1.24.0/22.

If this command is used with no additional arguments, the aggregate will be advertised as originating from the aggregating router and will have the atomic aggregate attribute set to show that information might be missing. The more specific routes will also be advertised along with the aggregate. This command has several optional arguments:

as−set: The **as−set** keyword creates an aggregate entry whose path consists of all elements contained in all

the paths that are being summarized.

summary−only: The **summary−only** keyword suppress advertisements of more specific routes to all neighbors.

suppress−map: The **suppress−map** keyword creates the aggregate route but suppresses advertisement of specified routes. The route map can be used to selectively suppress some more specific routes of the aggregate and allow others.

bgp confederation identifier: Used to specify a BGP confederation identifier.

bgp confederation peers: Used to configure the ASs that belong to the confederation. The autonomous systems specified in this command are visible internally to a confederation. Each autonomous system is fully meshed within itself.

clear ip bgp: This exec command is used to reset the BGP connection. When a configuration change is made to an established BGP peer, the BGP session must be reset using this command.

debug ip bgp: This exec command is used to display BGP events as the events occur. These events include state messages, routing updates, and so forth.

ip as−path access−list: This global configuration command allows the user to define an access list filter on both inbound and outbound BGP routes. The filter is an access list based on regular expressions. If the regular expression matches the representation of the autonomous system path of the route as an ASCII string, then the permit or deny condition applies.

neighbor distribute−list: This router configuration command applies a distribute list to either inbound or outbound routes to a particular neighbor. Using distribute lists is one of two ways to filter BGP advertisements. The other way is to use an AS−PATH filter with the **neighbor filter−list** command.

neighbor filter−list: This router configuration command applies the BGP filters to inbound and outbound BGP routes to or from a specified neighbor.

neighbor next−hop−self: This router configuration command forces the router to advertise itself, rather than the external peer, as the next hop to reach the route.

neighbor remote−as: This router configuration command adds a BGP neighbor to the neighbor table. The router will only exchange information with neighboring routers. If the specified neighbor is in the same autonomous system as the AS specified in the global **router bgp** configuration command, the neighbor is internal to the AS (IBGP neighbor); otherwise, the neighbor is external to the AS (EBGP neighbor).

neighbor route−map: This router configuration command applies a route map to incoming or outgoing routes to or from a particular neighbor.

neighbor route−reflector−client: This router configuration command configures the router as a BGP route reflector and configures the specified neighbor as the route−reflector−client.

network: This router configuration command specifies a network address that should be included in the BGP update.

no auto−summary: This router configuration command disables the summarization of subnet routes into network−level routes.

no synchronization: This router configuration command disables the synchronization between BGP and IGP. A BGP speaker will not advertise a route to an external neighbor unless the route is local or exists in the IGP routing table. The **no synchronization** command allows the router to advertise a network route without

first having that route present in the IGP routing table.

route−map: This global configuration command is used with BGP to control and modify routing information and to define the conditions by which routes are redistributed between routing domains.

The **route−map** command uses a *map tag*, which is a name that identifies the route map, and a *sequence number*, which indicates the position a new route map is to have in the list of route maps already configured with the same name.

The following is an example of a route map:

```
↓ Map tag
route−map localpref 10 ← Sequence number (first)
match ip address 1 ← Match criteria—the conditions that must be met
set local−preference 200 ← Set actions—the actions to perform
route−map localpref permit 20 ← Sequence number (second)
set local−preference 100
```

When BGP applies **route−map localpref** to routing updates, it applies the lowest sequence number first (in this case, 10). If the first set of conditions is not met, the second sequence is applied. This goes on until a match condition is found or there are no more conditions to apply. For this particular example, if the route matches the IP address defined in access−list 1, the local preference is set to 200. If the route does not match, sequence 20 is used, which applies a local preference of 100.

router bgp: This global configuration command enables the BGP routing process on the router.

show ip bgp: This exec command is used to display entries in the BGP routing table.

show ip bgp community: This command displays the routes that belong to specified BGP communities

show ip bgp filter−list: This exec command is used to display routes that conform to a specified filter list.

show ip bgp neighbors: This exec command is used to display information about the TCP and BGP connections to neighbors.

show ip bgp summary: This exec command is used to display the status of all BGP connections.

set community: This route map configuration command sets the BGP community attribute.

IOS Requirements

BGP first became available in IOS 10.0, however, all labs were performed using IOS 11.2.

Lab #48: BGP Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial port
- One Cisco router with one Ethernet port
- One Cisco router with one serial port and one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program for connecting to the console port of the router

- One Ethernet hub and two Ethernet cables
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Configuration Overview

This configuration will demonstrate basic routing using BGP. All routers will be configured for BGP and no IGP will be run. RouterA is in AS 100 and will be external BGP neighbors with RouterB in AS 200. RouterC is also in AS 200 and will be internal BGP neighbors with RouterC.

RouterA and RouterB are connected serially via a crossover cable, and RouterC is connected to RouterB via an Ethernet hub. RouterA will act as the DCE, supplying clock to RouterB. The IP addresses are assigned as per Figure 10−12. All routers are configured for BGP and have a loopback address defined. RouterA will advertise network 1.0.0.0.

Figure 10−12: Basic BGP configuration

Router Configurations

The configurations for the three routers in this example are as follows (key BGP configuration statements are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
interface Ethernet0/0
 no ip address
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
!
router bgp 100 ← Configures a BGP process for
autonomous system 100
 neighbor 192.1.1.2 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!interface Loopback0
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0/0
 ip address 193.1.1.1 255.255.255.0
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
!
!
router bgp 200 ← Configures a BGP process for autonomous
system 200
 neighbor 192.1.1.1 remote-as 100 ← Specifies the neighboring router and the
                                    autonomous system it is in
 neighbor 193.1.1.2 remote-as 200 ← IBGP neighbor
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
!
interface Ethernet0/0
 ip address 193.1.1.2 255.255.255.0
!
!
!
router bgp 200 ← Configures a BGP process for autonomous system 200
 neighbor 193.1.1.1 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterA, display the BGP neighbors with the command **show ip bgp neighbors**. The following is the output from this command, which has been truncated. Notice that RouterA has one external BGP neighbor RouterB (192.1.1.2) in AS 200. The router ID of this neighbor is 2.2.2.2, which is the loopback address on RouterB.

To identify itself to its neighbors, the BGP process uses a RouterID, which is an IP address. This address is either the loopback address or the highest IP address of an active interface on the router. The router ID is calculated at boot time and will remain until the BGP process is removed or the router is reloaded.

One of the most important things shown by the command **show ip bgp neighbors** is the line **BGP state** =. This shows the state of the BGP connection. Remember from earlier discussion that there are five possible states. The state should be Established, which means the session between the BGP peers is up and running. If any other state is shown, there is a problem.

```
RouterA#show ip bgp neighbors
BGP neighbor is 192.1.1.2,  remote AS 200, external link
 Index 1, Offset 0, Mask 0x2
  BGP version 4, remote router ID 2.2.2.2
  BGP state = Established, table version = 1, up for 00:10:12
  Last read 00:00:12, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Received 13 messages, 0 notifications, 0 in queue
  Sent 13 messages, 0 notifications, 0 in queue
  Connections established 1; dropped 0
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 192.1.1.1, Local port: 11000
Foreign host: 192.1.1.2, Foreign port: 179
```

Now from RouterA we will advertise network 1.0.0.0 via BGP to RouterB. In order for a router to advertise a network to another BGP speaker, two conditions must be met:

- The BGP process must be aware of the route, either through the use of the network command or by redistribution.
- The network to be advertised must be present in the IP routing table.

For the purposes of this lab, we will be using the network command under the BGP process. This takes care of the first rule, making the BGP process aware of the route. The network command gives you better control of what is being redistributed from the IGP into BGP, allowing the user to individually list the prefixes that need to be advertised via BGP. The maximum number of **network** statements that can be configured on a Cisco router is 200. If you have more than 200 networks to advertise, dynamic redistribution must be used.

The second rule is met because network 1.0.0.0 is a directly connected network; therefore, it is in the IP routing table.

Display the IP routing table on RouterA. Notice that Network 1.0.0.0 is in the IP routing table.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set
     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
```

On RouterA, add the command **network 1.0.0.0** under the BGP process.

```
RouterA#configure terminal
RouterA(config)#router bgp 100
RouterA(config-router)#network 1.0.0.0
```

Display the IP BGP table on RouterB with the command **show ip bgp**. The following is the output from the command. Notice that network 1.0.0.0 was learned via the next hop address of 192.1.1.1, which is the serial interface of RouterA. Also note that the entry is valid, which is indicated by the (*) and the best to use for that network, which is indicated by the (>), before the network number. The route entry originated from IGP and was advertised with a network router configuration command. This is indicated by the (i) after the AS path.

```
RouterB#show ip bgp
BGP table version is 2, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop      Metric      LocPrf      Weight      Path
*> 1.0.0.0        192.1.1.1     0           0                       100 i
```

From RouterB, display the IP routing table with the command **show ip route**. The output from the command is shown next. Note that the router has a route to network 1.0.0.0 via 192.1.1.1, which is RouterA's serial interface.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

B    1.0.0.0/8 [20/0] via 192.1.1.1, 00:12:01
     2.0.0.0/24 is subnetted, 1 subnets
C       2.2.2.0 is directly connected, Loopback0
C    192.1.1.0/24 is directly connected, Serial0/0
C    193.1.1.0/24 is directly connected, Ethernet0/0
```

Display the IP BGP table on RouterC with the command **show ip bgp**. The following is the output from the command. Note that network 1.0.0.0 is in the BGP table with a next hop of 192.1.1.1. The route is valid, indicated by the (*), and was learned via an internal BGP session. This is indicated by the (i) preceding the network number.

```
RouterC#show ip bgp
BGP table version is 1, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

  Network        Next Hop      Metric      LocPrf      Weight      Path
* i1.0.0.0       192.1.1.1     0           100         0           100 i
```

From RouterC, display the IP routing table with the command **show ip route**. The output from the command is shown next. Note that the router has no route to network 1.0.0.0.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
```

```
Gateway of last resort is not set

     3.0.0.0/24 is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
C    193.1.1.0/24 is directly connected, Ethernet0/0
```

The reasons that the route to network 1.0.0.0 is not in the IP routing table are twofold. First, the next hop to reach network 1.0.0.0 is via 192.1.1.1, which is not in RouterC's routing table. Remember from earlier discussion that the next−hop address is the IP address of the EBGP neighbor from which the route was learned. When routes are injected into the AS via EBGP, the next hop learned from EBGP is carried unaltered into IBGP.

The second reason is that, by default, BGP and IGP must be in sync since the 1.0.0.0 network is not being learned via an IGP, meaning that the BGP learned routes on RouterB are not being redistributed into an IGP and the two are not in sync.

Let's fix the next−hop address problem first. Force RouterB to advertise itself as the next hop for all BGP updates it sends to RouterC. To do this, add the command **neighbor 193.1.1.2 next−hop−self**, under the BGP routing process on RouterB.

```
RouterB#configure terminal
RouterB(config)#router bgp 200
RouterB(config−router)#neighbor 193.1.1.2 next−hop−self
```

Note There would be no need for this command if RouterB and RouterC were advertising connected networks to each other via an IGP. This command is useful in nonmeshed networks (such as frame relay or X.25) where BGP neighbors may not have direct access to all other neighbors on the same IP subnet.

When a configuration change is made to an established BGP peer, the session must be reset. From RouterC, reset the BGP connection with the command **clear ip bgp 193.1.1.1**. This command will reset only the specific neighbor. If you wished to reset all BGP neighbors, you could use the command **clear ip bgp ***.

Display the IP BGP table on RouterC with the command **show ip bgp**. The following is the output from the command. Notice the next hop IP address is now 193.1.1.1, which is RouterB's Ethernet interface.

```
RouterC#show ip bgp
BGP table version is 2, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

  Network        Next Hop      Metric      LocPrf      Weight      Path
*>i1.0.0.0       193.1.1.1     0           100         0           100 i
```

On RouterC, disable synchronization with the command **no synchronization** under the BGP routing process.

```
RouterC#conf terminal
RouterC(config)#router bgp 200
RouterC(config−router)#no synchronization
```

From RouterC, display the IP routing table with the command **show ip route**. The output from the command is shown next. Note that the router now has a route to network 1.0.0.0 in its IP routing table.

```
RouterC#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set
```

```
B    1.0.0.0/8 [200/0] via 193.1.1.1, 00:00:03
     3.0.0.0/24 is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
C    193.1.1.0/24 is directly connected, Ethernet0/0
```

BGP Summarization

In this section, let's take a look at how BGP can summarize multiple contiguous networks into one advertisement. The router advertises what is called a supernet, which is a network advertisement whose prefix contains fewer bits than the network's natural mask. For example, if RouterA contained network 192.1.24.0/24, 192.1.25.0/24, 192.1.26.0/24, and 192.1.27.0/24, these routes could all be advertised using the supernet 192.1.24.0/22 .

Figure 10−13 shows the concept of route aggregation, also referred to as supernetting. Four class C networks, each using a standard 24−bit mask, are shown in the diagram:

- 192.1.24.0/24
- 192.1.25.0/24
- 192.1.26.0/24
- 192.1.27.0/24

Without supernetting, a router advertising these four networks would need to send a separate route update for each of these networks. Supernetting allows a router to advertise more than one network with a single advertisement. In the case of our four networks, a router can advertise the 192.1.24.0 network with a 22−bit mask. Notice that the default 24−bit mask has been shortened to 22 bits. Supernetting works by reducing the number of subnet bits in a routing advertisement. This has the effect of matching several networks with a single routing update.

In Figure 10−13 we see that all four networks have an exact match for their first 22 bits. The remaining two bits of the original 24−bit mask are now part of the supernet. Notice that the .24, .25, .26, and .27 networks use all four combinations of the two remaining bits. Thus, a 22−bit subnet mask can be used to advertise four Class C networks with a single advertisement of 192.1.24.0/22 .

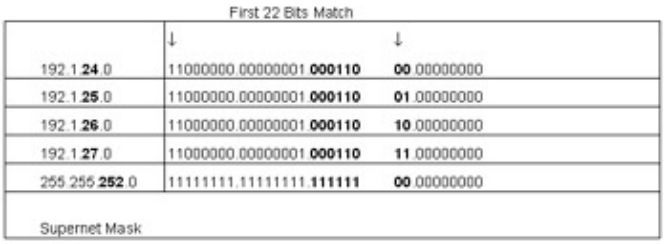


Figure 10−13: Supernetting

On RouterA, add four loopback interfaces with the following IP addresses: 192.1.24.1, 192.1.25.1, 192.1.26.1, and 192.1.27.1.

```
RouterA#configure terminal
RouterA(config)#interface loopback 1
RouterA(config−if)#ip add 192.1.24.1 255.255.255.0
RouterA(config)#interface loopback 2
RouterA(config−if)#ip add 192.1.25.1 255.255.255.0
RouterA(config)#interface Loopback 3
RouterA(config−if)#ip address 192.1.26.1 255.255.255.0
RouterA(config)#interface Loopback 4
RouterA(config−if)#ip address 192.1.27.1 255.255.255.0
```

Normally each network would need to be advertised under the BGP process; however, since we are summarizing all four networks into one announcement, only one **network** statement is need. This **network** statement will contain the mask that we wish to advertise.

On RouterA, add the following command: **network 192.1.24.0 mask 255.255.252.0.** Notice the mask is a 22−bit mask, which is smaller than the natural 24−bit mask, making this a supernet.

```
RouterA#configure terminal
RouterA(config)#router bgp 100
RouterA(config−router)#network 192.1.24.0 mask 255.255.252.0
```

Remember that the route will not install a prefix into the BPG table until a matching IGP prefix exits in the IP routing table. BGP assumes that networks defined with the **network** command are existing networks and verifies this by checking the routing table. If an exact match is not found in the routing table, the network will not be loaded in the BGP table.

Display the BGP table on RouterA with the command **show ip bgp**. Notice that no networks are present.

```
RouterA#show ip bgp
```

In order to get this network in the BGP table, we must define a static route so that it appears in the IP routing table. This is accomplished by defining a static route to a null interface 0.

```
RouterA#configure terminal
RouterA(config)#ip route 192.1.24.0 255.255.252.0 null 0
```

Display the BGP table on RouterA with the command **show ip bgp**. Notice the supernet 192.1.24.0/22 is now present.

```
RouterA#show ip bgp
BGP table version is 28, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

   Network          Next Hop      Metric      LocPrf      Weight      Path
*> 192.1.24.0/22    0.0.0.0       0                                 32768 i
```

BGP Aggregation

Aggregation applies to routes that exist in the BGP routing table. This is different than the network command discussed earlier, which applied to routes that were in the IP routing table. Aggregation can be performed if at least one more specific route of the aggregate appears in the BGP table.

In Figure 10−14, RouterB is aggregating the routes from AS 100 and AS 200 into one continuous block of addresses to be advertised to AS 400 and beyond. When aggregates are generated from more specific routes received from different neighbors, the router that performs the aggregation is the originator of the new route. Since aggregation causes a loss of information, BGP has defined an AS−SET, which is a mathematical set consisting of all elements that are contained in the paths that are being summarized.

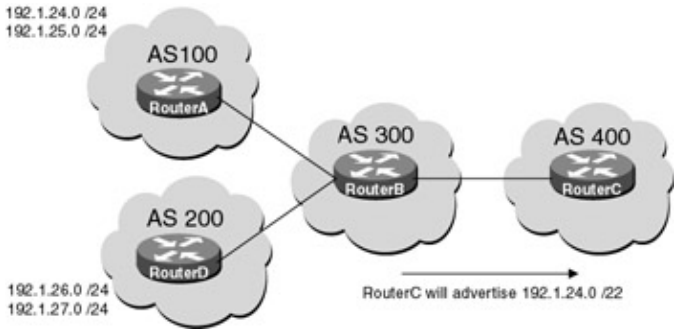


Figure 10−14: BGP route aggregation

For example, the aggregate advertised from RouterB would be:

↓ **AS SET**

```
192.1.24.0/22 300[100 200]i
```

Notice that the route is now originating from AS 300; however, with AS−SET the aggregate contains a set of all attributes (AS numbers) that existed in the individual routes being summarized.

To test this, we need to add an additional router to the testbed and make some changes to the configuration of RouterA. Attach a new Router (RouterD) serially to RouterB with a crossover cable. RouterD will act as the DCE supplying clock to RouterB. All IP addresses are as shown in Figure 10−15.

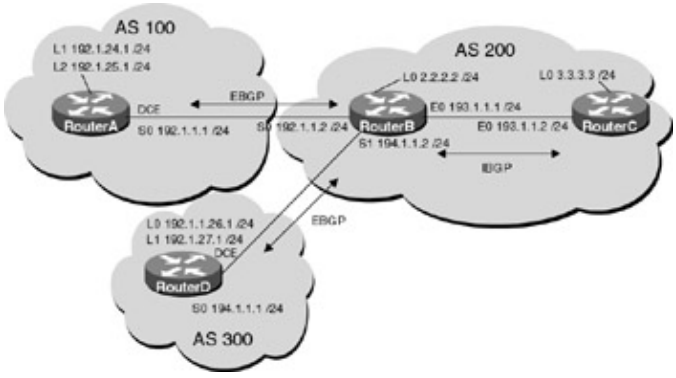


Figure 10−15: BGP route aggregation lab

Router Configurations

The configurations for the three routers are as follows (key BGP configurations are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0

!
interface Loopback1
 ip address 192.1.24.1 255.255.255.0
!
interface Loopback2
 ip address 192.1.25.1 255.255.255.0
!
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
 no fair-queue
!
router bgp 100 ← Configures a BGP process for autonomous system 200
 network 192.1.24.0 ← Specify the list of networks for the BGP routing
                      process to advertise
 network 192.1.25.0
 neighbor 192.1.1.2 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
```

```
end
```

RouterB

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!interface Loopback0
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0/0
 ip address 193.1.1.1 255.255.255.0
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
!
interface Serial1
 ip address 194.1.1.2 255.255.255.0
!
!
router bgp 200 ← Configures a BGP process for autonomous system 200
 neighbor 192.1.1.1 remote-as 100 ← Specifies the neighboring router and the
                                    autonomous system it is in
neighbor 193.1.1.2 remote-as 200 ← IBGP neighbor
neighbor 194.1.1.1 remote-as 300 ← IBGP neighbor
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.0
!
interface Loopback1
 ip address 192.1.26.1 255.255.255.0
!
interface Loopback2
 ip address 192.1.27.1 255.255.255.0
!
interface Serial0
 clockrate 500000 ← Acts as DCE providing clock
!
!
router bgp 300 ← Configures a BGP process for autonomous system 300

 network 192.1.26.0 ← Specify the list of networks for the BGP routing
```

```
                      process to advertise
 network 192.1.27.0
 neighbor 194.1.1.2 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Display the BGP table on RouterC with the command **show ip bgp**. The following is the output from the command. Notice that RouterC has learned about all four networks via IBGP from RouterB. Also note that the next−hop address is 193.1.1.1. This is because we have the **next−hop self** command under RouterB's BGP process.

```
RouterC#show ip bgp
BGP table version is 5, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop      Metric      LocPrf      Weight       Path
*>i192.1.24.0     193.1.1.1     0           100         0            100 i
*>i192.1.25.0     193.1.1.1     0           100         0            100 i
*>i192.1.26.0     193.1.1.1     0           100         0            300 i
*>i192.1.27.0     193.1.1.1     0           100         0            300 i
```

Now let's aggregate these four networks into one advertisement from RouterB. On RouterB, add the following command under the BGP process:

```
RouterB(config)#router bgp 200
RouterB(config-router)#aggregate-address 192.1.24.0 255.255.252.0 as-set
```

Display the BGP table on RouterC with the command **show ip bgp**. The following is the output from the command. Notice that RouterC has learned the aggregate address from RouterB: the {100,300} is the AS−SET information. This tells the router of all the elements contained in all the paths that are being summarized.

```
RouterC# show ip bgp
BGP table version is 6, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop      Metric      LocPrf      Weight      Path
*>i192.1.24.0     193.1.1.2     0           100         0           100 i
*>i192.1.24.0/22  193.1.1.2                 100         0           {100,300} i
*>i192.1.25.0     193.1.1.2     0           100         0           100 i
*>i192.1.26.0     193.1.1.2     0           100         0           300 i
*>i192.1.27.0     193.1.1.2     0           100         0           300 i
```

In this example, RouterC is receiving the aggregate address as well as all of the specific addresses. This is usually used when a customer is multihomed to a single provider. The provider would use the more specific routes to send traffic to the customer. The more specific routes allow the provider to make better routing decisions. The aggregate would only be sent towards the Internet, reducing the number of routes sent.

To only send the aggregate and suppress the more specific routes, the optional parameter summary−only is added to the aggregate command. This is used when more specific routes do not add any extra benefit, such as making better decisions in forwarding traffic as discussed earlier in the multihomed situation.

On RouterB, add the following command under the BGP process:

```
RouterB(config)#router bgp 200
RouterB(config-router)#aggregate-address 192.1.24.0 255.255.252.0 summary-only as-set
```

From RouterC, reset the BGP connection with the command **clear ip bgp 193.1.1.1**. This command will reset only the specified neighbor. If you wished to reset all of routers BGP neighbors, you could use the command **clear ip bgp ***.

Display the BGP table on RouterC with the command **show ip bgp**. The following is the output from the command. Notice that RouterC is now only receiving the aggregate address.

```
RouterC#show ip bgp
BGP table version is 14, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network              Next Hop     Metric     LocPrf     Weight     Path
*>i192.1.24.0/22     193.1.1.2                100        0         {100,300} i
```

Lab #49: BGP Route Reflectors

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers with one serial port
- One Cisco router with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program for connecting to the console port of the router
- Two Cisco DTE/DCE crossover cables
- One Cisco rolled cable

Route Reflector Overview

To prevent routing loops within an AS, BGP will not advertise to internal BGP peers routes that it has learned via other internal BGP peers. In Figure 10-16, RouterA will advertise all the routes it has learned via EBGP to RouterB. These routes will not be advertised to RouterC. This is because RouterB will not pass IBGP routes between RouterA and RouterC. In order for RouterC to learn about these routes, an IBGP connection between RouterA and RouterC is needed.

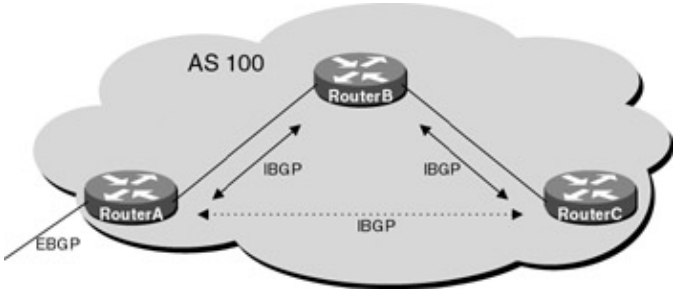


Figure 10-16: IBGP full mesh requirement

The full mesh requirement of IBGP creates the need for neighbor statements to be defined for each IBGP router. In an AS of a 100 routers, this would require 100 neighbor statements to be defined — as you can see, this does not scale well.

To get around this, a concept of a route reflector has been defined. A route reflector acts as a concentration router or focal point for all internal BGP (IBGP) sessions. Routers that peer with the route reflector are called "route reflector clients." The clients peer with the route reflector and exchange routing information. The route reflector then exchanges or "reflects" this information to all clients, thereby eliminating the need for a fully meshed environment.

In Figure 10−17, RouterA receives updates via EBGP and passes them to RouterB. RouterB is configured as a route reflector with two clients: RouterA and RouterC. When RouterB receives the routing updates from RouterA, it reflects the information to RouterC. An IBGP connection is not needed between RouterA and RouterC because RouterB is propagating or reflecting the information to RouterC.

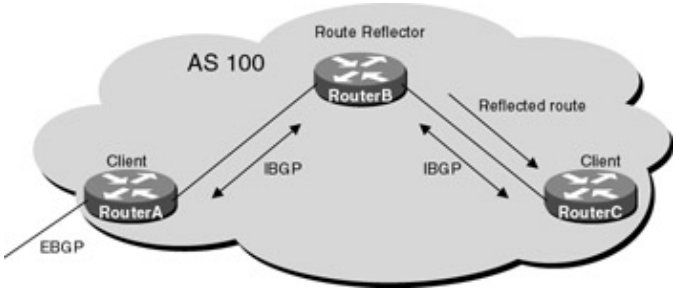


Figure 10−17: BGP route reflector

Configuration Overview

This configuration will use route reflectors to propagate routing information across an AS. All routers will be configured for BGP and OSPF. RouterA is in AS 100 and will be external BGP neighbors with RouterB, which is in AS 200. RouterC and RouterD are also in AS 200. RouterC will act as the route reflector with two clients: RouterB and RouterD.

All routers are connected serially via crossover cables. RouterB will act as the DCE supplying clock to RouterA and RouterC. RouterD will act as the DCE supplying clock for RouterC. The IP addresses are assigned as per Figure 10−18. All routers are configured for BGP and have loopback addresses defined. RouterA will advertise network 1.0.0.0.

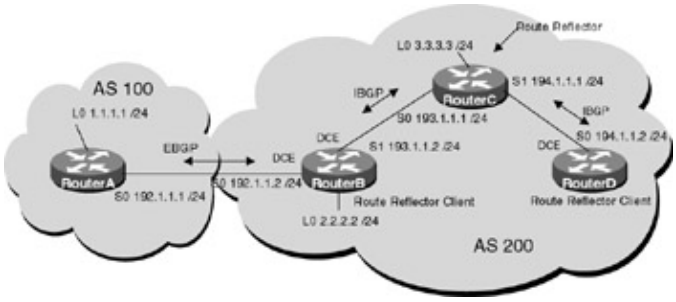


Figure 10−18: BGP route reflector lab

Router Configurations

The configurations for the four routers in this example are as follows (key BGP configurations are highlighted in bold).

RouterA

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
```

```
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
 no fair-queue
!
router bgp 100 ← Configures a BGP process for autonomous system 100
 network 1.0.0.0 ← Specify the list of networks for the BGP routing process
                   to advertise

 neighbor 192.1.1.2 remote-as 200 ← Specifies the neighboring router
and the autonomous system it is in

!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 clockrate 500000  ← Acts as DCE providing clock

!
interface Serial1
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
router ospf 64
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200 ← Configures a BGP process for autonomous system 200
 neighbor 192.1.1.1 remote-as 100
 neighbor 193.1.1.1 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
!
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 193.1.1.1 255.255.255.0
!
interface Serial1
 ip address 194.1.1.1 255.255.255.0
!
router ospf 64
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200 ← Configures a BGP process for autonomous system 200
 neighbor 193.1.1.2 remote-as 200
 neighbor 194.1.1.2 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 194.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
router ospf 64
network 0.0.0.0 255.255.255.255 area 0
!
```

```
router bgp 200 ← Configures a BGP process for autonomous system 100
 neighbor 194.1.1.1 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterB, display the BGP table with the command **show ip bgp**. The following is the output from the command. Note that RouterB has learned of network 1.0.0.0.

```
RouterB#show ip bgp
BGP table version is 49, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

  Network      Next Hop      Metric      LocPrf      Weight      Path
*>1.0.0.0      192.1.1.1     0                       0           100 I
```

From RouterC, display the BGP table with the command **show ip bgp**. The following is the output from the command. Note that RouterC also has learned network about 1.0.0.0.

```
RouterC#show ip bgp
BGP table version is 2, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

  Network      Next Hop      Metric      LocPrf      Weight      Path
*>i1.0.0.0     192.1.1.1     0           100         0           100 i
```

Display the routing table on RouterC with the command **show ip route**. The following is the output from the command. Notice that RouterC has not loaded the route into the IP routing table.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/65] via 193.1.1.2, 00:13:17, Serial0
     3.0.0.0/24 is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
     4.0.0.0/32 is subnetted, 1 subnets
O       4.4.4.4 [110/65] via 194.1.1.2, 00:13:18, Serial1
O    192.1.1.0/24 [110/128] via 193.1.1.2, 00:13:18, Serial0
C    193.1.1.0/24 is directly connected, Serial0
C    194.1.1.0/24 is directly connected, Serial1
```

This is because the IGP (in this case, OSPF) has not learned about the route, because we are not redistributing the BGP learned routes on RouterB into OSPF. Remember, BGP must be synchronized with the IGP. To disable synchronization, add the command **no synchronization** under the BGP process.

```
RouterC#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
RouterC(config)#router bgp 200
RouterC(config-router)#no synchronization
```

In order for the changes to take effect, the BGP neighbors must be reset. To do this, use the command **clear ip bgp *.** This causes the TCP session between neighbors to be reset, restarting the neighbor negotiations from scratch and invalidating the cache.

```
RouterC#clear ip bgp *
```

Now display the routing table on RouterC. Note that network 1.0.0.0 is now present.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

B    1.0.0.0/8 [200/0] via 192.1.1.1, 00:02:45
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/65] via 193.1.1.2, 00:26:28, Serial0
     3.0.0.0/24 is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
     4.0.0.0/32 is subnetted, 1 subnets
O       4.4.4.4 [110/65] via 194.1.1.2, 00:26:29, Serial1
O    192.1.1.0/24 [110/128] via 193.1.1.2, 00:26:29, Serial0
C    193.1.1.0/24 is directly connected, Serial0
C    194.1.1.0/24 is directly connected, Serial1
```

From RouterD, display the BGP table with the command **show ip bgp**. The following is the output from the command. Note that RouterD has not learned of any network via BGP.

```
RouterD#show ip bgp
```

This is because RouterB does not have an IBGP connection with RouterD and RouterC cannot advertise a route via IBGP that it learned from another IBGP neighbor.

There are two ways to fix this: establish an IBGP connection from RouterD to RouterB through the **neighbor** command, or make RouterC a route reflector. To make RouterC a route reflector for RouterB and RouterD, add the following commands to RouterC:

```
RouterC#conf terminal
RouterC(config)#router bgp 200
RouterC(config-router)#neighbor 193.1.1.2 route-reflector-client
RouterC(config-router)#neighbor 194.1.1.2 route-reflector-client
```

In order for the changes to take effect, the BGP neighbors must be reset. To do this, use the command **clear ip bgp *.** This causes the TCP session between neighbors to be reset, restarting the neighbor negotiations from scratch and invalidating the cache.

```
RouterC#clear ip bgp *
```

Display the BGP neighbor information on RouterC with the command **show ip bgp neighbor 194.1.1.2**. The following is the truncated output. Notice that neighbor 194.1.1.2 is now a route reflector client.

```
RouterC#show ip bgp neighbors 194.1.1.2
BGP neighbor is 194.1.1.2,  remote AS 200, internal link
 Index 0, Offset 0, Mask 0x0
```

```
  Route-Reflector Client
  BGP version 4, remote router ID 4.4.4.4
  BGP state = Established, table version = 11, up for 00:11:41
  Last read 00:00:41, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 5 seconds
  Received 78 messages, 0 notifications, 0 in queue
  Sent 79 messages, 0 notifications, 0 in queue
  Connections established 9; dropped 8
```

From RouterD, display the BGP table with the command **show ip bgp**. The following is the output from the command. Notice that RouterD's BGP process has now learned about network 1.0.0.0; however, the route will not get loaded into the IP routing table until synchronization is turned off.

```
RouterD#show ip bgp
BGP table version is 3, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

  Network      Next Hop      Metric      LocPrf      Weight      Path
*>i1.0.0.0     192.1.1.1     0           100         0           100 i
```

Lab #50: Manipulating BGP Path Selection

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, each with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program for connecting to the console port of the router
- Four Cisco DTE/DCE crossover cables
- One Cisco rolled cable

BGP Path Selection Overview

BGP uses a set of parameters (attributes) that describe the characteristics of a route. The attributes are sent in the BGP update packets with each route. The router uses these attributes to select the best route to the destination. In this lab, we will explore manipulating these attributes to control BGP path selections.

It is important to understand the BGP decision process in order to be able to correctly manipulate path selection. The following is the order of the BGP decision process used by the router in path selection:

1. If the next hop is unreachable, do not consider it.
2. Prefer the path that has the largest weight.
3. If the routes have the same weight, use the route with the highest local preference.
4. If the routes have the same local preference, prefer the route that was originated by BGP on this router.
5. If no route was originated, prefer the route with the shortest AS path.
6. If all paths are of the same AS length, prefer the route with lowest origin code (IGP < EGP < INCOMPLETE).
7. If the origin codes are the same, prefer the path with the lowest Multi-Exit Discriminator (MED).
8. If the MEDs are the same, prefer external paths over internal paths.
9. If they are still the same, prefer the path through the closest IGP neighbor.
10. If they are still the same, prefer the path with the lowest BGP Router ID.

Configuration Overview

This lab will demonstrate how an administrator can manipulate route selection through the use of BGP attributes. All routers will be configured for BGP. OSPF will be used as the IGP within AS 200. RouterA is in AS 100 and will be external BGP neighbors with RouterB and RouterC, which are in AS 200. RouterB and RouterC will run IBGP to RouterD, which is also in AS 200.

All routers are connected serially via crossover cables, RouterB will act as the DCE supplying clock to RouterA and RouterD. RouterC will also act as the DCE supplying clock for RouterD and RouterA. The IP addresses are assigned as per Figure 10−19. All routers are configured for BGP and have loopback addresses defined — RouterA's and RouterD's loopback addresses will be advertised via BGP.

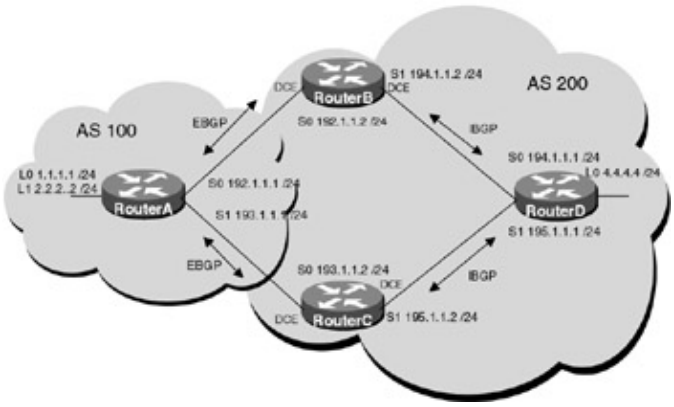


Figure 10−19: BGP path selection

Router Configurations

The configurations for the four routers in this example are as follows (key BGP configurations are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
interface Loopback1
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
interface Serial1
 ip address 193.1.1.1 255.255.255.0
!
router bgp 100 ← Configures a BGP process for autonomous system 100
 network 1.0.0.0
 network 2.0.0.0 ← Specify the list of networks for the
BGP routing process to advertise
 neighbor 192.1.1.2 remote-as 200
 neighbor 193.1.1.2 remote-as 200 ← Specifies the neighboring router
```

```
                                    and the autonomous system it is in

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock


!
interface Serial1
 ip address 194.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router ospf 64
 passive-interface Serial0/0
 network 194.1.1.0 0.0.0.255 area 0
 network 192.1.1.0 0.0.0.255 area 0
!
router bgp 200 ← Configures a BGP process for autonomous system 100
no synchronization ← Disables synchronization between BGP and your IGP
neighbor 192.1.1.1 remote-as 100
 neighbor 194.1.1.1 remote-as 200
 neighbor 195.1.1.2 remote-as 200 ← Specifies the neighboring router and
                                    the autonomous system it is in

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
interface Ethernet0/0
 no ip address
```

```
 shutdown
!
interface Serial0
 ip address 193.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 195.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
router ospf 64
 passive-interface Serial0/0
 network 195.1.1.0 0.0.0.255 area 0
 network 193.1.1.0 0.0.0.255 area 0
!
router bgp 200 ← Configures a BGP process for autonomous system 200
no synchronization ← Disables synchronization between BGP and your IGP
 neighbor 193.1.1.1 remote-as 100
 neighbor 194.1.1.2 remote-as 200
 neighbor 195.1.1.1 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in

!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0
 ip address 194.1.1.1 255.255.255.0
!
interface Serial1
 ip address 195.1.1.1 255.255.255.0
!
router ospf 64
 network 194.1.1.0 0.0.0.255 area 0
 network 195.1.1.0 0.0.0.255 area 0
 network 4.4.4.0 0.0.0.255 area 0
!
router bgp 200 ← Configures a BGP process for autonomous system 200
 no synchronization ← Disables synchronization between BGP and your IGP

 neighbor 194.1.1.2 remote-as 200
 neighbor 195.1.1.2 remote-as 200 ← Specifies the neighboring router and the
                                    autonomous system it is in
```

```
!
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display the BGP table on RouterD with the command **show ip bgp**. The following is the output from the command. RouterD is learning about network 1.0.0.0 and 2.0.0.0 via BGP from both RouterB and RouterC.

```
RouterD# show ip bgp
BGP table version is 11, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network      Next Hop      Metric      LocPrf      Weight      Path
*>i1.0.0.0      192.1.1.1     0           100         0           100 i
* i             193.1.1.1     0           100         0           100 i
*>i2.0.0.0      192.1.1.1     0           100         0           100 i
* i             193.1.1.1     0           100         0           100 i
```

Notice the best path (which is indicated by the >) is through RouterB (192.1.1.0). Remember the ten decision steps that BGP goes through to select the best path — since all other things were equal, the route from the router with the lowest RouterID is used.

This can be verified through the command **show ip bgp neighbors**. Notice that RouterB's RouterID is 194.1.1.2 and RouterC's RouterID is 195.1.1.2.

```
RouterD#show ip bgp neighbors
BGP neighbor is 194.1.1.2,  remote AS 200, internal link ← RouterB
 Index 0, Offset 0, Mask 0x0
  BGP version 4, remote router ID 194.1.1.2
  BGP state = Established, table version = 11, up for 00:11:56
  Last read 00:00:56, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 5 seconds
  Received 91 messages, 0 notifications, 0 in queue
  Sent 83 messages, 0 notifications, 0 in queue
  Connections established 6; dropped 5
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 194.1.1.1, Local port: 179
Foreign host: 194.1.1.2, Foreign port: 11006

BGP neighbor is 195.1.1.2,  remote AS 200, internal link ← RouterC
 Index 0, Offset 0, Mask 0x0
  BGP version 4, remote router ID 195.1.1.2
  BGP state = Established, table version = 11, up for 00:11:40
  Last read 00:00:40, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 5 seconds
  Received 103 messages, 0 notifications, 0 in queue
  Sent 91 messages, 0 notifications, 0 in queue
  Connections established 8; dropped 7
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 195.1.1.1, Local port: 179
Foreign host: 195.1.1.2, Foreign port: 11031
```

Local Preference Attribute

The local preference attribute is a degree of preference given to a BGP route to compare it with other routes to the same destination. This is the second highest attribute used in the BGP decision process (Cisco proprietary weight parameter is first). The local preference attribute only is local to the autonomous system and does not get passed to EBGP neighbors. The higher the local preference, the more preferred the route is.

In this exercise we will configure RouterC to set the local preference for network 1.0.0.0 learned from RouterA to 200. Since the default local preference is 100, all routers in AS 200 will prefer the path through RouterC to reach network 1.0.0.0.

In order to manipulate the local preference, we need to define what routes will be manipulated through the use of an access list, define the policy that will be applied to those routes through a route map, and then assign the route map to a BGP neighbor.

1. Add access−list 1 to RouterC, permitting network 1.0.0.0:

```
RouterC#configure terminal

RouterC(config)#access−list 1 permit 1.0.0.0 0.255.255.255
```

2. Define a route map named localpref that sets the local preference of the route to 200 if it matches access−list 1 and 100 if it does not.

```
RouterC#configure terminal
RouterC(config)#route−map localpref 10 ← If the IP address matches
                                        access−list 1, the local preference is set to 200
RouterC(config−route−map)#match ip address 1
RouterC(config−route−map)#set local−preference 200
RouterC(config−route−map)route−map localpref permit 20 ← If the IP address
                                                        does not match access−list 1, the lo
                                                        is set to 100
RouterC(config)# set local−preference 100
```

3. Apply the route map to inbound traffic from BGP neighbor 193.1.1.1 (RouterA).

```
RouterC#configure terminal
RouterC(config)#router bgp 200
RouterC(config−router)#neighbor 193.1.1.1 route−map localpref in
```

In order for the changes to take effect, the BGP neighbors must be reset. To do this, use the command **clear ip bgp *.** This causes the TCP session between neighbors to be reset, restarting the neighbor negotiations from scratch and invalidating the cache.

```
RouterC#clear ip bgp *
```

Display the BGP table on RouterD with the command **show ip bgp**. The following is the output from the command. Notice the local preference of the route learned from RouterC is now 200 and is the best route (indicated by the > sign).

```
RouterD#show ip bgp
BGP table version is 11, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete
Network      Next Hop     Metric     LocPrf     Weight    Path
*>i1.0.0.0   193.1.1.1    0          200        0         100 i ← Route learned
from Router C
* i2.0       193.1.1.1    0          100        0         100 i
*>i          192.1.1.1    0          100        0         100 i
```

Display the BGP table on RouterB with the command **show ip bgp**. The following is the output from the command. Notice that RouterB is also using the route advertised from RouterC to reach network 1.0.0.0.

```
RouterB#show ip bgp
BGP table version is 20, local router ID is 194.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete
Network      Next Hop     Metric     LocPrf     Weight    Path
*>i1.0.0.0   193.1.1.1    0          200        0         100 i ← Route learned
                                                                 from RouterC
*            192.1.1.1    0                     0         100 i
* i2.0.0.0   193.1.1.1    0          100        0         100 i
*>           192.1.1.1    0                     0         100 I
```

The Multi−Exit Discriminator (MED) Attribute

The Multi−Exit Discriminator (MED) attribute is the external metric of a route. Unlike the local preference attribute, the MED is exchanged between ASs; however, the MED that comes into an AS does not leave. As shown in the last section, local preference was used by the AS to influence its own outbound decision process. The MED can be used to influence the outbound decision of another AS. The lower the MED, the more preferred the route. In Figure 10−20, RouterA sets the MED attribute for network 1.0.0.0 to 50 before advertising it to RouterC and 100 before advertising it to RouterB.

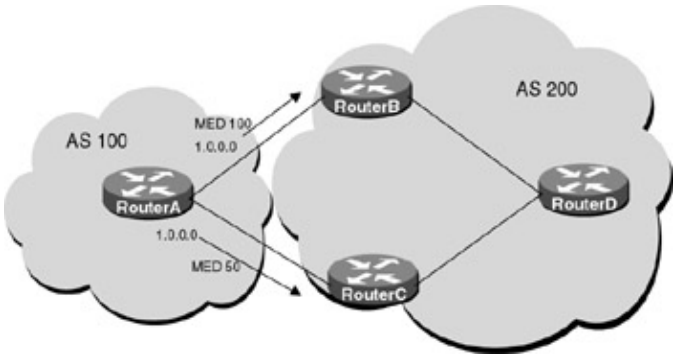


Figure 10−20: BGP Multi−Exit Discriminator (MED) attribute
The routers in AS 200 will prefer the route through RouterC because it has the lowest MED.

In order to manipulate the MED, we need to identify what networks will be manipulated through the use of an access list, define a policy that will be applied to those routes through a route map, and then assign the route map to a BGP neighbor.

Remove the **map** statement on RouterC:

```
RouterC#conf terminal
RouterC(config)#router bgp 200
RouterC(config−router)#no neighbor 193.1.1.1 route−map localpref in
```

1. Add access−list 1 to RouterA, permitting network 1.0.0.0:

   ```
   RouterA#configure terminal

   RouterA(config)#access−list 1 permit 1.0.0.0 0.255.255.255
   ```
2. Define two route maps, one named **set_med_**50 and the other named **set_med_**100. The first route map sets the MED attribute for network 1.0.0.0 to 50, and the latter sets the MED attribute to 100.

   ```
   RouterA#configuration terminal
   RouterA(config)#route−map set_med_50 10 ← The MED attribute for network
                                              1.0.0.0 is set to 50

   RouterA(config−route−map)#match ip address 1
   RouterA(config−route−map)#set metric 50
   RouterA(config−route−map)#exit
   RouterA(config)#route−map set_med_50 20 ← The MED attribute for all other
                                              networks is not set
   RouterA(config−route−map)#set metric
   ```

```
RouterA(config-route-map)#exit
RouterA(config)#route-map set_med_100 10 ← The MED attribute for network
                                           1.0.0.0 is set to 100
RouterA(config-route-map)#match ip address 1
RouterA(config-route-map)#set metric 100
RouterA(config-route-map)#exit
RouterA(config)#route-map set_med_100 20 ← The MED attribute for all other
                                           networks is not set

RouterA(config-route-map)#set metric
```

Apply route map **set_med_50** on outbound routing updates to RouterC (193.1.1.2) and route map **set_med_100** on outbound routing updates to RouterB (192.1.1.2).

```
RouterA#configure terminal
RouterA(config)#router bgp 100
RouterA(config-router)#neighbor 193.1.1.2 route-map set_med_50 out
RouterA(config-router)#neighbor 192.1.1.2 route-map set_med_100 out
```

In order for the changes to take effect, the BGP neighbors must be reset. To do this, use the command **clear ip bgp***. This causes the TCP session between neighbors to be reset, restarting the neighbor negotiations from scratch and invalidating the cache.

```
RouterC#clear ip bgp *
```

Display the BGP table on RouterB with the command **show ip bgp**. The following is the output from the command. Notice that the route to network 1.0.0.0 learned via 193.1.1.1 has a local preference of 50 and is the preferred route.

```
RouterB#show ip bgp
BGP table version is 30, local router ID is 194.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
Network       Next Hop     Metric     LocPrf     Weight     Path
*>i1.0.0.0    193.1.1.1    50         100        0          100 i ← Preferred
                                                                    route
*             192.1.1.1    100                   0          100 i
* i2.0.0.0    193.1.1.1    0          100        0          100 i
*>            192.1.1.1    0                     0          100 I
```

From RouterA, display the route maps that are being used with the command **show route-maps.** This command tells what access list is used by the match clause, and what set clause is applied and how many times it has been used. This command is very useful in troubleshooting possible route-map problems.

```
RouterA#show route-map
route-map set_med_50, permit, sequence 10
  Match clauses:
    ip address (access-lists): 1
  Set clauses:
    metric 50
  Policy routing matches: 0 packets, 0 bytes
route-map set_med_50, permit, sequence 20
  Match clauses:
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
route-map set_med_100, permit, sequence 10
  Match clauses:
    ip address (access-lists): 1
  Set clauses:
    metric 100
  Policy routing matches: 0 packets, 0 bytes
route-map set_med_100, permit, sequence 20
  Match clauses:
  Set clauses:
```

```
  Policy routing matches: 0 packets, 0 bytes
route-map med, permit, sequence 10
  Match clauses:
    ip address (access-lists): 1
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
```

AS Path Manipulation

BGP always prefers the route with the shortest AS path. In this exercise we will configure RouterA to prepend two extra AS path numbers to network 1.0.0.0 (AS 300 and AS 400) before advertising this network to RouterC and RouterB.

In order to manipulate the AS path information, we need to identify which routes will be manipulated through the use of an access list, define a policy that will be applied to those routes through a route map, and then assign the route map to a BGP neighbor.

1. Add access-list 1 to RouterA, permitting network 1.0.0.0:

   ```
   RouterA#configure terminal

   RouterA(config)#access-list 1 permit 1.0.0.0 0.255.255.255
   ```
2. Define a route map named AS_Path that prepends two additional AS path numbers (AS300 and AS400) to the route if it matches access list 1.

   ```
   RouterA(config)#route-map AS_Path permit 10
   RouterA(config-route-map)#match ip address 1
   RouterA(config-route-map)#set as-path prepend 300 400
   RouterA(config-route-map)#exit
   RouterA(config)#route-map AS_Path 20
   RouterA(config-route-map)#set as-path prepend
   RouterA(config-route-map)#exit
   ```

Apply the route map to outbound routing updates to BGP neighbor 193.1.1.2 (RouterC) and neighbor 192.1.1.2 (RouterB).

```
RouterA#configure terminal
RouterA(config)#router bgp 200
RouterA(config-router)#neighbor 193.1.1.2 route-map AS_Path out
RouterA(config-router)#neighbor 192.1.1.2 route-map AS_Path out
```

In order for the changes to take effect, the BGP neighbors must be reset. To do this, use the command **clear ip bgp ***. This causes the TCP session between neighbors to be reset, restarting the neighbor negotiations from scratch and invalidating the cache.

```
RouterA#clear ip bgp *
```

Display the BGP table on RouterB with the command **show ip bgp**. The following is the output from the command. Notice that the route to network 1.0.0.0 now has an AS path of [100 300 400].

```
RouterB#show ip bgp
BGP table version is 68, local router ID is 194.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network          Next Hop      Metric      LocPrf      Weight      Path
* i1.0.0.0       193.1.1.1     0           100         0           100 300 400 i
*>               192.1.1.1     0                       0           100 300 400 i
* i2.0.0.0       193.1.1.1     0           100         0           100 i
*>               192.1.1.1     0                       0           100 i
```

Route Filtering Based on Network Number

The router can filter routing updates to and from a particular neighbor based on the network number. The filter is made up of an access list that is applied to all BGP updates that are sent to or received from a particular neighbor.

In this exercise we will configure a distribute list on RouterA to prevent prefix 1.0.0.0/8 from being advertised into AS 200.

In order to filter routes based on network address, we need to identify network addresses through the use of an access list and apply that list to a BGP neighbor using a distribute list.

1. Define the access list on RouterA to deny network 1.0.0.0/8.

```
RouterA#configure terminal
RouterA(config)#no access-list 1 ← Removes the old access list
RouterA(config)#access-list 1 deny 1.0.0.0 0.255.255.255
RouterA(config)#access-list 1 permit any
```

2. Apply the distribution list to both BGP neighbors.

```
RouterA(config)#router bgp 100
RouterA(config-router)#neighbor 193.1.1.2 distribute-list 1 out
RouterA(config-router)#neighbor 192.1.1.2 distribute-list 1 out
```

In order for the changes to take effect, the BGP neighbors must be reset. To do this, use the command **clear ip bgp ***. This causes the TCP session between neighbors to be reset, restarting the neighbor negotiations from scratch and invalidating the cache.

```
RouterA#clear ip bgp *
```

Display the routes that are being advertised via BGP to neighbor 193.1.1.2 with the command **show ip bgp neighbors 193.1.1.2 advertised-routes**. The following is the output from the command. Notice that RouterA is now only advertising network 2.0.0.0.

```
RouterA#show ip bgp neighbors 193.1.1.2 advertised-routes
BGP table version is 3, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network       Next Hop       Metric       LocPrf       Weight       Path
*> 2.0.0.0    0.0.0.0        0                                      32768 i
```

Display the BGP table on RouterB with the command **show ip bgp**. The following is the output from the command. Notice that the route to network 1.0.0.0 is no longer in the BGP table.

```
RouterB#show ip bgp
BGP table version is 78, local router ID is 194.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network       Next Hop      Metric       LocPrf       Weight       Path
*> 2.0.0.0    192.1.1.1     0                         0            100 300 400 i
* i           193.1.1.1     0            100          0            100 300 400 i
```

BGP Soft Configuration

BGP soft configuration enables policies to be configured and activated without resetting the BGP and TCP session. This allows the new policy to take effect without significantly affecting the network. Without BGP soft configuration, BGP is required to reset the neighbor TCP connection in order for the new changes to take effect. This is accomplished using the **clear ip bgp** command, which was used throughout this chapter.

There are two types of BGP soft reconfiguration: outbound reconfiguration, which will make the new local outbound policy take effect without resetting the BGP session, and inbound soft reconfiguration, which enables the new inbound policy to take effect.

The problem with inbound reconfiguration is that in order to generate new inbound updates without resetting the BGP session, all inbound updates (whether accepted or rejected) need to be stored by the router. This is memory intensive, and wherever possible it should be avoided.

To avoid the memory overhead needed for inbound soft reconfiguration, the same outcome could be achieved by doing an outbound soft reconfiguration at the other end of the connection.

Outbound soft reconfiguration can be triggered with the following command:

```
clear ip bgp [*|address | peer-group] [soft out]
```

For inbound soft reconfiguration, an additional router command needs to be added before a soft reconfiguration can be issued. This command tells the router to start storing the received updates:

```
neighbor  [address | peer-group] soft-reconfiguration inbound
```

Inbound soft reconfiguration can than be triggered with the following command:

```
clear ip bgp [*|address | peer-group] [soft in]
```

Regular Expressions

In the previous section we looked at identifying routes based on IP address. In this section we will use regular expressions to identify routes based on AS path information. A regular expression is a pattern to match against an input string. When a regular expression is created, it specifies the pattern that a string must match. The following is a list of keyboard characters that have special meaning when used in regular expressions:

Character	Symbol	Meaning
Period	.	Match any character including white space.
Asterisk	*	Match zero or more sequences of the pattern.
Plus sign	+	Match one or more sequences of the pattern.
Question mark	?	Matches zero or one occurrences of the pattern.
Caret	^	Begins with.
Dollar sign	$	Ends with.
Underscore	_	Match the following.
Brackets	[]	Match a single value in range.
Hyphen	-	Separates the endpoints of a range.

Filtering Based on AS Path

For this exercise, let's configure a regular expression in conjunction with a filter list on RouterC that will prevent any network that passes through AS 300 from being sent via BGP to RouterD. Filtering routes based on AS path information can be very useful when all routes from a particular AS need to be filtered. If filtering based on AS path was not used, the administrator would have to list each route one by one or potentially filter on a prefix. AS path filtering provides an efficient alternative to this.

In order to filter routes based on AS path information, we need to identify the AS path based on the defined regular expression and apply this to a BGP neighbor through a filter list:

1. Define the regular expression to deny any route that passed through AS 300.

```
RouterC#configure terminal
RouterC(config)#ip as-path access-list 1 deny _300_ ←
Deny any route that passes through AS 300
RouterC(config)#ip as-path access-list 1 permit .*
```

Use the **show ip bgp regexp** command to see what routes the regular expression matches. The following is the output from the command. Note that network 2.0.0.0 is the only route that matches the regular expression (_300_). This command is very useful in verifying that the regular expression covers the routes that you intend it to.

```
RouterC#show ip bgp regexp _300_
BGP table version is 19, local router ID is 195.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network       Next Hop      Metric      LocPrf      Weight      Path
*> 2.0.0.0    193.1.1.1     0                       0           100 300 400 i
* I           192.1.1.1     0           100         0           100 300 400 i
```

2. Apply the filter list to BGP neighbor 195.1.1.1.

```
RouterC(config)#router bgp 200
RouterC(config-router)#neighbor 195.1.1.1 filter-list 1 out
```

In order for the changes to take effect, the BGP neighbor must be reset. To do this, use the command **clear ip bgp ***. This causes the TCP session between neighbors to be reset, restarting the neighbor negotiations from scratch and invalidating the cache.

```
RouterC#clear ip bgp *
```

Display the AS path access list on RouterC with the command **show ip as-path-access-list**. The following is the output from the command. This command is very useful in quickly determining what strings will be permitted or denied.

```
RouterC#show ip as-path-access-list
AS path access list 1
    deny _300_
    permit .*
```

Display the BGP filter list configured on RouterC with the command **show ip bgp filter-list 1**. The following is the output from the command. This command shows which routes conform to a specified filter list and therefore will be passed.

```
RouterC#show ip bgp filter-list 1
BGP table version is 5, local router ID is 195.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network       Next Hop       Metric       LocPrf       Weight       Path
*> 2.0.0.0   193.1.1.1       0                         0            100 i
* i          192.1.1.1       0            100          0            100 i
```

Display the BGP table on RouterD with the command **show ip bgp**. The following is the output from the command. Notice that the route to network 1.0.0.0 via RouterC is no longer present in the routing table.

```
RouterD#show ip bgp
BGP table version is 5, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network       Next Hop      Metric      LocPrf      Weight      Path
*>i1.0.0.0    192.1.1.1     0           100         0           100 300 400 i
* i2.0.0.0    193.1.1.1     0           100         0           100 i
```

```
*>i          192.1.1.1     0          100         0          100 I
```

The following is a list of the regular expressions and their significance:

Expression	Significance
_300_	Match any routes that pass via AS 300.
_300$	Match any routes that originated in AS 300.
^300_	Only match routes received from AS 300.
^300$	Only match routes that originated from AS 300 and did not pass through any other AS.
.*	All routes.

Lab #51: BGP Confederations

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface
- One Cisco router with one Ethernet interface
- One Cisco router with two serial interfaces
- One Cisco router with one Ethernet and two serial interfaces
- One Cisco router with two Ethernet and one serial interfaces
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for connecting to the console port of the router
- Three DTE/DCE crossover cables
- Two Ethernet cables and two hubs (or two Ethernet crossover cables)
- One Cisco rolled cable

Configuration Overview

BGP confederations are used to break an AS into multiple sub−ASs. This is another way to deal with the full IBGP mesh requirement. Within the sub−ASs, all IBGP rules apply; however, since each sub — AS is a different AS number, EBGP is used between them. This reduces the IBGP meshing in the domain.

Even though EBGP is used between sub−ASs, IBGP information such as next hop, MED, and local preference is preserved within the confederation. To the outside world, the confederation is seen as a single AS — the sub−ASs are hidden.

In Figure 10−21 all of the routers in the AS must have a full IBGP mesh, requiring n−1 or 3 peers per router.

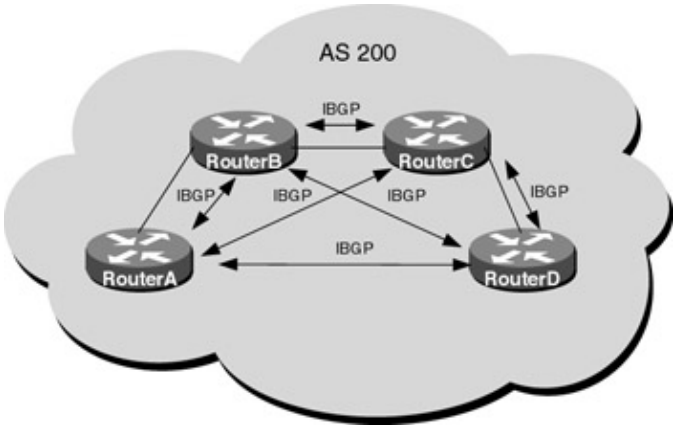


Figure 10−21: IBGP full mesh

In Figure 10−22, two sub−ASs are used: one containing RouterA and RouterB, and the other containing RouterC and RouterD. Now only the routers in each sub−AS need to have IBGP connections, reducing the

IBGP peer requirement to one. The two sub−ASs peer using EBGP.

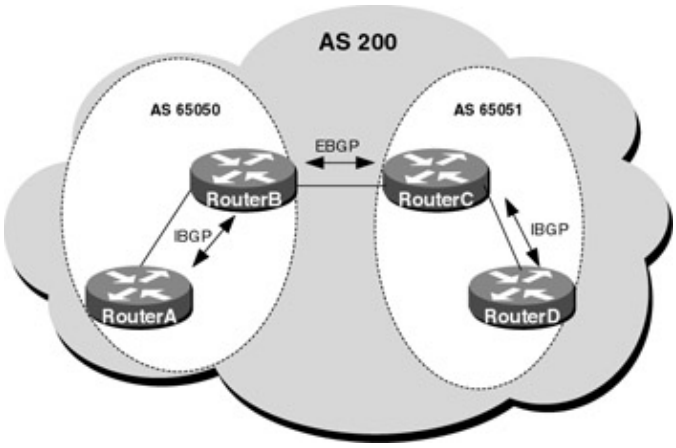


Figure 10−22: BGP confederation

This lab will demonstrate how confederations can be used to reduce the IBGP full mesh requirement. All routers will be configured for BGP. OSPF will be used as the IGP within the sub−ASs. RouterA and RouterB are in sub−AS 65050 and RouterC and RouterD are in sub−AS 65051. RouterB will run EBGP to RouterC as well as EBGP to RouterE in AS 100.

RouterA is connected serially via a crossover cable to RouterB, which will act as the DCE supplying clock to RouterA. RouterB is connected via Ethernet to RouterC, and serially to RouterE via a crossover cable. RouterC is connected to RouterD via Ethernet and to RouterE via a serial crossover cable.

The sub−AS numbers are chosen from the private address pool, which ranges from 64512−65535. OSPF process 64 is run in both sub−ASs. They are independent from one another. This is another major benefit of confederations. The IGP is independent, and therefore a change in one sub−AS will not affect the other sub−AS.

RouterB and RouterC are running EBGP to RouterE in AS 100. The command **bgp confederation identifier 200** is used to present themselves as being part of AS 200. RouterE is running normal EBGP and has no visibility of the sub−ASs in confederation 200.

The IP addresses are assigned as per Figure 10−23. All routers are configured for BGP and have loopback addresses defined.

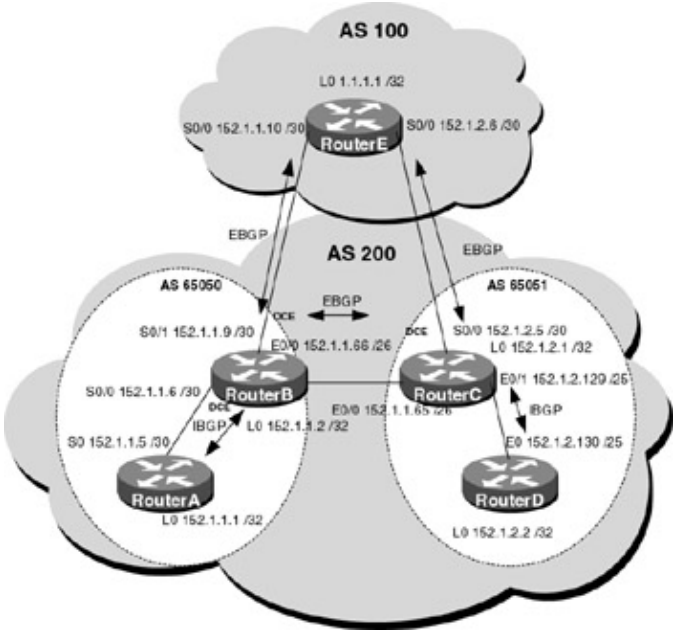


Figure 10−23: Physical connectivity and IP addressing

Router Configurations

The configurations for the five routers in this example are as follows (key confederation commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 152.1.1.1 255.255.255.255
!
!
interface Serial0
 ip address 152.1.1.5 255.255.255.252
!
router ospf 64
network 152.1.1.0 0.0.0.255 area 0
!
router bgp 65050 ← Configures a BGP process
 no synchronization
 network 152.1.1.1 mask 255.255.255.255
neighbor 152.1.1.6 remote-as 65050
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 !
end
```

RouterB

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
!
interface Loopback0
 ip address 152.1.1.2 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0/0
 ip address 152.1.1.66 255.255.255.192
 no ip directed-broadcast
!
interface Serial0/0
 ip address 152.1.1.6 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 clockrate 1000000
!
interface Serial0/1
```

```
 ip address 152.1.1.9 255.255.255.252
 no ip directed-broadcast
 clockrate 1000000
!
router ospf 64
 redistribute connected subnets
 network 152.1.1.6 0.0.0.0 area 0
!
router bgp 65050
 no synchronization
 bgp confederation identifier 200 ← Used to identify itself as being part of
                                    confederation 200
 bgp confederation peers 65051 ← Used to preserve all attributes IBGP
                                 attributes while traversing the EBGP session
 network 152.1.1.2 mask 255.255.255.255
 neighbor 152.1.1.5 remote-as 65050
 neighbor 152.1.1.10 remote-as 100
 neighbor 152.1.1.65 remote-as 65051
 neighbor 152.1.1.65 next-hop-self
!
ip classless
no ip http server
!
!
line con 0
transport input none
line aux 0
line vty 0 4
!
end
```

RouterC

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterC
!
interface Loopback0
 ip address 152.1.2.1 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0/0
 ip address 152.1.1.65 255.255.255.192
 no ip directed-broadcast
!
interface Serial0/0
 ip address 152.1.2.5 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 clockrate 1000000
!
interface Ethernet1/0
 ip address 152.1.2.129 255.255.255.128
 no ip directed-broadcast
!
router ospf 64
 redistribute connected subnets
 network 152.1.2.129 0.0.0.0 area 0
!
router bgp 65051
 no synchronization
 bgp confederation identifier 200 ← Used to identify itself as being part of
```

```
                                    confederation 200
 bgp confederation peers 65050 ← Used to preserve all attributes IBGP
                                 attributes while traversing the EBGP session
 network 152.1.2.1 mask 255.255.255.255
 neighbor 152.1.1.66 remote-as 65050
 neighbor 152.1.1.66 next-hop-self
 neighbor 152.1.2.6 remote-as 100
 neighbor 152.1.2.130 remote-as 65051
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterD

```
Building configuration...

Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterD
!
interface Loopback0
 ip address 152.1.2.2 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0
 ip address 152.1.2.130 255.255.255.128
 no ip directed-broadcast
!
router ospf 64
 network 152.1.2.0 0.0.0.255 area 0
!
router bgp 65051
 no synchronization
 network 152.1.2.2 mask 255.255.255.255
 neighbor 152.1.2.129 remote-as 65051
!
ine con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterE

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterE
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
```

```
 no ip directed-broadcast
!
interface Serial0
 ip address 152.1.1.10 255.255.255.252
!
interface Serial1
 ip address 152.1.2.6 255.255.255.252
!
router bgp 100
 network 1.1.1.1 mask 255.255.255.255
 neighbor 152.1.1.9 remote-as 200
 neighbor 152.1.2.5 remote-as 200
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

Display the BGP table on RouterE with the command **show ip bgp**. The following is the output from the command. Notice that RouterE has two paths for each network, both via AS 200.

All sub−AS information is hidden from RouterE.

```
RouterE#SHO IP BGP
BGP table version is 15, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network            Next Hop      Metric      LocPrf      Weight      Path
*> 1.1.1.1/32      0.0.0.0                   0                       32768 i
*  152.1.1.1/32    152.1.2.5                                         0 200 i
*>                 152.1.1.9                                         0 200 i
*  152.1.1.2/32    152.1.2.5                                         0 200 i
*>                 152.1.1.9                 0                       0 200 i
*>  152.1.2.1/32   152.1.1.9                                         0 200 i
*>                 152.1.2.5                 0                       0 200 i
*  152.1.2.2/32    152.1.1.9                                         0 200 i
*>                 152.1.2.5                                         0 200 i
```

Display the BGP table on RouterB. With the command **show ip bgp.** The following is the output. Notice that the path to network 152.1.2.2 is via (65051).

```
RouterB#show ip bgp
BGP table version is 11, local router ID is 152.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network            Next Hop      Metric      LocPrf      Weight      Path
*  1.1.1.1/32      152.1.1.65    0           100         0           (65051) 100 i
*>                 152.1.1.10    0                       0           100 i
*>i152.1.1.1/32    152.1.1.10    0                       0           i
*> 152.1.1.2/32    0.0.0.0       0                                   32768 i
*> 152.1.2.1/32    152.1.1.65    0           100         0           (65051) i
*> 152.1.2.2/32    152.1.1.65    0           100         0           (65051) i
```

Even though EBGP is used between the sub−ASs, routing inside the confederation behaves just like routing in a single AS. The IBGP attributes are preserved when crossing the sub−AS boundary.

Display the BGP table on RouterC. Notice that the best path to reach network 1.1.1.1 is via 152.1.2.6.

```
RouterC#show ip bgp
BGP table version is 9, local router ID is 152.1.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network            Next Hop      Metric     LocPrf     Weight      Path
*  1.1.1.1/32      152.1.1.66    0           100         0         (65050) 100 i
*>                 152.1.2.6     0                       0         100 i
*> 152.1.1.1/32    152.1.1.66    0           100         0         (65050) i
*> 152.1.1.2/32    152.1.166     0           100         0         (65050) i
*> 152.1.2.1/32    0.0.0.0       0                                  32768 i
*>i152.1.2.2/32    152.1.2.130   0           100         0          i
```

Configure RouterE to set the MED for network 1.1.1.1 to 50 when it advertises the route to RouterC. The following is the configuration information that must be added to RouterE.

```
RouterE(config)#route−map med permit 10
RouterE(config−route−map)#match ip address 1
RouterE(config−route−map)#set metric 50

RouterE(config)#access−list 1 permit 1.1.1.1 255.255.255.255

RouterE(config)#router bgp 100
RouterE(config−router)#neighbor 152.1.2.5 route−map med out
```

Display the BGP table on RouterC. Notice that the best path to reach network 1.1.1.1 is now via 152.1.1.66. The MED is used in the decision process even though it appears that the route via 152.1.1.66 has a longer AS path (65050) 100 vs. 100 via 152.1.2.6.

The reason for this is the path length of the internal route via (65050) 100 is considered to be the same length as the path via 152.1.2.6. Sub−ASs are not counted in calculating the path length.

```
RouterC#sho ip bgp
BGP table version is 19, local router ID is 152.1.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network            Next Hop      Metric     LocPrf     Weight      Path
*> 1.1.1.1/32      152.1.1.66    0           100         0         (65050) 100 i
*                  52.1.2.6      50                      0         100 i
*> 152.1.1.1/32    152.1.1.66    0           100         0         (65050) i
*> 152.1.1.2/32    152.1.1.66    0           100         0         (65050) i
*> 152.1.2.1/32    0.0.0.0       0                                 32768 i
*>i152.1.2.2/32    152.1.2.130   0           100         0         i
```

Display the BGP table on RouterB. The following is the output. Notice that the MED attribute has been carried across the sub−AS.

```
RouterB#show ip bgp
BGP table version is 6, local router ID is 152.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network            Next Hop      Metric     LocPrf     Weight      Path
*> 1.1.1.1/32      152.1.1.10    0                       0         100 i
*                  152.1.1.65    50         100          0       (65051) 100 i
*>i152.1.1.1/32    152.1.1.5     0          100          0       i
*> 152.2.1/32      152.1.1.65    0          100          0       (65051) i
*> 152.1.2.2/32    152.1.1.65    0          100          0       (65051) i
```

Lab #52: BGP Communities

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface
- One Cisco router with an Ethernet interface
- One Cisco router with two serial interfaces
- One Cisco router with one Ethernet and two serial interfaces
- One Cisco router with one serial and two Ethernet interfaces
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for connecting to the console port of the router
- Three DTE/DCE crossover cables
- Two Ethernet cables and two hubs (or two Ethernet crossover cables)
- One Cisco rolled cable

Configuration Overview

A BGP community is a group of routers that share a common property. For example, a group of routers that are in the same AS, that don't want to propagate network X outside of the AS, could tag the prefix with the NO_EXPORT community attribute. When a router receives a route tagged with the NO_EXPORT attribute, the prefix will not be sent outside of the AS. The use of communities simplifies routing policies by identifying routes based on logical properties rather than IP prefix.

The community attribute is optional and transitive. "Optional" means that all implementations of BGP may not recognize the community. "Transitive" means that the community value should be passed to BGP neighbors.

There are two types of communities: well−known communities, which are reserved, and private communities, which are defined for local use.

An example of well−known communities are as follows:

- **NO_EXPORT:** Routes that carry this community value should not be advertised outside of the local AS or local confederation.
- **NO_ADVERTISE:** Routes that carry this community value should not be advertised to any BGP peer.

Private community attributes can also be defined. The common practice is to use the first two octets of the community for the AS number and the last two octets to define a policy. The community is written in decimal notation — for example, AS.policy.

This lab will demonstrate how communities can be used to simplify routing policies. All routers will be configured for BGP. OSPF will be used as the IGP within the ASs. RouterA, RouterB, RouterC, and RouterD are in AS 200.

RouterA is connected serially via a crossover cable to RouterB, which will act as the DCE supplying clock to RouterA. RouterB is connected via Ethernet to RouterC, and serially to RouterE via a crossover cable. RouterC is connected to RouterD via Ethernet and to RouterE via a serial crossover cable.

OSPF process 64 is run in AS 200 as the IGP. RouterB is configured as the route reflector for the AS. All routers in AS 200 will IBGP peer with RouterB. RouterB and RouterC are running EBGP to RouterE in AS 100. The IP addresses are assigned as per Figure 10−24.

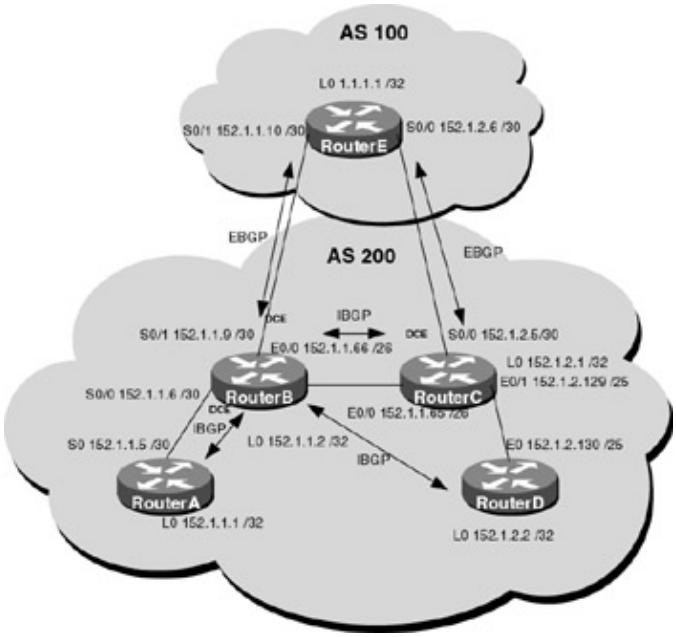


Figure 10−24: BGP communities

Router Configurations

The configurations for the five routers in this example are as follows.

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Loopback0
 ip address 152.1.1.1 255.255.255.255
!
!
interface Serial0
 ip address 152.1.1.5 255.255.255.252
!
router ospf 64
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200
 no synchronization
 network 152.1.1.1 mask 255.255.255.255
neighbor 152.1.1.2 remote−as 200
 neighbor 152.1.1.2 update−source Loopback0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 12.0
```

```
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
ip subnet-zero
!
interface Loopback0
 ip address 152.1.1.2 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0/0
 ip address 152.1.1.66 255.255.255.192
no ip directed-broadcast
!
interface Serial0/0
 ip address 152.1.1.6 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 clockrate 1000000
!
interface Serial0/1
 ip address 152.1.1.9 255.255.255.252
 no ip directed-broadcast
 clockrate 1000000
!
router ospf 64
 passive-interface Serial0/1
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200
 no synchronization
 network 152.1.1.2 mask 255.255.255.255
 neighbor 152.1.1.1 remote-as 200
 neighbor 152.1.1.1 update-source Loopback0
 neighbor 152.1.1.1 route-reflector-client
 neighbor 152.1.1.10 remote-as 100
 neighbor 152.1.2.1 remote-as 200
 neighbor 152.1.2.1 update-source Loopback0
 neighbor 152.1.2.1 route-reflector-client
 neighbor 152.1.2.2 remote-as 200
 neighbor 152.1.2.2 update-source Loopback0
 neighbor 152.1.2.2 route-reflector-client
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterC
```

```
!
ip subnet-zero
!
interface Loopback0
 ip address 152.1.2.1 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0/0
 ip address 152.1.1.65 255.255.255.192
 no ip directed-broadcast
!
interface Serial0/0
 ip address 152.1.2.5 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 clockrate 1000000
!
interface Ethernet1/0
 ip address 152.1.2.129 255.255.255.128
 no ip directed-broadcast
!
router ospf 64
 passive-interface Serial0/0
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200
 no synchronization
 network 152.1.2.1 mask 255.255.255.255
 neighbor 152.1.1.2 remote-as 200
 neighbor 152.1.1.2 update-source Loopback0
neighbor 152.1.2.6 remote-as 100
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterD
!
ip subnet-zero
!
interface Loopback0
 ip address 152.1.2.2 255.255.255.255
 no ip directed-broadcast
 ip ospf interface-retry 0
!
interface Ethernet0
 ip address 152.1.2.130 255.255.255.128
 no ip directed-broadcast
 ip ospf interface-retry 0
!
```

```
router ospf 64
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200
 no synchronization
 network 152.1.2.2 mask 255.255.255.255
 neighbor 152.1.1.2 remote-as 200
 neighbor 152.1.1.2 update-source Loopback0
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterE

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterE
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
 no ip directed-broadcast
!
interface Serial0
 ip address 152.1.1.10 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
!
interface Serial1
 ip address 152.1.2.6 255.255.255.252
 no ip directed-broadcast
!
router bgp 100
no synchronization
 network 1.1.1.1 mask 255.255.255.255
 neighbor 152.1.1.9 remote-as 200
 neighbor 152.1.2.5 remote-as 200
!
ip classless
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display the BGP table on RouterE with the command **show ip bgp**. The following is the output from the command. Notice that RouterE has a route to network 152.1.1.1 via RouterC and RouterB.

```
RouterE#show ip bgp
BGP table version is 16, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
```

```
Origin codes: i − IGP, e − EGP, ? − incomplete

Network            Next Hop      Metric      LocPrf      Weight      Path
*> 1.1.1.1/32      0.0.0.0       0                                   32768 i
*  152.1.1.1/32    152.1.2.5                               0           200 i
*>                 152.1.1.9                               0           200 i
*  152.1.1.2/32    152.1.2.5                               0           200 i
*>                 152.1.1.9     0                         0           200 i
*  152.1.2.1/32    152.1.1.9                               0           200 i
*>                 152.1.2.5     0                         0           200 i
*  152.1.2.2/32    152.1.1.9                               0           200 i
*>                 152.1.2.5                               0           200 i
```

Configure RouterA so that when it advertises network 152.1.1.1, it applies the NO_EXPORT community attribute. This will prevent the route from being advertised outside of the AS.

To do this, you first need to identify the prefix using an access list, define the community that will be assigned to that prefix with a route map, and apply the route map to a neighbor:

1. Define an access list to permit prefix 152.1.1.1/32.

   ```
   RouterA(config)#access−list 3 permit 152.1.1.1 0.0.0.0
   ```
2. A route map named Set_Community is defined to match on prefix 152.1.1.1/32 and set its community attribute to no−export.

   ```
   RouterA(config)#route−map Set_Community permit 10
   RouterA(config−route−map)#match ip address 3
   RouterA(config−route−map)#set community no−export
   ```
3. The last step is to apply the route map to a neighbor. The **send−community** keyword must be assigned to a neighbor session in order to enable the community attribute to be sent to a specified neighbor.

   ```
   RouterA(config−router)#neighbor 152.1.1.2 route−map Set_Community out
   RouterA(config−router)#neighbor 152.1.1.2 send−community
   ```

Use the **show ip bgp community no−export** on RouterB to verify that network 152.1.1.1/32 has the community attribute set. The following is the output from the command:

```
BGP table version is 10, local router ID is 152.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network              Next Hop      Metric      LocPrf      Weight      Path
*>i152.1.1.1/32    152.1.1.1     0           100                     0i
```

Display the BGP table on RouterE. The following is the output. Notice that RouterE now only has a route to network 152.1.1.1 via RouterC. RouterA is no longer advertising network 152.1.1.1 to RouterE.

```
RouterE#show ip bgp
BGP table version is 6, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network           Next Hop      Metric      LocPrf      Weight      Path
*> 1.1.1.1/32     0.0.0.0       0                                   32768 i
*> 152.1.1.1/32   152.1.2.5                               0           200 i
*  152.1.1.2/32   152.1.1.9     0                         0           200 i
*>                152.1.2.5                               0           200 i
*  152.1.2.1/32   152.1.1.9                               0           200 i
*>                152.1.2.5     0                         0           200 i
*  152.1.2.2/32   152.1.1.9                               0           200i
*>                152.1.2.5                               0           200 i
```

The reason that RouterC is still advertising the network is that the community attribute was not passed by RouterB. The send community option in the neighbor router subcommand is needed to cause the community to be sent to the BGP neighbors. The following commands enable RouterB to send community information to RouterD and RouterC:

```
RouterB(config−router)#neighbor 152.1.2.1 send−community
RouterB(config−router)#neighbor 152.1.2.2 send−community
```

Use the **show ip bgp community no−export** on RouterC to verify that network 152.1.1.1/32 has the community attribute set. The following is the output from the command:

```
BGP table version is 25, local router ID is 152.1.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network          Next Hop      Metric      LocPrf      Weight      Path
*>i152.1.1.1/32   152.1.1.1     0           100                     0 i
```

Display the BGP table on RouterE. The following is the output. Notice that RouterE no longer has network 152.1.1.1 in the table.

```
RouterE#show ip bgp
BGP table version is 18, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network          Next Hop      Metric      LocPrf      Weight      Path
*> 1.1.1.1/32     0.0.0.0       0                               32768 i
* 152.1.1.2/32   152.1.1.9      0                       0       200 i
*>               152.1.2.5                             0       200 i
* 152.1.2.1/32   152.1.1.9                             0       200 i
*>               152.1.2.5      0                      0       200 i
* 152.1.2.2/32   152.1.2.5                             0       200 i
*>               152.1.1.9                             0       200 i
```

Display the BGP table on RouterA with the command **show ip bgp**. The following is the output from the command. Notice that RouterA has network 152.1.2.2 in its table.

```
RouterA#show ip bgp
BGP table version is 28, local router ID is 152.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network          Next Hop      Metric      LocPrf      Weight      Path
*>i1.1.1.1/32     152.1.1.10    0           100         0           100 i
*> 152.1.1.1/32   0.0.0.0       0                                   32768 i
*>152.1.1.2/32    152.1.1.2     0           100         0           i
*>i152.1.2.1/32   152.1.2.1     0           100         0           i
*>i152.1.2.2/32   152.1.2.2     0           100         0           i
```

Configure RouterD so that when it advertises network 152.1.2.2 it applies the NO_ADVERTISE community attribute. This will prevent the route from being advertised by RouterB to the route reflector clients.

To do this, you first need to identify the prefix using an access list, define the community that will be assigned to that prefix with a route map, and apply the route map to a neighbor.

1. Define an access list to permit prefix 152.1.2.2/32.

   ```
   RouterD(config)#access−list 3 permit 152.1.2.2 0.0.0.0
   ```
2. A route map named No_Advertise is defined to match on prefix 152.1.2.2/32 and set its community attribute to no−advertise.

```
RouterD(config)#route-map No_Advertise permit 10
RouterD(config-route-map)#match ip address 3
RouterA(config-route-map)#set community no-advertise
```

3. The last step is to apply the route map to a neighbor. The **send-community** keyword must be assigned to a neighbor session in order to enable the community attribute to be sent to a specified neighbor.

```
RouterD(config-router)#neighbor 152.1.1.2 route-map No_Advertise out
RouterD(config-router)#neighbor 152.1.1.2 send-community
```

Use the **show ip bgp community no-advertise** on RouterB to verify that network 152.1.2.3/32 has the community attribute set. The following is the output from the command:

```
RouterB#show ip bgp community no-advertise
BGP table version is 9, local router ID is 152.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network          Next Hop      Metric      LocPrf      Weight      Path
*>i152.2.2.2/32   152.1.2.2     0           100                     0i
```

Display the BGP table on RouterA with the command **show ip bgp**. The following is the output from the command. Notice that RouterA no longer has network 152.1.2.2 in its table.

```
RouterA#show ip bgp
BGP table version is 29, local router ID is 152.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network          Next Hop      Metric      LocPrf      Weight      Path
*>i.1.1.1.1/32    152.1.1.10    0           100         0           100 i
*> 152.1.1.1/32   0.0.0.0       0                                   32768 i
*>i152.1.1.2/32   152.1.1.2     0           100         0           i
*>i152.1.2.1/32   152.1.2.1     0           100         0           i
```

Display the BGP table on RouterC with the command **show ip bgp**. The following is the output from the command. Notice that RouterC no longer has network 152.1.2.2 in its table. A route carrying the no-advertise community, when received, will not be advertised to any BGP peer. The only router that will have network 152.1.2.2 in its BGP table is RouterB.

```
RouterC#show ip bgp
BGP table version is 26, local router ID is 152.1.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network          Next Hop      Metric      LocPrf      Weight      Path
* i1.1.1.1/32     152.1.1.10    0           100         0           100 i
*>                152.1.2.6     0                       0           100 i
*>i152.1.1.1/32   152.1.1.1     0           100         0           i
*>i152.1.1.2/32   152.1.1.2     0           100         0           i
*> 152.1.2.1/32   0.0.0.0       0                                   32768 i
```

Lab #53: BGP Backdoor Links

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface

- One Cisco router with an Ethernet interface
- One Cisco router with two serial interfaces
- One Cisco router with one Ethernet and two serial interfaces
- One Cisco router with one serial and two Ethernet interfaces
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program for connecting to the console port of the router
- Three DTE/DCE crossover cables
- Two Ethernet cables and two hubs (or two Ethernet crossover cables)
- One Cisco rolled cable

Configuration Overview

Cisco provides a way to force IGP routes to take precedence over EBGP routes. The concept is called "backdoor links." EBGP routes can be tagged as backdoor routes, which sets the distance of these routes to the same as BGP local or 200. Since the distance is then higher than the IGP route, the backdoor IGP route is preferred.

In Figure 10−25, AS 200 is running OSPF on a link between it and AS 300 and is running EBGP to AS 100. RouterB will receive advertisements about network 152.1.2.2 and 152.1.2.1 from AS 100 via EBGP and from AS 300 via OSPF. Since the administrative distance of EBGP (20) is lower than OSPF (110), RouterB will use the BGP link to AS 100 to reach the two networks.

Figure 10−25: BGP backdoor links

This lab will demonstrate how BGP backdoor links can be used to prefer IGP learned routes over EBGP routes. All routers will be configured for BGP, and OSPF will be used as the IGP within the ASs and between AS 200 and AS 300.

RouterA is connected serially via a crossover cable to RouterB, which will act as the DCE supplying clock to RouterA. RouterB is connected via Ethernet to RouterC and serially to RouterE via a crossover cable. RouterC is connected to RouterD via Ethernet and to RouterE via a serial crossover cable.

OSPF process 64 is run in AS 200 and AS 300 as the IGP. RouterB and RouterC are running EBGP to RouterE in AS 100. The IP addresses are assigned as per Figure 10−25.

Router Configurations

The configurations for the five routers in this example are as follows.

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
interface Loopback0
 ip address 152.1.1.1 255.255.255.255
!
!
interface Serial0
 ip address 152.1.1.5 255.255.255.252
!
router ospf 64
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200
 no synchronization
neighbor 152.1.1.2 remote-as 200
 neighbor 152.1.1.2 update-source Loopback0
!
line con 0
line 1 16
 line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
interface Loopback0
 ip address 152.1.1.2 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0/0
 ip address 152.1.1.66 255.255.255.192
 no ip directed-broadcast
!
interface Serial0/0
 ip address 152.1.1.6 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 clockrate 1000000
!
interface Serial0/1
 ip address 152.1.1.9 255.255.255.252
 no ip directed-broadcast
 clockrate 1000000
!
router ospf 64
 passive-interface Serial0/1
```

```
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 200
 no synchronization
network 152.1.1.4 mask 255.255.255.252
 neighbor 152.1.1.1 remote-as 200
 neighbor 152.1.1.1 update-source Loopback0
 neighbor 152.1.1.1 next-hop-self
 neighbor 152.1.1.10 remote-as 100
!
ip classless
no ip http server
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterC
!
interface Loopback0
 ip address 152.1.2.1 255.255.255.255
 no ip directed-broadcast
!
interface Ethernet0/0
 ip address 152.1.1.65 255.255.255.192
 no ip directed-broadcast
!
interface Serial0/0
 ip address 152.1.2.5 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 clockrate 1000000
!
interface Ethernet1/0
 ip address 152.1.2.129 255.255.255.128
 no ip directed-broadcast
!
router ospf 64
 passive-interface Serial0/0
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 300
 no synchronization
 network 152.1.2.1 mask 255.255.255.255
 neighbor 152.1.2.2 remote-as 300
 neighbor 152.1.2.2 update-source Loopback0
 neighbor 152.1.2.2 next-hop-self
 neighbor 152.1.2.6 remote-as 100
!
ip classless
no ip http server
!
line con 0
 line aux 0
line vty 0 4
```

```
 login
!
end
```

RouterD

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterD
!
interface Loopback0
 ip address 152.1.2.2 255.255.255.255
 no ip directed-broadcast
 ip ospf interface-retry 0
!
interface Ethernet0
 ip address 152.1.2.130 255.255.255.128
 no ip directed-broadcast
 ip ospf interface-retry 0
!
!
router ospf 64
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 300
 no synchronization
 neighbor 152.1.2.1 remote-as 300
 neighbor 152.1.2.1 update-source Loopback0
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterE

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterE
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
 no ip directed-broadcast
!
interface Serial0
 ip address 152.1.1.10 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
!
interface Serial1
ip address 152.1.2.6 255.255.255.252
 no ip directed-broadcast
!
```

```
router bgp 100
 no synchronization
 network 1.1.1.1 mask 255.255.255.255
 neighbor 152.1.1.9 remote-as 200
 neighbor 152.1.2.5 remote-as 300
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display the routing table on RouterC. The following is the output. Note that network 152.1.1.4/30 is in the routing table as an EBGP learned route.

```
RouterC#sho ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/32 is subnetted, 1 subnets
B       1.1.1.1 [20/0] via 152.1.2.6, 00:00:46
     152.1.0.0/16 is variably subnetted, 9 subnets, 4 masks
C       152.1.2.128/25 is directly connected, Ethernet1/0
O       152.1.1.8/30 [110/58] via 152.1.1.66, 00:00:46, Ethernet0/0
O       152.1.2.2/32 [110/11] via 152.1.2.130, 00:00:46, Ethernet1/0
O       152.1.1.1/32 [110/59] via 152.1.1.66, 00:00:46, Ethernet0/0
O       152.1.1.2/32 [110/11] via 152.1.1.66, 00:00:46, Ethernet0/0
C       152.1.2.1/32 is directly connected, Loopback0
B       152.1.1.4/30 [20/0] via 152.1.2.6, 00:00:47
C       152.1.2.4/30 is directly connected, Serial0/0
C       152.1.1.64/26 is directly connected, Ethernet0/0
O    192.1.1.0/24 [110/68] via 152.1.1.66, 00:00:47, Ethernet0/0
```

In order to get RouterC to prefer the OSPF learned route, the BGP learned prefix must be tagged as a backdoor route. To tag the network prefix as a backdoor route perform the following on RouterC:

```
RouterC(config)#router bgp 300
RouterC(config-router)#network 152.1.1.4 mask 255.255.255.252 backdoor
```

Display the routing table on RouterC. The following is the output. Note that network 152.1.1.4/30 is now in the routing table as an OSPF learned route.

```
RouterC#sho ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/32 is subnetted, 1 subnets
```

```
B       1.1.1.1 [20/0] via 152.1.2.6, 00:03:19
     152.1.0.0/16 is variably subnetted, 9 subnets, 4 masks
C       152.1.2.128/25 is directly connected, Ethernet1/0
O       152.1.1.8/30 [110/58] via 152.1.1.66, 00:03:19, Ethernet0/0
O       152.1.2.2/32 [110/11] via 152.1.2.130, 00:03:19, Ethernet1/0
O       152.1.1.1/32 [110/59] via 152.1.1.66, 00:03:19, Ethernet0/0
O       152.1.1.2/32 [110/11] via 152.1.1.66, 00:03:19, Ethernet0/0
C       152.1.2.1/32 is directly connected, Loopback0
O       152.1.1.4/30 [110/58] via 152.1.1.66, 00:00:11, Ethernet0/0
C       152.1.2.4/30 is directly connected, Serial0/0
C       152.1.1.64/26 is directly connected, Ethernet0/0
O    192.1.1.0/24 [110/68] via 152.1.1.66, 00:03:19, Ethernet0/0
```

Troubleshooting BGP

The Cisco IOS provides many tools for troubleshooting routing protocols. The following is a list of key commands along with sample output from each that will aid in troubleshooting BGP.

{show ip bgp} This exec command displays all the entries in the BGP routing table. This command is helpful in determining if a route has been learned by the BGP process.

```
RouterA#show ip bgp
      ↓ Internal version number of the table. This number is incremented
        whenever the table changes
BGP table version is 3, local router ID is 5.5.5.5 ← IP address of the router
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network        Next Hop      Metric      LocPrf      Weight      Path
*> 1.0.0.0     0.0.0.0       0                       32768       i
*> 2.0.0.0     0.0.0.0       0                       32768       i
```

{show ip bgp filter−list} This exec command displays all routes that conform to a specified filter list. The following is a sample output from the command:

```
RouterC#show ip bgp filter−list 1
BGP table version is 5, local router ID is 195.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network        Next Hop      Metric      LocPrf      Weight      Path
*> 2.0.0.0     193.1.1.1     0                       0           100 i
* i           192.1.1.1     0           100         0           100 i
```

{show ip bgp neighbors} This exec command displays information about the TCP and BGP connections to neighbors. This command can be used with the argument **received routes or advertised−routes**, which displays all updates that are sent to or received from a particular neighbor. In order to display the received routes, inbound soft reconfiguration must be configured on the router.

```
RouterC#show ip bgp neighbors 193.1.1.1 received−routes
BGP table version is 3, local router ID is 195.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network        Next Hop      Metric      LocPrf      Weight      Path
*> 2.0.0.0     193.1.1.1     0                       0           100 300 400 I

RouterA#show ip bgp neighbors 193.1.1.2 advertised−routes
BGP table version is 5, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network        Next Hop      Metric      LocPrf      Weight      Path
*> 2.0.0.0     0.0.0.0       0                                   32768 i
```

{show ip bgp paths} This exec command displays all BGP paths in the database and the number of routes using each path. A regular expression can be added to the command to search for a particular AS or string of ASs.

```
RouterB#show ip bgp paths 400
Address        Hash        Refcount        Metric              Path
0x6069719C     219            1              0             100 300 400 i
0x60764F18     219            1              0             100 300 400 I
```

{show ip bgp regexp} This exec command displays all routes matching the regular expression. This command can quickly tell you if your regular expression is matching the routes that you require.

```
RouterB#show ip bgp regexp  _400_
BGP table version is 12, local router ID is 194.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i − internal
Origin codes: i − IGP, e − EGP, ? − incomplete

Network        Next Hop      Metric      LocPrf      Weight      Path
* i2.0.0.0     193.1.1.1     0           100         0           100 300 400 i
*>             192.1.1.1     0                       0           100 300 400 I
```

{show ip bgp summary} This exec command shows the status of all BGP connections. This command displays all of the neighbor routers that are attached and shows the length of time that the BGP session has been in state established, or the current state if it is not established.

```
RouterC#show ip bgp summary
BGP table version is 19, main routing table version 19 ← Indicates last
                                                         version of BGP
                                                         database that was
                                                         injected into main
                                                         routing table
1 network entries (2/3 paths) using 260 bytes of memory
2 BGP path attribute entries using 252 bytes of memory
0 BGP route−map cache entries using 0 bytes of memory
1 BGP filter−list cache entries using 16 bytes of memory

Neighbor     V   AS   MsgRcvd   MsgSent   TblVer   InQ   OutQ   Up/Down State
193.1.1.1    4   100  7106      7088      19       0      0     02:01:46
193.1.1.2    4   200  7092      7096      19       0      0     04:17:33
195.1.1.1    4   200  7072      093       19       0      0     04:17:36
```

Conclusion

BGP is an exterior gateway protocol (EGP), which means that it performs routing between multiple autonomous systems or domains. It was developed to replace the Exterior Gateway Protocol (EGP). "EGP" is a particular instance of an exterior gateway protocol (also EGP) as the standard exterior gateway routing protocol used in the global Internet. BGP solves serious problems that were present with EGP and scales to Internet growth more efficiently. BGP has been deployed extensively on routers within the Internet today.

Chapter 11: Route Redistribution

Overview

Topics Covered in This Chapter

- Redistributing RIP and IGRP
- Redistributing IGRP and EIGRP
- Redistributing RIP and OSPF
- Redistributing IGRP and OSPF
- Detailed troubleshooting examples

Introduction

Route redistribution is required when networks running multiple routing protocols need to be integrated. This chapter covers the interaction of RIP, IGRP, EIGRP, and OSPF routing protocols as well as how to successfully redistribute routes from one protocol to another.

Commands Discussed in This Chapter

- **area** area−id **range** address mask
- **default−metric** number
- **default−metric** bandwidth delay reliability loading mtu
- **distribute−list** access−list−number| name **in** [type number]
- **distribute−list** access−list−number| name **out** [type number]
- **redistribute protocol** [process−id] {level−1 | level−1−2 | level−2} [metric metric−value]
- **show ip protocols**
- **summary−address** address mask {level−1 | level−1−2 | level−2} prefix mask [not−advertise]

Definitions

area range: This router configuration command is used to consolidate and summarize routes at an area border router (ABR).

default−metric: This router configuration command is used to set the metric value for all routes being redistributed into IGRP, EIGRP, BGP, EGP, and OSPF. The default−metric command is used in conjunction with the redistribute router configuration command, setting the metric to the same value for all redistributed routes.

distribute−list in: This router configuration command is used to filter networks received in routing updates.

distribute−list out: This router configuration command is used to suppress networks sent out in routing updates.

redistribute protocol: This router configuration command is used to redistribute routes from one routing domain into another routing domain.

show ip protocols: This exec command will display the parameters and current state of all active routing protocol processes.

summary−address: This router configuration command is used to create aggregate addresses for IS−IS or OSPF. This command allows multiple groups of addresses to be summarized by an ASBR in one advertisement.

IOS Requirements

All of the labs in this chapter were done using IOS 11.2.

Lab #54: Redistributing RIP and IGRP

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, two with one serial port and two with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Three DTE/DCE crossover cables
- A Cisco rolled cable used for console port access

Configuration Overview

This configuration will demonstrate redistribution between two distance vector routing protocols, RIP and IGRP. NetworkA has just been acquired by NetworkB; the two networks are running different routing protocols. NetworkA is running RIP on RouterA and RouterB, and NetworkB is running IGRP on RouterC and RouterD. In order for the two networks to communicate, RIP is run between RouterB and RouterC.

All routers are connected serially via crossover cables. RouterB will act as the DCE supplying clock to RouterA and RouterC. RouterC will act as the DCE supplying clock to RouterD. The IP addresses are assigned as per Figure 11−1.

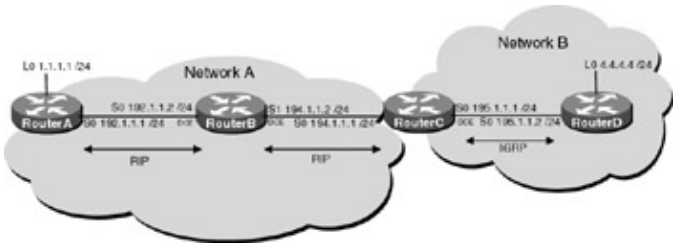


Figure 11−1: Redistributing RIP and IGRP

Router Configurations

The configurations for the four routers in this example are as follows (key routing configurations commands are highlighted in bold).

RouterA

```
Version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
```

```
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
interface Serial1
 no ip address
 shutdown
!
router rip ← Enables RIP on the Router
 network 192.1.1.0
 network 1.0.0.0
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 194.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
router rip ← Enables the RIP routing process on the router
 network 192.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
network 194.1.1.0
!
line con 0
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
!
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 194.1.1.1 255.255.255.0
!
interface Serial1
 ip address 195.1.1.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
router rip ← Enables the RIP routing process on the router
 network 194.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
!
router igrp 100 ← Enables the RIP routing process on the router
 network 195.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 195.1.1.2 255.255.255.0
!
interface Serial1
 no ip address
 shutdown
```

```
!
router igrp 100 ← Enables the RIP routing process on the router
 network 195.1.1.0
 network 4.0.0.0 ← Specifies what interfaces will receive and send RIP routing
                   updates. It also specifies what networks will be advertised
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

Display the IP routing table on RouterC with the command **show ip route;** what follows is the output from the command. Notice that RouterC has learned all of NetworkA's routes via RIP.

```
RouterC#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route

Gateway of last resort is not set

R    1.0.0.0/8 [120/2] via 194.1.1.2, 00:00:02, Serial0
I    4.0.0.0/8 [100/8976] via 195.1.1.2, 00:00:51, Serial1
R    192.1.1.0/24 [120/1] via 194.1.1.2, 00:00:02, Serial0
C    194.1.1.0/24 is directly connected, Serial0
C    195.1.1.0/24 is directly connected, Serial1
```

Display the IP routing table on RouterB with the command **show ip route;** what follows is the output from the command. Notice that RouterB has not learned any routes from NetworkB. The reason for this is that RouterB and RouterC are running RIP between them, not IGRP. If they were running IGRP, we would see the exact opposite: RouterB would see all of NetworkB's routes and RouterC would not see any of NewtorkA's routes.

```
RouterB#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default

Gateway of last resort is not set

R    1.0.0.0 [120/1] via 192.1.1.1, 00:00:05, Serial0
C    192.1.1.0 is directly connected, Serial0
C    194.1.1.0 is directly connected, Serial1
```

Remove RIP from network 194.1.1.0 on RouterB and RouterC.

```
RouterB#configure terminal
RouterB(config)#router rip
RouterB(config−router)#no network 194.1.1.0

RouterC#configure terminal
RouterC(config)#router rip
RouterC(config−router)#no network 194.1.1.0
```

Enable IGRP on network 194.1.1.0 on both RouterB and RouterC.

```
RouterB#configure terminal
RouterB(config)#router igrp 100
RouterB(config-router)#network 194.1.1.0

RouterC#configure terminal
RouterC(config)#router igrp 100
RouterC(config-router)#network 194.1.1.0
```

Display the IP routing table on RouterC with the command **show ip route;** what follows is the output from the command. Notice that RouterC no longer has any of NetworkA's routes.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

I    4.0.0.0/8 [100/8976] via 195.1.1.2, 00:00:56, Serial1
C    194.1.1.0/24 is directly connected, Serial0
C    195.1.1.0/24 is directly connected, Serial1
```

Display the IP routing table on RouterB with the command **show ip route;** what follows is the output from the command. Notice that RouterB has now learned NetworkB's routes via IGRP. This is because RouterB is now participating in the IGRP domain.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

R    1.0.0.0 [120/1] via 192.1.1.1, 00:00:26, Serial0
I    4.0.0.0 [100/10976] via 194.1.1.1, 00:00:04, Serial1
C    192.1.1.0 is directly connected, Serial0
C    194.1.1.0 is directly connected, Serial1
I    195.1.1.0 [100/10476] via 194.1.1.1, 00:00:04, Serial1
```

In order for RouterC to learn the RIP routes from NetworkA, we must use route redistribution. *Route redistribution* is the process of taking routes learned from one routing protocol, such as RIP, and injecting them into a different routing protocol, such as IGRP.

Since RouterB has all of the IGRP routes from NetworkA, we only need to redistribute the RIP learned routes on RouterB into IGRP. This is referred to as one-way redistribution, as opposed to mutual redistribution, where both routing protocols are redistributed into one another.

On RouterB, enable the redistribution of RIP into IGRP.

```
RouterB(config)#router igrp 100
RouterB(config-router)#redistribute rip
```

Display the routing table on RouterC with the command **show ip route;** what follows is the output from the command. Notice that RouterC still does not see any of NetworkA's routes. Why is this?

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
```

```
       U - per-user static route

Gateway of last resort is not set

I    4.0.0.0/8 [100/8976] via 195.1.1.2, 00:00:07, Serial1
I    192.1.1.0/24 [100/10476] via 194.1.1.2, 00:00:47, Serial0
C    194.1.1.0/24 is directly connected, Serial0
C    195.1.1.0/24 is directly connected, Serial1
```

From RouterC, display the IGRP routing updates with the command **debug ip igrp transactions;** what follows is the output from the command. Notice that RouterC is receiving an IGRP update for route 1.0.0.0. However, the route is marked inaccessible; this is why it is not being loaded in the IP routing table.

```
RouterC#
IGRP: received update from 194.1.1.2 on Serial0
      network 1.0.0.0, metric -1 (inaccessible)
      network 192.1.1.0, metric 10476 (neighbor 8476)
IGRP: received update from 195.1.1.2 on Serial1
      network 4.0.0.0, metric 8976 (neighbor 501)
```

The reason that the route is being advertised as inaccessible from RouterB is metrics. RIP and IGRP use totally different metrics to convey route preference. When we redistribute RIP into IGRP, we need to tell the router what the metric will be; otherwise, it marks the route as inaccessible.

We need to tell RouterB what it should set the metric to when it redistributes RIP routes into IGRP. There are several ways that this can be done. The first and simplest is to set a default metric that will be applied to any route that gets distributed into IGRP. To do this, add the following command to RouterB.

```
RouterB(config)#router igrp 100
RouterB(config-router)#default-metric 10000 100 255 1 1500
```

IGRP uses five metrics to calculate the cost of the route: bandwidth, delay, reliability, load, and MTU.

Display the routing table on RouterC with the command **show ip route;** what follows is the output from the command. Notice that all of NetworkA's routes are being learned via IGRP.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

I    1.0.0.0/8 [100/8576] via 194.1.1.2, 00:01:22, Serial0
I    4.0.0.0/8 [100/8976] via 195.1.1.2, 00:00:55, Serial1
I    192.1.1.0/24 [100/10476] via 194.1.1.2, 00:01:22, Serial0
C    194.1.1.0/24 is directly connected, Serial0
C    195.1.1.0/24 is directly connected, Serial1
```

The problem with using the default metric command is that it assigns this metric to all redistributed routes regardless of what protocol they originated from or how far away they actually are. The default metric can be set on a per-protocol basis by adding the metric to the end of the redistribution command. For example, if we only wanted to set the default metric for RIP routes being redistributed into IGRP, we would use the redistribute command as shown.

```
RouterB#configure termiinal
RouterB(config)#router igrp 100
RouterB(config-router)#redistribute rip metric 10000 100 255 1 1500 ← Default metric
```

This approach offers more flexibility, allowing each protocol that is being redistributed to have different metrics. For each protocol, however, we are assigning the same metric to every redistributed route. Route maps can be used to assign different metrics to routes learned from the same routing protocol. For example, we can assign different metrics for network 1.0.0.0 and network 192.1.1.0.

From RouterB, remove the default metric from the IGRP routing process.

```
RouterB#configure terminal
RouterB(config)#router igrp 100
RouterB(config-router)#no default-metric 10000 100 255 1 1500
```

Setting the metric for a particular route is a three-step process. We need to identify the network through the use of an access list, define the metric that will be applied to the routes through a route map, and assign the route map to a redistribution statement.

1. Add access list 1 to RouterB, permitting network 1.0.0.0.

   ```
   RouterB#configure terminal

   RouterB(config)#access-list 1 permit 1.0.0.0
   ```
2. Define a route map named **rip_to_igrp** that sets the five IGRP metrics of the route to "56 100 255 1 1500" if it matches access list 1 and "10000 100 255 1 1500" if it does not.

   ```
   RouterB#configure terminal

   RouterB(config)#route-map
   RouterB(config)#route-map rip_to_igrp 10
   RouterB(config-route-map)#match ip address 1
   RouterB(config-route-map)#set metric 56 100 255 1 1500
   RouterB(config-route-map)#exit
   RouterB(config)#route-map rip_to_igrp 20
   RouterB(config-route-map)#set metric  10000 100 255 1 1500
   ```
3. Apply the route map to routes being redistributed from RIP to IGRP.

   ```
   RouterB#configure terminal
   RouterB(config)#router igrp 100
   RouterB(config-router)#redistribute rip route-map rip_to_igrp
   ```

When a RIP route is redistributed into IGRP, the route map, rip_to_igrp, is consulted. If the route matches access list 1 (1.0.0.0), then the metrics are set to "56 100 255 1 1500;" if it does not, the metrics are set to "10000 100 255 1 1500."

Display the IP routing table on RouterC, noticing that the metrics have changed for network 1.0.0.0 but have remained the same for network 192.1.1.0.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

I    1.0.0.0/8 [100/180671] via 194.1.1.2, 00:00:05, Serial0
I    4.0.0.0/8 [100/8976] via 195.1.1.2, 00:00:05, Serial1
I    192.1.1.0/24 [100/8576] via 194.1.1.2, 00:00:05, Serial0
C    194.1.1.0/24 is directly connected, Serial0
C    195.1.1.0/24 is directly connected, Serial1
```

Up to this point, we have only been dealing with one-way redistribution; the next topic to be covered is

mutual redistribution. *Mutual redistribution* is when each routing protocol is redistributed into the other. In this example, RIP is redistributed into IGRP and IGRP is redistributed into RIP on RouterB. In order for RouterA to have visibility to NetworkB, RouterB must redistribute the IGRP routes into RIP.

Redistribute the IGRP routes into RIP on RouterB with the following commands.

```
RouterB#configure terminal
RouterB(config)#router rip
RouterB(config-router)#redistribute igrp 100 metric 3
```

Display the IP routing table on RouterA. Notice RouterA is now receiving all routes via RIP.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
R    4.0.0.0/8 [120/3] via 192.1.1.2, 00:00:10, Serial0
C    192.1.1.0/24 is directly connected, Serial0
R    194.1.1.0/24 [120/3] via 192.1.1.2, 00:00:10, Serial0
R    195.1.1.0/24 [120/3] via 192.1.1.2, 00:00:10, Serial0
```

Care must be taken when using mutual redistribution because routing loops can occur. For example, RouterB is advertising network 4.0.0.0, which it learned via IGRP to RouterA via RIP. What would happen if RouterA advertised the route back to RouterB ?

The rule of split horizons prevents this; however, what if split horizons were disabled on RouterA? RouterB would redistribute the RIP learned route into IGRP and cause a routing loop.

Lab #55: Redistributing IGRP and EIGRP

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, two with one serial port and two with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Three DTE/DCE crossover cables
- A Cisco rolled cable used for console port access

Configuration Overview

This configuration will demonstrate redistribution between two Cisco proprietary protocols, EIGRP and IGRP. NetworkB has just been acquired by NetworkA; the two networks are running different routing protocols. NetworkA is running EIGRP on RouterA and RouterB, and NetworkB is running IGRP on RouterC and RouterD. In order for the two network to communicate, EIGRP is run between RouterB and RouterC.

All routers are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to

RouterA and RouterC. RouterC will act as the DCE supplying clock to RouterD. The IP addresses are assigned as per Figure 11−2.

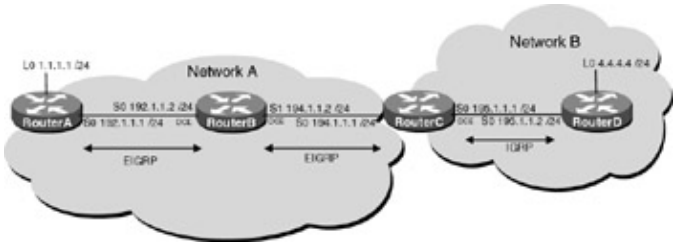


Figure 11−2: Redistribution between EIGRP and IGRP

Router Configurations

The configurations for the four routers in this example are as follows (key routing configurations commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
!
interface Serial0
 ip address 192.1.1.1 255.255.255.0
!
router eigrp 100
 network 192.1.1.0
 network 1.0.0.0
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 11.2
service udp−small−servers
service tcp−small−servers
!
hostname RouterB
!
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 192.1.1.2 255.255.255.0
```

```
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock
!
interface  Serial1
 ip address 194.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router eigrp 100
 network 192.1.1.0
 network  194.1.1.0
!
line con 0
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
!
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 194.1.1.1 255.255.255.0
!
interface Serial1
 ip address 195.1.1.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router eigrp 100
 network 194.1.1.0
!
router igrp 200
 network 195.1.1.0
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
version 11.2
no service udp-small-servers
no service  tcp-small-servers
!
hostname RouterD
!
```

```
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 195.1.1.2 255.255.255.0
!
!
router igrp 200
 network 195.1.1.0
 network 4.0.0.0
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

Display the IP routing table on RouterC with the command **show ip route;** what follows is the output from the command. Notice that RouterC has learned all of NetworkA's routes via EIGRP.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

D    1.0.0.0/8 [90/2809856] via 194.1.1.2, 00:02:07, Serial0
I    4.0.0.0/8 [100/8976] via 195.1.1.2, 00:00:10, Serial1
D    192.1.1.0/24 [90/2681856] via 194.1.1.2, 00:02:07, Serial0
C    194.1.1.0/24 is directly connected, Serial0
C    195.1.1.0/24 is directly connected, Serial1
```

Display the IP routing table on RouterB with the command **show ip route;** what follows is the output from the command. Notice that RouterB has not learned any routes from NetworkB.

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

D    1.0.0.0 [90/2297856] via 192.1.1.1, 00:02:47, Serial0
C    192.1.1.0 is directly connected, Serial0
C    194.1.1.0 is directly connected, Serial1
```

Display the IP routing table on RouterD with the command **show ip route;** what follows is the output from the command. Notice that RouterD has not learned any routes from NetworkA.

```
RouterD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
```

```
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     4.0.0.0/24 is subnetted, 1 subnets
C       4.4.4.0 is directly connected, Loopback0
C    195.1.1.0/24 is directly connected, Serial0
```

Why is this, doesn't IGRP and EIGRP redistribute automatically, since they are similar protocols? The reason that the redistribution is not occurring automatically is that the AS numbers are not the same. In order for mutual redistribution to be automatic, both EIGRP and IGRP must have the same AS number.

Change the AS number for the IGRP process on RouterC and RouterD to 100.

```
RouterC#conf terminal
RouterC(config)#no router igrp 200
RouterC(config)#router igrp 100
RouterC(config−router)#network 195.1.1.0
RouterD#conf terminal
RouterD(config)#no router igrp 200
RouterD(config)#router igrp 100
RouterD(config−router)#network 195.1.1.0
RouterD(config−router)# network 4.0.0.0
```

Display the IP routing table on RouterD with the command **show ip route;** what follows is the output from the command. Notice that RouterD has now learned all of routes on NetworkA.

```
RouterD#sho ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

I    1.0.0.0/8 [100/12976] via 195.1.1.1, 00:00:09, Serial0
     4.0.0.0/24 is subnetted, 1 subnets
C       4.4.4.0 is directly connected, Loopback0
I    192.1.1.0/24 [100/12476] via 195.1.1.1, 00:00:09, Serial0
I    194.1.1.0/24 [100/10476] via 195.1.1.1, 00:00:09, Serial0
```

We could have left the AS numbers different; however, we would then have had to use the redistribution command under the routing process.

Lab #56: Redistributing RIP and OSPF

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, two with one serial port and two with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program

- Three DTE/DCE crossover cables
- A Cisco rolled cable used for console port access

Configuration Overview

This configuration will demonstrate redistribution between a link state routing protocol (OSPF) and a distance vector routing protocol (RIP). NetworkB has just been acquired by NetworkA; the two networks are running different routing protocols. NetworkA is running OSPF on RouterA, RouterB, and RouterC, and NetworkB is running RIP on RouterD. In order for the two network to communicate, RIP is run between RouterC and RouterD.

All routers are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. RouterC will act as the DCE supplying clock to RouterD.

RouterA's serial and Ethernet interfaces are in OSPF area 1 along with RouterB interface S0. RouterC's interface S0 is in OPSF area 0 along with interface S1 on RouterB. RouterD is running RIP on all networks, and RouterC is performing mutual redistribution between OSPF and RIP. The IP addresses are assigned as per Figure 11−3.

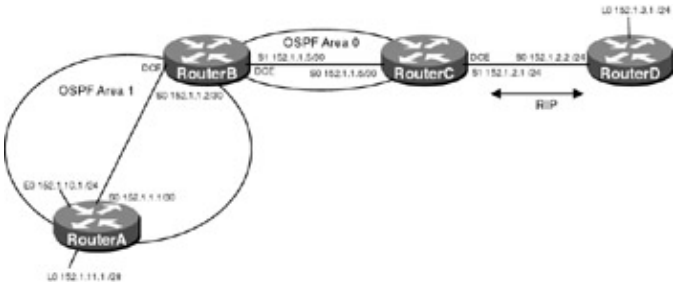


Figure 11−3: Redistribution between OSPF and RIP

Router Configurations

The configurations for the four routers in this example are as follows (key routing configurations commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 152.1.11.1 255.255.255.240
!
interface Ethernet0
 ip address 152.1.10.1 255.255.255.0
 no keepalive
!
interface Serial0
 ip address 152.1.1.1 255.255.255.252
!
router ospf 64
network 152.1.1.0 0.0.0.3 area 1
 network 152.1.10.1 0.0.0.16 area 1
!
no ip classless
!
line con 0
```

```
line 1 16
line aux 0
line vty 0 4
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!
interface Serial0
 ip address 152.1.1.2 255.255.255.252
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock
!
interface Serial1
 ip address 152.1.1.5 255.255.255.252
 clockrate 500000 ← Acts as DCE providing clock
!
!
router ospf 64
 network 152.1.1.0 0.0.0.3 area 1
 network 152.1.1.4 0.0.0.3 area 0
!
line con 0
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

RouterC

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
!
!
!
interface Serial0
 ip address 152.1.1.6 255.255.255.252
!
interface Serial1
 ip address 152.1.2.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router ospf 64
 redistribute rip
network 152.1.1.4 0.0.0.3 area 0
 default-metric 64
!
router rip
 redistribute ospf 64 ← Redistributes routes from OSPF process 64 to RIP
passive-interface Serial0
network 152.1.0.0
 default-metric 2 ← Sets the metric to 2 on any routes that are redistributed
                    into RIP
```

```
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
!
interface Loopback0
 ip address 152.1.3.1 255.255.255.0

!
interface Serial0
 ip address 152.1.2.2 255.255.255.0
!
!
router rip
 network 152.1.0.0
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Display the IP routing table on RouterC. What follows is the output from the command. Note that RouterC has learned about network 152.1.3.0, the loopback interface of RouterD, via RIP. It has also learned about network 152.1.10.0/24 and network 152.1.1.0/30 via OSPF. The routes are OSPF interarea routes, because they originated from OSFP area 1.

RouterC has not learned about network 152.1.11.0/28, the loopback interface on RouterA. The reason for this is that the network is not configured for OSPF, so it is not being advertised.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
     152.1.0.0/16 is variably subnetted, 5 subnets, 2 masks
O IA    152.1.10.0/24 [110/138] via 152.1.1.5, 00:02:36, Serial0 ← OSPF inter area route
O IA    152.1.1.0/30 [110/128] via 152.1.1.5, 00:05:39, Serial0
R       152.1.3.0/24 [120/1] via 152.1.2.2, 00:00:05, Serial1
```

```
C       152.1.2.0/24 is directly connected, Serial1
C       152.1.1.4/30 is directly connected, Serial0
```

To fix this problem, we could simply run OSPF on the network and then it would be advertised. The other option is to redistribute connected subnets on RouterA into OSPF.

Add the following command to the OSPF process on RouterA.

```
RouterA#configure terminal
RouterA(config)#router ospf 64
RouterA(config-router)#redistribute connected subnets
```

Now display the IP routing table on RouterC. What follows is the output from the command. Note that RouterC now sees the route; however, it is a OSPF external route because it was redistributed into the domain.

```
RouterC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 6 subnets, 3 masks
O E2    152.1.11.0/28 [110/20] via 152.1.1.5, 00:01:43, Serial0
O IA    152.1.10.0/24 [110/138] via 152.1.1.5, 00:09:21, Serial0
O IA    152.1.1.0/30 [110/128] via 152.1.1.5, 00:09:21, Serial0
R       152.1.3.0/24 [120/1] via 152.1.2.2, 00:00:16, Serial1
C       152.1.2.0/24 is directly connected, Serial1
C       152.1.1.4/30 is directly connected, Serial0
```

Now Display the IP routing table on RouterA. What follows is the output from the command. Note that RouterA is not receiving the route to network 152.1.3.0/24.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 4 subnets, 3 masks
C       152.1.11.0/28 is directly connected, Ethernet0
C       152.1.10.0/24 is directly connected, Loopback0
C       152.1.1.0/30 is directly connected, Serial0
O IA    152.1.1.4/30 [110/128] via 152.1.1.2, 00:01:53, Serial0
```

The RIP learned routes have not been successfully redistributed into OSPF. The reason for this lies with the current configuration on RouterC. Only routes with a 16-bit mask (class B) will be redistributed into OSPF. The networks on RouterD have been subnetted using a 24-bit mask.

In order to have the subnet redistributed, you must specify this in the configuration. Add the following command under the OSPF routing process on RouterC.

```
RouterC#configure terminal
RouterC(config)#router ospf 64
RouterC(config-router)#redistribute rip subnets
```

Display the IP routing table on RouterA. Notice that RouterA now has routes to networks 152.1.2.0 and 152.1.3.0. Also note that the routes are OSPF external routes (OE2), because they were learned from another domain.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 6 subnets, 3 masks
C       152.1.11.0/28 is directly connected, Loopback0
C       152.1.10.0/24 is directly connected, Ethernet0
C       152.1.1.0/30 is directly connected, Serial0
O E2    152.1.3.0/24 [110/64] via 152.1.1.2, 00:01:05, Serial0
O E2    152.1.2.0/24 [110/64] via 152.1.1.2, 00:01:05, Serial0
O IA    152.1.1.4/30 [110/128] via 152.1.1.2, 00:18:54, Serial0
```

Display the IP routing table on RouterD; what follows is the output. Note that RouterD has only learned about one network, 152.1.10.0/24 . The reason for this is that all of the other networks are subnetted past the 24−bit boundary; remember that RIP is a classful protocol and does not pass subnet information.

```
RouterD#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
R       152.1.10.0/24 [120/2] via 152.1.2.1, 00:00:27, Serial0
C       152.1.3.0/24 is directly connected, Loopback0
C       152.1.2.0/24 is directly connected, Serial0
C       152.1.4.0/28 is directly connected, Loopback1
```

In order to get the routes redistributed into RIP, either we can create static routes to the two networks using a 24−bit mask and redistribute the routes into RIP or we can summarize the routes in OSPF.

Let's examine the first option. Create two static routes on RouterC using a 24−bit mask.

```
RouterC#configure terminal
RouterC(config)#ip route 152.1.11.0 255.255.255.0 s0
RouterC(config)#ip route 152.1.1.0 255.255.255.0 s0
```

Redistribute the static routes into RIP on RouterC.

```
RouterC#configure terminal
RouterC(config−router)#redistribute static
```

Display the IP routing table on RouterD. Notice that RouterD now has routes to both networks.

```
RouterD#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
```

```
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 6 subnets, 2 masks
R       152.1.11.0/24 [120/1] via 152.1.2.1, 00:00:14, Serial0
R       152.1.10.0/24 [120/2] via 152.1.2.1, 00:00:14, Serial0
R       152.1.1.0/24 [120/1] via 152.1.2.1, 00:00:14, Serial0
C       152.1.3.0/24 is directly connected, Loopback0
C       152.1.2.0/24 is directly connected, Serial0
C       152.1.4.0/28 is directly connected, Loopback1
```

Now remove the static routes and the **redistribute static** command from RouterC.

```
RouterC#conf terminal
RouterC(config)#no ip route 152.1.1.0 255.255.255.0 Serial0
RouterC(config)#no ip route 152.1.11.0 255.255.255.0 Serial0
RouterC(config-router)#no redistribute static
```

Now let's examine option 2, summarizing the routes using a 24−bit mask in OSPF. To do this, we need to need to use the OSPF **area range** command and the OSPF **summary−address** command. Remember from Chapter 8 that the OSPF **area range** command is used to summarize routes from nonbackbone OSPF areas into area 0 and that the OSPF **summary−address** command is used to summarize external routes on an ASBR.

For this lab, we have both types: network 152.1.11.0/28 is an external route because it was redistributed into the OSPF process, and network 152.1.1.0/30 is a nonbackbone OSPF area (area1).

Note The summary−address command summarizes only routes from other routing protocols that are being redistributed into OSPF. The area range command is used for route summarization between OSPF areas. The area range command is used on area border routers (ABRs), and the summary−address command is used on autonomous system border routers (ASBRs).

Add the OSPF **summary−address** command under the OSPF process on RouterA. The command will be used to summarize network 152.1.11.0/28 to network 152.1.11.0/24 so that it will be propagated to RouterD.

```
RouterA#conf terminal
RouterA(config)#router ospf 64
RouterA(config-router)#summary-address 152.1.11.0 255.255.255.0
```

Display the IP routing table on RouterD. Note that RouterD now has a route to network 152.1.11.0/24 .

```
RouterD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 5 subnets, 2 masks
R       152.1.11.0/24 [120/2] via 152.1.2.1, 00:00:02, Serial0
R       152.1.10.0/24 [120/2] via 152.1.2.1, 00:00:02, Serial0
C       152.1.3.0/24 is directly connected, Loopback0
C       152.1.2.0/24 is directly connected, Serial0
C       152.1.4.0/28 is directly connected, Loopback1
```

Add the OSPF **area range** command under the OSPF process on RouterB. The command will be used to summarize network 152.1.1.0/30 to network 152.1.11.0/24 so that it will be propagated to RouterD.

```
RouterB#configure terminal
RouterB(config)#router ospf 64
RouterB(config-router)#area 1 range 152.1.1.0 255.255.255.0
```

Display the IP routing table on RouterD. Note that RouterD now has a route to network 152.1.1.0/24.

```
RouterD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 6 subnets, 2 masks
R       152.1.11.0/24 [120/2] via 152.1.2.1, 00:00:02, Serial0
R       152.1.10.0/24 [120/2] via 152.1.2.1, 00:00:02, Serial0
R       152.1.1.0/24 [120/2] via 152.1.2.1, 00:00:02, Serial0
C       152.1.3.0/24 is directly connected, Loopback0
C       152.1.2.0/24 is directly connected, Serial0
C       152.1.4.0/28 is directly connected, Loopback1
```

Care must be taken when using mutual redistribution in order to prevent routing loops. Metrics and split horizons will help to prevent routing loops, but it is a good idea to configure distribution lists so that the router cannot advertise invalid routes. To demonstrate this, disable split horizons on RouterC's interface s1.

```
RouterC#configure terminal
RouterC(config-if)#no ip split-horizon
```

Display the RIP update packets from RouterC with the **debug ip rip** command on RouterD. What follows is the output from the command; notice that RouterC is now advertising network 152.1.3.0, which is a directly connected network on RouterD. Although RouterD will not install this route in its routing table, because it prefers the directly connected network, it is good design to filter out the route using a distribute list on RouterC.

```
RouterD#debug ip rip
RIP: received v1 update from 152.1.2.1 on Serial0
     152.1.11.0 in 2 hops
     152.1.10.0 in 2 hops
     152.1.1.0 in 2 hops
     152.1.3.0 in 2 hops
     152.1.2.0 in 1 hops
```

Add a distribute list to RouterC permitting only networks 152.1.11.0, 152.1.1.0, and 152.1.10.0 to be advertised out via RIP. This a two-step process: first an access list must be set up to identify and permit or deny a network according to network address, and then the access list must be applied to routing updates through the use of a distribute list.

1. Define an access list on RouterC permitting networks 152.1.11.0, 152.1.1.0, and 152.1.10.0.

   ```
   RouterC#configure terminal
   RouterC(config)#access-list 1 permit 152.1.1.0 0.0.0.255
   RouterC(config)#access-list 1 permit 152.1.11.0 0.0.0.255
   RouterC(config)#access-list 1 permit 152.1.10.0 0.0.0.255
   ```
2. Apply the access list to routing updates through the use of a distribute list.

   ```
   RouterC#configure terminal
   RouterC(config-router)#distribute-list 1 out
   ```

Display the RIP update packets from RouterC with the **debug ip rip** command on RouterD. What follows is the output from the command; notice that RouterC is now advertising only the three networks.

```
RIP: received v1 update from 152.1.2.1 on Serial0
     152.1.11.0 in 2 hops
     152.1.10.0 in 2 hops
     152.1.1.0 in 2 hops
```

Lab #57: Redistributing IGRP and OSPF

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers, two with one serial port and two with two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Three DTE/DCE crossover cables
- A Cisco rolled cable used for console port access

Configuration Overview

This configuration will demonstrate redistribution between a link state routing protocol (OSPF) and a distance vector routing protocol (IGRP). NetworkB has just been acquired by NetworkA; the two networks are running different routing protocols. NetworkA is running OSPF on RouterA, RouterB, and RouterC, and NetworkB is running IGRP on RouterD. In order for the two networks to communicate, IGRP is run between RouterC and RouterD.

All routers are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA and RouterC. RouterC will act as the DCE supplying clock to RouterD.

RouterA's serial and Ethernet interfaces are in OSPF area 1 along with RouterB interface S0. RouterC's interface S0 is in OPSF area 0 along with interface S1 on RouterB. RouterD is running IGRP on all networks, and RouterC is performing mutual redistribution between OSPF and IGRP. The IP addresses are assigned as per Figure 11−4.

Figure 11−4: Redistribution between OSPF and IGRP

Router Configurations

The configurations for the four routers in this example are as follows (key routing configuration commands are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
```

```
no service tcp-small-servers
!
hostname RouterA
!
interface Loopback0
 ip address 152.1.11.1 255.255.255.240
!
interface Ethernet0
 ip address 152.1.10.1 255.255.255.0
 no keepalive
!
interface Serial0
 ip address 152.1.1.1 255.255.255.252
!
!
router ospf 64
 network 152.1.1.0 0.0.0.3 area 1
 network 152.1.10.1 0.0.0.16 area 1
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Serial0
 ip address 152.1.1.2 255.255.255.252
 no fair-queue
 clockrate 500000 ← Acts as DCE providing clock
!
interface Serial1
 ip address 152.1.1.5 255.255.255.252
 clockrate 500000 ← Acts as DCE providing clock
!
!

router ospf 64
 network 152.1.1.0 0.0.0.3 area 1
 network 152.1.1.4 0.0.0.3 area 0
!
line con 0
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

RouterC

```
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
```

```
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 152.1.1.6 255.255.255.252
!
interface Serial1
 ip address 152.1.2.1 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router ospf 64
redistribute igrp 100 ← Redistributes routes from IGRP process 100 into OSPF
network 152.1.1.4 0.0.0.3 area 0
default-metric 64 ← Sets the metric on any routes that are redistributed into
                    OSPF

!
router igrp 100
 redistribute ospf 64 ← Redistributes routes from OSPF process 64 into IGRP
 network 152.1.0.0
 default-metric 1000 10 1 255 1500 ← Sets the metric to any routes that are
                                     redistributed into IGRP

!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
!
interface Loopback0
 ip address 152.1.3.1 255.255.255.0
!
!
interface Serial0
 ip address 152.1.2.2 255.255.255.0
!
interface Serial1
 no ip address
!
router igrp 100
 network 152.1.0.0
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
```

```
!
end
```

Monitoring and Testing the Configuration

Display the IP routing table on RouterC. What follows is the output from the command; note that RouterC has learned about network 152.1.3.0, the loopback interface of RouterD, via IGRP. It has also learned about network 152.1.10.0/24 and network 152.1.1.0/30 via OSPF. The routes are OSPF interarea routes, because they originated from OSFP area 1.

RouterC has not learned about network 152.1.11.0/28, the loopback interface on RouterA. The reason for this is that the network is not configured for OSPF, so it is not being advertised.

```
RouterC#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 5 subnets, 2 masks
O IA    152.1.10.0/24 [110/138] via 152.1.1.5, 00:00:33, Serial0
O IA    152.1.1.0/30 [110/128] via 152.1.1.5, 00:00:33, Serial0
I       152.1.3.0/24 [100/8976] via 152.1.2.2, 00:00:21, Serial1
C       152.1.2.0/24 is directly connected, Serial1
C       152.1.1.4/30 is directly connected, Serial0
```

To fix this problem, we could simply run OSPF on the network and then it would be advertised. The other option is to redistribute connected subnets on RouterA into OSPF.

Add the following command to the OSPF process on RouterA.

```
RouterA#configure terminal
RouterA(config)#router ospf 64
RouterA(config−router)#redistribute connected subnets
```

Now display the IP routing table on RouterC. What follows is the output from the command. Note that RouterC now sees the route; however, it is an OSPF external route because it was redistributed into the domain.

```
RouterC#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 6 subnets, 3 masks
O E2    152.1.11.0/28 [110/20] via 152.1.1.5, 00:00:13, Serial0
O IA    152.1.10.0/24 [110/138] via 152.1.1.5, 00:00:14, Serial0
O IA    152.1.1.0/30 [110/128] via 152.1.1.5, 00:00:14, Serial0
I       152.1.3.0/24 [100/8976] via 152.1.2.2, 00:01:14, Serial1
C       152.1.2.0/24 is directly connected, Serial1
C       152.1.1.4/30 is directly connected, Serial0
```

Now display the IP routing table on RouterA.. What follows is the output from the command; note that RouterA is not receiving the route to network 152.1.3.0/24 .

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 4 subnets, 3 masks
C       152.1.11.0/28 is directly connected, Loopback0
C       152.1.10.0/24 is directly connected, Ethernet0
C       152.1.1.0/30 is directly connected, Serial0
O IA    152.1.1.4/30 [110/128] via 152.1.1.2, 00:01:02, Serial0
```

The IGRP learned routes have not been successfully redistributed into OSPF. The reason for this lies with the current configuration on RouterC. Only routes with a 16−bit mask (class B) will be redistributed into OSPF. The networks on RouterD have been subnetted using a 24−bit mask.

In order to have the subnet redistributed, you must specify this in the configuration. Add the following command under the OSPF routing process on RouterC.

```
RouterC#configure terminal
RouterC(config)#router ospf 64
RouterC(config−router)#redistribute igrp 100 subnets
```

Display the IP routing table on RouterA; what follows is the output. Notice that RouterA now has routes to networks 152.1.2.0 and 152.1.3.0. Also note that the routes are OSPF external routes (OE2), because they were learned from another domain.

```
RouterA#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     152.1.0.0/16 is variably subnetted, 6 subnets, 3 masks
C       152.1.11.0/28 is directly connected, Loopback0
C       152.1.10.0/24 is directly connected, Ethernet0
C       152.1.1.0/30 is directly connected, Serial0
O E2    152.1.3.0/24 [110/64] via 152.1.1.2, 00:00:04, Serial0
O E2    152.1.2.0/24 [110/64] via 152.1.1.2, 00:00:04, Serial0
O IA    152.1.1.4/30 [110/128] via 152.1.1.2, 00:00:04, Serial0
```

Display the IP routing table on RouterD; what follows is the output. Note that RouterD has learned about only one network, 152.1.10.0/24 . The reason for this is that all of the other networks are subnetted past the 24−bit boundary; remember that RIP is a classful protocol and will not pass subnet information.

```
RouterD#show ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     152.1.0.0/24 is subnetted, 3 subnets
```

```
I       152.1.10.0 [100/12000] via 152.1.2.1, 00:00:31, Serial0
C       152.1.3.0 is directly connected, Loopback0
C       152.1.2.0 is directly connected, Serial0
```

In order to get the routes redistributed into IGRP, either we can create static routes to the two networks using a 24−bit mask and redistribute the routes into IGRP or we can summarize the routes in OSPF.

Let's examine the first option. Create two static routes on RouterC using a 24−bit mask.

```
RouterC#configure terminal
RouterC(config)#ip route 152.1.11.0 255.255.255.0 s0
RouterC(config)#ip route 152.1.1.0 255.255.255.0 s0
```

Redistribute the static routes into IGRP on RouterC.

```
RouterC#configure terminal
RouterC(config−router)#redistribute static
```

Display the IP routing table on RouterD. Notice that RouterD now has routes to both networks.

```
RouterD#sho ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR

Gateway of last resort is not set

     152.1.0.0/24 is subnetted, 5 subnets
I       152.1.11.0 [100/12000] via 152.1.2.1, 00:00:03, Serial0
I       152.1.10.0 [100/12000] via 152.1.2.1, 00:00:03, Serial0
I       152.1.1.0 [100/12000] via 152.1.2.1, 00:00:03, Serial0
C       152.1.3.0 is directly connected, Loopback0
C       152.1.2.0 is directly connected, Serial0
```

Now remove the static routes and the effects of the **redistribute static** command from the IGRP routing process on RouterC.

```
RouterC#conf terminal
RouterC(config)#no ip route 152.1.1.0 255.255.255.0 Serial0
RouterC(config)#no ip route 152.1.11.0 255.255.255.0 Serial0
RouterC(config−router)#no redistribute static
```

Now let's examine option 2, summarizing the routes using a 24−bit mask in OSPF. To do this, we need to use the OSPF **area range** command and the OSPF **summary−address** command. Remember from Chapter 8 that the OSPF **area range** command is used to summarize routes from nonbackbone OSPF areas into area 0 and that the OSPF **summary−address** command is used to summarize external routes on an ASBR.

For this lab we have both types; network 152.1.11.0/28 is an external route because it was redistributed into the OSPF process, and network 152.1.1.0/30 is a nonbackbone OSPF area (area1).

Note The summary−address command summarizes only routes from other routing protocols that are being redistributed into OSPF. The area range command is used for route summarization between OSPF areas. The area range command is used on area border routers (ABR), and the summary−address command is used on autonomous system border routers (ASBR).

Add the OSPF **summary−address** command under the OSPF process on RouterA. The command will be used to summarize network 152.1.11.0/28 to network 152.1.11.0/24 so that it will be propagated to RouterD.

```
RouterA#conf terminal
RouterA(config)#router ospf 64
RouterA(config-router)#summary-address 152.1.11.0 255.255.255.0
```

Display the IP routing table on RouterD. What follows is the output; note that RouterD now has a route to network 152.1.11.0/24.

```
 RouterD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     152.1.0.0/24 is subnetted, 4 subnets
I       152.1.11.0 [100/12000] via 152.1.2.1, 00:00:09, Serial0
I       152.1.10.0 [100/12000] via 152.1.2.1, 00:00:09, Serial0
C       152.1.3.0 is directly connected, Loopback0
C       152.1.2.0 is directly connected, Serial0
```

Add the OSPF **area range** command under the OSPF process on RouterB. The command will be used to summarize network 152.1.1.0/30 to network 152.1.11.0/24 so that it will be propagated to RouterD.

```
RouterB#configure terminal
RouterB(config)#router ospf 64
RouterB(config-router)#area 1 range 152.1.1.0 255.255.255.0
```

Display the IP routing table on RouterD. What follows is the output; note that RouterD now has a route to network 152.1.1.0/24 .

```
RouterD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

     152.1.0.0/24 is subnetted, 5 subnets
I       152.1.11.0 [100/12000] via 152.1.2.1, 00:00:02, Serial0
I       152.1.10.0 [100/12000] via 152.1.2.1, 00:00:02, Serial0
I       152.1.1.0 [100/12000] via 152.1.2.1, 00:00:02, Serial0
C       152.1.3.0 is directly connected, Loopback0
C       152.1.2.0 is directly connected, Serial0
```

Care must be taken when using mutual redistribution in order to prevent routing loops. Metrics and split horizons will help to prevent routing loops, but it is a good idea to configure distribution lists so that the router cannot advertise invalid routes. To demonstrate this, disable split horizons on RouterC's interface s1.

```
RouterC#configure terminal
RouterC(config-if)#no ip split-horizon
```

Display the IGRP update packets being sent from RouterC with the **debug ip igrp transactions** command. What follows is the output from the command; notice that RouterC is now advertising network 152.1.3.0, which is a directly connected network on RouterD. Although RouterD will not install this route in its routing table, because it prefers the directly connected network, it is good design to filter out the route using a distribute list on RouterC.

```
RouterC#
IGRP: sending update to 255.255.255.255 via Serial1 (152.1.2.1)
      subnet 152.1.11.0, metric=10000
      subnet 152.1.10.0, metric=10000
      subnet 152.1.1.0, metric=10000
      subnet 152.1.3.0, metric=8976
      subnet 152.1.2.0, metric=8476
```

Add a distribute list to RouterC permitting only networks 152.1.11.0, 152.1.1.0, and 152.1.10.0 to be advertised out via IGRP. This a two−step process: first, an access list must be set up to identify and permit or deny a network according to its network address, and then the access list must be applied to routing updates through the use of a distribute list.

1. Define an access list on RouterC permitting networks 152.1.11.0, 152.1.1.0, and 152.1.10.0.

   ```
   RouterC#configure terminal
   RouterC(config)#access−list 1 permit 152.1.1.0 0.0.0.255
   RouterC(config)#access−list 1 permit 152.1.11.0 0.0.0.255
   RouterC(config)#access−list 1 permit 152.1.10.0 0.0.0.255
   ```
2. Apply the access list to routing updates through the use of a distribute list under the IGRP routing process.

   ```
   RouterC#configure terminal
   RouterC(config−router)#distribute−list 1 out
   ```

Display the IGRP update packets being sent from RouterC with the **debug ip igrp transactions** command on RouterD. What follows is the output from the command; notice that RouterC is now advertising only the three networks.

```
RouterC#
IGRP: sending update to 255.255.255.255 via Serial1 (152.1.2.1)
      subnet 152.1.11.0, metric=10000
      subnet 152.1.10.0, metric=10000
      subnet 152.1.1.0, metric=10000
```

Troubleshooting Route Redistribution

{show ip protocols} This exec command displays the parameters and current state of the active routing protocol process. The output shows the routing protocol used, timer information, inbound and outbound filter information, the protocols being redistributed, and the networks that the protocol is routing for. This command is very useful in quickly determining what routing protocols are running, what (if any) protocols are being redistributed, and what the redistribution metric is.

```
Routing Protocol is "ospf 64"
  Sending updates every 0 seconds
  Invalid after 0 seconds, hold down 0, flushed after 0
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default redistribution metric is 64
  Redistributing: ospf 64, igrp 100
  Routing for Networks:
    152.1.1.4/30
  Routing Information Sources:
    Gateway         Distance      Last Update
    152.1.11.1           110      00:15:27
    152.1.1.5            110      00:15:27
  Distance: (default is 110)
```

{debug ip igrp transactions} This exec command displays transaction information on Interior Gateway Routing Protocol (IGRP) routing transactions. The information contains the source and the destination of each update, the routes that are received or being advertised, and the metric of each route.

```
RouterA#debug ip igrp transactions
IGRP: sending update to 255.255.255.255 via Ethernet0 (148.1.1.1)
      network 10.0.0.0, metric=501
      network 152.1.0.0, metric=10576
      network 192.1.1.0, metric=8476
      network 193.1.1.0, metric=10476
```

{debug ip rip} This exec command displays information on RIP routing transactions. The output shows whether the router is sending or receiving an update, the networks contained in the update, and the metric or hop count for each.

```
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (148.1.1.1)
 network 10.0.0.0, metric 1
 network 192.1.1.0, metric 1
network 148.1.0.0, metric 1
RIP: received v1 update from 192.1.1.2 on Serial0
193.1.1.0 in 1 hops
```

{show ip access−list} This privileged exec command displays the contents of all current IP access lists. The command can also display specific access lists by number.

```
RouterA#show ip access−lists 1
Standard IP access list 1
permit 150.1.1.0, wildcard bits 0.0.0.255
```

Conclusion

Redistributing routes between protocols can be tricky; in order to be successful, you need to have a complete understanding of the protocols and how they interact with one another. Also understand how route distribution and route advertisement can be controlled with distribute lists, route maps, and passive interfaces. Remember, mutual redistribution can cause routing loops, so care must be taken.

Chapter 12: IP Access Lists

Overview

Topics Covered in This Chapter

- Detailed overview
- Access list terminology
- Standard IP access lists
- Extended IP access lists
- Extended access lists with the Established option
- Dynamic IP access lists
- Controlling VTY access
- How lock−and−key works
- Time of day access lists

Introduction

A router uses access lists to police traffic that comes into or leaves a specified interface. Below are some of the ways that access lists can be used:

- Deny or permit ingress or egress traffic from crossing specified interfaces.
- Define interesting traffic for DDR applications.
- Filter contents of routing updates.
- Control virtual terminal line access.
- Provide traffic flow control.

This chapter will explore controlling packet movement through the router using static and dynamic access lists.

Overview

Access lists (often referred to as access control lists) provide basic traffic−filtering capabilities. Access lists can be used to control access into or out of the network or to filter packets at ingress and egress router ports.

Access lists filter network traffic by determining whether a packet should be forwarded or dropped by the router based on predefined criteria. These criteria are defined by an access list, which in turn is applied to an interface.

Depending on the access list defined, the match criteria can be quite simple (standard access) or fairly complex (extended access list). Access lists provide a means of examining packets at the ingress and egress router ports, enabling the router to manipulate these packets based on certain criteria.

Access List Terminology

When dealing with access lists on a Cisco router, it is important to understand the terminology used:

- **Wildcard mask:** A wildcard mask specifies which bits in an IP address should be ignored when comparing that address with another IP address. A 1 in the wildcard mask means to ignore that bit position when comparing to another IP address and a 0 (zero) specifies that the bit position must match.

Note With a standard access list, if you omit the wildcard mask for an entry, 0.0.0.0 is assumed to be the mask.

For example, access list 1 below specifies that in order for traffic to be permitted, the first three octets must match (150.1.1). This access list permits hosts 1−255 on network 150.1.1.0.

```
access-list 1 permit 150.1.1.0 0.0.0.255
```

150	1	1	0	← IP address
10010110	00000001	00000001	00000000	← Binary representation of IP address
00000000	00000000	00000000	11111111	← Binary representation of Wildcard mask
10010110	00000001	00000001	xxxxxxx	
↑	↑	↑	↑	
150	1	1	x	← When comparing IP addresses, match the first three octets and ignore the last octet.

- **Inbound & outbound:** When applying an access list to an interface, the user specifies whether the access list is applied to inbound or outbound (or both) traffic. By default, the access list is applied to outbound traffic.

The direction of traffic flow is relative to the router interface. For example, in Figure 12−1, RouterA wishes to deny all traffic from host 150.1.1.2 destined for PCA (152.1.1.2). There are two places that an access list could be applied on RouterA: An inbound access list could be applied on the serial interface or an outbound access list could be applied to the Ethernet interface. It is good design to apply the access lists closest to the traffic that will be denied.

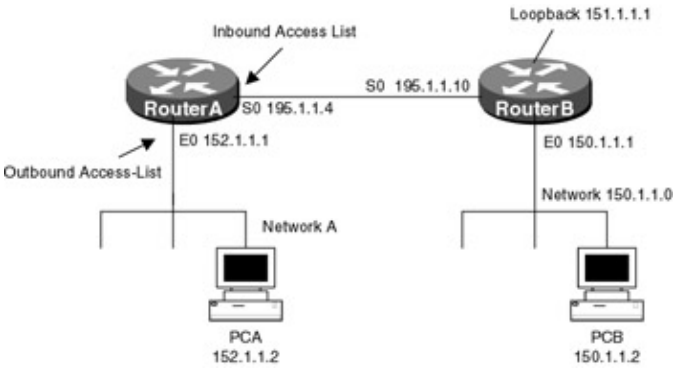


Figure 12−1: Access list terminology

Note Outbound access lists are applied to the packet after a routing decision is made and the packet is routed to the proper interface. At this point, the packet is applied against the access list and either forwarded or dropped. It makes no sense to route a packet through the router only to be dropped at the egress port. It is good design to apply access lists on the interface closest to the traffic that will be denied.

Commands Discussed in This Chapter

- **access−class** access−list−number {in | out}
- **access−enable** [host] [timeout minutes]
- **access−list** access−list−number {deny | permit} source [source−wildcard]
- **access−list** access−list−number {deny | permit}protocol source source−wildcard destination destination−wildcard [precedence precedence] [tos tos] [log]
- **access−list** access−list−number [**dynamic** dynamic−name [timeout minutes]]
- **access−template** [access−list−number | name] [dynamic−name] [source] [destination] [timeout minutes]
- **autocommand**
- **clear access−list counters** {access−list−number | name}

- **clear access−template** [access−list−number | name] [dynamic−name] [source] [destination]
- **ip access−group** {access−list−number | name}{in | out}
- **ip telnet source−interface**
- **periodic**
- **show access−lists** [access−list−number | name]
- **show ip access−list** [access−list−number | name]
- **time−range**
- **username** [name] **password** [password]

Definitions

access−class: The access−class command is used to restrict incoming and outgoing VTY connections between a particular VTY line on a Cisco router and a specific address defined by the access list. This line configuration command applies a specific access list [1−99] to a VTY line. Incoming connects are restricted using the keyword **in** and outgoing connections are restricted using the keyword **out.**

access−enable: This exec command is used to create a temporary access list entry in a dynamic access list, based on predefined criteria. The **host** keyword tells the IOS only to allow access for the particular host that originated the telnet session. The **timeout** keyword specifies an idle timeout period. If the access list entry is not used within this period of time, the access entry is deleted. If the entry is deleted, the user will have to reauthenticate in order to gain access to the network.

access−list [1−99]: An access list defined with a number ranging from 1−99 is a standard access list. A standard access list is used to permit or deny packets solely based on source IP address. The source address is the number of the network or host from which the packet is being sent. The source address is followed by a wildcard mask, which is used to specify what bit positions are ignored and what positions must match. The wildcard mask is defined in more detail later in this chapter. This is a global configuration command.

access−list [100−199]: An access list defined with a number ranging from 100−199 is an extended access list. An extended access list can be configured to be dynamic or static. Static is the default and can be changed using the keyword **dynamic**. Both static and dynamic access lists will be covered in detail later in this chapter. An extended access list is used to permit or deny packets based on multiple factors (protocol, source IP address, destination IP address, precedence, TOS, and port number), providing much more granularity then a standard access list. The extended access list permits logging, which creates an informational logging message about any packet that matches the list. This option is very useful when troubleshooting extended access lists.

access−list [100−199] [dynamic]: An access list defined with a number ranging from 100−199 using the **dynamic** keyword is a dynamic extended access list. A dynamic extended access list, also referred to as lock−and−key security, permits access on a per−user basis to a specific source/destination address through the use of authentication. This is a global configuration command.

access−template: This exec command allows you to manually place a temporary dynamic access list on the router.

autocommand: This global configuration command is used to automatically execute a command when a user connects to a particular line. This command will be used later in this chapter when configuring dynamic access lists.

clear access−list counters: This exec command clears all access list counters. Some access lists keep counters on the number of packets that match each line of a particular list. The access list name or number can be specified. By default, the system will clear all counters.

clear access−template: This exec command clears dynamic access list entries.

ip access−group: The ip access−group command is used to permit or deny incoming or outgoing packets on a particular interface based on criteria defined in the access list. This interface configuration command applies a specific access list [1−199] to a router interface. Incoming packets are filtered using the keyword **in** and outgoing connections are filtered using the keyword **out**.

ip telnet source−interface: This global configuration command allows the user to select an address of an interface as the source address for telnet connections. By default, the source address is the address of the closest interface to the destination.

periodic or absolute: In time−range configuration mode, you can specify when the function it will be applied to will be in effect. Specify some combination of these commands; multiple **periodic** statements are allowed; only one **absolute** statement is allowed.

show access−lists: This exec command displays the contents of current access lists. The access list number can be specified. By default, the system displays all access lists.

show ip access−list: This exec command displays the contents of all current IP access lists. The access list number can be specified. By default, the system displays all extended and standard access lists.

time−range: This command identifies the time range by a meaningful name.

username: This global configuration command is used to define a user−based authentication system.

IOS Requirements

Access lists first appeared in IOS 10.0; however, some of the commands described in this chapter require later IOS versions.

Lab #58: Standard IP Access Lists

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each having one Ethernet port and one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Configuration Overview

This configuration will demonstrate packet filtering using standard access lists. As per Figure 12−2, RouterA will permit all traffic from network 150.1.1.0 and deny traffic from all other networks.

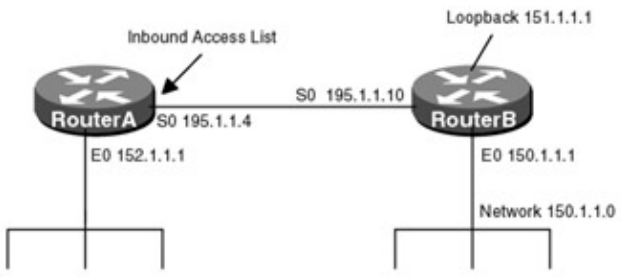


Figure 12−2: Access lists

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 12−2. RouterB has a loopback interface (IP address 151.1.1) defined to provide a test point.

An inbound access list will be applied to the serial interface of RouterA, permitting packets from network 150.1.1.0. All other packets will be denied. RouterB will ping RouterA's serial interface (195.1.1.4) using the extended **ping** command to source the packet from multiple IP addresses.

Note Access lists are a sequential collection of permit and deny statements that apply to IP addresses. The router checks addresses against the access list conditions one by one. The order of the conditions is critical because the first match in an access list is used, after which the router stops testing conditions. If no match is found, the packet is denied because of the implicit deny all at the end of each access list.

Router Configurations

The configurations for the two routers in this example are as follows (key access list configurations for RouterA are highlighted in bold).

RouterA

```
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
!interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip access−group 1 in ← Applies access list 1 to all inbound traffic on serial 0
!
no ip classless
ip route 150.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                           dynamic routing protocol is configured
ip route 151.1.1.1 255.255.255.255 Serial0 ← Static route is used because no
                                             dynamic routing protocol is
                                             configured
access−list 1 permit 150.1.1.0 0.0.0.255 ← Defines access list 1, permitting
                    Wildcard mask á        traffic from network 150.1.1.0

(Note: all other access implicitly denied) ← All access lists end with an
                                             implied deny all
!
line con 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp−small−servers
```

```
service tcp-small-servers
!
hostname RouterB
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 151.1.1.1 255.255.255.0
!
interface Ethernet0/0
 ip address 150.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0/0
 ip address 195.1.1.10 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

To test the configuration, ping RouterA (195.1.1.4) using the extended **ping** command on RouterB, source the packet from the loopback interface (151.1.1.1). To use this command, simply type in **ping** at the privileged level.

```
routerB#ping
Protocol [ip]:
Target IP address: 195.1.1.4
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 151.1.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
```

Monitor incoming packets on RouterA using the **debug ip packet** command. Below is the output from the **debug ip packet** command on RouterA. Notice that the packet is being denied and an ICMP host unreachable message is sent back to RouterB.

```
IP: s=151.1.1.1 (Serial0), d=195.1.1.4, len 100, access denied
IP: s=195.1.1.4 (local), d=151.1.1.1 (Serial0), len 56, sending ← Host
                                                                  unreachable
                                                                  message
```

Below is the output from the command **show access-list 1** on RouterA. Note that the wildcard mask permits all hosts on network 150.1.1.0.

```
RouterA#show ip access-lists 1
Standard IP access list 1
permit 150.1.1.0, wildcard bits 0.0.0.255
```

Lab #59: Extended IP Access Lists

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each having one Ethernet port and one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Configuration Overview

This configuration will demonstrate packet filtering using extended access lists. RouterA will permit all traffic from PCC (150.1.1.2) to PCA (152.1.1.2) and deny all traffic from PCC (150.1.1.2) to PCB (152.1.1.3). An extended access list is used because we are filtering on both source and destination IP addresses.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 12−3. RouterA and RouterB have secondary IP addresses defined on their Ethernet interfaces, which will be used as test points.

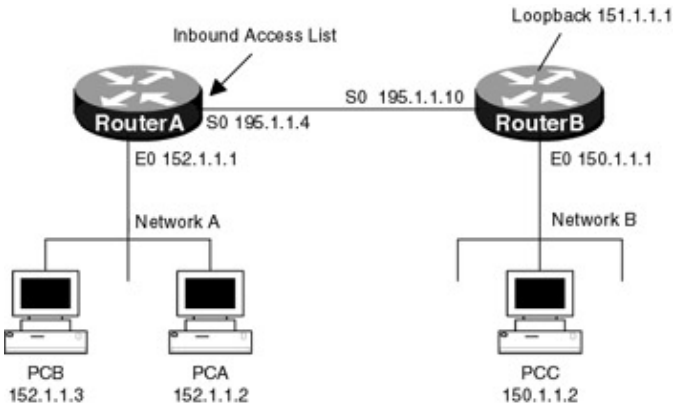


Figure 12−3: Extended IP access list

An inbound access list will be applied to the serial interface of RouterA, permitting packets from PCC 150.1.1.2 destined to PCA and denying packets from PCC to PCB.

Note When creating an access list, all entries are sequentially placed in the order of which they are entered. Any subsequent entries are placed at the end of the list. When creating access lists, it is a good idea to edit them offline and either cut and paste them to a router configuration or TFTP them from a server.

Router Configurations

The configurations for the two routers in this example are as follows (key access list configurations for RouterA are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
```

```
interface Ethernet0
ip address 152.1.1.2 255.255.255.0 secondary ← Secondary IP addresses are used
                                                as test points
 ip address 152.1.1.3 255.255.255.0 secondary
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip access-group 100 in ← Applies access list 100 to all inbound traffic on
                          serial 0
!
no ip classless
ip route 150.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                           dynamic routing protocol is
                                           configured
ip route 151.1.1.1 255.255.255.255 Serial0
                                             Host specific is the same as
                                           ↓ wildcard mask 0.0.0.0
access-list 100 permit ip host 150.1.1.2 host 152.1.1.2 log ← Generates an
                 á Permit any IP packet     informational logging message
                    from 150.1.1.2 to        about any packet that matches
                    152.1.1.2                the entry
access-list 100 deny  ip host 150.1.1.2 host 152.1.1.3 log ← Generates an
!                á Deny any IP packet       informational logging message
                    from 150.1.1.2 to        about any packet that matches
                    152.1.1.3                the entry
(Note: all other access implicitly denied) ← All access lists end with an
                                             implied deny all
!
!
line con 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
!interface Loopback0
 ip address 151.1.1.1 255.255.255.0 ← Virtual interface used as a test point
!
interface Ethernet0/0
 ip address 150.1.1.2 255.255.255.0 secondary ← Secondary IP addresses are used
                                                as test points
 ip address 150.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0/0
 clockrate 500000 ← Acts as DCE providing clock
!
no ip classless
ip route 152.1.1.0 255.255.255.0 Serial0/0 ← Static route is used because no
                                             dynamic routing protocol is
                                             configured
!
line con 0
line aux 0
line vty 0 4
```

```
 login
!
end
```

Monitoring and Testing the Configuration

The following examples all use the extended **ping** command on RouterB to source the packets from the Secondary IP addresses defined in the configuration. This is used instead of multiple PC's on RouterB's LAN.

1. From RouterB, ping 152.1.1.3 using source address 150.1.1.2.

   From the output of the **debug ip packet** command on RouterA, we see that the packet is being denied and an ICMP host unreachable message is being sent.

   ```
   IP: s=150.1.1.2 (Serial0), d=152.1.1.3, len 100, access denied
   IP: s=195.1.1.4 (local), d=150.1.1.2 (Serial0), len 56, sending ← ICMP host unreachable
   ```

   The following is the output from the **show ip access−list** command. Notice that the output shows what access lists are defined and the number of matches against each one.

   ```
   RouterA#show ip access−lists
   Extended IP access list 100
   permit ip host 150.1.1.2 host 152.1.1.2 log (5 matches)
   deny   ip host 150.1.1.2 host 152.1.1.3 log (105 matches)
   ```

   The following is the output from the **log** option, which generates an informational logging message about any packet that matches the extended access list. The logging option is a keyword, which can be added to the end of each access list statement. The information−logging message is an excellent tool to use when troubleshooting access lists.

   ```
   SEC−6−IPACCESSLOGDP: list 100 denied icmp 150.1.1.2 −> 152.1.1.3 (0/0), 4 packets
   ```

2. From RouterB, ping 152.1.1.3 using source address 150.1.1.2.

   From the output of the **debug ip packet** command on RouterA, we see that the packet is being permitted.

   ```
   IP: s=150.1.1.2 (Serial0), d=152.1.1.2, len 100, rcvd 7
   ```

   The following is the output from the **show ip access−list** command. Notice that the output shows what access lists are defined and the number of matches against each.

   ```
   RouterA#show ip access−lists
   Extended IP access list 100
       permit ip host 150.1.1.2 host 152.1.1.2 log (308 matches)
       deny   ip host 150.1.1.2 host 152.1.1.3 log
   ```

Lab #60: Extended Access List with Established Option

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable

- One Cisco rolled cable

Overview

An extended access list with the keyword **established** allows inside users to connect to the outside network while still denying outside users from accessing the inside network. This is not possible with a standard access list or an extended access list that does not use the **established** keyword.

The problem is that a standard access list and a basic extended access list deny all traffic that matches the access list criteria, even if the packet is a response. In Figure 12−4, a standard access list is applied to the serial interface of RouterA, denying all outside traffic from establishing a connection with PCA. Although this is very secure, it prevents PCA from making a connection to any server on the Internet because the response packets will be denied by the access list on RouterA.

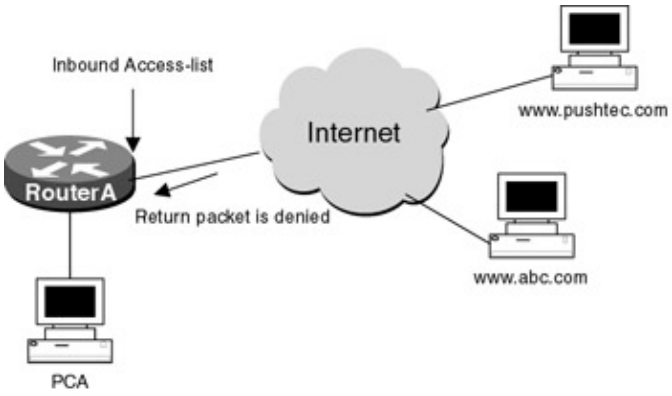


Figure 12−4: Established connections

Note Remember when creating a standard or extended access list, by default an implicit deny all is placed at the end. This means that any packet that does not match the conditions of the entries in the access list is denied.

An extended access list using the keyword **established** allows return packets from established connections to be permitted through the access list. The router looks at the TCP datagram and if the ACK or RST bits are set, the packet is permitted:

1. PCA sends a HTTP pack to server **http://www.pushtec.com/**.
2. Server pushtec.com responds.
3. RouterA applies the access list against the packet; if the return packet has the ACK or RST bit set, it is permitted.

If server **http://www.pushtec.com/** is trying to establish a connection to PCA, the SYN bit would be set and the packet would be denied. The SYNchronizing segment is the first segment sent by the TCP protocol. It is used to synchronize the two ends of a connection in preparation for opening a new connection.

Configuration Overview

This configuration will demonstrate extended access lists using the **established** keyword. The access list on RouterA will deny all nonestablished IP traffic destined for NetworkA.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 12−5. An inbound access list will be applied to the serial interface of RouterA.

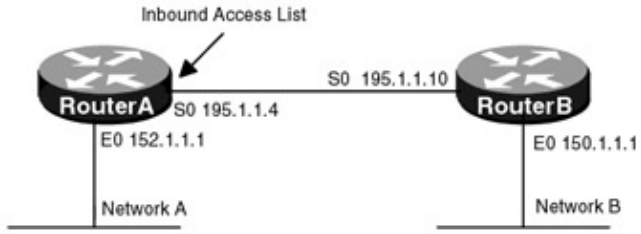


Figure 12−5: Extended access list with established option

Router Configurations

The configurations for the two routers in this example are as follows (key access list configurations for RouterA are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
ip telnet source-interface Ethernet0 ← Sources all telnet packets from E0 ip
                                       add 152.1.1.1
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip access-group 100 in ← Applies access list 100 to all inbound traffic on
                          serial
 no fair-queue
!
interface Serial1
 no ip address
 shutdown
!

ip route 150.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                           dynamic routing protocol is
                                           configured
ip route 151.1.1.1 255.255.255.255 Serial0 ← Static route is used because no
                                             dynamic routing protocol is
                                             configured
access-list 100 permit tcp any host 152.1.1.1 established log ← Permit any
                                                                traffic
                                                                destined for
                                                                152.1.1.1 with
                                                                the ACK or RST
                                                                bit set
access-list 100 deny ip any any log ← This statement is not needed—by default,
                                      all traffic is denied. However, by adding
                                      this statement we can monitor how many
                                      packets matched the access list and were
                                      denied
(Note: all other access implicitly denied) ← All access lists end with an
                                              implied deny all

!
```

```
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0
  ip address 150.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.10 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 no ip address
 shutdown
!
ip route 152.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                           dynamic routing protocol is
                                           configured
!
!
line con 0
line aux 0
line vty 0 4
 login ← Allows telnet access to the router
!
end
```

Monitoring and Testing the Configuration

To test this configuration, establish a telnet connection from RouterA to RouterB. RouterA is configured to use the IP address of the Ethernet interface to source all telnet packets. This is accomplished using the command **ip telnet source-interface e0** on RouterA:

1. Use the **debug ip packet detailed** command to monitor IP packets coming into or leaving RouterA.
2. Telnet from RouterA (152.1.1.1) to RouterB (150.1.1.1).

   The following is the output from the **debug ip packet detailed** command on RouterA. Note that the all the packets from 150.1.1.1 to 152.1.1.1 have the ACK or RST bit set.

```
IP: s=152.1.1.1 (local), d=150.1.1.1 (Serial0), len 44, sending
    TCP src=11004, dst=23, seq=682801374, ack=0, win=4288 SYN
IP: s=150.1.1.1 (Serial0), d=152.1.1.1, len 44, rcvd 4
    TCP src=23, dst=11004, seq=426675527, ack=682801375, win=2144 ACK SYN
IP: s=152.1.1.1 (local), d=150.1.1.1 (Serial0), len 40, sending
    TCP src=11004, dst=23, seq=682801375, ack=426675528, win=4288 ACK
    TCP src=11004, dst=23, seq=682801375, ack=426675528, win=4288 ACK PSH
IP: s=152.1.1.1 (local), d=150.1.1.1 (Serial0), len 40, sending
    TCP src=11004, dst=23, seq=682801384, ack=426675528, win=4288 ACK
```

```
IP: s=150.1.1.1 (Serial0), d=152.1.1.1, len 52, rcvd 4
    TCP src=23, dst=11004, seq=426675528, ack=682801375, win=2144 ACK PSH
```

The following is the output from the **log** option, which generates an informational logging message about any packet that matches the extended access list. The response packet from 150.1.1.1 matched access list 100 and was permitted.

```
%SEC-6-IPACCESSLOGP: list 100 permitted tcp 150.1.1.1(23) -> 152.1.1.1(11002),
1 packet
```

3. Use the **show ip access-list** command to display what access lists are configured and how many packets matched the criteria. Below is the output from the command. Note that six packets were permitted and one packet was denied. This command is very useful in troubleshooting access lists, allowing you to quickly see what (if any) line in the access list the packets matched.

```
RouterA#show ip access-lists
Extended IP access list 100
    permit tcp any host 152.1.1.1 established log (6 matches)
    deny   ip any any log (1 match)
```

4. Now let's try to establish a telnet connection in the opposite direction, from RouterB (150.1.1.1) to RouterA (152.1.1.1). RouterB is configured to use the IP address of the Ethernet interface to source all telnet packets.
5. Use the **debug ip packet detailed** command to monitor IP packets coming into or leaving RouterA. Below is the output from the command. Notice the SYN bit is set. The access list only permits established connections, which are determined by RouterA if the ACK or RST bit is set in the TCP header. The SYNchronizing segment is the first segment sent by the TCP protocol. It is used to synchronize the two ends of a connection in preparation for opening a new connection. The ACKnowledgement bit is set by the receiver to indicate to the sender successful reception of information. The RST bit or reset bit indicates to reset the connection.

```
IP: s=150.1.1.1 (Serial0), d=152.1.1.2, len 44, access denied
    TCP src=11004, dst=23, seq=2826185914, ack=0, win=2144 SYN
IP: s=195.1.1.4 (local), d=150.1.1.1 (Serial0), len 56, sending
    ICMP type=3, code=13
```

Lab #61: Dynamic IP Access Lists

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each with one Ethernet port and one serial port
- Cisco IOS 11.1 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Overview

Dynamic access lists, referred to as lock-and-key security, permit or deny traffic based on user authentication. For a user to gain access to a host through a router, the user must first telnet to the router and be authenticated. After the user is authenticated, a temporary access list is created allowing the user to reach the destination host.

With lock-and-key security, you specify which users are permitted to which source/destination hosts, as shown in Figure 12-6. Based on user authentication, a temporary entry is created in the access list.

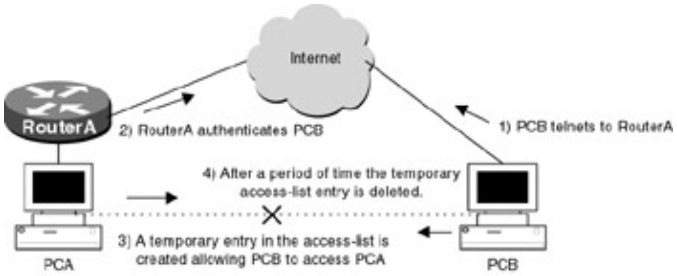


Figure 12−6: Lock−and−key security

How Lock−and−Key Works

1. PCB telnets to RouterA connecting via the virtual terminal port.
2. RouterA opens the telnet session and prompts for a username and password. If the user passes the authentication process, access through the router is allowed.
3. PCB is then automatically logged out of the telnet session and a temporary entry is created in the dynamic access list. The temporary access list entry is based on user authentication and predefined in the configuration. For example, you can configure it so that PCB only has telnet access to PCA, with all other access being denied.
4. Traffic from PCB can then pass through RouterA.
5. The router deletes the temporary access list after a predefined timeout period. Once the temporary access list is deleted, the user must be reauthenticated by RouterA before passing traffic.

Configuration Overview

This configuration, shown in Figure 12−7, will demonstrate lock−and−key security using a dynamic extended access list. RouterA will authenticate incoming telnet sessions based on username and password. When RouterB logs in to RouterA and is authenticated, a 5−minute temporary access list entry is created allowing RouterB (150.1.1.1) to telnet to RouterA (152.1.1.1).

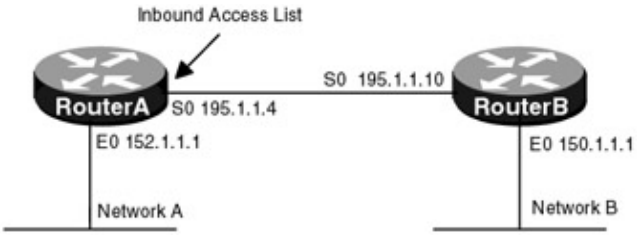


Figure 12−7: Lock−and−key security

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 12−7. An inbound access list will be applied to the serial interface of RouterA.

Router Configurations

The configurations for the two routers in this example are as follows (key access list configurations for RouterA are highlighted in bold).

RouterA

```
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
username pcb password 0 pcb ← Establish a username-based authentication system
```

```
                  Causes the specified command to be issued automatically after
               ↓ the user logs in
username pcb autocommand access-enable timeout 5 ← If the access list entry is
                           á Adds the               not accessed within 5
                             temporary entry        minutes, it is automatically
                             to access list         deleted and requires the
                                                    user to authenticate again
ip telnet source-interface Ethernet0 ← Sources all telnet packets from E0 ip
                                       add 152.1.1.1
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip access-group 100 in ← Applies access list 100 to all inbound traffic on
                          serial 0
 no fair-queue
!
interface Serial1
 no ip address
 shutdown
!

no ip classless
ip route 150.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                           dynamic routing protocol is
                                           configured
                                ↓ Dynamic name
access-list 100 dynamic tempaccess permit tcp host 150.1.1.1 host 152.1.1.1 eq telnet log
                                           á Only permit telnet
                                                       traffic between
                                                       150.1.1.1 and 152.1.1.1
access-list 100 permit tcp any host 195.1.1.4 eq telnet log ← Telnet access
                                                              must be allowed
                                                              to enable user
                                                              authentication
access-list 100 deny ip any any log ← This statement is not needed—by default,
                                      all traffic is denied. However, by adding
                                      this statement we can monitor how many
                                      packets matched the access list and were
                                      denied
(Note: all other access implicitly denied) ← All access-lists end with an
                                              implied deny all
!
line con 0
line aux 0
line vty 0 4
  login local ← Enables local password checking at login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0
  ip address 150.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
```

```
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.10 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 no ip address
 shutdown
!
ip route 152.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                            dynamic routing protocol is
                                            configured

!
line con 0
line aux 0
line vty 0 4
login ← Allows telnet access to the route
!
end
```

Monitoring and Testing the Configuration

To test this configuration, establish a telnet connection from RouterB to RouterA (195.1.1.4). Log into RouterA with username pcb and password pcb. The following sample display is what users will see if they are authenticated. Notice that the telnet connection is closed immediately after the password is entered. After authentication, RouterA creates a temporary entry in access list 100.

```
RouterB#telnet 195.1.1.4
Trying 195.1.1.4 ... Open
User Access Verification
Username: pcb
Password: pcb
[Connection to 195.1.1.4 closed by foreign host]
```

The following sample is from the **show ip access−list** command on RouterA. Notice that the temporary entry has been added to the access list 100.

```
RouterA#show ip access−lists
Extended IP access list 100
    Dynamic tempaccess permit tcp host 150.1.1.1 host 152.1.1.1 eq telnet log
       permit tcp host 150.1.1.1 host 152.1.1.1 eq telnet log idle−time 5 min.
    permit tcp any host 195.1.1.4 eq telnet log (72 matches)
    deny   ip any any log (1 match)
```

Establish a telnet connection from RouterB (150.1.1.1) to 152.1.1.1. RouterB is configured to use the IP address of the Ethernet interface to source all telnet packets. This is accomplished using the command **ip telnet source−interface e0** on RouterA.

The following is a sample of the output from the access list log command on RouterA. The log command is an option, which is added to the access list during configuration. The log keyword generates an informational logging message about any packet that matches the extended access list. The telnet packet from 150.1.1.1 matched the temporary entry added to access list 100.

```
%SEC−6−IPACCESSLOGP: list 100 permitted tcp 150.1.1.1(11010) −> 152.1.1.1(23), 1 packet
```

The telnet session was permitted. Now, on RouterA, remove the temporary entry from the access list with the exec command:

```
clear access−template 100 tempaccess 150.1.1.1 0.0.0.0 152.1.1.1 0.0.0
```

Use the **show ip access−list** command to display what access lists are configured and how many packets matched the criteria. Below is the output from the command. Notice that the temporary entry has been removed from the dynamic access list.

```
RouterA#show ip access−lists
Extended IP access list 100
    Dynamic tempaccess permit tcp host 150.1.1.1 host 152.1.1.1 eq telnet log
    permit tcp any host 195.1.1.4 eq telnet log (124 matches)
    deny   ip any any log (1 match)
```

Telnet from RouterB (150.1.1.1) to 152.1.1.1. Notice that this time the connection has failed. PCB will have to reauthenticate with RouterA in order to gain telnet access to 152.1.1.1.

```
RouterB#telnet 152.1.1.1
Trying 152.1.1.1 ...
% Destination unreachable; gateway or host down
```

Lab #62: Controlling VTY Access

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each with one Ethernet port and one serial port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Overview

This lab demonstrates using access lists to control VTY connections to the router. In a production environment, it is imperative to limit router access to authorized personnel. This can be accomplished using password authentication and access control lists.

Access control lists allow you to specify what stations are able to gain telnet access to your router based on source IP address. Router access should be limited to specific workstations. It is good design to set up a bastion host and only allow that specific IP address VTY access to all of your routers. Network administrators telnet to the bastion host and then out to the specific router. A bastion host is a computer that forms part of a security firewall and runs applications that communicate with computers outside an organization.

Configuration Overview

This configuration will demonstrate controlling VTY access to a router using a standard access list. RouterA will only allow VTY access from host 150.1.1.1 (RouterB's Ethernet interface). All other VTY sessions will be denied.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 12−8. An inbound access list will be applied to the serial interface of RouterA.

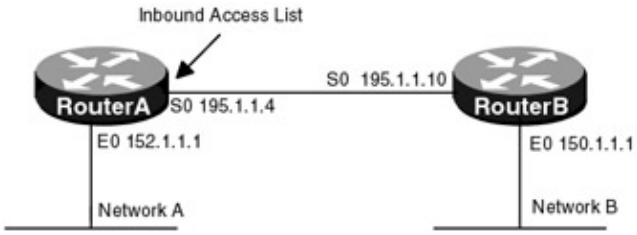


Figure 12−8: VTY access control

Router Configurations

The configurations for the two routers in this example are as follows (key access list configurations for RouterA are highlighted in bold).

RouterA

```
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Ethernet0
  ip address 152.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 no fair-queue
!
interface Serial1
 no ip address
 shutdown
!
no ip classless
ip route 150.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                           dynamic routing protocol is
                                           configured
access-list 1 permit 150.1.1.1 ← Standard access list permits access from host
                                  150.1.1.1
(Note: all other access implicitly denied) ← All access lists end with an
                                             implied deny all
!
line con 0
line aux 0
line vty 0 4
 access-class 1 in ← Applies standard access list 1 to all inbound VTY
                     connections
 password cisco
 login
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
ip telnet source-interface Ethernet0
!
```

```
interface Ethernet0
  ip address 150.1.1.1 255.255.255.0
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 195.1.1.10 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 no ip address
 shutdown
!
ip route 152.1.1.0 255.255.255.0 Serial0 ← Static route is used because no
                                           dynamic routing protocol is
                                           configured
!
line con 0
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

To test this configuration, establish a telnet connection from RouterB to RouterA (195.1.1.4). The source address of the telnet packet is 150.1.1.1, which is defined on RouterB using the command **ip telnet source−interface Ethernet0**.

The following is a sample of what the user will see if the telnet connection is successful:

```
RouterB#telnet 195.1.1.4
Trying 195.1.1.4 ... Open

User Access Verification
Password:
```

On RouterB, edit the configuration so that all telnet packets are sourced from the serial interface. To do this, perform the following in global configuration mode:

```
RouterA(config)#
no ip telnet source−interface Ethernet0
ip telnet source−interface serial0
```

Now, from RouterB, telnet to RouterA (195.1.1.4). The connection is refused because the source address of the telnet packet does not match the access control list on RouterA.

```
RouterB#telnet 152.1.1.1
Trying 152.1.1.1 ...
% Connection refused by remote host
```

Lab #63: Time−of−Day Access Lists

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each with one serial port

- Cisco IOS 12.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- One Cisco rolled cable

Configuration Overview

This lab demonstrates using time−of−day access lists to control VTY connections to the router. In a production environment, it is imperative to limit router access to authorized personnel. This can be accomplished using password authentication and access control lists.

Time−of−day access lists, available in IOS 12.0, allow you to control access, policy routing, DDR, CAR, and IOS Firewall by the time of day. The time range allows the network administrator to define when the permit or deny statements in the access list are in effect. Prior to this feature, access list statements were always in effect once they were applied. This new feature provides the administrator with greater flexibility in controlling resources.

To enable time−of−day access lists, you create a time range that defines specific times of the day and week. This range, which is identified by name, can then be referenced by a function (such as CAR), so that those time restrictions are imposed on the function itself.

This configuration will demonstrate controlling VTY access to a router using a time−of−day extended access list. RouterB will only allow VTY access from host 152.1.2.1 from 9:00 to 17:00 Monday through Friday.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 12−9. An inbound access class list will be applied to the VTY lines of RouterB.

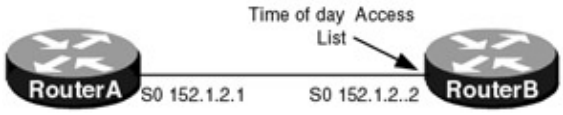


Figure 12−9: Time−of−day access list

Router Configurations

The configurations for the two routers in this example are as follows (key access list configurations for RouterB are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Serial0
 ip address 152.1.2.1 255.255.255.252
!
!
line con 0
line 1 16
 line aux 0
line vty 0
!
end
```

RouterB

```
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
username cisco  password 0 cisco
!
ip subnet-zero
no ip domain-lookup
!
interface Serial0/0
 ip address 152.1.2.2 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 1000000 ← Acts as DCE providing clock
!
ip classless
no ip http server
!
access-list 101 permit tcp any any eq telnet time-range deny_user ← Extended
                                                                    access list
                                                                    101 permits
                                                                    telnet
                                                                    access from
                                                                    9:00 to
                                                                    17:00 on
                                                                    weekdays
!
line con 0
 transport input none
line aux 0
line vty 0 4
 access-class 101 in ← Applies extended access list 101 to all inbound VTY
                       connections
 login local
!
time-range deny_user ← Defines a time range called deny_user
  periodic weekdays 9:00 to 17:00 ← Defines a period of time
!
end
```

Monitoring and Testing the Configuration

The time-of-day access list works off the system clock, so the correct time and date need to be configured on the router. To do this, use the **clock set** command. The following command sets the system clock on RouterB:

```
RouterB#clock set 15:56:00 20 december 2000
```

Use the **show clock** command on RouterB to verify that the time and date are set correctly. Below is the output from the command on RouterB:

```
RouterB#show clock
15:58:37.987 UTC Wed Dec 20 2000
```

Display access list 101 on RouterB with the command **show access-list 101**. Below is the output from the command. Notice that the access list is active. This is because the system time is 15:58:37.987 UTC Wed Dec 20 2000, which falls within the time range specified.

```
RouterB#show access-lists 101
```

```
Extended IP access list 101
    permit tcp any any eq telnet time-range deny_user (active) (2 matches)
```

Now try to telnet to RouterB from RouterA, with the command **telnet 152.1.2.2**. Below is the output from the command. Notice that the telnet was successful.

```
RouterA#telnet 152.1.2.2
Trying 152.1.2.2 ... Open

 User Access Verification

 Username:
```

Now change the system clock on RouterB to 18:00.

```
RouterB#clock set 18:00:00 20 December 2000
```

Display access list 101 on RouterB, with the command **show access-list 101**. Below is the output from the command. Notice that the access list is now inactive. This is because the system time does not fall within the time range specified.

```
RouterB#show access-lists 101
Extended IP access list 101
    permit tcp any any eq telnet time-range deny_user (inactive) (4 matches)
```

Now try to telnet to RouterB from RouterA with the command **telnet 152.1.2.2**. Below is the output from the command. Notice that the telnet is not successful.

```
RouterA#telnet 152.1.2.2
Trying 152.1.2.2 ...
% Connection refused by remote host
```

Troubleshooting IP Access Lists

The Cisco IOS provides many tools for troubleshooting access lists. Below is a list of key commands along with sample output from each.

{show access-lists} This privileged exec command displays the contents of all current access lists. The command can also display specific access lists by number.

```
RouterA# show access-lists 1
Standard IP access list 1
    permit 150.1.1.1
```

{show ip access-list} This privileged exec command displays the contents of all current IP access lists. The command can also display specific access lists by number.

```
RouterA#show ip access-lists 1
Standard IP access list 1
permit 150.1.1.0, wildcard bits 0.0.0.255
```

{clear access list counters} Some access lists keep counters that count the number of packets that pass each line of an access list. The **show access-lists** command displays the counters as a number of matches. Use the **clear access-list counters** command to reset the counters for a particular access list to 0.

```
RouterA#clear access-list counters
```

{debug ip packet} This exec command displays information about packets that are received, generated, or forwarded by the router. When access lists are applied to the router, the output from the debug command will tell you whether or not the packet was permitted or denied.

```
RouterA#Debug ip packet
IP: s=150.1.1.2 (Serial0), d=152.1.1.3, len 100, access denied
IP: s=195.1.1.4 (local), d=150.1.1.2 (Serial0), len 56, sending
```

Conclusion

Access lists are an integral part of the Cisco IOS, allowing the router to make decisions based on defined criteria. This chapter explores filtering packets and controlling access to the router using standard and extended access lists. However, access lists are used for many other things not covered in this chapter, such as filtering router updates, determining DDR interesting traffic, and flow control, to name a few. These advanced topics will be covered in their respective chapters.

Chapter 13: Policy−based Routing

Overview

Topics Covered in This Chapter

- Detailed policy routing overview
- Policy routing terminology
- Policy routing based on source IP address
- Policy routing based on packet size
- Policy routing based on application
- Load balancing across default routes
- Detailed troubleshooting examples

Introduction

Policy−based routing affords network administrators increased control over packet forwarding and routing that goes beyond the capabilities of traditional routing protocols. Traditionally, routers forward packets according to destination addresses using routing tables derived from routing protocols such as OSPF or RIP.

Policy−based routing goes far beyond traditional routing, allowing network administrators to select forwarding paths not only according to destination addresses but to protocol type, packet size, application, or source address. Policies can be defined to load−balance traffic across multiple routers or provide Quality of Service (QOS) by forwarding packets over various links according to traffic profiles. Policy−based routing provides the network administrator with a mechanism to specify what path a packet will take. This freedom is greatly needed in today's high−performance and complex internetworks.

This chapter will explore controlling packet forwarding through a router, using policy−based routing.

Policy Routing Overview

Policy routing provides a mechanism for forwarding packets based on criteria defined by the network administrator. Policy−based routing uses **match** and **set** clauses to achieve path selection.

In Figure 13−1, all traffic from PCA destined for the order entry server will go over the ISDN link and all other traffic from PCA as well as other nodes will be routed over the 512K leased line.

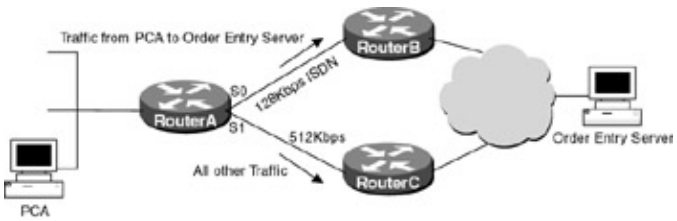


Figure 13−1: Policy−based routing

RouterA's policy would be quite simple: Any traffic from PCA to the order entry system will have the next hop set to RouterB. All other traffic will not match the criteria and will therefore be routed according to the entries in the routing table. The routing table will use the 512 Kbps link because the routers are using OSPF as their routing protocol. Since OSPF uses bandwidth as its metric for selecting the best path, the 512 Kbps link will be preferred over the 128 Kbps ISDN link.

Note Policy routing is set on the interface that receives the packet, not on the interface that the packet is sent out. In Figure 13−1, the policy is applied to RouterA's Ethernet interface.

Policy Routing Terminology

Match Clause: The match clause evaluates packets on the ingress router port. The match criteria can be as simple as matching source and destination addresses using an ACL (access control list) or as complex as matching packet sizes according to specified minimum and maximum packet lengths.

Multiple match clauses can be used to provide even more granularity in route selection. The match clauses are serviced in order, and all must match for the set clause to be applied. If no match is found or if the policy is made to deny instead of permit, then the packet is routed according to the destination address using an entry from the routing table.

Set Clause: The set clause defines the route that the packet will transverse if the match clause is met. Multiple set clauses can be used and are evaluated in the order that follows; if the first one is not available (interface is down), then the next one is tried until the end of the list is reached.

1. Next hop interfaces
2. Next hop IP address
3. Next hop default interfaces
4. Next hop default IP address

If the end of the list is reached and no set clause is applied, then the packet is routed through the normal destination−based routing process.

Note Packet forwarding based on policy routing will override packet forwarding based on routing table entries to the same destination.

Commands Discussed in This Chapter

- **ip local policy route−map** *map−tag*
- **ip policy route−map** [*map−tag*]
- **match ip address** {*access−list−number* | *name*} [*access−list−number*]
- **match length** *min max*
- **route−map** *map−tag* [permit | deny] [*sequence−number*]
- **set default interface** *type number*
- **set interface** *type number*
- **set ip default next−hop** *ip−address*
- **set ip next−hop** *ip−address* [. . . *ip−address*]
- **show ip policy**
- **show route−map command**

Definitions

ip local policy route−map: Packets that are generated by the router are not normally policy−routed. This global configuration command enables the router to policy−route locally generated packets.

ip policy route−map: This interface configuration command enables policy routing on a particular interface and identifies the route map that will be applied to the packet.

match ip address: This route map configuration command bases the policy match condition on a predefined access list.

match length: This route map configuration command bases the policy match condition on layer three packet length. This is often used to direct interactive traffic, which tends to mean routing small−sized packets over different links than large−sized bulk traffic.

route−map: This global configuration command defines the name of the route map and whether the packet matching the criteria is policy−routed or not. If the match criteria are met for the route map and the permit key word is specified, the packet is policy−routed. If the match criteria are met and the deny key word is specified, then the packet is not policy−routed. Several route maps can be defined using the same name. An optional number can be placed at the end of the route map indicating the order in which the route map will be checked. In this example, two route maps are defined with the name "lab1". One is defined with the item number 10, and the other with the item number 20. An incoming packet is checked against item 10 of route map lab1; if the packet does not match the IP address, then the packet is checked against item 20.

```
route−map lab1 permit 10 ← Item 10 of route−map lab1
match ip address 1
set interface Serial0
!
route−map lab1 permit 20 ← Item 20 of route−map lab1
match ip address 2
set interface Serial1
```

set default interface: This route−map configuration command indicates what default interface packets that pass the match clause are sent to. This command gives certain packets a different default route. If the router has no explicit route in the routing table for the destination address, then it will forward the packets out the set default interface.

set interface: This route map configuration command indicates to what output interface packets that pass the match clause are sent. Multiple output interfaces can be configured; if the first interface is down, then the optionally specified interface is tried in turn.

set ip default next−hop: This route map configuration command indicates to what default next−hop packets that pass the match clause are sent. If the router has no explicit route for the destination, then it will route the packet to the set default next hop. This is often used to load−balance between two different service providers. When using the default next hop set command, the routing table will always be used first to route the packet. If an explicit route does not exist in the routing table, then the packet is forwarded using the policy set default.

set ip next−hop: This route map configuration command indicates the next hop IP address that output packets that pass the match clause are forwarded to. Multiple next hop addresses can be configured; if the first next hop specified is down, the optionally specified IP addresses are tried in turn.

show ip policy: This exec command displays what policies are applied to what interfaces.

show route−map: This exec command displays the match and set conditions for all configured route maps. The command can also be used to display a specific route map, by specifying the route map name after the command.

> Note Multiple match commands can be used, but all match commands must "pass" to cause the packet to be routed according to the set actions given with the set commands.

IOS Requirements

Policy−based routing first appeared in IOS 11.0; however, IOS 11.2 was used for all configurations.

Lab #64: Policy Routing Based on Source IP address

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each having one Ethernet port and two serial ports
- Cisco IOS 11.0 or higher
- A PC running a terminal emulation program for connecting to the console port of the routers
- Two Cisco DTE/DCE crossover cables
- A Cisco rolled cable

Configuration Overview

This configuration will demonstrate routing packets by source IP address using policy−based routing. As per Figure 13−2, RouterA will route all traffic from 192.1.1.1 out interface S0 and all traffic from 192.1.1.2 out interface S1.

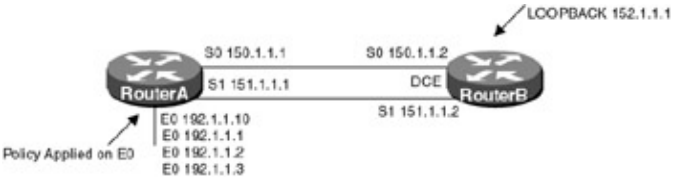


Figure 13−2: Source address policy routing

RouterA and RouterB are connected serially via two crossover cables. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 13−2. RouterB has a loopback interface (IP address 152.1.1.1) defined to provide a test point. RouterA has multiple secondary interfaces defined on Ethernet 0, which will also be used as test points. Both RouterA and RouterB will be configured for RIP.

The IP policy route−map **lab1** will be applied to the Ethernet interface of RouterA, setting the next hop interface to S0 for packets coming from 192.1.1.1 and the next hop interface to S1 for packets coming from 192.1.1.2. All other packets will be routed using the normal destination−based routing process.

Router Configurations

The configurations for the two routers in this example are as follows (key policy routing configurations are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Ethernet0
 ip address 192.1.1.1 255.255.255.0 secondary ← Secondary IP addresses are used
                                                as test points
 ip address 192.1.1.2 255.255.255.0 secondary
 ip address 192.1.1.3 255.255.255.0 secondary
 ip address 192.1.1.10 255.255.255.0
 ip policy route-map lab1 ← Enables policy routing on interface E0 and
                            identifies the route map lab1, which will be
                            applied to all incoming packets
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
```

```
                interface to stay up when it is not attached to a hub

!
interface Serial0
 ip address 150.1.1.1 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 151.1.1.1 255.255.255.0
!
router rip ← Enables the RIP routing process on the router
 network 192.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
 network 150.1.0.0
 network 151.1.0.0
!
ip local policy route-map lab1 ← Packets that are generated by the router are
                                 not normally policy-routed. This command
                                 enables the router to policy-route packets
                                 that are sourced from the router
no ip classless
access-list 1 permit 192.1.1.1 ← Access list defines what source IP address of
                                 192.1.1.1
access-list 2 permit 192.1.1.2
route-map lab1 permit 10 ← Defines the route map lab1, the number specifies the
                           order of the route maps. This is referred to as item
                           10 of route map lab1

 match ip address 1 ← The match criteria is the IP address from access list 1
 set interface Serial0 ← The set clause sets the next hop interface to S0
!
route-map lab1 permit 20 ← Defines the route map lab1, the number specifies the
                           order of the route maps. This is referred to as item
                           20 of route map lab1

 match ip address 2 ← The match criteria is the IP address from access list 2
 set interface Serial1 ← The set clause sets the next hop interface to S1
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 152.1.1.1 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 150.1.1.2 255.255.255.0
 clockrate 500000
!
interface Serial1
 ip address 151.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as the DCE providing clock
```

```
!
router rip ← Enables the RIP routing process on the router

 network 152.1.0.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
 network 151.1.0.0
!
!
line con 0
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

To test the configuration, trace the route to 152.1.1.1 from RouterA using the extended traceroute command to source the packets from 192.1.1.1. To use this command, simply type **traceroute ip** at the privileged router prompt.

```
RouterA#traceroute ip

Target IP address: 152.1.1.1
Source address: 192.1.1.2
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 152.1.1.1
```

What follows is the output from the traceroute command. Note that the packet was sent out RouterA's S0 interface connected to 150.1.1.2.

```
RouterA#
Tracing the route to 152.1.1.1

  1 150.1.1.2 8 msec 8 msec *
```

Now perform another traceroute sourcing the packet from 192.1.1.2.

```
RouterA#
Tracing the route to 152.1.1.1

  1 151.1.1.2 8 msec 8 msec *
```

Notice that the packet was sent out interface S1, which is exactly how the policy is defined.

On RouterA monitor the policy routing with the **debug ip policy** privileged command. From RouterA, ping 152.1.1.1 using the extended ping command to source the packet from 192.1.1.1. To use this command, simply type **ping** at the privileged level.

```
routerB#ping
Protocol [ip]:
```

```
Target IP address: 152.1.1.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.1.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
```

What follows is the output from the debug command; notice that the packet matched item10 of route map **lab1** and was forwarded via interface S0.

```
RouterA#
IP: s=192.1.1.1 (local), d=152.1.1.1, len 100, policy match
IP: route map lab1, item 10, permit
IP: s=192.1.1.1 (local), d=152.1.1.1 (Serial0), len 100, policy routed
IP: local to Serial0 152.1.1.1
```

From RouterA, ping 152.1.1.1 now, sourcing the packet from 192.1.1.2. What follows is the output from the debug command; notice that packet matched item 20 of route map **lab1** and was forwarded via interface S1.

```
RouterA#
IP: s=192.1.1.2 (local), d=152.1.1.1, len 100, policy match
IP: route map lab1, item 20, permit
IP: s=192.1.1.2 (local), d=152.1.1.1 (Serial1), len 100, policy routed
IP: local to Serial1 151.1.1.2
```

From RouterA, ping 152.1.1.1, sourcing the packet from 192.1.1.3, which is not defined in the match criteria. What follows is the output from the debug command; note that the match criteria were not met, so the router forwarded the packet using normal destination−based routing process.

```
RouterA#
IP: s=192.1.1.10 (local), d=255.255.255.255 (Ethernet0), len 92, policy rejected
                                                                 normal
                                                                 forwarding
```

Now on RouterA remove the set clause for item 10 of route map lab1, with the following commands:

```
RouterA(config)#route−map lab1 permit 10
RouterA(config−route−map)#no set interface Serial0
```

From RouterA, ping 152.1.1.1, sourcing the packet from 192.1.1.1. What follows is the output from the debug command; note that source address matches item 10 of route−map lab1. Since no set policy is defined, however, the packet is rejected and the router forwards the packet normally.

```
RouterA#
IP: s=192.1.1.1 (local), d=152.1.1.1, len 100, policy match
IP: route map lab1, item 10, permit
IP: s=192.1.1.1 (local), d=152.1.1.1, len 100, policy rejected − normal
                                                                 forwarding
```

Lab #65: Policy Routing Basedon Packet Size

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each having one Ethernet port and two serial ports
- Cisco IOS 11.0 or higher
- A PC running a terminal emulation program for connecting to the console port of the routers
- Two Cisco DTE/DCE crossover cables
- A Cisco rolled cable

Configuration Overview

This configuration will demonstrate routing packets based on packet size using policy−based routing. As per Figure 13−3, RouterA will route all traffic with a datagram size from 64 to 100 bytes out interface S0 and all packets with a datagram size of 101 to 1,000 bytes out interface S1; all other packet sizes will be routed normally.

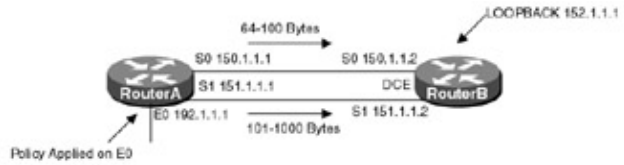


Figure 13−3: Packet size policy routing

RouterA and RouterB are connected serially via crossover cables; RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 13−3. RouterB has a loopback interface (IP address 152.1.1.1) defined to provide a test point. Both RouterA and RouterB will be configured using RIP as their routing protocol.

The IP policy route−map **lab1** will be applied to the Ethernet interface of RouterA. This policy will set the next hop IP address to 150.1.1.2 for packets with a datagram size from 64 to 100 bytes and the next hop IP address to 151.1.1.2 for packets coming with a datagram size from 101 to 1,000 bytes. All other packets will be routed through the normal destination−based routing process.

Router Configurations

The configurations for the two routers in this example are as follows (key policy routing configurations are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Ethernet0
 ip address 192.1.1.1 255.255.255.0
 ip policy route-map lab1 ← Enables policy routing on interface E0 and
                            identifies the route map lab1, which will be
                            applied to the packet
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub

!
interface Serial0
```

```
 ip address 150.1.1.1 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 151.1.1.1 255.255.255.0
!
router rip ← Enables the RIP routing process on the router
 network 192.1.1.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
 network 150.1.0.0
 network 151.1.0.0
!
ip local policy route-map lab1 ← Packets that are generated by the router are
                                 not normally policy-routed. This command
                                 enables the router to policy-route packets
                                 that are sourced from the router
no ip classless
route-map lab1 permit 10 ← Defines the rout map lab1, the number specifies the
                           order of the route maps. This is referred to as item
                           10 of route map lab1

match length 3 100 ← The match criteria is the packet length
set ip next-hop 150.1.1.2 ← The set clause sets the next hop IP address
!
route-map lab1 permit 20 ← Defines the route map lab1 the number specifies the
                           order of the route maps. This is referred to as item
                           20 of route map lab1
match length 101 1000 ← The match criteria is the packet length
set ip next-hop 151.1.1.2 ← The set clause sets the next hop IP address
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname routerb
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 152.1.1.1 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 150.1.1.2 255.255.255.0
 clockrate 500000
!
interface Serial1
 ip address 151.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
router rip ← Enables the RIP routing process on the router

 network 152.1.0.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
```

```
 network 151.1.0.0
 network 101.0.0.0
!
!
line con 0
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterA, monitor the policy routing using the **debug ip policy** command. Using the extended ping command on RouterA change the ping packet size to 64 bytes and ping 152.1.1.1. What follows is the output from the debug command; note that the 64−byte packet matched item 10 of route map lab1 and was forwarded to 150.1.1.2.

```
RouterA#
IP: s=151.1.1.1 (local), d=152.1.1.1, len 64, policy match
IP: route map lab1, item 10, permit
IP: s=151.1.1.1 (local), d=152.1.1.1 (Serial0), len 64, policy routed
IP: local to Serial0 150.1.1.2
```

From RouterA, ping 152.1.1.1, increasing the packet size to 101 bytes; this packet will match item 20 of route map lab1. What follows is the output from the debug command; note that the 101−byte packet matches item 20 and was forwarded to 151.1.1.2.

```
RouterA#
IP: s=151.1.1.1 (local), d=152.1.1.1, len 101, policy match
IP: route map lab1, item 20, permit
IP: s=151.1.1.1 (local), d=152.1.1.1 (Serial1), len 101, policy routed
IP: local to Serial1 151.1.1.2
```

From RouterA, ping 152.1.1.1, increasing the packet size to 1,001 bytes; this packet will not match any items in route map lab1. What follows is the output from the debug command; note that the 1,001−byte packet does not match any clauses, so the router forwarded the packet normally.

```
RouterA#
IP: s=151.1.1.1 (local), d=152.1.1.1, len 1001, policy rejected − normal forwarding
IP: s=151.1.1.1 (local), d=152.1.1.1, len 1001, policy rejected  − normal forwarding
```

Lab #66: Policy Routing Based on Application

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each having one Ethernet port and two serial port
- Cisco IOS 11.0 or higher
- A PC running a terminal emulation program for connecting to the router console port
- Two Cisco DTE/DCE crossover cable
- Cisco rolled cable

Configuration Overview

This configuration will demonstrate routing packets based on source and destination port number (FTP, WWW) using policy−based routing. As per Figure 13−4, RouterA will route all Web traffic (port 80) out interface S0 and all telnet traffic (port 23) out interface S1; all other packet types will be routed normally.

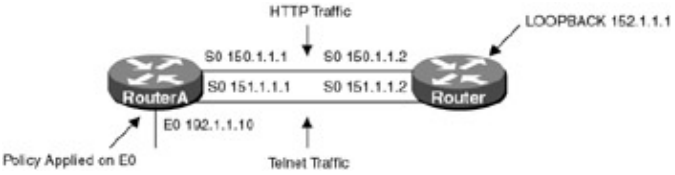


Figure 13−4: Application−level policy routing

RouterA and RouterB are connected serially via crossover cables; RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 13−4. RouterB has a loopback interface (IP address 152.1.1) defined to provide a test point.

The IP policy route−map **lab1** will be applied to the Ethernet interface of RouterA, setting the next hop IP address to 150.1.1.2 for HTTP traffic and the next hop IP address to 151.1.1.2 for telnet traffic. All other packets will be routed through the normal destination−based routing process.

Router Configurations

The configurations for the two routers in this example are as follows (key policy routing configurations are highlighted in bold).

RouterA

```
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterA
!

interface Ethernet0
 ip address 192.1.1.1 255.255.255.0
  ip policy route-map lab1 ← Enables policy routing on interface E0 and
                              identifies the route map lab1, which will be
                              applied to the packet
 no keepalive ← Disables the keepalive on the Ethernet interface, allows the
                interface to stay up when it is not attached to a hub
!
interface Serial0
 ip address 150.1.1.1 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 151.1.1.1 255.255.255.0
!
router rip ← Enables the RIP routing process on the router
 network 192.1.1.0 ← Specifies what interfaces will receive and send RIP
                      routing updates. It also specifies what networks will be
                      advertised
 network 150.1.0.0
 network 151.1.0.0
!
ip local policy route-map lab1 ← Packets that are generated by the router are
                                  not normally policy-routed. This command
                                  enables the router to policy-route packets
                                  that are sourced from the router
```

```
no ip classless
access-list 101 permit tcp any any eq www ← Access list 101 sets the match
                                              criteria to www traffic
access-list 102 permit tcp any any eq telnet ← Access list 102 sets the match
                                                 criteria to telnet traffic
route-map lab1 permit 10 ← Defines the route map lab1, the number specifies the
                           order of the route maps. This is referred to as item
                           10 of route map lab1

 match ip address 101 ← This defines the match criteria tied to access list 101
 set ip next-hop 150.1.1.2 ← Sets the next hop IP address
!
route-map lab1 permit 20 ← Defines the route map lab1, the number specifies the
                           order of the route maps.  This is referred to as
                           item 20 of route map lab1

match ip address 102 ← This defines the match criteria tied to access list 102
set ip next-hop 151.1.1.2 ← Sets the next hop IP address
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname routerb
interface Loopback0 ← Defines a virtual interface that will be used as a test
                      point
 ip address 152.1.1.1 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 150.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
interface Serial1
 ip address 151.1.1.2 255.255.255.0
 clockrate 500000  ← Acts as DCE providing clock
!
router rip ← Enables the RIP routing process on the router

 network 152.1.0.0 ← Specifies what interfaces will receive and send RIP
                     routing updates. It also specifies what networks will be
                     advertised
 network 151.1.0.0
!
line con 0
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

From RouterA, monitor the policy routing using the **debug ip policy** command. Telnet from RouterA to 152.1.1.1. What follows is the output from the debug command; note that the telnet packet matched item 20 of route map lab1 and was forwarded to 151.1.1.2.

```
IP: s=151.1.1.1 (local), d=152.1.1.1, len 44, policy match
IP: route map lab1, item 20, permit
IP: s=151.1.1.1 (local), d=152.1.1.1 (Serial1), len 44, policy routed
IP: local to Serial1 151.1.1.2
```

From RouterA, use the extended telnet command to send a HTTP packet to 152.1.1.2. To use this command, simply type in **Telnet 152.1.1.1 www** at the privileged level. What follows is the output from the debug command; note that the HTTP packet matched item 10 of route map lab1 and was forwarded to 150.1.1.2.

```
IP: s=151.1.1.1 (local), d=152.1.1.1, len 44, policy match
IP: route map lab1, item 10, permit
IP: s=151.1.1.1 (local), d=152.1.1.1 (Serial0), len 44, policy routed
IP: local to Serial0 150.1.1.2
```

Lab #67: Load Balancing Across Default Routes

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers each having one Ethernet port and two serial ports
- One Cisco router with one Ethernet port
- Cisco IOS 11.0 or higher
- A PC running a terminal emulation program for connecting to the console port of the routers
- Two Cisco DTE/DCE cross over cables
- Cisco rolled cable
- One Ethernet crossover cable or an Ethernet hub and two straight−through Ethernet cables

Configuration Overview

This configuration provides two end users with equal access to two different service providers. As per Figure 13−5, RouterA will route packets arriving on Ethernet 0 from the source 192.1.1.11 to default interface S0 if no explicit route for the packets' destination is in the routing table. Packets arriving from 192.1.1.12 are sent to default interface S1 if the router has no explicit route for the packets destination.

Figure 13−5: Load balancing across default routes

This lab uses the default interface command, which differs from the next hop interface and next hop IP address set commands we used in previous labs. The next hop set commands send the matching packet out that interface or to that IP address regardless of the routing table. The default interface command only sends the packet out that particular interface if there is no explicit route in the routing table.

Note When using the default interface set command, the router will first check the routing table for an explicit route. If there is no explicit route available to the destination address of the packet being considered for policy routing, then the router will route the packet out the default interface.

Router Configurations

The configurations for the two routers in this example are as follows (key policy routing configurations are highlighted in bold).

RouterA

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname RouterA
!
!
interface Ethernet0
ip address 192.1.1.10 255.255.255.0
 ip policy route-map lab1 ← Enables policy routing on interface E0 and
                            identifies the route map lab1, which will be
                            applied to the packet
!
interface Serial0
 ip address 150.1.1.1 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 151.1.1.1 255.255.255.0
!
router rip
 network 150.1.0.0
 network 151.1.0.0
 network 192.1.1.0
!
no ip classless
access-list 1 permit 192.1.1.11
access-list 2 permit 192.1.1.12
route-map lab1 permit 10 ← Defines the route map lab1, the number specifies the
                           order of the route maps. This is referred to as item
                           10 of route map lab1
 match ip address 1 ← This defines the match criteria tied to access list 1
 set default interface Serial0 ← Sets the default interface to S0
!
route-map lab1 permit 20 ← Defines the route map lab1, the number specifies the
                           order of the route maps. This is referred to as item
                           10 of route map lab1
 match ip address 2 ← This defines the match criteria tied to access list 1
 set default interface Serial1 ← Sets the default interface to S0
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname routerb
!
!
!
```

```
interface Loopback0
 ip address 152.1.1.1 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 150.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock

!
interface Serial1
 ip address 151.1.1.2 255.255.255.0
 clockrate 500000 ← Acts as DCE providing clock
!
router rip
 passive-interface Serial0 ← Prevents RIP updates from being sent to RouterA
 passive-interface Serial1
 network 152.1.0.0
 network 151.1.0.0
network 150.1.0.0
!
!
line con 0
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

RouterC

```
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname routerc
!
interface Ethernet0
 ip address 192.1.1.12 255.255.255.0 secondary
 ip address 192.1.1.11 255.255.255.0
!
interface Serial0
no ip address
shutdown
!
ip route 0.0.0.0 0.0.0.0 192.1.1.10 ← Sets the default route
!
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

When using the default interface set command, the router will first check the routing table for an explicit route. RouterA does not have an explicit route to 152.1.1.1 because RouterB suppresses RIP updates with the

passive interface commands.

From RouterA, monitor the policy routing using the **debug ip policy** command. From RouterC, ping 152.1.1.1 using the extended ping command to source the packet from 192.1.1.11. What follows is the output from the debug command on RouterA; note that the source address 192.1.1.11 matched item 10 of route map lab1 and was forwarded out interface S1.

```
IP: s=192.1.1.11 (Ethernet0), d=152.1.1.1, len 100, policy match
IP: route map lab1, item 10, permit
IP: s=192.1.1.11 (Ethernet0), d=152.1.1.1 (Serial0), len 100, policy routed
IP: Ethernet0 to Serial0 152.1.1.1
```

From RouterA, ping 152.1.1.1, sourcing the packet from 192.1.1.12. What follows is the output from the debug command on RouterA; note that the source address 192.1.1.12 matched item 20 of route map lab1 and was forwarded out interface S0.

```
IP: s=192.1.1.12 (Ethernet0), d=152.1.1.1, len 100, policy match
IP: route map lab1, item 20, permit
IP: s=192.1.1.12 (Ethernet0), d=152.1.1.1 (Serial1), len 100, policy routed
IP: Ethernet0 to Serial1 152.1.1.1
```

On RouterB remove the passive interface commands to allow RIP updates to be sent to RouterA. Now that RouterA has a route for 152.1.1.1 learned via RIP, it will not policy−route the packet. Remember when using the default interface set command, the router will first check the routing table for an explicit route. If the router has a route to the destination, the packet is forwarded using that route; if there is no explicit route available to the destination address, then the router will route the packet out the default interface, which is set using policy routing.

```
routerb(config)#router rip
routerb(config−router)#no passive−interface s0
routerb(config−router)#no passive−interface s1
```

From RouterC, ping 152.1.1.1. What follows is the output from the **debug ip policy** command on RouterA; note that the packet matched item 20 in route map lab1. However, the set policy was rejected because the routing table has an explicit route to 152.1.1.1.

```
IP: s=192.1.1.12 (Ethernet0), d=152.1.1.1, len 100, policy match
IP: route map lab1, item 20, permit
IP: s=192.1.1.12 (Ethernet0), d=152.1.1.1 (Serial1), len 100, policy rejected −
                                                            normal forwarding
```

Troubleshooting Policy Routing

The Cisco IOS provides many tools for troubleshooting policy routing. What follows is a list of key commands along with sample output from each.

{show ip policy} This privileged exec command displays which route map is used on which interface.

```
RouterA#show ip policy
Interface      Route map
Ethernet0      lab1
```

{show route−map} This privileged exec command displays configured route maps. This command allows you to view the policies defined by each route map. The command also shows how many packets matched the policy clauses.

```
RouterA#show route−map
route−map lab1, permit, sequence 10
  Match clauses:
    ip address (access−lists): 1
```

```
  Set clauses:
    default interface Serial0
  Policy routing matches: 129 packets, 14526 bytes
route-map lab1, permit, sequence 20
  Match clauses:
    ip address (access-lists): 2
  Set clauses:
    default interface Serial1
  Policy routing matches: 205 packets, 23370 bytes
```

{debug ip policy} This exec command helps you determine what policy routing is doing. It displays information about whether a packet matches the criteria, and if so, the resulting routing information for the packet. The first line indicates that a packet matched the policy. The second line indicates the item of the route map that the packet matched. In this case, the packet matches item 20 in route map lab1. Line three indicates that the packet was policy-routed out interface S0.

```
IP: s=192.1.1.11 (Ethernet0), d=152.1.1.1, len 100, policy match
IP: route map lab1, item 10, permit
IP: s=192.1.1.11 (Ethernet0), d=152.1.1.1 (Serial0), len 100, policy routed
IP: Ethernet0 to Serial0 152.1.1.1
```

{show ip local policy} This exec command displays any route maps used for local policy routing. By default, packets that are generated by the router are not policy-routed. Local policy routing must be enabled on the router using the IP local policy route-map command.

```
RouterA#show ip local policy
Local policy routing is enabled, using route map lab1
route-map lab1, permit, sequence 10
  Match clauses:
    ip address (access-lists): 1
  Set clauses:
    default interface Serial0
  Policy routing matches: 129 packets, 14526 bytes
route-map lab1, permit, sequence 20
  Match clauses:
    ip address (access-lists): 2
  Set clauses:
    default interface Serial1
  Policy routing matches: 205 packets, 23370 bytes
```

Conclusion

Policy-based routing provides network administrators a way to implement packet forwarding according to other criteria than traditional destination-based routing. The following are some of the potential applications for policy routing:

- Carrier selection is available for WAN transmissions or internal data path selection for Internet access.
- ISPs can use policy routing to provide equal access to multiple carrier networks.
- Policy-based routing can be used to set either the precedence or type-of-service bits in an IP datagram, which can be used to provide Quality of Service (QOS) across the backbone.
- Policy-based routing can be used to separate high- and low-priority traffic over separate links.

Chapter 14: Cisco Discovery Protocol

Overview

Topics Covered in This Chapter

- CDP overview
- Cisco CDP WAN configuration
- Cisco CDP LAN configuration
- CDP troubleshooting

Introduction

Cisco Discovery Protocol (CDP) is a Cisco proprietary protocol that is used for neighbor discovery. CDP is supported across the entire Cisco product line. CDP is very helpful in debugging situations. For example, it can be used to verify that a given router is connected to the proper port number on its neighbor. This chapter will examine CDP in detail.

Cisco Discovery Protocol Overview

CDP runs on all Cisco routers and switches. It can run over any physical media and over any protocol. Unlike a routing protocol that shows a next−hop destination port for all known networks, CDP will only show information for directly connected neighbors. It is most useful for verifying that a router is connected to the proper port of its neighbor.

Figure 14−1 gives an overview of the information that CDP can provide. A CDP−enabled router will be able to learn directly connected neighbor port and hostname information. Additional information such as the neighbor's hardware model number and capabilities are also reported.

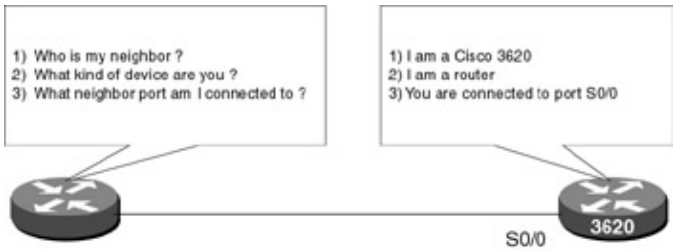


Figure 14−1: CDP overview

How Does CDP Work?

A CDP−enabled router sends out a periodic multicast packet containing a CDP update. The time between these CDP updates is determined by the **cdp timer** command, the timer value default being 60 seconds.

The following code shows a captured CDP packet. A Network Associates sniffer was put on an Ethernet LAN that also had several Cisco routers connected to it. As can be seen from the packet trace, the router sending the packet includes important information including:

- Router hostname (Cisco1)
- Router port information (Ethernet 0/0)
- IOS version information (11.2(7a)P)
- IOS platform information (C3620−I−M)
- Hardware version information (Cisco 3600)

Although neighbor router IOS version, IOS platform, and hardware version are not critical pieces of information, neighbor router hostname and neighbor router port information are critical for debug purposes. The use of the **show cdp neighbor** command is most useful in debug situations where one needs to verify what router and router port a given router is connected to.

```
Packet 1 captured at 12/21/1998 12:19:37 AM; Packet size is 318(0x13e)bytes
        Relative time: 000:00:35.858
        Delta time: 0.000.000
  ETHER: Address: 00-E0-1E-5B-0A-81 —->01-00-0C-CC-CC-CC
  Logical Link Control
        SSAP Address: 0xAA, CR bit = 0 (Command)
        DSAP Address: 0xAA, IG bit = 0 (Individual address)
        Unnumbered frame: UI
  SubNetwork Access Protocol
        Organization code: 0x00000c
        Type: Custom Defined
  Data:
  0000: 01 b4 aa 2b 00 01 00 0a 43 69 73 63 6f 31 00 02 | ..a+ ....Cisco1..
  0010: 00 11 00 00 00 01 01 01 cc 00 04 c1 01 01 01 00 | .........I..A....
  0020: 03 00 0f 45 74 68 65 72 6e 65 74 30 2f 30 00 04 | ...Ethernet0/0..
  0030: 00 08 00 00 00 01 00 05 00 e4 43 69 73 63 6f 20 | .........aCisco
  0040: 49 6e 74 65 72 6e 65 74 77 6f 72 6b 20 4f 70 65 | Internetwork Ope
  0050: 72 61 74 69 6e 67 20 53 79 73 74 65 6d 20 53 6f | rating System So
  0060: 66 74 77 61 72 65 20 0a 49 4f 53 20 28 74 6d 29 | ftware .IOS (tm)
  0070: 20 33 36 30 30 20 53 6f 66 74 77 61 72 65 20 28 |  3600 Software (
  0080: 43 33 36 32 30 2d 49 2d 4d 29 2c 20 56 65 72 73 | C3620-I-M), Vers
  0090: 69 6f 6e 20 31 31 2e 32 28 37 61 29 50 2c 20 53 | ion 11.2(7a)P, S
  00a0: 48 41 52 45 44 20 50 4c 41 54 46 4f 52 4d 2c 20 | HARED PLATFORM,
  00b0: 52 45 4c 45 41 53 45 20 53 4f 46 54 57 41 52 45 | RELEASE SOFTWARE
  00c0: 20 28 66 63 31 29 0a 43 6f 70 79 72 69 67 68 74 |  (fc1).Copyright
  00d0: 20 28 63 29 20 31 39 38 36 2d 31 39 39 37 20 62 |  (c) 1986-1997 b
  00e0: 79 20 63 69 73 63 6f 20 53 79 73 74 65 6d 73 2c | y cisco Systems,
  00f0: 20 49 6e 63 2e 0a 43 6f 6d 70 69 6c 65 64 20 57 |  Inc..Compiled W
  0100: 65 64 20 30 32 2d 4a 75 6c 2d 39 37 20 30 38 3a | ed 02-Jul-97 08:
  0110: 32 35 20 62 79 20 63 63 61 69 00 06 00 0e 63 69 | 25 by ccai....ci
  0120: 73 63 6f 20 33 36 32 30                         | sco 3620
```

Commands Discussed in This Chapter

- **cdp enable**
- **cdp run**
- **cdp timer**
- **clear cdp counters**
- **clear cdp table**
- **show cdp interface**
- **show cdp neighbor**
- **show cdp traffic**
- **debug cdp [packets] [ip] [adjacency] [events]**

Definitions

cdp enable: This interface command is used to enable CDP on a particular interface. Since CDP is enabled by default, this command will not be shown in the router configuration.

cdp run: This global command enables CDP on the entire router. Using the **no cdp run** command will disable any CDP on the router. Since CDP is enabled by default, the **cdp run** command will not shown in the router configuration.

cdp timer: This global command specifies how often the router sends CDP updates. The default time between CDP updates is 60 seconds.

clear cdp counters: This privileged exec command causes the router's CDP traffic counters to be reset.

clear cdp table: This privileged exec command causes the router's CDP table to be cleared. When this occurs, the **show cdp neighbor** command will not show any information until another CDP update is received from a neighbor router.

show cdp interface: This privileged exec command will show the status of CDP for each interface on the router.

show cdp neighbor: This privileged exec command causes the router to display neighbor information for all directly attached routers.

show cdp traffic: This privileged exec command will show how many CDP packets have been sent and received by the router. It also shows how many errored CDP packets have been received.

debug cdp [packets] [ip] [adjacency] [events]: This debug command will cause the router to display debugging information for a variety of CDP events.

IOS Requirements

CDP is supported in Cisco IOS releases 10.3 and higher.

Lab #68: Cisco CDP WAN Example

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, one of which must have two serial ports. The other two routers can have one serial port.
- Cisco IOS 10.3 or higher
- A PC running a terminal emulation program. The PC should be connected to one of the three routers using a Cisco rolled cable.
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable.

Configuration Overview

This configuration will demonstrate the basics of CDP. It will allow us to see the difference between information supplied by CDP and information supplied by a routing protocol such as RIP.

The three routers are serially connected as shown in Figure 14−2. RouterB will act as the DCE supplying clock to RouterA and RouterC. A PC running a terminal emulation program should be connected to the console port of one of the three routers using a Cisco rolled cable.

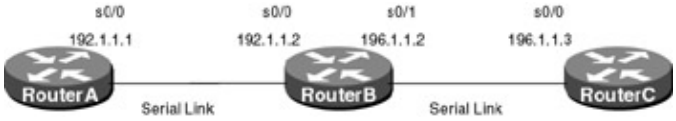


Figure 14−2: CDP WAN example
Note

Keep in mind that CDP will only supply information for directly connected neighbors. This is in contrast to a routing protocol, which will provide information that allows the router to determine the next interface hop to all known networks.

Router Configuration

The configurations for the three routers in this example are as follows. Notice that since CDP is enabled by default, there are no specific CDP commands in the configuration:

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation ppp
!
router rip
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation ppp
 clockrate 500000
!
interface Serial0/1
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
 clockrate 19200
!
router rip
 network 192.1.1.0
 network 196.1.1.0
!
no ip classless
```

```
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
enable password cisco
!
interface Serial0/0
 ip address 196.1.1.3 255.255.255.0
 encapsulation ppp
!
router rip
 network 196.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's examine some commands that enable us to monitor the status and results of CDP. The first important command is **show cdp traffic**. This command will display the number of CDP packets that have been received and sent by the router since the last **clear cdp counter** command:

```
RouterA#sh cdp traffic
CDP counters :
     Packets output: 16, Input: 11
     Hdr syntax: 0, Chksum error: 0, Encaps failed: 4
     No memory: 0, Invalid packet: 0, Fragmented: 0
```

The **show cdp** command will display how often CDP updates are sent (60 seconds) as well as how long CDP incoming information is kept until it is discarded (180 seconds).

```
RouterB#sh cdp
Global CDP information:
     Sending CDP packets every 60 seconds
     Sending a holdtime value of 180 seconds
```

The **show cdp neighbor** command will display information on directly connected neighbors of the router, provided that CDP is enabled on these interfaces. In the following example, we see that interface S0/0 on RouterA is connected to interface S0/0 on RouterB:

```
Cisco1#sh cdp neigh
```

```
Capability Codes: R − Router, T − Trans Bridge, B − Source Route Bridge
                  S − Switch, H − Host, I − IGMP, r − Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
RouterB            Ser 0/0         120            R         3620      Ser 0/0
```

Notice how the **show cdp neighbor** command output for router RouterB shows two directly connected neighbors, RouterA and RouterC:

```
RouterB#show cdp neighbor
Capability Codes: R − Router, T − Trans Bridge, B − Source Route Bridge
                  S − Switch, H − Host, I − IGMP, r − Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
RouterA              Ser 0/0         174            R         3620       Ser 0/0
RouterC              Ser 0/1         125            R         3620       Ser 0/0
```

The **show cdp neighbor detail** command provides additional information such as what IOS version and platform the neighboring device is running:

```
RouterA#sh cdp neighbor detail
Device ID: RouterB
Entry address(es):
  IP address: 192.1.1.2
Platform: cisco 3620,  Capabilities: Router
Interface: Serial0/0 ,  Port ID (outgoing port): Serial0/0
Holdtime : 174 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620−I−M), Version 11.2(7a)P, SHARED PLATFORM,
RELEASE SOFTWARE (fc1)
Copyright (c) 1986−1997 by cisco Systems, Inc.
Compiled Wed 02−Jul−97 08:25 by ccai
```

You can use the **show cdp interface** command to verify that CDP is enabled on the desired interfaces. If an interface does not have CDP enabled, it will not have an entry when using this command:

```
RouterA#sh cdp interface
Ethernet0/0 is administratively down, line protocol is down
  Encapsulation ARPA
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial0/0  is up, line protocol is up
  Encapsulation PPP
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial0/1  is administratively down, line protocol is down
  Encapsulation HDLC
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
```

Once again, keep in mind the advantages and limitations of CDP. CDP will only show directly connected neighbors. Recall that in this example, the **show cdp neighbor** command issued on router RouterA only shows RouterB as being directly connected. The following **show ip route** command output shows that the routing protocol RIP also has learned about the 196.1.1.0 network (which is RouterC):

```
RouterA#sh ip route
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, * − candidate default
       U − per−user static route, o − ODR
```

```
Gateway of last resort is not set

     192.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.1.1.0/24 is directly connected, Serial0/0
C       192.1.1.2/32 is directly connected, Serial0/0
R    196.1.1.0/24 [120/1 ] via 192.1.1.2, 00:00:08, Serial0/0
```

CDP Debug Commands

Several debug commands are available for advanced monitoring and troubleshooting of CDP.

The following screen print shows all CDP debugging enabled:

```
RouterC#sh debug
CDP:
  CDP packet info debugging is on
  CDP events debugging is on
  CDP neighbor info debugging is on
  CDP IP info debugging is on
```

CDP debug packet events will show packets being sent out and received by the router. The CDP debug information will show what router sent traffic to the router being monitored by the debug command. The following example shows a CDP packet being received from RouterB. Notice that the information being received is already known to the router. This is signified by the line "Entry found in cache."

```
CDP−PA: Packet received from RouterB on interface Serial0/0
**Entry  found in cache**
```

The following example shows what occurs when the router sends out a CDP packet:

```
CDP−PA: Packet sent out on Serial0/0
```

An interesting experiment can be tried by the reader that will highlight the details of how CDP functions. Using the current three−router configuration, turn on CDP debugging with the **debug cdp** command. Then pull the cable on serial 0/0 on RouterA. The following screen print should be similar to what you will see (remember to use the **term mon** command to direct output to the screen if you are not connected to the console connector on the router):

First you will see the router declare the line protocol and the interface down:

```
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial0/0, changed state to
down
%LINK−3−UPDOWN: Interface Serial0/0, changed state to down
```

CDP will then declare the interface to be in a failed state:

```
Dec 27 09:14:05: CDP−AD: Interface Serial0/0 going down
Dec 27 09:14:05: CDP−EV: Encapsulation on interface Serial0/0 failed
```

Try typing the **show cdp neighbor** command every few seconds. You will notice that the neighbor information does not change even though the interface is down. This is because of the holdtime value used by CDP. By default, CDP will hold an incoming packet's information for 180 seconds before discarding it. The following screen print shows that there are still 24 seconds remaining before the CDP process on RouterA will delete the neighbor entry for RouterB.

```
RouterA#sh cdp neigh
Capability Codes: R − Router, T − Trans Bridge, B − Source Route Bridge
                  S − Switch, H − Host, I − IGMP, r − Repeater
```

```
Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
RouterB              Ser 0/0         24             R        3620      Ser 0/0
```

As shown in the following example, the holdtime will eventually decrease to zero:

```
RouterA#sh cdp neigh
Capability Codes: R − Router, T − Trans Bridge, B − Source Route Bridge
                  S − Switch, H − Host, I − IGMP, r − Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
RouterB              Ser 0/0         0              R        3620      Ser 0/0
```

When the holdtime expires, the router will then age out the entry. Notice in the following screen print that there is no longer an entry for any neighbor router:

```
RouterA#sh cdp neigh
Capability Codes: R − Router, T − Trans Bridge, B − Source Route Bridge
                  S − Switch, H − Host, I − IGMP, r − Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
```

CDP will alert you to an aged entry via the following message:

```
Dec 27 09:16:33: CDP−AD: Aging entry for RouterB, on interface Serial0/0
```

When you reconnect the cable going to interface serial 0/0 on router RouterA, you will see the interface go to an up state. CDP will start to send out packets. Notice that the first entry received will not be found in the CDP cache, since the old entry was already aged out.

```
%LINK−3−UPDOWN: Interface Serial0/0, changed state to up
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial0/0 , changed state to up
Dec 27 09:17:06: CDP−AD: Interface Serial0/0  coming up
Dec 27 09:17:06: CDP−PA: Packet sent out on Serial0/0
Dec 27 09:17:06: CDP−PA: Packet received from Cisco2 on interface Serial0/0
Dec 27 09:17:06: **Entry NOT found in cache**
```

The **show cdp neighbor** command will now show an entry for directly connected neighbor RouterB:

```
RouterA#sh cdp neigh
Capability Codes: R − Router, T − Trans Bridge, B − Source Route Bridge
                  S − Switch, H − Host, I − IGMP, r − Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
RouterB              Ser 0/0         171            R        3620      Ser 0/0
```

Lab #69: Cisco CDP LAN Example

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having an Ethernet port
- Cisco IOS 10.3 or higher
- A PC running a terminal emulation program
- Three Ethernet cables
- An Ethernet hub
- An optional LAN sniffer to trace the CDP packets

Configuration Overview

This configuration will show how CDP works on a shared−media Ethernet LAN.

The three routers are all connected to the same Ethernet hub, as shown in Figure 14−3. An optional LAN sniffer can also be connected into the Ethernet hub. The LAN sniffer can be used to capture the CDP packets.

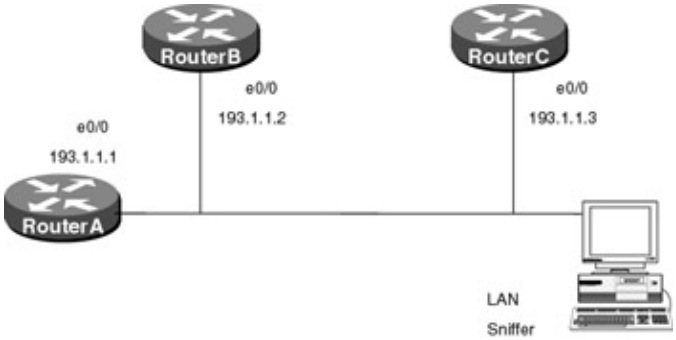


Figure 14−3: CDP LAN example

Note Keep in mind that CDP will only supply information for directly connected neighbors. This is in contrast to a routing protocol such as RIP, which will provide information that allows the router to determine the next interface hop to all known networks.

Router Configuration

The configurations for the three routers in this example are as follows. Notice that since CDP is enabled by default, there are no specific CDP commands in the configuration:

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
enable password cisco
!
!
interface Ethernet0/0
 ip address 193.1.1.1 255.255.255.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterB
```

```
!
enable password cisco
!
!
interface Ethernet0/0
 ip address 193.1.1.2 255.255.255.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
enable password cisco
!
!
interface Ethernet0/0
 ip address 193.1.1.3 255.255.255.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

The key CDP monitoring and debug commands were covered in the previous section. Since all the routers in this configuration are connected to the same LAN, each of the three routers will display the same neighbor table, as shown in the following **show cdp neighbor** command:

```
RouterA#sh cdp neigh
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
RouterC              Eth 0/0         121           R        3620      Eth 0/0
RouterB              Eth 0/0         177           R        3620      Eth 0/0
```

Conclusion

In this chapter, we examined the Cisco Discovery Protocol. CDP is a media- and protocol-independent proprietary protocol used for neighbor discovery. CDP does not replace a routing protocol. CDP will only

show information on directly connected neighbors. CDP is particularly useful in determining what neighbor router and port a given router is connected to.

Chapter 15: Network Address Translation

Overview

Topics Covered in This Chapter

- Detailed NAT overview
- NAT terminology
- Static inside source address translation
- Dynamic inside source address translation
- Overloading an inside global address
- Translating overlapping addresses
- Destination address rotary translation
- Changing translation timeouts
- Detailed troubleshooting examples

Introduction

Network Address Translation (NAT) is a router function that provides the translation from one IP address to another. Address translation is required for customers who have private (or unregistered) addresses and wish to access a public service (where publicly registered addresses are used). This chapter will explore NAT capabilities available through the Cisco IOS.

Network Address Translation Overview

One of the greatest problems facing the Internet today is the issue of address depletion. Network Address Translation promises to relieve some of this pressure by allowing organizations to reuse globally unique registered IP addresses in other parts of their network.

NAT allows organizations to reuse registered IP addresses within multiple domains, as long as the addresses are translated to globally unique Internet registered addresses before they leave that domain. Figure 15−1 shows how basic NAT works. Both stub networks are using the class A address 10.0.0.0 for their internal network. Each organization is assigned an Internet registered unique class C address. This address is used when traffic wishes to flow off the private intranet onto the public Internet.

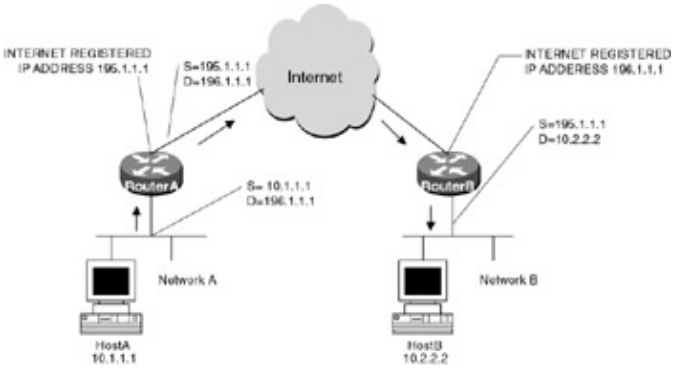


Figure 15−1: Network Address Translation

In Figure 15−1, when HostA (10.1.1.1) wishes to send a packet to HostB (10.2.2.2), it uses HostB's globally unique address 196.1.1.1 as the packet's destination. When the packet arrives at RouterA, the source address of 10.1.1.1 is translated to the globally unique address of 195.1.1.1. When the packet arrives at RouterB, the

destination address is translated to the unregistered IP address 10.2.2.2. Likewise, packets on the return path go through similar address translation.

This requires no additional configuration to hosts on the internal network; as far as HostA is concerned, 196.1.1.1 is the IP address of the HostB (10.2.2.2) on Network B. As far as HostB is concerned, 195.1.1.1 is the IP address of HostA (10.1.1.1) on Network A.

NAT Terminology

When dealing with NAT on a Cisco router, it is important to understand the terminology used, as illustrated in Figure 15−2.

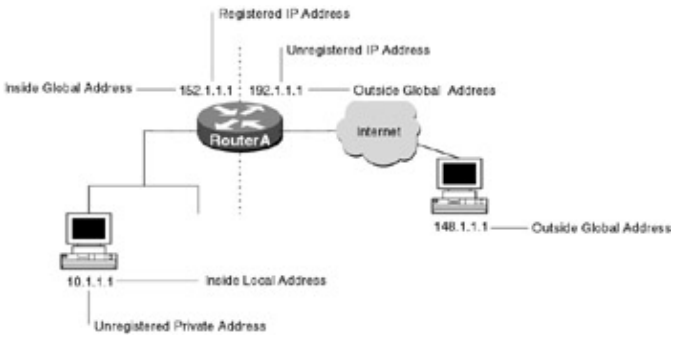


Figure 15−2: NAT terminology

Inside local address: The IP address that is assigned to a host on the inside network. This address is probably not an IP address assigned by the Network Information Center (NIC) or service provider.

Inside global address: An NIC−registered IP address that is used to represent one or more inside local IP addresses to the outside world.

Outside local address: The IP address of an outside host as it appears to the inside network. Not necessarily a legitimate address, it was allocated from address space routable on the inside.

Outside global address: The IP address assigned to a host on the outside network by the host's owner. The address was allocated from globally routable address or network space.

Commands Discussed in This Chapter

clear ip nat translations

debug ip nat

ip nat {inside | outside}

ip nat inside destination list {*access−list−number* | *name*} *pool name*

ip nat inside source {*list* {*access−list−number* | *name*} *pool name* [overload] | static *local−ip global−*ip}

ip nat outside source {*list* {*access−list−number* | *name*} *pool name* | static *global−ip local−ip*}

ip nat pool name *start−ip end−ip* {*netmask* | *prefix−length prefix−length*} [type rotary]

ip nat translation {timeout | udp−timeout | dns−timeout | tcp−timeout | finrst−timeout} *seconds*

show ip nat statistics

show ip nat translations

Definitions

clear ip nat: This exec command is used to clear all or specific active NAT translations.

ip nat: This command is used to enable Network Address Translation for packets originating from (**inside**) or destined to (**outside**) interfaces.

ip nat inside destination list: This global command enables Network Address Translation of the inside destination address. This command can be configured for both dynamic and static address translations.

ip nat inside source: This global command enables Network Address Translation of the inside source address. This command can be configured for both dynamic and static address translations.

ip nat outside source: This global command enables Network Address Translation of outside source addresses. This command can be configured for both dynamic and static address translations.

ip nat pool name: This global command defines a pool of IP addresses used for network translations. The pool could define either an inside global pool, an outside local pool, or a rotary pool.

ip nat translation: This global command is used to change the amount of time after which Network Address Translations time out**.**

show ip nat statistics: This command is used to display Network Address Translation statistics.

show ip nat translations: This command displays all active Network Address Translations.

IOS Requirements

NAT first became available in IOS 11.2.

Lab #70: Static Inside Source Address Translation

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- Cisco IOS 11.2 or higher
- A PC running a terminal emulation program
- A PC with an Ethernet NIC or additional router
- Two Ethernet cables and an Ethernet hub, one Cisco DTE/DCE crossover cable

Configuration Overview

This configuration will demonstrate Network Address Translation of an unregistered inside IP address to a globally unique outside address, as shown in Figure 15−3. RouterA will translate the inside source address of 10.1.1.1 to the globally unique address of 195.1.1.1.

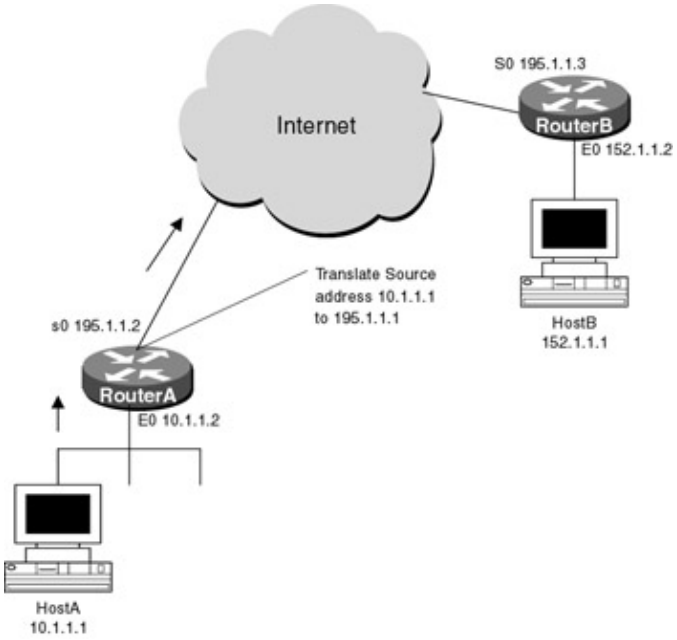


Figure 15−3: Inside source address translation

RouterA and RouterB are connected serially via a crossover cable. RouterA will act as the DCE supplying clock to RouterB. The IP addresses are assigned as per Figure 15−4. A PC with an Ethernet NIC (or an additional router) is connected to an Ethernet LAN attached to RouterA. RouterA is configured for NAT and will translate source IP address 10.1.1.1 to 195.1.1.1.

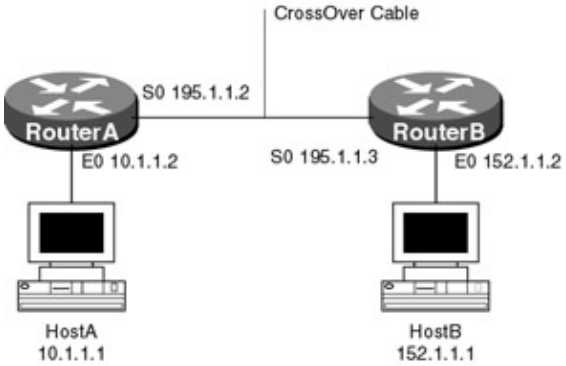


Figure 15−4: Inside source address translation

Router Configurations

The configurations for the two routers in this example are as follows (key NAT configurations for RouterA are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname routerA
!
ip nat inside source static 10.1.1.1 195.1.1.1 ← Translates the inside source
                                                  address 10.1.1.1 to 195.1.1.1
!
interface Ethernet0
  ip address 10.1.1.2 255.255.255.0
 ip nat inside ← Marks the interface as connected to the inside
 !
interface Serial0
  ip address 195.1.1.2 255.255.255.0
 ip nat outside ← Marks the interface as connected to the outside
```

```
Clock rate 500000
 !
no ip classless
ip route 152.1.1.1 255.255.255.255 Serial0
!
line con 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Ethernet0/0
  ip address 152.1.1.1 255.255.255.0
!
interface Serial0/0
 ip address 195.1.1.3 255.255.255.0
 !
line con 0
line aux 0
line vty 0 4
 password cisco
 login
```

Monitoring and Testing the Configuration

From HostA, ping HostB (152.1.1.1) and analyze the packets coming from RouterB with the **debug ip packet** command. What follows is the output from the command; note that the source address of the ICMP Ping packet is 195.1.1.1.

```
IP: s=195.1.1.1 (Serial0/0), d=152.1.1.1, len 104, rcvd 4 ← ICMP ECHO
IP: s=152.1.1.1 (local), d=195.1.1.1 (Serial0/0 ), len 104 ← ICMP ECHO REPLY
```

From the **debug ip nat** output on RouterA, we can see that the source IP address 10.1.1.1 has been translated to 195.1.1.1. We also see this is a two−way process; the return packet that has the destination IP address 195.1.1.1 is changed back to 10.1.1.1.

```
NAT: s=10.1.1.1->195.1.1.1, d=152.1.1.1 [2542]
NAT*: s=152.1.1.1, d=195.1.1.1->10.1.1.1 [2542]
```

In the preceding section, we covered a one−to−one mapping between an inside local address and an inside global address. This method is very inefficient and does not scale, because each registered IP address can only be used by one end station. *Static* translation is most often used when a host on the inside needs to be accessed by a fixed IP address from the outside world.

Figure 15−5 shows an example of when static address mapping is required. HostA wishes to access files on the FTP server; however, the FTP server resides on an inside network and does not have a unique globally significant IP address. Static mapping is used to define the globally significant address of 195.1.1.1 to the locally significant address of 10.1.1.1.

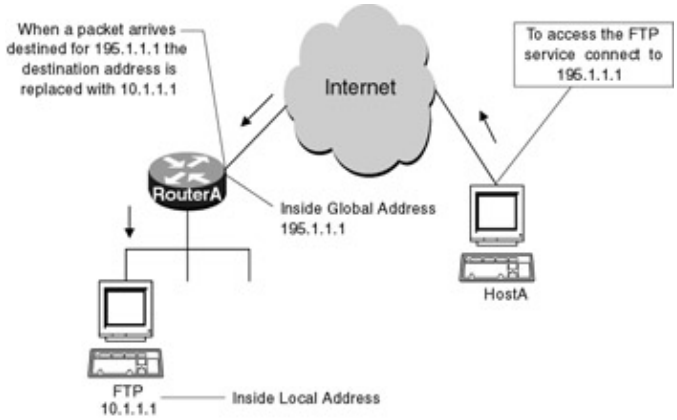


Figure 15−5: Static mapping

Lab #71: Dynamic Inside Source Address Translation

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- Cisco IOS 11.2 or higher
- One PC running a terminal emulation program
- One Cisco DTE/DCE crossovercable

Overview

The other type of inside address translation is *dynamic* translation, which establishes a mapping between a group of inside local addresses and a pool of global addresses. This is very useful when you have a large group of unregistered users who wish to access off−net services.

Dynamic inside address translation dynamically translates an unregistered IP address to a registered IP address, using a predefined pool. This is a one−to−one relationship; as an outside connection is requested, an IP address is used from the pool. When the connection is finished, the globally significant IP address is released back into the pool, where it can be used for another connection. Dynamic address translation is very efficient, because the same global IP address can be used over and over as needed, by multiple end stations. This is in contrast to the previous static translation, where only one particular end station can use the global address.

Figure 15−6 shows three workstations on a LAN, all of which need access to the outside network. As packets arrive at RouterA, the source address is translated to an Internet registered address, using the predefined pool. This is still a one−to−one mapping; you need an Internet registered IP address for each workstation that wishes to communicate outside the private network. However, not all PCs will access the Internet at the same time. For example, depending on the traffic pattern, 10 registered IP addresses possibly could service 40 PCs.

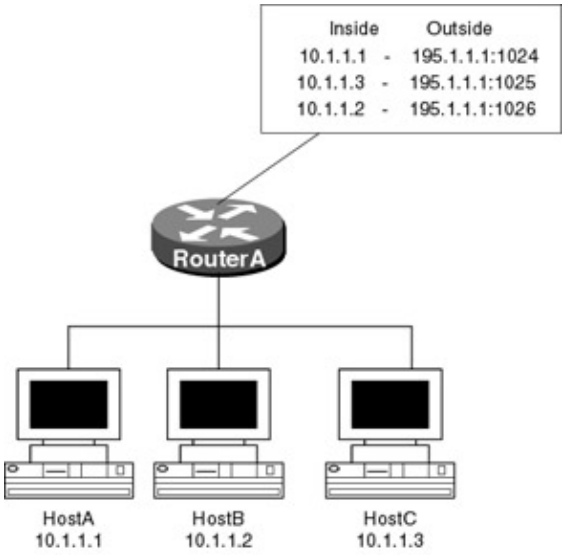


Figure 15−6: Dynamic address translation

Note Although dynamic address translation is more scalable, more efficient, and simpler to administer, outside users cannot access inside addresses, because there is no static mapping between IP addresses. After each session is closed, the global IP address is released back into the pool, where it can be used by other sessions. Each end station can and most likely will be mapped to a different global address when it opens a new connection. Therefore, it is impossible to reference a particular inside address with a global address.

This problem can be avoided by using a combination of dynamic and static translations. All hosts that need to be accessed by outside users, such as FTP and HTTP servers, will be configured using static translations, while all other end stations will use dynamic translations.

Configuration Overview

This configuration will demonstrate dynamic translation of inside source addresses to outside global addresses. RouterA will translate any source address within the range of 10.1.1.1 to 10.1.1.3 to any of the three global addresses defined in the address pool "globalpool."

Two Cisco routers are connected serially. RouterA is connected to RouterB via a crossover cable. RouterB acts as the DCE providing clock for RouterA. A PC running a terminal emulation program is connected to the console port of RouterA. All IP addresses are as per Figure 15−7.

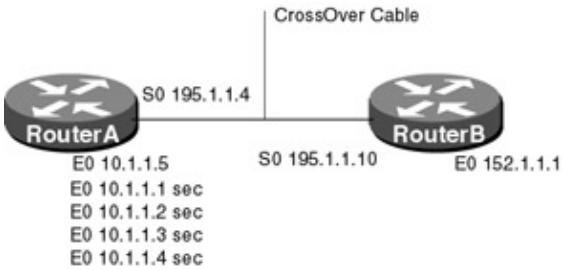


Figure 15−7: Dynamic address translation

RouterA is configured for Network Address Translation and will dynamically translate any inside source address within the range specified by access−list 1 to a unique Internet registered global address, which is predefined by the pool "globalpool."

Router Configurations

The configurations for the two routers in this example are as follows (key NAT configurations for RouterA are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname routerA
!
                     ↓ Name of the pool
ip nat pool globalpool 195.1.1.1 195.1.1.3 netmask 255.255.255.0 ← Defines
                                                                    the pool of
                                                                    address
                            List 1 reference access-list 1 and defines which
                          ↓ addresses will be translated
ip nat inside source list 1 pool globalpool ← Globalpool references the pool of
                                              addresses defined in the previous
                                              line
!
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0 secondary
 ip address 10.1.1.2 255.255.255.0 secondary
 ip address 10.1.1.3 255.255.255.0 secondary → Secondary IP addresses are used
                                                as test points
 ip address 10.1.1.4 255.255.255.0 secondary
 ip address 10.1.1.5 255.255.255.0
 ip nat inside → Defines the inside interface
!
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip nat outside → Defines the outside interface
!
no ip classless
ip route 152.1.1.1 255.255.255.255 Serial0
access-list 1 permit 10.1.1.2
access-list 1 permit 10.1.1.3
access-list 1 permit 10.1.1.1 → Access list 1 defines which inside source
                                 addresses will be translated
access-list 1 permit 10.1.1.4
!
line con 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Ethernet0/0
  ip address 152.1.1.1 255.255.255.0
!
interface Serial0/0
 ip address 195.1.1.10 255.255.255.0
 clock rate 500000 → Defines the clock rate for the DCE interface

 !
line con 0
```

```
line aux 0
line vty 0 4
 password cisco
 login
```

Monitoring and Testing the Configuration

To test the configuration, use the extended ping command on RouterA. This command will allow you to source the ping packet from any active IP address on the router. To use this command, simply type in **ping** at the privileged level.

```
routerA#ping
Protocol [ip]:
Target IP address: 152.1.1.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 10.1.1.2
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
```

The following examples all use the extended ping command on RouterA to source the packets from the secondary IP addresses defined in the configuration. This is used instead of multiple PCs on RouterA's LAN.

1. From RouterA ping 152.1.1.1 using source address 10.1.1.2
2. From RouterA ping 152.1.1.1 using source address 10.1.1.1
3. From RouterA ping 152.1.1.1 using source address 10.1.1.3

From the **debug ip nat** translation's output on RouterA, we see that the source address 10.1.1.2 has been translated to 195.1.1.1, which is the first address in the pool. The global IP addresses from the pool are assigned in the order that they are requested.

```
NAT: s=10.1.1.2->195.1.1.1, d=152.1.1.1 [20]
NAT: s=10.1.1.1->195.1.1.2, d=152.1.1.1 [25]
NAT: s=10.1.1.3->195.1.1.3, d=152.1.1.1 [35]
```

The following output from the **debug ip nat translation** command on RouterA shows what happens when a fourth end station wishes to access the outside network but all of the global addresses are being used.

```
NAT: translation failed (L), dropping packet s=10.1.1.4 d=152.1.1.1
```

From the preceding examples, you can see that although dynamic address translation provides more efficient use of global addresses than do static translations, each translation still requires its own address. Therefore, the network administrator must accurately gauge the amount of off−net traffic and define the address pool accordingly.

Lab #72 Overloading an Inside Global Address

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- Cisco IOS 11.2 or higher
- One PC running a terminal emulation program
- Cisco DTE/DCE crossover cable

Overview

The Cisco IOS allows you to overload a global address, thereby bypassing the need for a one−to−one mapping between the local address and the global address. This greatly reduces the number of registered IP addresses needed.

When overloading is configured, the router maintains enough information from higher−level protocols (for example, TCP or UDP port numbers) to translate the global address back to the correct local address. When multiple local addresses map to one global address, the TCP or UDP port numbers of each inside host are used to distinguish between the local addresses.

In Figure 15−8 all of the local addresses on the LAN are translated to one global IP address, 195.1.1.1. The router reuses the inside global address for each translation and uses the TCP or UDP port number to differentiate between end stations.

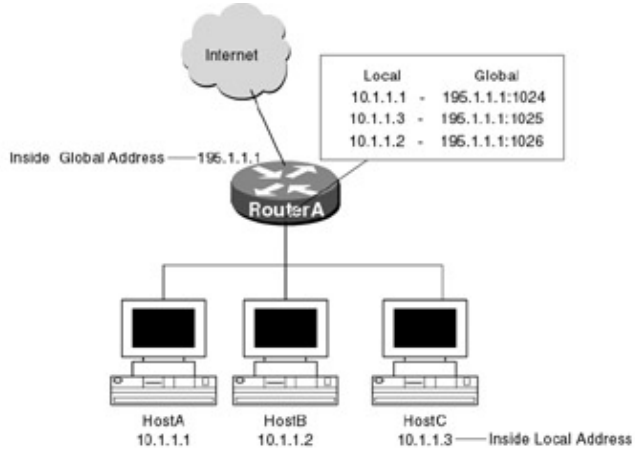


Figure 15−8: Overloading an inside global address

The following steps are taken by RouterA when overloading is enabled:

1. HostA (10.1.1.1) opens a connection to Host 152.1.1.1 on the Internet.
2. The first packet that the RouterA receives from HostA causes the router to check its NAT table.
3. If no translation exists, RouterA replaces the source address with the global address of 195.1.1.1.
4. When the router receives a packet from host 152.1.1.1 destined for 195.1.1.1, it performs a NAT table lookup using the protocol, inside global address and port number, and outside address and port number as a key. With this key, RouterA is able to translate the destination address 195.1.1.1 to inside local address 10.1.1.1, and it forwards the packet to Host 10.1.1.1.

The following is output from the **show ip nat translations** command on RouterA; notice the port number after the address. The port number 1029 after the inside global address is the ephemeral port that HostA chooses; port number 23 after the outside address is the well−known port for telnet.

```
routerA# show ip nat translations

Pro I    inside global      Inside local       Outside local      Outside global
icmp     195.1.1.1:256      10.1.1.1:256       152.1.1.1:256      152.1.1.1:256
tcp      195.1.1.1:1029     10.1.1.1:1029      152.1.1.1:23       152.1.1.1:23
```

Configuration Overview

This configuration will demonstrate overloading one outside global address. RouterA will translate any source address within the range of 10.1.1.1 to 10.1.1.3 to the global address 195.1.1.1.

Two Cisco routers are connected serially. RouterA is connected to RouterB via a crossover cable. RouterB acts as the DCE providing clock for RouterA. A PC running a terminal emulation program is connected to the console port of RouterA. All IP address are as shown in Figure 15−9.

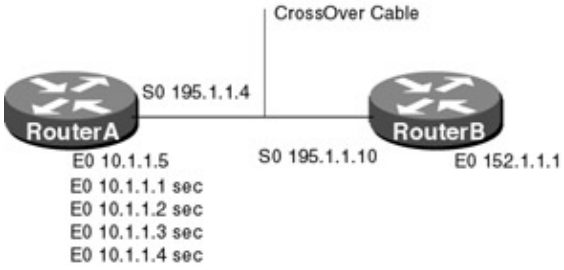


Figure 15−9: Overloading an inside global address

RouterA is configured for Network Address Translation and will dynamically translate any inside source address within the range specified to the unique Internet registered global address 195.1.1.1.

Router Configurations

The configurations for the two routers in this example are as follows (key NAT configurations for RouterA are highlighted in bold).

RouterA

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname routerA
!
                     ↓ Name of the pool
ip nat pool globalpool 195.1.1.1 195.1.1.1 netmask 255.255.255.0 ← Defines
                                                                    range on
                                                                    pool, in
                                                                    this case
                                                                    there is
                                                                    only one
                                                                    address in
                                                                    the pool
                            List 1 references access list 1 and defines which
                      ↓ address will be translated
ip nat inside source list 1 pool globalpool  overload ← Allows multiple inside
                                  á Defines what       local addresses to be
                                    global address     translated to one
                                    to use             outside global address
!
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0 secondary
 ip address 10.1.1.2 255.255.255.0 secondary
 ip address 10.1.1.3 255.255.255.0 secondary → Secondary IP addresses are used
                                               as test points
 ip address 10.1.1.4 255.255.255.0 secondary
 ip address 10.1.1.5 255.255.255.0
 ip nat inside → Defines the inside interface
!
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip nat outside → Defines the outside interface
```

```
 !
no ip classless
ip route 152.1.1.1 255.255.255.255 Serial0
access-list 1 permit 10.1.1.2
access-list 1 permit 10.1.1.3
access-list 1 permit 10.1.1.1 → Access list 1 defines which inside source
                                 addresses that will be translated
access-list 1 permit 10.1.1.4
!
line con 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Ethernet0/0
  ip address 152.1.1.1 255.255.255.0
!
interface Serial0/0
 ip address 195.1.1.10 255.255.255.0
clock rate 500000
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
```

Monitoring and Testing the Configuration

To test the configuration, ping RouterB (195.1.1.3) and, using the extended ping command on RouterA, source the packet from 10.1.1.1 and 10.1.1.2. Monitor the translation using the command **debug ip nat**.

What follows is the output from the command; notice that both the inside

source addresses 10.1.1.1 and 10.1.1.2 have been translated to 195.1.1.1.

```
NAT: s=10.1.1.1->195.1.1.1, d=195.1.1.3 [5]
NAT: s=10.1.1.2->195.1.1.1, d=195.1.1.3 [10]
```

Now show the NAT table using the command **show ip nat translations.** What follows is the output from the command; notice the port number after each IP address. This port number and address are used as a key to map return packets to the correct inside local IP address.

```
RouterA#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
icmp 195.1.1.1:9       10.1.1.2:4         195.1.1.3:4        195.1.1.3:9
icmp 195.1.1.1:8       10.1.1.2:3         195.1.1.3:3        195.1.1.3:8
icmp 195.1.1.1:7       10.1.1.2:2         195.1.1.3:2        195.1.1.3:7
icmp 195.1.1.1:6       10.1.1.2:1         195.1.1.3:1        195.1.1.3:6
icmp 195.1.1.1:5       10.1.1.2:0         195.1.1.3:0        195.1.1.3:5
icmp 195.1.1.1:4       10.1.1.1:4         195.1.1.3:4        195.1.1.3:4
```

```
icmp 195.1.1.1:3       10.1.1.1:3         195.1.1.3:3        195.1.1.3:3
icmp 195.1.1.1:2       10.1.1.1:2         195.1.1.3:2        195.1.1.3:2
icmp 195.1.1.1:1       10.1.1.1:1         195.1.1.3:1        195.1.1.3:1
icmp 195.1.1.1:0       10.1.1.1:0         195.1.1.3:0        195.1.1.3:0
```

Lab #73: Translating Overlapping Addresses

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- Cisco IOS 11.2 or higher
- One PC running a terminal emulation program
- Two PCs with Ethernet network interface cards, one running a DNS server daemon
- Four Ethernet cables and two Ethernet hubs, one Cisco DTE/DCE crossover cable

Overview

Overlapping occurs when an inside local address overlaps with an address of the destination that you are trying to reach. In Figure 15−10, HostA (148.1.1.1) opens a connection to HostB by name, requesting a name−to−address lookup from the DNS server. The DNS server responds with the address of HostB, 148.1.1.1. The inside local address overlaps with the outside address.

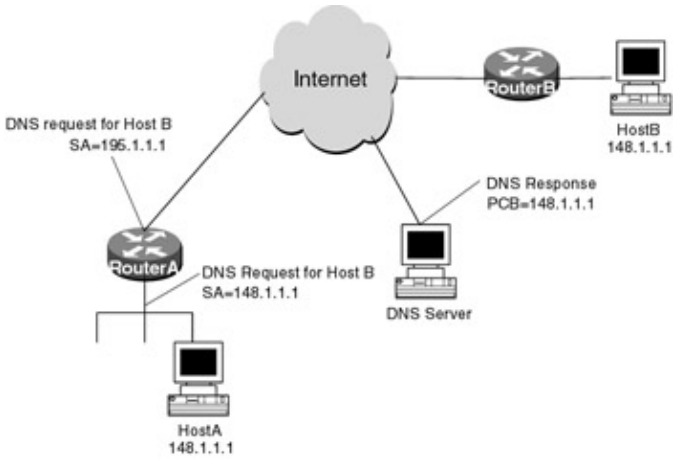


Figure 15−10: IP address overlapping

The Cisco IOS solves this problem by translating the outside global address to an outside local address.

The following steps are taken by RouterA:

1. HostA opens a connection to HostB using its name, and a request is sent to the DNS server for a name−to−address resolution.
2. The DNS server responds, resolving HostB to IP address148.1.1.1.
3. RouterA intercepts the packet and translates the global source address to a local address from the outside local address pool.
4. RouterA keeps a simple table mapping the global address to the outside local address.
5. When HostA sends a packet to HostB, the destination IP address is the outside local address.
6. When RouterA receives a packet destined for the outside local address, it translates the local address back to the global address.

The following is output from the **show ip nat translations** command on RouterA; the outside global address 148.1.1.1 is mapped to outside local address 2.2.2.2, which is defined in the router configuration.

```
RouterA#show ip nat translations

Pro     Inside global      Inside local      Outside local      Outside global
--- ---           ---                          2.2.2.2            148.1.1.1
tcp     195.1.1.1:1071      10.1.1.1:1071    148.1.1.1:23        148.1.1.1:23
```

Configuration Overview

This configuration demonstrates outside global address translation. RouterA monitors all DNS responses, and if the resolved address overlaps with the inside local address (10.1.1.1), RouterA translates that address to 2.2.2.2.

Two Cisco routers are connected serially. RouterA is connected to RouterB via a crossover cable. RouterB acts as the DCE providing clock for RouterA. A PC running a terminal emulation program is connected to the console port of RouterA. All IP address are as Figure 15−11.

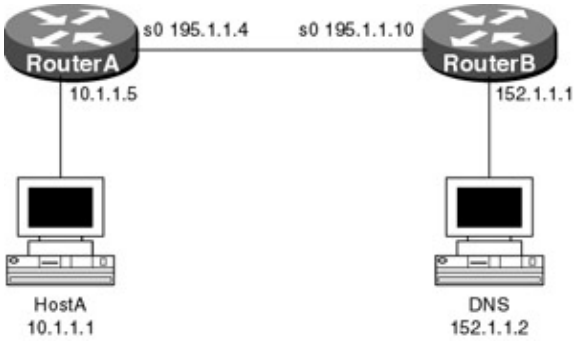


Figure 15−11: IP address overlapping

HostA is configured with a default route of 10.1.1.5 and a DNS entry of 152.1.1.2. RouterA is configured for Network Address Translation and will monitor all DNS responses. If the resolved address overlaps with 10.1.1.1, it will statically translate the address of the resolved host to 2.2.2.2.

The second workstation is configured as a domain name server and will resolve the name HostB to 10.1.1.1.

Router Configurations (Static Mapping)

The following configuration defines a static mapping between the outside global address of 10.1.1.1 and the outside local address of 2.2.2.2.

RouterA

```
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname routerA
!
!
ip nat pool globalpool 195.1.1.1 195.1.1.3 netmask 255.255.255.0
ip nat inside source list 1 pool globalpool overload
ip nat outside source static 10.1.1.1 2.2.2.2 ← Defines translation from the
                                                outside global address 10.1.1.1
                                                to the outside local address of
                                                2.2.2.2
!
interface Ethernet0
 ip address 10.1.1.2 255.255.255.0 secondary
 ip address 10.1.1.3 255.255.255.0 secondary
 ip address 10.1.1.4 255.255.255.0 secondary
 ip address 10.1.1.5 255.255.255.0
```

```
 ip nat inside ← Defines the inside interface
 !
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip nat outside ← Defines the outside interface
 !
no ip classless
ip route 152.1.1.1 255.255.255.255 Serial0
access−list 1 permit 10.1.1.2
access−list 1 permit 10.1.1.3
access−list 1 permit 10.1.1.1
access−list 1 permit 10.1.1.4
!
line con 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp−small−servers
service tcp−small−servers
!
hostname RouterB
!
enable password cisco
!
interface Ethernet0/0
  ip address 152.1.1.1 255.255.255.0
!
interface Serial0/0
 ip address 195.1.1.10 255.255.255.0
clock rate 500000
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
```

Router Configurations (Dynamic Mapping)

The following configuration defines a dynamic mapping between a pool of outside local addresses to a group of outside global addresses defined by an access list.

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname routerA
!
!ip nat pool globalpool 195.1.1.1 195.1.1.3 netmask 255.255.255.0

                    ↓ Pool Name
ip nat pool outsidelocal 2.2.2.1 2.2.2.4 netmask 255.255.255.0
                          á Pool Range
```

```
ip nat inside source list 1 pool globalpool overload

                                ↓ References the outside local pool
ip nat outside source list 2 pool outsidelocal ← (If the outside global source
                         á Specifies what         address matches access list 1
                           addresses should       change to one of the addresses
                           be changed             defined in pool outsidelocal)
!
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0 secondary
 ip address 10.1.1.2 255.255.255.0 secondary
 ip address 10.1.1.3 255.255.255.0 secondary
 ip address 10.1.1.4 255.255.255.0 secondary
 ip address 10.1.1.5 255.255.255.0
 ip nat inside ← Defines the inside interface
 !
interface Serial0
 ip address 195.1.1.4 255.255.255.0
 ip nat outside ← Defines the outside interface
 !
no ip classless
ip route 152.1.1.1 255.255.255.255 Serial0
access-list 2 permit 10.1.1.1
access-list 2 permit 10.1.1.2 ← If the outside global source address matches
                                 one of these change
access-list 2 permit 10.1.1.3
access-list 2 permit 10.1.1.4
no cdp run
!
line con 0
line vty 0 4
 login
!
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Ethernet0/0
  ip address 152.1.1.1 255.255.255.0
!
interface Serial0/0
 ip address 195.1.1.10 255.255.255.0
clock rate 500000
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
```

Monitoring and Testing the Configuration

To test the configuration, ping HostB from HostA using the domain name. Use the **Debug ip nat** command and the **show ip nat translations** command to verify that the translation is working properly.

What follows is the output from the **debug ip nat** command; note that the DNS response is translated to 2.2.2.2.

```
RouterArA#deb ip nat
01:04:23: NAT: i: udp (10.1.1.1, 1082) -> (10.10.3.111, 53) [62735]
01:04:23: NAT: s=10.1.1.1->195.1.1.1, d=10.10.3.111 [62735]
01:04:23: NAT: o: udp (10.10.3.111, 53) -> (195.1.1.1, 1082) [9227]
01:04:23: NAT: DNS resource record 10.1.1.1 -> 2.2.2.2
01:04:23: NAT: s=10.10.3.111, d=195.1.1.1->10.1.1.1 [9227]
01:04:23: NAT: o: icmp (10.1.1.100, 256) -> (10.1.1.1, 256) [21]
01:04:24: NAT: o: icmp (10.1.1.100, 256) -> (10.1.1.1, 256) [22]
01:04:25: NAT: o: icmp (10.1.1.100, 256) -> (10.1.1.1, 256) [23]
01:04:26: NAT: o: icmp (10.1.1.100, 256) -> (10.1.1.1, 256) [24]
```

What follows is the output from the **show ip nat translations** on RouterA; note that the overlapping outside global address of 10.1.1.1 is translated to 2.2.2.2.

```
routerA#show ip nat translations
Pro Inside global       Inside local       Outside local      Outside global
--- 195.1.1.1          10.1.1.1        ---          ---
--- ---                 ---                2.2.2.2             10.1.1.1
```

Lab #74: Destination Address Rotary Translation

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with one Ethernet port and one serial port
- Cisco IOS 11.2 or higher
- One PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable

Overview

Network address rotary translation can be used as a means to provide load sharing among multiple highly utilized hosts. Figure 15−12 illustrates this feature. Company X has multiple FTP servers that are accessed by customers to download software. The NAT translation is transparent to the user; they simply FTP to the virtual IP address 152.1.1.10.

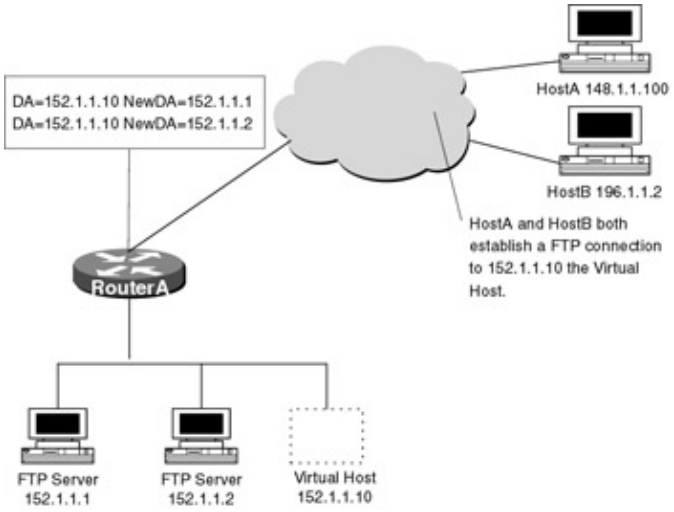


Figure 15−12: Load balancing using NAT

When RouterA receives a packet destined for the virtual IP address, it translates the destination address to the first FTP server. When the next FTP connection is established to the virtual IP address, RouterA translates the destination address to the second FTP server. These translations occur in a round−robin fashion, providing equal load balancing across multiple FTP servers.

RouterA takes the following steps when translating a rotary address:

1. User A (148.1.1.100) establishes a connection to virtual host 152.1.1.10.
2. RouterA receives the packet destined for the virtual host 152.1.1.10 and translates the destination address to next real host from the pool, in this case FTP server 152.1.1.1.
3. FTP server 152.1.1.1 receives the packet and responds.
4. RouterA receives the response packet from FTP server 152.1.1.1 and performs a NAT table lookup using the inside local address and port number and the outside address and port number as the key.
5. RouterA then translates the source address to the address of the virtual host and forwards the packet.
6. User B (196.1.1.2) establishes a connection to virtual host 152.1.1.10.
7. RouterA receives the packet destined for the virtual host 152.1.1.10 and translates the destination address to next real host from the pool, in this case FTP server 152.1.1.2.
8. RouterA receives the response packet from FTP server 152.1.1.2, performs the NAT lookup, translates the source address to the virtual address, and forwards the packet.

Configuration Overview

This configuration will demonstrate load sharing using destination address rotary translation. RouterA will translate destination addresses of any packet that matches access list 2 using real host addresses from the rotary pool "loadsharing."

The pool defines the addresses of the real hosts, and the access−list defines the virtual address. If a translation does not already exist, TCP packets from serial 0 (the outside interface) whose destination address match access−list 2 are translated to an address from the pool.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. The IP addresses are assigned as per Figure 15−13. Secondary IP addresses are used on RouterA as test points only.

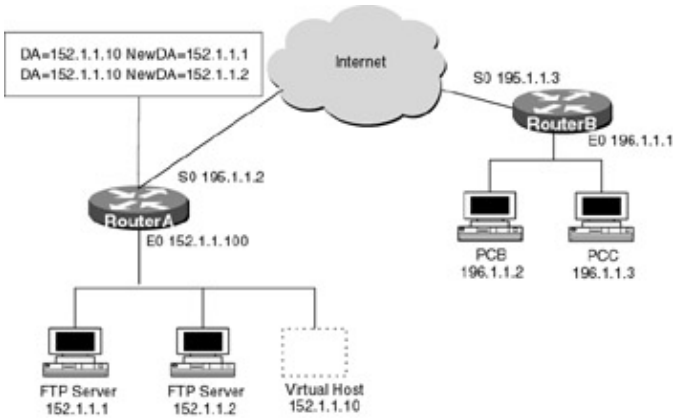


Figure 15−13: Destination address rotary translation
RouterA is configured for destination address rotary translation. From RouterB, telnet to virtual host 152.1.1.10. Instead of using multiple PCs off of the router's Ethernet, configure secondary IP addresses. RouterA will also be configured to allow VTY sessions, so that we can establish a telnet session to the secondary IP address on RouterA.

Router Configurations

The configurations for the two routers in this example are as follows (key NAT configurations for RouterA are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
                                                                    Defines
                                                                    the Pool
                  ↓ Pool Name    ↓ Pool Range                    ↓ as rotary
ip nat pool loadsharing 152.1.1.1 152.1.1.2 prefix-length 24 type rotary
ip nat inside destination list 2 pool loadsharing ← If the destination address
                              á References access      matches access list 2,
                                list 2                 replace with an IP
                                                       address from pool
                                                       "loadsharing"
!
interface Ethernet0
 ip address 152.1.1.1 255.255.255.0 secondary ← Secondary IP address used for
                                                 test point
ip address 152.1.1.2 255.255.255.0 secondary ← Secondary IP address used for
                                               test point
ip address 152.1.1.100 255.255.255.0
 ip nat inside ← Defines the inside interface
!
interface Serial0
 ip address 195.1.1.2 255.255.255.0
 ip nat outside ← Defines the Outside interface
!
no ip classless
access-list 2 permit 152.1.1.10 ← Defines what destination address will be
                                   translated
!
line con 0
line vty 0 4
 password cisco ← Sets the VTY password to cisco
 login ← Allows telnet access into the router
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
!
interface Ethernet0/0
  ip address 196.1.1.1 255.255.255.0
 !
interface Serial0/0
 ip address 195.1.1.3 255.255.255.0
 clockrate 500000 ← Acts as DCE, providing clock
```

Monitoring and Testing the Configuration

Perform the following steps to test the configuration:

1. On RouterA, enter the command **debug ip nat**.
2. On RouterB, telnet to IP address 152.1.1.10.

   The following is the output from the **debug ip nat** command on RouterA. The first line is the translation from destination 152.1.1.10 to the first address of the pool 152.1.1.1. The next line is the return packet from 152.1.1.1. Note that RouterA translated the source address to the virtual IP address 152.1.1.10 before forwarding the packet to RouterB.

   ```
   NAT: s=195.1.1.3, d=152.1.1.10->152.1.1.1 [0]
   NAT: s=152.1.1.1->152.1.1.10, d=195.1.1.3 [0]
   ```
3. On RouterB, telnet again to IP address 152.1.1.10.

   The following is the output from the **debug ip nat** command on RouterA. Note that this time destination address 152.1.1.10 is translated to the second address in the pool (152.1.1.2).

   ```
   NAT: s=195.1.1.3, d=152.1.1.10->152.1.1.2 [0]
   NAT: s=195.1.1.3, d=152.1.1.10->152.1.1.2 [0]
   ```
4. Show the NAT table on RouterA using the command **show ip nat translations.** The following is the output from the command. Note that each address is followed by the port number, and this combined with the protocol type is used as a key to translate the return packet back.

   ```
   Pro Inside global      Inside local       Outside local      Outside global
   tcp 152.1.1.10:23      152.1.1.2:23       195.1.1.3:26658    195.1.1.3:26658
   tcp 152.1.1.10:23      152.1.1.1:23       195.1.1.3:26146    195.1.1.3:26146
   ```

Changing Translation Timeouts

Dynamic translation will time out after a period of inactivity; by default, simple translation not configured for overloading will time out after 24 hours. To change the default timeout period, perform the following command in global configuration mode:

```
ip nat translation timeout  { seconds} ← Command changes the timeout value for
                                         dynamic address translations that do
                                         not use overloading
```

When overloading is configured, Cisco IOS allows finer control over translation entry timeouts because each entry contains more information about the traffic that is using it. The UDP, TCP, DNS, and finish timers shown here can be changed with the following global configuration commands:

```
ip nat translation udp-timeout {seconds}← Changes the UDP timeout value; the
                                          default is 5 minutes
ip nat translation dns-timeout {seconds} ← Changes the DNS timeout value; the
                                           default is 1 minute
ip nat translation tcp-timeout (seconds) ← Changes the TCP timeout value; the
                                           default is 24 hours
ip nat translation finrst-timeout (seconds) ← Changes the Finish and reset
                                              timeouts; the default is 1 minute
```

Troubleshooting NAT

The Cisco IOS provides many tools for troubleshooting Network Address Translation. What follows is a list of commands along with a sample output from each.

{show ip nat statistics} This command displays the number of active translations along with the number of translations that have expired. An expired translation is a translation that has been inactive for a period of time and has been removed from the table. The command also shows the inside and outside configured interfaces.

```
RouterA#show ip nat statistics
Total active translations: 0 (0 static, 0 dynamic; 0 extended)
Outside interfaces: Serial0
Inside interfaces: Ethernet0
Hits: 20  Misses: 20
Expired translations: 20
Dynamic mappings:
-- Inside Source
access-list 1 pool pool refcount 0
 pool pool: netmask 255.255.255.0
        start 195.1.1.1 end 195.1.1.1
        type generic, total addresses 1, allocated 0 (0%), misses 0
```

{show ip nat translations} This command displays all active translations, the protocol of the packet translated, the inside local address, the inside global address, the outside local address, and the outside global address.

From the following output, we can see that a ping packet (protocol icmp) with the inside local address of 10.1.1.1 has been translated to the inside global address of 195.1.1.1. The number after the IP address is the port number; this is used in this particular translation because the router is configured for overloading.

```
RouterA#show ip nat translations
Pro   Inside global      Inside local      Outside local      Outside global
icmp  195.1.1.1:4        10.1.1.1:4        195.1.1.3:4        195.1.1.3:4
icmp  195.1.1.1:3        10.1.1.1:3        195.1.1.3:3        195.1.1.3:3
icmp  195.1.1.1:2        10.1.1.1:2        195.1.1.3:2        195.1.1.3:2
icmp  195.1.1.1:1        10.1.1.1:1        195.1.1.3:1        195.1.1.3:1
icmp  195.1.1.1:0        10.1.1.1:0        195.1.1.3:0        195.1.1.3:0
```

{show ip nat translations verbose} This command is an extension of the previous command that displays more detailed information about how long ago the translation was created and how long ago the translation was last used.

From the following output, we can see that the translation was created 1 minute and 31 seconds ago and last used 31 seconds ago.

```
RouterA#show ip nat translations verbose
Pro   Inside global      Inside local       Outside local      Outside global
icmp  195.1.1.1:4        10.1.1.1:4         195.1.1.3:4        195.1.1.3:4
     create 00:01:31, use 00:00:31, left 00:00:28, flags: extended
icmp  195.1.1.1:3        10.1.1.1:3         195.1.1.3:3        195.1.1.3:3
     create 00:00:31, use 00:00:31, left 00:00:28, flags: extended
```

{clear ip nat translation} This command is used to clear all or specific active translations. The following is a list of extensions that can be used with this command.

*	The asterisk clears all dynamic translations
Inside	Clears specific inside address and port translations
Outside	Clears specific outside address and port translations
TCP	Clears specific inside address by protocol

UDP	Clear specific inside address by protocol

{clear ip nat statistics} This command is used to clear the counters for all NAT statistics.

{debug ip nat} This command is used to verify the operation of the NAT feature by displaying information about every packet that is translated by the router. The command will also display information about certain errors or exceptional conditions, such as the failure to allocate a global address.

From the following output of the command, we can see that the source address 10.1.1.1 has been translated to the global address 195.1.1.1.

```
NAT: s=10.1.1.1->195.1.1.1, d=195.1.1.3 [35]
```

Conclusion

This chapter explores Network Address Translation (NAT). NAT allows the addresses inside one stub domain to be reused by any other stub domain. NAT allows organizations to appear from the outside as if they are using different IP address space than what it is actually used, thereby reducing the need for unique, registered IP addresses. Network Address Translation can also save private network administrators from having to renumber hosts and routers that do not confirm to global IP addressing. NAT is defined in RFC 1631.

Chapter 16: Hot Standby Router Protocol

Overview

Topics covered in This Chapter

- Detailed HSRP Overview
- Basic HSRP Configuration (One HSRP Group)
- Basic HSRP Configuration Using the Track Option
- Multi−group HSRP Configuration
- Detailed Troubleshooting Examples

Introduction

This chapter will explore configuring and troubleshooting Cisco's Hot Standby Router Protocol (HSRP). HSRP provides high network availability by protecting against a single router failure. Without HSRP, the failure of a single default gateway router could isolate all hosts.

Overview

The majority of today's TCP/IP LAN networks rely on the use of a default gateway (which is statically configured in the host) in order to route packets to hosts on other networks. The default gateway is usually a router connected to the Internet or the company's intranet. Each host on the LAN is configured to forward packets to this destination if the host they are trying to reach is not on the same network. This provides for a single point of failure on the network; if the gateway is down, all the hosts on the LAN are isolated from the rest of the

network. To combat this problem, many companies install redundant gateways, the problem with which is that the user host is pointed at one gateway. If this router should fail, users must change their statically configured default gateway.

HSRP resolves this problem by allowing the network administrator to configure a set of routers to work together to present the appearance of a single default gateway. The routers in an HSRP group share a virtual Mac address and IP address, which is used by hosts on the LAN as the default gateway. The HSRP protocol selects which router is active; the active router receives and routes packets that are destined for the group's Mac address.

HSRP uses multicast UDP−based hello packets to communicate with other routers that are part of the same HSRP group. Each router in the group watches for hello packets from the active and standby routers. If the active router becomes unavailable, the standby will assume the active role and route the packets for the network.

Commands Discussed in This Chapter

debug standby

show standby

standby [*group−number*] ip [*ip−address* [secondary]]

standby [*group−number*] priority *priority−number*

standby [*group−number*] timers *hellotime holdtime* standby [*group−number*] preempt

standby [*group−number*] track *type number* [*interface−priority*]

standby ip: Used to activate HSRP.

Definitions

standby priority: Used to set the HSRP priority on an interface; the HSRP member with the highest priority (assuming preemption is enabled) becomes the active router.

standby timers: Used to configure the time between hello packets (hello time) and the amount of time after not hearing a hello packet from a HSRP neighbor that the router declares the neighbor down (holdtime).

standby preempt: Indicates that, when the local router's standby priority is higher than that of the current active router, the local router should attempt to assume control as the active router.

standby track: Used to configure an interface to change its HSRP priority according to the availability of another interface.

IOS Requirements

The HSRP feature set was first introduced in IOS 10.0.

Lab #75: Basic HSRP Configuration (One HSRP Group)

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers
- Cisco IOS 10.0 or higher
- A PC with an Ethernet NIC running a TCP/IP protocol stack
- One Ethernet hub
- Ethernet cables

Figure 16−1 shows a basic HSRP design where RouterA and RouterB are in one HSRP group. The hosts on the LAN use the default gateway of 192.1.1.10, which is a virtual IP address.

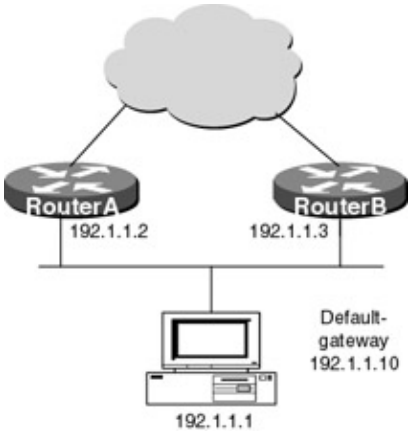


Figure 16−1: One HSRP group

The object of this configuration is to enable HSRP on both routers; if the active router fails, the standby router should take over and route for the 192.1.1.X network. No reconfiguration is needed on the hosts.

Router Configuration

The following are the configurations for RouterA and RouterB (key HSRP commands are highlighted in bold).

RouterA

```
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
!
interface Ethernet0
 ip address 192.1.1.2 255.255.255.0
 no ip redirects
 standby priority 200
 standby preempt
 standby ip 192.1.1.10
!
interface Serial0
 no ip address
 shutdown
 no fair−queue
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp−small−servers
service tcp−small−servers
!
hostname RouterB
!
```

```
!
interface Ethernet0
 ip address 192.1.1.3 255.255.255.0
 no ip redirects
 standby priority 150
 standby preempt
 standby ip 192.1.1.10
!
interface Serial0
 no ip address
 shutdown
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

In this configuration, RouterA is the active router because the priority is set to 200. If the standby router does not receive a hello packet from the active router in 10 seconds (which is the default holdtime), the standby router becomes active. To test this configuration, remove the Ethernet cable from RouterA. The following output is from the command **debug standby** on RouterB**.** Notice that the last incoming hello message was received from RouterA at 00:31:52; ten seconds after this, at 00:32:02, RouterB becomes the active router.

```
00:31:52: SB0:Ethernet0 Hello in 192.1.1.2 Active pri 200 hel 3 hol 10 ip 192.1.1.10
00:31:52: SB0:Ethernet0 Hello out 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
00:31:55: SB0:Ethernet0 Hello out 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
00:31:58: SB0:Ethernet0 Hello out 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
00:32:01: SB0:Ethernet0 Hello out 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
00:32:02: SB0: Ethernet0 state Standby -> Active
00:32:02: SB: Ethernet0 changing MAC address to 0000.0c07.ac00
```

Basic HSRP Configuration Usingthe Track Option

The Track option under HSRP allows the router to track other interfaces, so that if one of the other interfaces goes down, the device's Hot Standby priority is lowered. When the priority is lowered below that of the router in standby, the standby router becomes active. This is critical if the primary router loses its connection to the Internet or the company's intranet. Without tracking, the primary router would still receive packets from the hosts on the LAN even though it does not have a route to the outside world. If the two routers are running an Interior Gateway Protocol (IGP), the primary router will route the packets via the secondary router (however, no ICMP redirects will occur); if not, the packets will be dropped.

Figure 16−2 shows a basic HSRP design where RouterA and RouterB are in one HSRP group. RouterA is monitoring the status of its serial interface; with the use of the track command, when the interface goes down, RouterA lowers its priority and RouterB becomes the active router.

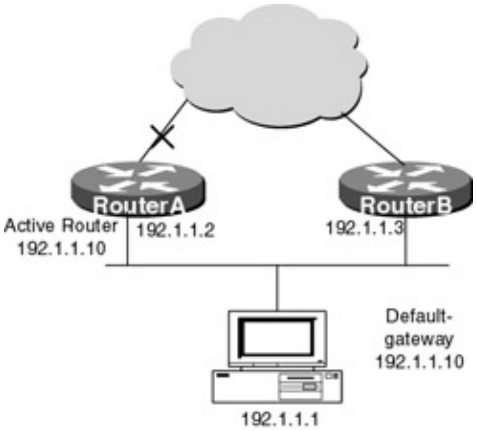


Figure 16−2: Basic HSRP

Router Configuration

The following is the configuration for RouterA to track the serial interfaces in case of failure.

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
interface Ethernet0
 ip address 192.1.1.2 255.255.255.0
 no ip redirects
 standby priority 200
 standby preempt
 standby ip 192.1.1.10
 standby track Serial0 51
!
interface Serial0
 ip address 172.1.1.1 255.255.255.252
 no fair−queue
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
 login
!
end
```

The **standby track serial 0 51** command decreases the priority of RouterA by 51 if the serial interface should go down. Remember from the previous configuration that the priority of RouterA was set 50 points higher than RouterB. The following debug output (**debug standby**) shows what will happen when serial 0 on RouterA fails.

At time 00:15:50, the serial interface on RouterA goes down, and at time 00:15:52 RouterA decreases the HSRP priority by 50 points, making it 149, and RouterA resigns as the active speaker.

```
 SB0:Ethernet0 Hello out 192.1.1.2 Active pri 200 hel 3 hol 10 ip 192.1.1.10
00:15:48: SB0:Ethernet0 Hello in 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial0, changed state to down
00:15:51: SB0:Ethernet0 Hello out 192.1.1.2 Active pri 200 hel 3 hol 10 ip 192.1.1.10
00:15:51: SB0:Ethernet0 Hello in 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
```

```
%LINK-3-UPDOWN: Interface Serial0, changed state to down
00:15:52: SB0: Ethernet0 Now 0/1 tracked interfaces up
00:15:52: SB0: Ethernet0 Priority was 200 now 149, configured as 200
00:15:52: SB0:Ethernet0 Hello out 192.1.1.2 Active pri 149 hel 3 hol 10 ip 192.1.1.10
00:15:52: SB0:Ethernet0 Coup in 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
00:15:52: SB0: Ethernet0 state Active -> Speak
00:15:52: SB0:Ethernet0 Resign out 192.1.1.2 Speak pri 149 hel 3 hol 10 ip 192.1.1.10
```

Lab #76: Multigroup HSRP Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers
- Cisco IOS 10.0 or higher
- Two PCs with Ethernet NICs running a TCP/IP protocol stack
- One Ethernet hub
- Ethernet cables

Overview

A router can be in multiple HSRP groups at one time. This allows the LAN administrator to load−balance across all of the routers while still providing redundancies in case of a failure. In Figure 16−3, each PC on the LAN uses a different default gateway; HostA uses default gateway 192.1.1.10, which is the IP address of HSRP group 1, and HostB uses 192.1.1.11, which is the IP address of HSRP group 2.

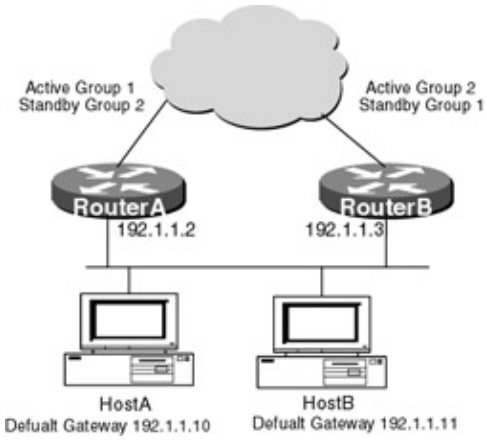


Figure 16−3: Multiple HSRP groups

When there are no failures, the traffic on the LAN is split equally between both routers. When a failure occurs, however, the standby router becomes active for that group and all traffic is routed to that interface.

Figure 16−3 shows two routers each in two standby groups; RouterA is active for group one and standby for group two, while RouterB is active for group two and standby for group one.

Router Configuration

The following are the configurations for RouterA and RouterB (key HSRP commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
interface Ethernet0
 ip address 192.1.1.2 255.255.255.0
 no ip redirects
 standby preempt
 standby 1 priority 200
 standby 1 ip 192.1.1.10
 standby 2 priority 150
 standby 2 ip 192.1.1.11
!
interface Serial0
 no ip address
 shutdown
 no fair-queue
!
no ip classless
!
line con 0
line 1 16
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0
 ip address 192.1.1.3 255.255.255.0
 no ip redirects
  standby preempt
  standby 1 priority 150
 standby 1 ip 192.1.1.10
 standby 2 priority 200
 standby 2 ip 192.1.1.11

!
interface Serial0
 no ip address
 shutdown
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

The following is the output from the **show standby** command on RouterA. Notice that RouterA's state is active for group 1 and standby for group 2.

```
RouterA#sho standby
Ethernet0/0 – Group 1
  Local state is Active, priority 200
  Hellotime 3 holdtime 10
  Next hello sent in 00:00:02.496
  Hot standby IP address is 192.1.1.10 configured
  Active router is local
  Standby router is 192.1.1.3 expires in 00:00:08
Ethernet0/0  – Group 2
  Local state is Standby, priority 150
  Hellotime 3 holdtime 10
  Next hello sent in 00:00:02.496
  Hot standby IP address is 192.1.1.11 configured
  Active router is 192.1.1.3 expires in 00:00:07
  Standby router is local
```

Troubleshooting HSRP

HSRP has only one debug command, which is **debug standby;** however, the output of this command will tell you a lot.

{debug standby} From the Cisco command line, enter **debug standby**. The following is sample output from this command. The debug shows the source of the hello packet, whether it was received or sent, and on what interface. It also shows the active and standby routers as well as priority and hello and hold timers.

From this debug message, you can tell that RouterA is the active router with a priority of 200 and that RouterB is sending hello messages and is the standby router with a priority of 150.

```
RouterA#debug standby
Hot standby protocol debugging is on
SB0:Ethernet1/0 Hello out 192.1.1.2 Active pri 200 hel 3 hol 10 ip 192.1.1.10
SB0:Ethernet1/0 Hello in 192.1.1.3 Standby pri 150 hel 3 hol 10 ip 192.1.1.10
```

{show standby} From the Cisco command line, enter the **show standby** command. The following is sample output from this command. The command shows the local state of the router, whether it is active or standby, the router's priority, and whether or not it can preempt. It also shows the hellotime, the holdtime, the standby IP address, and the address of the standby router.

```
RouterA#sho standby
Ethernet1/0 – Group 0
  Local state is Active, priority 200, may preempt
  Hellotime 3 holdtime 10
  Next hello sent in 00:00:01
  Hot standby IP address is 192.1.1.10 configured
  Active router is local
  Standby router is 192.1.1.3 expires in 00:00:09
```

Conclusion

The HSRP provides high network availability and is very useful for hosts that do not support routing protocols and cannot automatically switch to a new default gateway (router) if the primary router fails.

Chapter 17: Network Time Protocol

Overview

Topics Covered in This Chapter

- NTP technology overview
- Configuring a Cisco router as an NTP time server
- Configuring NTP peers
- Cisco NTP authentication
- Configuring Cisco NTP for LAN broadcasts

Introduction

Network Time Protocol (NTP) is a TCP/IP protocol that is designed to distribute accurate time throughout a network. NTP uses UDP as its transport mechanism. This chapter will explore the NTP capabilities available through the Cisco IOS.

Network Time Protocol (NTP) Overview

Every Cisco router has a system clock that can store the current date and time. NTP addresses the problem of how to synchronize all routers in a network to a common clock source. In addition to synchronizing all routers, NTP can be used to synchronize all system clocks in a given network. This applies to all workstations as well as any other systems that have a system clock. NTP client software is available for a wide variety of workstations and servers. There are several reasons why it is important that all routers in a network share a common time:

- Debug and event timestamps: Debug and event timestamps collected from several different routers are meaningful unless they are all referenced to a common time.
- Transaction processing: These types of transactions need to be accurately timestamped.
- Simulation: Complex transactions are often divided between multiple systems. Common clocks are needed between these multiple systems to ensure a proper sequence of events.
- System maintenance: Performing a function such as reloading all routers in a network at the same time requires a common clock across the entire network.

NTP will usually be able to synchronize all system clocks within 10 milliseconds of each other over a wide area network.

How Does NTP Work?

This section will give a high−level simplified overview of how NTP works. NTP is fully described in RFC 1305.

NTP addresses the basic problem shown in Figure 17−1. Two routers, RouterA and RouterB, are connected via a serial link. Both RouterA and RouterB have independent system clocks. How can we devise a protocol to automatically synchronize the system clocks of both routers?

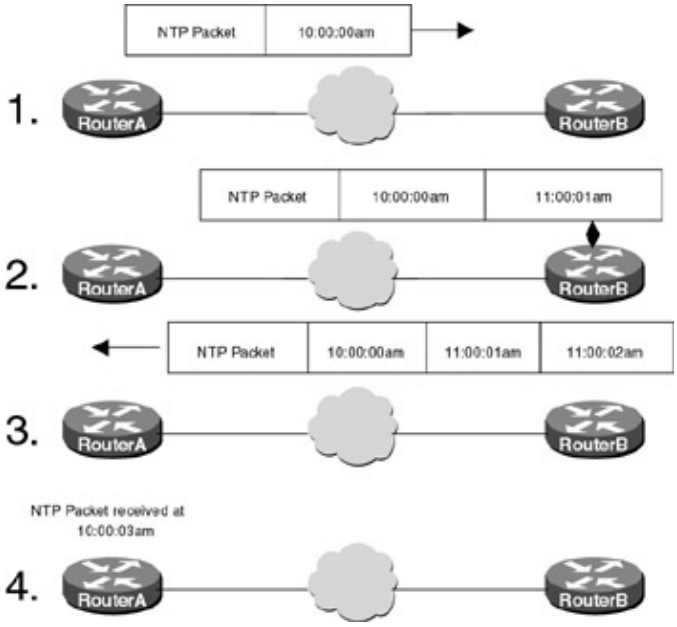


Figure 17−1: NTP functional overview
Let's assume the following:

- Before RouterA and RouterB synchronize their system clocks, RouterA's clock is set for 10 A.M. and RouterB's clock is set for 11 A.M.
- RouterB will be our time server, meaning that RouterA will set its clock to match the clock of RouterB.
- The delay in getting a packet from RouterA to RouterB is 1 second in each direction.
- The delay in getting a packet through RouterB is also 1 second. Clock synchronization occurs in the following way:

    1. RouterA sends an NTP packet to RouterB, which has a timestamp corresponding to the time the packet left RouterA. This timestamp will be 10:00:00 A.M.
    2. RouterB provides a timestamp when the packet arrives. This timestamp will be 11:00:01 A.M.
    3. RouterB provides another timestamp when the packet leaves for RouterA. This timestamp will be 11:00:02 A.M.
    4. When the response packet is received at RouterA, RouterA provides another timestamp. This timestamp will be 10:00:03 A.M.

  RouterA now has enough information to calculate two important items:
- The round−trip delay of the packet exchange
- The difference in clocks between RouterA and RouterB

RouterA will now be able to set its clock to agree with RouterB's clock.

One should keep in mind that this is just a high level overview of how NTP works. As outlined in RFC 1305, NTP uses complex algorithms to ensure clock accuracy.

NTP Implementation

In a real−world application, you would usually not use a router as an NTP server. The accuracy of the clock on a router is far lower than can be achieved using commercial time products. One such product is the DATUM Tymserve 2000 Network Time Server.

This device is a combination of a GPS receiver and an NTP server. It receives stratum 1 clocking signals from the GPS network. The Tymserve is connected to an Ethernet LAN where it can respond to NTP requests from peers and clients.

The low cost of these types of devices can also allow for distributed clocking systems where a Tymserve 2000 is located locally, such as on a college campus, eliminating the need for clock synchronization over the wide area.

Commands Discussed in This Chapter

- **ntp access−group {query−only | serve−only | serve | peer}** *access−list number*
- **ntp authenticate**
- **ntp authentication−key** *number* **md5** *value*
- **ntp broadcast [version** *number***]**
- **ntp broadcast client**
- **ntp broadcast delay** *microseconds*
- **ntp clock period** *value*
- **ntp disable**
- **ntp master [***stratum***]**
- **ntp peer** *ip−address* **[version** *number***] [key** *keyid***] [source** *interface***] [prefer]**
- **ntp server** *ip−address* **[version** *number***] [key** *keyid***] [source** *interface***] [prefer]**
- **ntp source** *type number*
- **ntp trusted−key** *key−number*
- **ntp update−calendar**
- **show ntp status**
- **show ntp association**
- **show ntp association detail**

Definitions

ntp access−group: This global command is used to control access to the routers NTP services.

ntp authenticate: Use this global command to enable NTP authentication.

ntp authentication−key: This global command defines an authentication key for use with NTP.

ntp broadcast: This interface command is used to specify that a specific interface should send NTP broadcast packets.

ntp broadcast client: This interface command allows the router to receive NTP broadcast packets on the specified interface.

ntp broadcastdelay: This global command is used to set the estimated round−trip delay between the router and the NTP server.

ntp clock−period: THIS GLOBAL COMMAND SHOULD NOT BE ENTERED. THE ROUTER WILL AUTOMATICALLY GENERATE THIS COMMAND WHEN NTP SYNCHRONIZES THE SYSTEM.

ntp disable: Prevents the specified interface from receiving NTP packets.

ntp master: This global command is used to configure the router as an NTP clock master. This command should only be used when an external NTP source is not available or for test purposes.

ntp peer: This global command causes the routers system clock to synchronize to a peer or to synchronize a peer.

ntp server: This global command causes the router's system clock to be synchronized by a time server.

ntp source: This command will force the router to use a particular source address in its NTP packets.

ntp trusted−key: This global command is used to authenticate the router to a specific authentication key.

ntp update−calendar: This global command causes NTP to periodically update the calendar on a Cisco 7XXX series router.

show ntp status: This exec command is used to display NTP information for the router. This command can show if the router is synchronized to an NTP peer or an NTP server.

show ntp association [detail]: This exec command displays NTP information such as polling cycles.

IOS Requirements

The labs in this chapter were done using IOS version 11.2. Most of the NTP functionality that is discussed in this chapter can be tested using IOS 10.0 and above. The reader should check the Cisco Web site bug reports for their particular IOS. Some IOS versions have NTP functionality issues.

Lab #77: Cisco NTP Using Time Servers

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable.

Configuration Overview

This configuration will demonstrate two Cisco routers synchronizing their time clocks to a Cisco router acting as an NTP time server. Three Cisco routers are connected serially. RouterA is connected to RouterB via a crossover cable. RouterB is connected to RouterC via a crossover cable. RouterB acts as a DCE, supplying clock to both RouterA and RouterC. IP addresses are assigned as per diagram in Figure 17−2. A PC running a terminal emulation program is connected to the console port of RouterA.

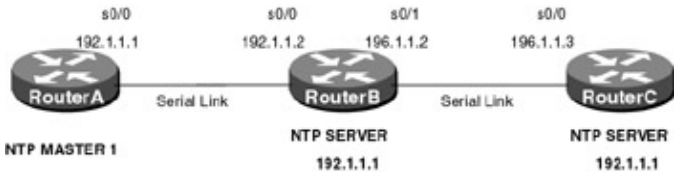


Figure 17−2: NTP time server lab

RouterA is configured as the NTP clock master. Both RouterB and RouterC are configured to synchronize to RouterA via the **ntp server** statement.

Note NTP convergence can take up to half an hour. This means that when changing the system clock on the NTP master, it can take up to half an hour for all other clocks in the configuration to synchronize. This is caused by NTP viewing a clock change as an instability in the clocking system. NTP waits for a stable system before synchronizing and propagating any changes.

Note A Cisco router will not have a valid date and time set when it is powered on. After power on, the clock

gets set to March 1, 1993. The clock must be given a valid setting before NTP takes effect. The clock can be set via the clock set command whose syntax is as follows:clock set hh:mm:ss day month year

Router Configuration

The configurations for the three routers in this example are as follows (key NTP commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
no ip domain-lookup
!
interface Ethernet0/0
 ip address 10.10.3.100 255.255.255.0
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation ppp
!
router rip
 network 10.0.0.0
 network 192.1.1.0
!
ip classless
!
line con 0
 exec-timeout 120 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
ntp master 1
end
```

RouterB

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
no ip domain-lookup
!
```

```
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation ppp
 clockrate 500000
!
interface Serial0/1
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
 clockrate 19200
!
router rip
 network 192.1.1.0
 network 196.1.1.0
!
ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
ntp clock-period 17179866
ntp server 192.1.1.1
end
```

RouterC

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
no ip domain-lookup
!
interface Serial0/0
 ip address 196.1.1.3 255.255.255.0
 encapsulation ppp
!
router rip
 network 196.1.1.0
!
no ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 30 0
 password cisco
 login
!
ntp clock-period 17179864
ntp server 192.1.1.1
end
```

Monitoring the Configuration

After RouterA has converged and stabilized its new clock time, it will start to propagate its clock settings. The **show ntp status** command can be used to monitor the synchronization state of each router.

The **show ntp status** output from RouterA shows that the router is synchronized. Notice that the reference is listed as being local since this router is configured as a clock master. Because we had the command **ntp master 1** on RouterA, the router declares itself as a stratum 1 source.

```
RouterA#sh ntp status
Clock is synchronized, stratum 1, reference is .LOCL.
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA879B74.9051655D (11:28:52.563 UTC Wed Mar 3 1999)
clock offset is 0.0000 msec, root delay is 0.00 msec
root dispersion is 0.02 msec, peer dispersion is 0.02 msec
RouterA#
```

The **show ntp status** output from routers RouterB and RouterC are shown below. Notice that both routers are in a synchronized state. Each router is reported to be operating at stratum 2. This is correct since they are synchronized to a stratum 1 source. Finally, notice that the clock reference is listed as being 192.1.1.1. This is the interface address of RouterA, our NTP master.

```
RouterB#sh ntp status
Clock is synchronized, stratum 2, reference is 192.1.1.1
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA879BA3.91C82791 (11:29:39.569 UTC Wed Mar 3 1999)
clock offset is −2.0598 msec, root delay is 1.97 msec
root dispersion is 3.02 msec, peer dispersion is 0.93 msec

RouterC#sh ntp status
Clock is synchronized, stratum 2, reference is 192.1.1.1
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA879B9C.90A0FB53 (11:29:32.564 UTC Wed Mar 3 1999)
clock offset is −0.7317 msec, root delay is 74.55 msec
root dispersion is 2.52 msec, peer dispersion is 1.75 msec
```

Another useful command is the **show ntp associations** command. The output for this command is shown below for each of the three routers. This command shows how often the router sends/receives NTP updates. It also shows when the last update was received. RouterB, for example, last received an NTP update 14 seconds ago.

```
RouterA#sh ntp associations

      address         ref clock     st  when  poll reach  delay  offset    disp
*~127.127.7.1      .LOCL.            0    43    64  377     0.0    0.00     0.0
 * master (synced), # master (unsynced), + selected, − candidate, ~ configured


RouterB#sh ntp associations

      address         ref clock     st  when  poll reach  delay  offset    disp
*~192.1.1.1        .LOCL.            1    14    64  377     2.0   −2.06     0.9
 * master (synced), # master (unsynced), + selected, − candidate, ~ configured


RouterC#sh ntp assoc

      address         ref clock     st  when  poll reach  delay  offset    disp
*~192.1.1.1        .LOCL.            1    39    64  377    74.6   −0.73     1.8
 * master (synced), # master (unsynced), + selected, − candidate, ~ configured
```

Lab #78: Cisco NTP Using Time Servers and Peers

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- Two Cisco DTE/DCE crossover cables. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable.

Configuration Overview

This configuration will demonstrate one Cisco router synchronizing its time clock to a Cisco router NTP time server. A second Cisco router will synchronize its clock to the Cisco NTP server as a peer connection. Three Cisco routers are connected serially. RouterA is connected to RouterB via a crossover cable. RouterB is connected to RouterC via a crossover cable. RouterB acts as a DCE, supplying clock to both RouterA and RouterC. IP addresses are assigned as shown in Figure 17−3. A PC running a terminal emulation program is connected to the console port of RouterA.

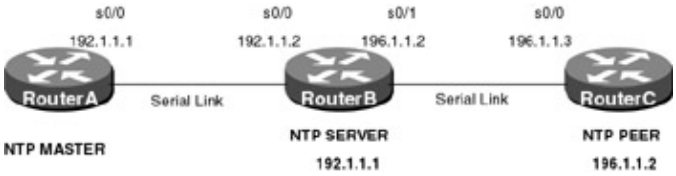


Figure 17−3: NTP server and peer lab

RouterA is configured as the NTP clock master. RouterB is configured to synchronize to RouterA via the **ntp server** statement. RouterC is configured to synchronize to RouterB as a peer. Since RouterB is already synchronized to RouterA, RouterC will always synchronize to RouterB.

Note NTP convergence can take up to half an hour. This means that when changing the system clock on the NTP master, it can take up to half an hour for all other clocks in the configuration to synchronize. This is caused by NTP viewing a clock change as an instability in the clocking system. NTP waits for a stable system before synchronizing and propagating any changes.

Note A Cisco router will not have a valid date and time set when it is powered on. After power on, the clock gets set to March 1, 1993. The clock must be given a valid setting before NTP takes effect. The clock can be set via the clock set command, with syntax is as follows:clock set hh:mm:ss day month year

Router Configuration

The configurations for the three routers in this example are as follows (key NTP commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
```

```
!
no ip domain-lookup
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation ppp
!
router rip
network 192.1.1.0
!
ip classless
!
line con 0
 exec-timeout 120 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
ntp master 1
end
```

RouterB

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
no ip domain-lookup
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation ppp
 clockrate 500000
!
interface Serial0/1
 ip address 196.1.1.2 255.255.255.0
 encapsulation ppp
 clockrate 19200
!
router rip
 network 192.1.1.0
 network 196.1.1.0
!
ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
```

```
!
ntp clock-period 17179854
ntp server 192.1.1.1
end
```

RouterC

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
no ip domain-lookup
!
interface Serial0/0
 ip address 196.1.1.3 255.255.255.0
 encapsulation ppp
!
router rip
 network 196.1.1.0
!
no ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 30 0
 password cisco
 login
!
ntp clock-period 17179866
ntp peer 196.1.1.2
end
```

Monitoring the Configuration

After RouterA has converged and stabilized its new clock time, it will start to propagate its clock settings. The **show ntp status** command can be used to monitor the synchronization state of each router.

The **show ntp status** output from RouterA shows that the router is synchronized. Notice that the reference is listed as being local since this router is configured as a clock master. Because we had the command **ntp master 1** on RouterA, the router declares itself as a stratum 1 source.

```
RouterA#sh  ntp status
Clock is synchronized, stratum 1, reference is .LOCL.
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA879FB4.904CEDDC (11:47:00.563 UTC Wed Mar 3 1999)
clock offset is 0.0000 msec, root delay is 0.00 msec
root dispersion is 0.02 msec, peer dispersion is 0.02 msec
```

Notice that the **show ntp status** output from routers RouterB and RouterC are not identical. RouterB is synchronized to RouterA. RouterB is listed as being a stratum 2 source. Its reference is **192.1.1.1** (RouterA). RouterC is listed as a stratum 3 source. This is due to RouterC synchronizing to RouterB. Notice that RouterC is using RouterB as its reference (**196.1.1.2**).

```
RouterB#sh ntp status
Clock is synchronized, stratum 2, reference is 192.1.1.1
nominal freq is 250.0000 Hz, actual freq is 250.0002 Hz, precision is 2**24
reference time is BA879FA0.8FD33F2F (11:46:40.561 UTC Wed Mar 3 1999)
clock offset is −2.7416 msec, root delay is 1.94 msec
root dispersion is 4.50 msec, peer dispersion is 1.72 msec
RouterC#sh ntp status
Clock is synchronized, stratum 3, reference is 196.1.1.2
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA879FE4.8DF8FEFE (11:47:48.554 UTC Wed Mar 3 1999)
clock offset is −0.8742 msec, root delay is 70.74 msec
root dispersion is 4.84 msec, peer dispersion is 0.31 msec
```

Let's take a look at the **show ntp associations** command as shown in the following screen prints. This command shows how often the router sends/receives NTP updates. It also shows when the last update was received. RouterC for example last received an NTP update 10 seconds ago.

```
RouterA#sh ntp associations

      address         ref clock     st  when  poll reach  delay  offset    disp
*~127.127.7.1      .LOCL.            0    25    64  377     0.0    0.00     0.0
 * master (synced), # master (unsynced), + selected, − candidate, ~ configured

RouterB#sh ntp associations

      address         ref clock     st  when  poll reach  delay  offset    disp
*~192.1.1.1        .LOCL.            1    61    64  377     1.9   −2.74     1.7
 * master (synced), # master (unsynced), + selected, − candidate, ~ configured

RouterC#sh ntp associations

      address         ref clock     st  when  poll reach  delay  offset    disp
*~196.1.1.2        192.1.1.1         2    10    64  377    68.8   −0.87     0.3
 * master (synced), # master (unsynced), + selected, − candidate, ~ configured
```

Lab #79: Cisco NTP with Authentication

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having two serial ports
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable.

Configuration Overview

This configuration will demonstrate the authentication features of NTP. Cisco's NTP implementation includes powerful authentication capabilities that ensure that NTP updates are being sent from a trusted source. Two Cisco routers are connected serially. RouterA is connected to RouterB via a crossover cable. RouterB acts as a DCE, supplying clock to RouterA. IP addresses are assigned as shown in Figure 17−4. A PC running a terminal emulation program is connected to the console port of RouterA.

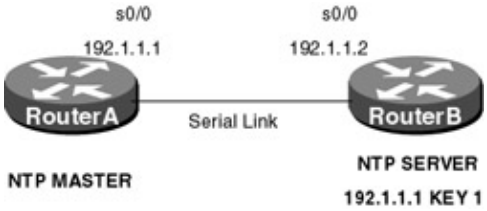


Figure 17−4: NTP authentication

RouterA is configured as the NTP clock master. RouterB is configured to synchronize to RouterA via the **ntp server** statement. RouterA and RouterB are configured for NTP authentication. This authentication uses MD5 security to ensure the validity of NTP packets being sent between the two routers.

Note NTP convergence can take up to half an hour. This means that when changing the system clock on the NTP master, it can take up to half an hour for all other clocks in the configuration to synchronize. This is caused by NTP viewing a clock change as an instability in the clocking system. NTP waits for a stable system before synchronizing and propagating any changes.

Note This exercise should be tried in two steps. First enter the configurations for RouterA and RouterB, with both RouterA and RouterB having a different authentication key value. The command syntax is:

```
ntp authentication−key number md5 value.
```

Note As an example, enter the command for RouterA as ntp authentication−key 1 md5 cisco and enter the command for RouterB as ntp authentication−key 1 md5 bay. Doing this will cause the authentication to fail between the two routers. This will demonstrate how an authentication failure will cause NTP to not work between the two routers. Next, change the two authentication keys to the same value for both routers and verify that the two routers do synchronize their time via NTP. The authentication key will appear encrypted as soon as it is typed in. This will cause a clear−text key such as cisco to appear in the following manner when viewing the configuration via the show run command:

```
ntp authentication−key 1 md5 045802150C2E 7
```

Note A Cisco router will not have a valid date and time set when it is powered on. After power on, the clock gets set to March 1, 1993. The clock must be given a valid setting before NTP takes effect. The clock can be set via the clock set command with syntax is as follows:

```
clock set hh:mm:ss day month year
```

Router Configuration

The configurations for the two routers in this example are as follows (key NTP commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
enable password cisco
!
no ip domain−lookup
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation ppp
```

```
!
router rip
 network 192.1.1.0
!
ip classless
!
line con 0
 exec-timeout 120 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
ntp authentication-key 1 md5 045802150C2E 7
ntp authenticate
ntp trusted-key 1
ntp master
end
```

RouterB

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
no ip domain-lookup
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation ppp
 clockrate 500000
!
router rip
 network 192.1.1.0
!
ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
ntp authentication-key 1 md5 121A0C041104 7
ntp authenticate
ntp trusted-key 1
ntp clock-period 17179827
ntp server 192.1.1.1 key 1
end
```

Monitoring the Configuration

After RouterA has converged and stabilized its new clock time, it will start to propagate its clock settings. The **show ntp status** command can be used to monitor the synchronization state of each router.

The **show ntp status** output from router RouterA shows that the router is synchronized. Notice that the reference is listed as being local since this router is configured as a clock master. RouterA declares itself as a stratum 8 source since we did not specify any stratum in the **ntp master** command.

```
RouterA#sh ntp status
Clock is synchronized, stratum 8, reference is 127.127.7.1
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA2DE8FD.D1BC0A57 (10:35:41.819 UTC Fri Dec 25 1998)
clock offset is 0.0000 msec, root delay is 0.00 msec
root dispersion is 0.02 msec, peer dispersion is 0.02 msec
```

RouterB is also synchronized. It is a stratum 9 source since it is referenced to RouterA, which is a stratum 8 source.

```
RouterB#sh ntp status
Clock is synchronized, stratum 9, reference is 192.1.1.1
nominal freq is 250.0000 Hz, actual freq is 250.0006 Hz, precision is 2**24
reference time is BA2DE912.F26F9533 (10:36:02.947 UTC Fri Dec 25 1998)
clock offset is 0.1931 msec, root delay is 0.73 msec
root dispersion is 1.79 msec, peer dispersion is 1.57 msec
```

Let's take a look at the **show ntp associations** command as shown in the following screen prints. This command shows how often the router sends/receives NTP updates. It also shows when the last update was received. RouterB, for example, last received an NTP update 27 seconds ago.

```
RouterA#sh ntp assoc

      address         ref clock     st  when  poll reach  delay  offset    disp
*~127.127.7.1      127.127.7.1       7    20    64  377     0.0    0.00     0.0
 * master (synced), # master (unsynced), + selected, - candidate, ~ configured


RouterB#sh ntp assoc

      address         ref clock     st  when  poll reach  delay  offset    disp
*~192.1.1.1        127.127.7.1       8    27   128  377     0.7    0.19     1.6
 * master (synced), # master (unsynced), + selected, - candidate, ~ configured
```

The **ntp association detail** command contains additional important information regarding the current NTP status. For example, the following screen print shows the time that is being broadcast to other systems highlighted in bold.

```
RouterA#sh ntp assoc detail
127.127.7.1 configured, our_master, sane, valid, stratum 7
ref ID 127.127.7.1, time BA2DE8FD.D1BC0A57 (10:35:41.819 UTC Fri Dec 25 1998)
our mode active, peer mode passive, our poll intvl 64, peer poll intvl 64
root delay 0.00 msec, root disp 0.00, reach 377, sync dist 0.015
delay 0.00 msec, offset 0.0000 msec, dispersion 0.02
precision 2**24, version 3
org time BA2DE8FD.D1BC0A57 (10:35:41.819 UTC Fri Dec 25 1998)
rcv time BA2DE8FD.D1BC0A57 (10:35:41.819 UTC Fri Dec 25 1998)
xmt time BA2DE8FD.D1BDEB65 (10:35:41.819 UTC Fri Dec 25 1998)
filtdelay =     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
filtoffset =    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
filterror =     0.02    0.99    1.97    2.94    3.92    4.90    5.87    6.85
Reference clock status:  Running normally
Timecode:
```

Lab #80: Cisco NTP Using LAN Broadcasts

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers, each having one Ethernet port
- Cisco IOS 10.0 or higher
- A PC running a terminal emulation program
- An Ethernet hub
- Three Ethernet cables connecting each router to the Ethernet hub
- An optional LAN sniffer, which is connected into the Ethernet hub. This will allow traces to be taken that will show the NTP packets being sent on the network.

Configuration Overview

This configuration demonstrates the broadcast capabilities of NTP. NTP updates will be broadcast on an Ethernet LAN to two Cisco routers. We will see that the broadcast configuration is less complex than the previous configurations due to the fact that NTP peer and server IP addresses are no longer needed. Three Cisco routers are all connected to the same Ethernet LAN. All three routers' IP addresses reside on the same network. IP addresses are assigned as shown in Figure 17−5. A PC running a terminal emulation program is connected to the console port of RouterA. An optional LAN sniffer can be connected on the LAN to allow for capture and analysis of the NTP packets.

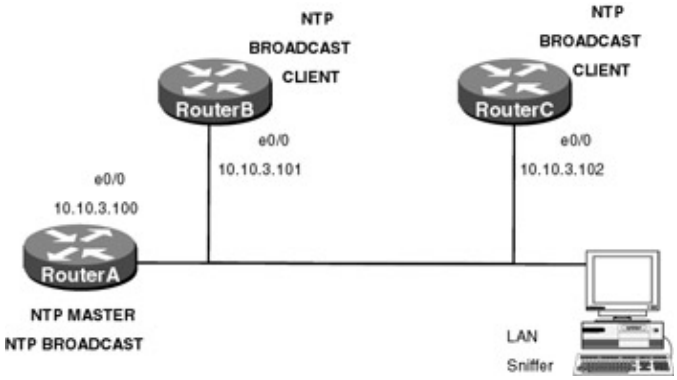


Figure 17−5: NTP LAN broadcast lab

RouterA is configured as the NTP clock master. RouterB is configured to synchronize to RouterA via the **ntp server** statement. RouterA and RouterB are configured for NTP authentication. This authentication uses MD5 security to ensure the validity of NTP packets being sent between the two routers.

Note NTP convergence can take up to half an hour. This means that when changing the system clock on the NTP master, it can take up to half an hour for all other clocks in the configuration to synchronize. This is caused by NTP viewing a clock change as an instability in the clocking system. NTP waits for a stable system before synchronizing and propagating any changes.

Note A Cisco router will not have a valid date and time set when it is powered on. After power on, the clock gets set to March 1 1993. The clock must be given a valid setting before NTP takes effect. The clock can be set via the clock set command, with syntax is as follows: **clock set** hh:mm:ss day month year

Note Due to IOS issues, the ntp broadcast statement under the Ethernet interface of RouterA must be reentered every time the clock is changed.

NTP Packet Capture

The following trace was taken with a Network Associates Sniffer Ethernet analyzer. It shows a completely decoded NTP packet that was sent on the LAN of this example.

```
Packet 5 captured at 11/28/1998 11:35:30 PM; Packet size is 90(0x5a)bytes
        Relative time: 000:00:49.927
        Delta time: 37.671.943
  Ethernet Version II
        Address: 00-E0-1E-5B-0A-81 --->FF-FF-FF-FF-FF-FF
        Ethernet II Protocol Type: IP
  Internet Protocol
        Version(MSB 4 bits): 4
        Header length(LSB 4 bits): 5 (32-bit word)
        Service type: 0x00
              000.  . . . . = 0 - Routine
               . . . 0  . . . . = Normal delay
               . . . . 0 . . .  = Normal throughput
               . . . . .0.. = Normal reliability
        Total length: 76 (Octets)
        Fragment ID: 1278
        Flags summary: 0x00
              0 . . .   . . . . = Reserved
              .0..  . . . . = May be fragmented
              ..0.  . . . . = Last fragment
              Fragment offset(LSB 13 bits): 0 (0x00)
        Time to live: 255 seconds/hops
        IP protocol type: UDP (0x11)
        Checksum: 0xA935
        IP address 10.10.3.100 ->BROADCAST
        No option
  User Datagram Protocol
        Port Network Time Protocol ---> Network Time Protocol
        Total length: 56 (Octets)
        Checksum: 0xF9EF
  Network Time Protocol
        Leap Indicator: 0 - No Warning
        Version Number: 3
        Mode: 5 - Broadcast
        Stratum: 8 - Secondary Reference
        Poll Interval: 6 (Sec)
        Precision: 232 (Sec)
        Root Delay: 0.0 (Sec.200PicoSec)
        Root Dispersion: 0.2 (Sec.200PicoSec)
        Reference Source Address: 127.127.7.1
        Reference Timestamp: 3121240502.2318470671 (Sec.200PicoSec)
        Originate Timestamp: 0.0 (Sec.200PicoSec)
        Receive Timestamp: 0.0 (Sec.200PicoSec)
        Transit Timestamp: 3121240534.2318467748 (Sec.200PicoSec)
```

Router Configuration

The configurations for the three routers in this example are as follows (key NTP commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
```

```
!
enable password cisco
!
no ip domain-lookup
!
interface Ethernet0/0
 ip address 10.10.3.100 255.255.255.0
 ntp broadcast
!
ip classless
!
line con 0
 exec-timeout 120 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
ntp master
end
```

RouterB

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
no ip domain-lookup
!
interface Ethernet0/0
 ip address 10.10.3.101 255.255.255.0
 ntp broadcast client
!
ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
```

```
!
hostname RouterC
!
!
no ip domain-lookup
!
interface Ethernet0/0
 ip address 10.10.3.102 255.255.255.0
 ntp broadcast client
!
no ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
exec-timeout 30 0
 password cisco
 login
!
end
```

Monitoring the Configuration

After RouterA has converged and stabilized its new clock time, it will start to propagate its clock settings. The **show ntp status** command can be used to monitor the synchronization state of each router.

The **show ntp status** output from router RouterA shows that the router is synchronized.

```
RouterA#sh ntp status
Clock is synchronized, stratum 8, reference is 127.127.7.1
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA374A04.95680411 (13:20:04.583 UTC Fri Jan 1 1999)
clock offset is 0.0000 msec, root delay is 0.00 msec
root dispersion is 0.02 msec, peer dispersion is 0.02 msec
RouterB#sh ntp status
Clock is synchronized, stratum 9, reference is 10.10.3.100
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA374A2B.9511C731 (13:20:43.582 UTC Fri Jan 1 1999)
clock offset is 3.4407 msec, root delay is 0.93 msec
root dispersion is 382.71 msec, peer dispersion is 379.24 msec
RouterC#sh ntp status
Clock is synchronized, stratum 9, reference is 10.10.3.100
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**24
reference time is BA374A2B.96B65B60 (13:20:43.588 UTC Fri Jan 1 1999)
clock offset is -4.7316 msec, root delay is 0.78 msec
root dispersion is 134.98 msec, peer dispersion is 130.20 msec
```

Let's take a look at the **show ntp associations** command as shown in the following screen prints. This command shows how often the router sends/receives NTP updates. It also shows when the last update was received. RouterC, for example, last received an NTP update 8 seconds ago.

```
RouterC#sh ntp assoc

      address         ref clock     st  when  poll reach  delay  offset    disp
* 10.10.3.100      127.127.7.1       8    52    64  376     0.8   -4.73   130.2
 * master (synced), # master (unsynced), + selected, - candidate, ~ configured
```

The **ntp association detail** command contains additional important information regarding the current NTP status. For example, the following screen print shows the time that is being broadcast to other systems, highlighted in bold.

```
RouterC#sh ntp assoc detail
10.10.3.100 dynamic, our_master, sane, valid, stratum 8
ref ID 127.127.7.1, time BA374A04.95680411 (13:20:04.583 UTC Fri Jan 1 1999)
our mode bdcast client, peer mode bdcast, our poll intvl 64, peer poll intvl 64
root delay 0.00 msec, root disp 0.03, reach 376, sync dist 130.615
delay 0.78 msec, offset −4.7316 msec, dispersion 130.20
precision 2**24, version 3
org time BA374A2B.9562442A (13:20:43.583 UTC Fri Jan 1 1999)
rcv time BA374A2B.96B65B60 (13:20:43.588 UTC Fri Jan 1 1999)
xmt time BA374983.F15A0680 (13:17:55.942 UTC Fri Jan 1 1999)
filtdelay =     0.78    0.78    2.43    0.78    2.73    1.17    3.08    0.00
filtoffset =   −4.73   −0.25   −7.38   −0.27   −7.28   −0.11   −7.15    0.00
filterror =     0.99    1.97    2.58    2.59    2.61    2.62    2.64 16000.0
```

Conclusion

This chapter explores the Network Time Protocol (NTP). NTP is used to synchronize all systems in a network to the same clock. There are several reasons why clock synchronization is important, such as being able to debug and timestamp events at the same time for all systems in a network.

NTP is defined in RFC 1305 and uses a simple method to determine both the round−trip delay between two systems as well as the time difference between two systems.

This chapter contains detailed lab exercises using a Cisco router as an NTP server. NTP servers, peers, authentication, and broadcasting are presented.

In the real world, a stand−alone GPS referenced NTP server can be used to provide network synchronization.

Chapter 18: Novell IPX

Overview

Topics Covered in This Chapter

- IPX technology overview
- IBP over Frame Relay
- IPX configuration using RIP/SAP
- Configuring EIGRP on an IPX network
- IPX dial backup
- Static SAP entries
- SAP access lists
- Troubleshooting IPX

Introduction

Novell IPX, although less popular than it once was, is still a widely deployed networking protocol. This chapter will explore the IPX protocol and examine how it is supported by the Cisco IOS.

Novell IPX Overview

Novell Netware is both an operating system and a networking protocol. Novell IPX is based on the Xerox Network System (XNS) protocols. When we use the term "IPX," we refer to the entire Novell protocol stack, just as we use IP to refer to the entire TCP/IP suite.

IPX Addressing

Addressing in a Novell IPX network is different than addressing in an IP network. An IPX address is in the format of network.node.socket, as shown in Figure 18−1. The three parts of the IPX address are as follows:

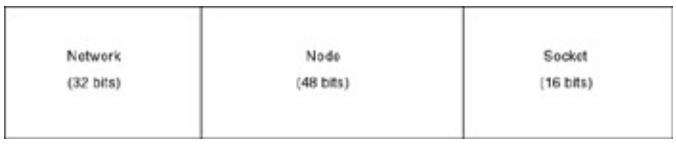


Figure 18−1: IPX addressing

- **Network portion:** Every IPX network is assigned a globally unique 32−bit network number.
- **Node portion:** Each IPX device (usually a workstation, router, or server) is assigned a 48−bit node address. This 48−bit address is taken from the MAC address of the device itself. This is an attractive feature of IPX. Having the node address of an IPX device use the device's MAC address means that IPX will not need ARP (Address Resolution Protocol). An IPX−enabled device wishing to send a datagram to another IPX device only needs to know the IPX address of the end station since the MAC address of the end station is embedded in the IPX address. The sending station has enough information to create an Ethernet or token ring frame since the destination MAC address is already known. With IP, you know an end device's IP address but do not know the end device's MAC address that is needed to build the Ethernet, token ring, or FDDI data link frame. IP uses ARP to find the MAC address of a destination device or network whose IP address is already known.
- **Socket portion:** The socket is a 16−bit number that identifies a software process using IPX in the end station. Some socket numbers are reserved and some are available for use by the end station.

We see in Figure 18−1 that the network address is 32 bits, the node address is 48 bits, and the socket number is 16 bits. A complete IPX address is a 96−bit number expressed as a 12−byte hexadecimal number.

IPX Protocol Stack

Figure 18−2 shows the IPX protocol stack. The key portions of the protocol stack can be described as follows.

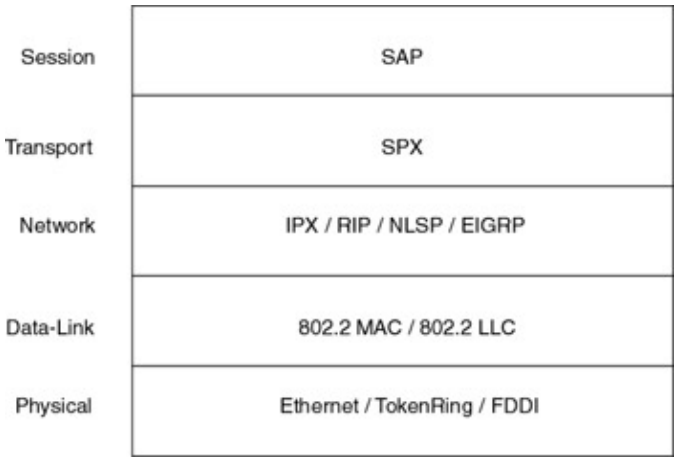


Figure 18−2: IPX protocol stack

IPX Network Layer

IPX (Internetwork Packet Exchange) provides network layer connectionless datagram delivery to support Novell Netware. The minimum IPX packet size is 30 bytes and the maximum packet size is 65,535 bytes. An IPX packet has a 30−byte header, as shown in Figure 18−3.

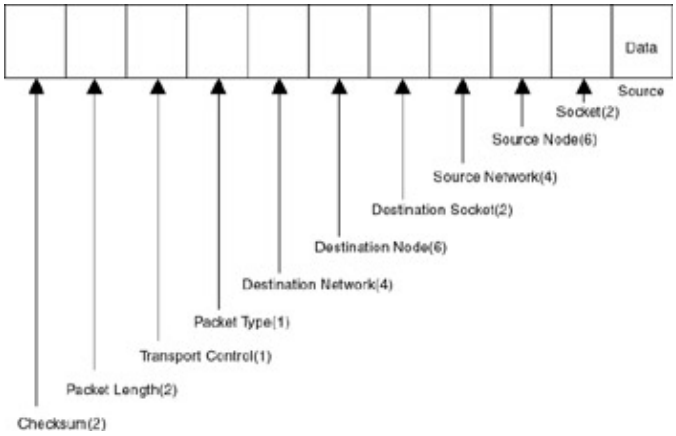


Figure 18−3: IPX packet structure

IPX Transport Layer

Novell IPX uses SPX as its transport layer protocol. SPX is a connection−oriented protocol. No data transfer can take place between two end stations using SPX until a connection has been built.

SAP (Service Advertising Protocol)

A SAP is used to advertise and distribute Novell server information. Netware servers and routers broadcast a SAP message every 60 seconds. This message advertises what services they provide.

There are three types of SAP packets:

- **Periodic updates:** A periodic update is used by a server when it has a service to advertise. The server sends a SAP broadcast with the service's name, service type, and full IPX address (network.node.socket). Routers listen to and store these broadcasts. Routers periodically broadcast

these updates to all directly connected neighbors.

- **Service queries:** A service query is used by a Netware client to locate a server. This kind of query is often referred to as a get nearest server (GNS) query. This service query is a broadcast and does not go off the local network. The query will be answered by the local router, which has stored the periodic updates that it has received from servers on the network.
- **Service responses:** A service response is a response to a service query. This is usually a response from a router.

IPX Routing Protocols

IPX uses three different routing protocols to propagate routing information:

- **IPX RIP:** IPX RIP is a distance vector protocol. It has many similarities to IP RIP. IPX RIP differs from IP RIP because the IPX end station requests route information, whereas with an IP network, the end station will have a default route to the nearest router. IPX updates are broadcast every 60 seconds. IPX RIP uses two metrics. The first metric is delay, referred to as ticks. The second metric is hop count. The route with the lowest delay will be given the preference over the route with the lowest hop count. If two routes exist with the same tick value, the router will use the hop count of each route to determine the best route.
- **NLSP:** NLSP is a link state protocol. It provides load balancing across equal−cost paths and features much faster convergence times than IPX RIP.
- **EIGRP:** This is the Cisco proprietary Enhanced IGRP. This routing protocol features automatic redistribution between RIP/SAP and EIGRP.

RIP/SAP Operation

RIP and SAP work closely together in an IPX network. Figure 18−4 shows how an IPX end station locates a server and finds a route to that server, and the steps are detailed below:

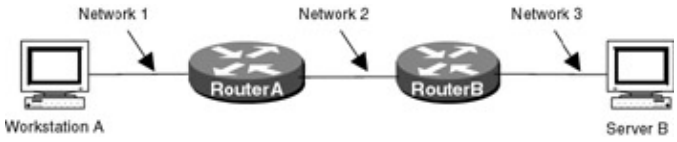


Figure 18−4: SAP operation

1. Workstation A sends a get nearest server (GNS) request to find a server.
2. RouterA has cached a SAP update from RouterB advertising ServerB. RouterA sends a SAP reply to Workstation A.
3. Workstation A now has the IPX address of ServerB. Workstation A knows that ServerB is on Network 3. Workstation A now needs to find a route to Network 3.
4. RouterA knows about Network 3 via RIP updates. RouterA sends a RIP response packet to Workstation A.

IPX Encapsulation Types

Figure 18−5 shows the different IPX Ethernet encapsulation types. When running on a local area network, IPX runs on Ethernet, token ring, and FDDI. IPX can use four different Ethernet encapsulation types on a local area network. This means that four different MAC frames can be used on an IPX network. If two workstations on a NetWare LAN use different Ethernet encapsulation types, they cannot talk to each other directly — their traffic has to go through a router.

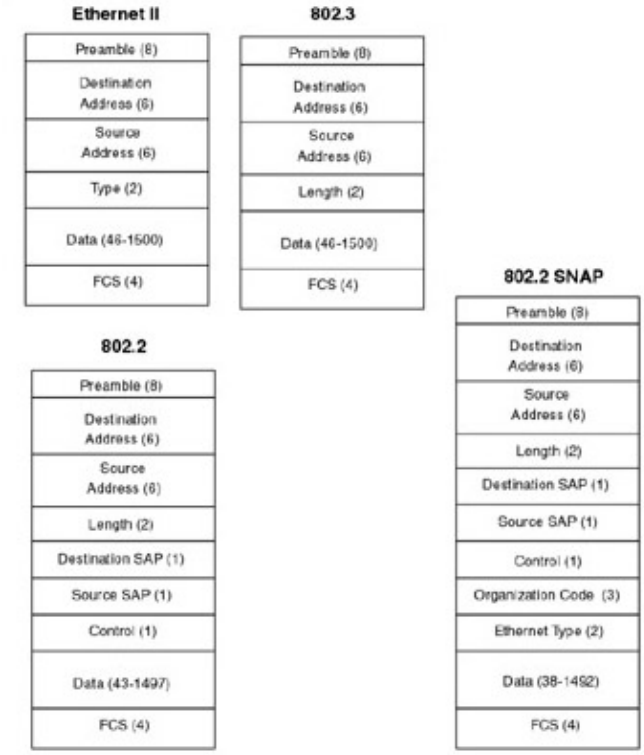


Figure 18−5: IPX Ethernet encapsulation types
The four Ethernet encapsulation types are as follows:

- **Ethernet II:** This encapsulation type is referred to by Cisco as ARPA encapsulation.
- **802.2:** This encapsulation type is referred to by Cisco as SAP encapsulation.
- **802.3:** This encapsulation type is referred to by Cisco as Novell−Ether encapsulation. This is the default encapsulation on a Cisco Ethernet interface.
- **SNAP:** This encapsulation type is referred to by Cisco as SNAP encapsulation.

IPX supports two different token ring encapsulation types:

- Cisco SAP, which is the default
- Cisco SNAP

IPX supports three different FDDI encapsulation types:

- Cisco SNAP
- Cisco SAP
- Novell−FDDI Raw

Commands Discussed in This Chapter

- **access−list** *access−list−number* **[deny|permit]** *network*[.*node*] [*network−mask.node−mask*] [*service−type*[*server−name*]]
- **debug ipx routing activity**
- **debug ipx routing events**
- **debug ipx sap activity**
- **debug ipx sap events**
- **distribute−list in**
- **distribute−list out**
- **ipx network** *network* **[encapsulation *encapsulation−type* [secondary]]**
- **ipx output−sap−filter** *access−list−number*

- **ipx route** {*network* [network−*mask*] | **default**} {*network.node* | *interface*} [ticks] [hops] [**floating−static**]
- **ipx router [eigrp** *autonomous−system−number* | **nlsp [***tag***]** | **rip]**
- **ipx routing [***node***]**
- **ipx sap** *service−type name network.node socket hop−count*
- **ipx split−horizon eigrp** *autonomous−system−number*
- **network [***network−number*] |**all]**
- **ping [ipx] [***network.node***]**
- **show access−list**
- **show ipx eigrp interfaces [***type number***] [***as−number***]**
- **show ipx eigrp neighbors [servers] [***autonomous−system−number* | *interface***]**
- **show ipx interface [***type number***]**
- **show ipx interface brief**
- **show ipx route [***network***] [default] [detailed]**
- **show ipx servers [unsorted** | [**sorted [name** | **net** | **type]] [regexp** *name***]**
- **show ipx traffic**

Definitions

access−list: This global configuration command defines access lists for SAP filters, route filters, and NLSP filters.

debug ipx routing activity: This debug command displays information on IPX routing activity.

debug ipx routing events: This debug command displays information on IPX routing activity.

debug ipx sap activity, debug ipx sap events: These debug commands provide information on IPX SAP packets

distribute−list in: This router configuration command filters IPX network information as it comes into a router.

distribute−list out: This router configuration command filters IPX network information as it leaves a router.

ipx network: This interface configuration command enables IPX routing on the selected interface. It can also be used to select the encapsulation type on a LAN interface.

ipx route: This global configuration command adds a static route to the routing table.

ipx router: This global configuration command is used to specify what type of routing protocol to use. Valid options are RIP, EIGRP, and NLSP.

ipx routing: This global configuration command enables IPX routing on a router. You can optionally set the IPX node number for this router.

ipx sap: This global configuration command creates static SAP entries in the router's IPX server table.

ipx split−horizon eigrp: This interface command configures the status of split horizon.

network: This router configuration command specifies what networks should be included in routing updates.

ping ipx: This exec command is used to verify IPX network reachability.

show access−list: This exec command displays information on access lists that have been defined on the router.

show ipx eigrp interfaces: This exec command displays information on any interfaces that have been enabled for the EIGRP routing protocol.

show ipx eigrp neighbor: This exec command will display information on neighbor routers that have been discovered by EIGRP.

show ipx interface: This exec command displays information on router interfaces that have been configured for the IPX protocol.

show ipx interface brief: This exec command provides a summary of all router interfaces that have been configured for the IPX protocol.

show ipx route: This exec command displays the IPX route table for the router.

show ipx servers: This exec command displays all IPX servers that have either been discovered through SAP advertisements or have been statically configured.

show ipx traffic: This exec command shows information on transmitted and received IPX protocol packets. The output is sorted by IPX packet type such as RIP, EIGRP, SAP, and so forth.

IOS Requirements

These labs were done using IOS 11.2.

Lab #81: IPX Configuration with IPX RIP/SAP

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers. One of the routers must have two serial interfaces, and the other two routers must have one serial interface and one Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports the IPX protocol.

Configuration Overview

This lab will demonstrate IPX configuration and monitoring. As shown in Figure 18−6, this lab defines five IPX networks. RouterA, RouterB, and RouterC are each given IPX node numbers a.a.a, b.b.b, and c.c.c respectively.

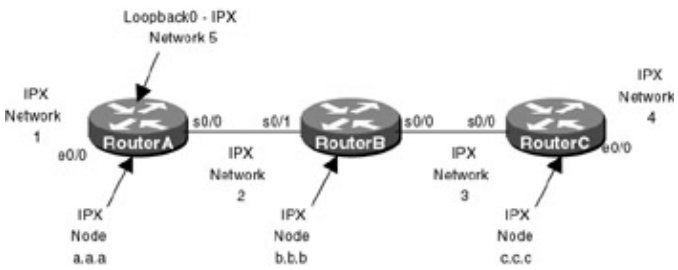


Figure 18−6: IPX with RIP/SAP

The three routers are connected as shown in Figure 18−6. RouterB acts as a DCE and supplies clocking to RouterA and RouterC.

Router Configuration

The configurations for the three routers in this example are as follows (key IPX commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
ipx routing 000a.000a.000a ← Enable IPX routing. Define the IPX node to be
                             000a.000a.000a
!
interface Loopback0
 no ip address
 ipx network 5 ← Make this interface IPX network 5
!
interface Ethernet0/0
 no ip address
 no keepalive
 ipx network 1 ← Make this interface IPX network 1
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 2 ← Make this interface IPX network 2
 no fair-queue
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
ipx routing 000b.000b.000b ← Enable IPX routing. Define the IPX node to be
                             000b.000b.000b
!
interface Serial0/0
 no ip address
```

```
 encapsulation ppp
 ipx network 3 ← Make this interface IPX network 3
 no fair-queue
 clockrate 64000
!
interface Serial0/1
 no ip address
 encapsulation ppp
 ipx network 2 ← Make this interface IPX network 2
 clockrate 64000
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
ipx routing 000c.000c.000c ← Enable IPX routing. Define the IPX node to be
                             000c.000c.000c
!
interface Ethernet0/0
 no ip address
 no keepalive
 ipx network 4 ← Make this interface IPX network 4
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 3 ← Make this interface IPX network 3
 no fair-queue
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Notice that there are no routing protocols explicitly configured for any of the three routers. This is because with the IPX protocol, IPX RIP is enabled by default.

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Type the **show ipx interface brief** command to display the status of all interfaces on the router. We see that there are three interfaces on this router that are running the IPX protocol. These are Ethernet0/0 on IPX network 1, Serial0/0 on IPX network 2, and the Loopback 0 interface on IPX network 5. We see that each of these interfaces is in up/up state. Notice that the encapsulation on the

Ethernet0/0 interface is set to NOVELL−ETHER. Recall from the introduction section in this chapter that the default encapsulation on an Ethernet interface is NOVELL−ETHER.

```
RouterA#show ipx interface brief
Interface            IPX Network Encapsulation Status                IPX State
Ethernet0/0          1           NOVELL-ETHER  up                    [up]
Serial0/0            2           PPP           up                    [up]
Loopback0            5           UNKNOWN       up                    [up]
```

Type the **show ipx route** command to display the routing table for this router. The routing table shows us that three IPX networks are directly connected, Network 1 is on Ethernet0, Network 2 is on Serial 0, and Network 5 is on Loopback 0. RouterA has learned about two networks via the IPX RIP routing protocol. Network 3 is 1 hop and 7 ticks away and Network 4 is 2 hops and 13 ticks away.

```
RouterA#show ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          1 (NOVELL-ETHER),  Et0/0
C          2 (PPP),           Se0/0
C          5 (UNKNOWN),       Lo0

             Tick Count       Next hop address
             ↓                  ↓
R          3 [07/01] via        2.000b.000b.000b,   49s, Se0/0
                  á
                  Hop count to destination network

            Tick count       Next hop address
              ↓                  ↓
R          4 [13/02] via        2.000b.000b.000b,   50s, Se0/0
                  á
                  Hop count to destination network
```

The **show interface s 0/0** command usually shows an IP address. Since there is no IP enabled on this router, the show command does not display a network address. We see that IPXCP is opened. This is the IPX Network Control Protocol.

```
RouterA#show int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open
  Open: CDPCP, IPXCP ← No IP is enabled on this interface
  Last input 00:00:01, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     99 packets input, 3888 bytes, 0 no buffer ← Packet's input
     Received 99 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     100 packets output, 3902 bytes, 0 underruns ← Packet's output
     0 output errors, 0 collisions, 16 interface resets
     0 output buffer failures, 0 output buffers swapped out
     31 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Specific IPX information for the interface can be displayed with the **show ipx int s 0/0** command.

```
RouterA#show ipx int s 0/0
Serial0/0 is up, line protocol is up
  IPX address is 2.000a.000a.000a [up] ← IPX address


      A WAN interface has a default IPX delay of 6
                                          ↓
  Delay of this IPX network, in ticks is 6 throughput 0 link delay 0
  IPXWAN processing not enabled on this interface.
  IPX SAP update interval is 1 minute(s)
  IPX type 20 propagation packet forwarding is disabled
  Incoming access list is not set
  Outgoing access list is not set
  IPX helper access list is not set
  SAP GNS processing enabled, delay 0 ms, output filter list is not set
  SAP Input filter list is not set
  SAP Output filter list is not set
  SAP Router filter list is not set
  Input filter list is not set
  Output filter list is not set
  Router filter list is not set
  Netbios Input host access list is not set
  Netbios Input bytes access list is not set
  Netbios Output host access list is not set
  Netbios Output bytes access list is not set
  Updates each 60 seconds, aging multiples RIP: 3 SAP: 3
  SAP interpacket delay is 55 ms, maximum size is 480 bytes
  RIP interpacket delay is 55 ms, maximum size is 432 bytes
  Watchdog processing is disabled, SPX spoofing is disabled, idle time 60
  IPX accounting is disabled
  IPX fast switching is configured (enabled)
  RIP packets received 9, RIP packets sent 9 ← RIP is running on this interface
  SAP packets received 1, SAP packets sent 1 ← SAP is running on this interface
```

Type **show interface e 0/0** to display information on the Ethernet interface of the router. There are two important items to note here:

1. There is no IP address on this interface since the IP protocol is not configured on this router.
2. The MAC address of the interface is 00e0.1e5b.2601. The interface did not take the 000a.000a.000a MAC address that has been assigned to the Serial0 interface of this router by the command **IPX Routing 000A.000A.000A** in the router's configuration.

```
RouterA#show int e 0/0
Ethernet0/0 is up, line protocol is up
  Hardware is AmdP2, address is 00e0.1e5b.2601 (bia 00e0.1e5b.2601)
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 128/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive not set
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:54, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     576 packets output, 96038 bytes, 0 underruns
     576 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     576 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Specific IPX information for the interface can be displayed with the **show ipx int e 0/0** command.

```
RouterA#show ipx int e 0/0
Ethernet0/0 is up, line protocol is up
  IPX address is 1.00e0.1e5b.2601, NOVELL-ETHER [up] ← Default IPX encapsulation
      A LAN interface has a default IPX delay of 1
                                          ↓
Delay of this IPX network, in ticks is 1 throughput 0 link delay 0
  IPXWAN processing not enabled on this interface.
  IPX SAP update interval is 1 minute(s)
  IPX type 20 propagation packet forwarding is disabled
  Incoming access list is not set
  Outgoing access list is not set
  IPX helper access list is not set
  SAP GNS processing enabled, delay 0 ms, output filter list is not set
  SAP Input filter list is not set
  SAP Output filter list is not set
  SAP Router filter list is not set
  Input filter list is not set
  Output filter list is not set
  Router filter list is not set
  Netbios Input host access list is not set
  Netbios Input bytes access list is not set
  Netbios Output host access list is not set
  Netbios Output bytes access list is not set
  Updates each 60 seconds, aging multiples RIP: 3 SAP: 3
  SAP interpacket delay is 55 ms, maximum size is 480 bytes
  RIP interpacket delay is 55 ms, maximum size is 432 bytes
  IPX accounting is disabled
  IPX fast switching is configured (enabled)
  RIP packets received 0, RIP packets sent 200
  SAP packets received 0, SAP packets sent 166
```

RouterB and RouterC should be reachable from RouterA. IPX has limited test functionality as compared to IP. With IPX, you can only ping another IPX interface. Use the **ping ipx 2.b.b.b** command to verify that you can reach RouterB.

```
RouterA#ping ipx 2.b.b.b ← ping RouterB

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 2.000b.000b.000b, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

Use the **ping ipx 3.c.c.c** command to verify that you can reach RouterC.

```
RouterA#ping ipx 3.c.c.c ← ping RouterC

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 3.000c.000c.000c, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms
```

Now connect to RouterC. Type the **show ipx route** command to display the IPX routing table for RouterC. We see that RouterC has two directly connected IPX networks: Network 3 and Network 4. Three networks have been learned via IPX RIP: Network 1, Network 2, and Network 4. All of these RIP routes have a next hop address of RouterB.

```
RouterC#show ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses
```

```
5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          3 (PPP),           Se0/0
C          4 (NOVELL-ETHER),  Et0/0

      The next hop address for all remote networks is RouterB
                                  ↓
R          1 [13/02] via        3.000b.000b.000b,   57s, Se0/0
R          2 [07/01] via        3.000b.000b.000b,   58s, Se0/0
R          5 [13/02] via        3.000b.000b.000b,   58s, Se0/0
```

Type the **show ipx interface brief** command to display the status of each interface on the router. We see that RouterC has two IPX networks configured. Network 4 is assigned to interface Ethernet0/0 and Network 3 is assigned to interface S0/0. We see that both networks are in an up/up status.

```
RouterC#show ipx interface brief
Interface            IPX Network Encapsulation Status                IPX State
Ethernet0/0          4           NOVELL-ETHER  up                    [up]
Serial0/0            3           PPP           up                    [up]
```

Both RouterB and RouterA should be reachable via an IPX ping. Try to ping Network 3 on RouterB.

```
RouterC#ping ipx 3.b.b.b

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 3.000b.000b.000b, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

Make sure that both interfaces on IPX Network 2 are also reachable.

```
RouterC#ping ipx 2.b.b.b  ← ping RouterB

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 2.000b.000b.000b, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms

RouterC#ping ipx 2.a.a.a ← ping RouterA

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 2.000a.000a.000a, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/56 ms
```

The **show ipx traffic** command is very useful. It gives detailed information on the number of IPX packets that have been sent or received on the router.

```
RouterC#show ipx traffic
System Traffic for 0.0000.0000.0001 System-Name: RouterC
Rcvd:   36 total, 0 format errors, 0 checksum errors, 0 bad hop count,
        0 packets pitched, 36 local destination, 0 multicast
Bcast:  16 received, 29 sent
Sent:   50 generated, 0 forwarded
        0 encapsulation failed, 0 no route
SAP:    1 SAP requests, 0 SAP replies, 0 servers
        0 SAP Nearest Name requests, 0 replies
        0 SAP General Name requests, 0 replies
        5 SAP advertisements received, 4 sent
        2 SAP flash updates sent, 0 SAP format errors
RIP:    1 RIP requests, 0 RIP replies, 5 routes
        9 RIP advertisements received, 18 sent
        2 RIP flash updates sent, 0 RIP format errors
```

```
Echo:   Rcvd 5 requests, 15 replies
        Sent 15 requests, 5 replies
        0 unknown: 0 no socket, 0 filtered, 0 no helper
        0 SAPs throttled, freed NDB len 0
Watchdog:
        0 packets received, 0 replies spoofed
Queue lengths:
        IPX input: 0, SAP 0, RIP 0, GNS 0
        SAP throttling length: 0/(no limit), 0 nets pending lost route reply
        Delayed process creation: 0
EIGRP:  Total received 0, sent 0
        Updates received 0, sent 0
        Queries received 0, sent 0
        Replies received 0, sent 0
        SAPs received 0, sent 0
NLSP:   Level-1 Hellos received 0, sent 0
        PTP Hello received 0, sent 0
        Level-1 LSPs received 0, sent 0
        LSP Retransmissions: 0
        LSP checksum errors received: 0
        LSP HT=0 checksum errors received: 0
        Level-1 CSNPs received 0, sent 0
        Level-1 PSNPs received 0, sent 0
        Level-1 DR Elections: 0
        Level-1 SPF Calculations: 0
        Level-1 Partial Route Calculations: 0
```

Now let's connect to RouterB. Use the **show ipx interface brief** command to display a summary of all interfaces on the router. We see that there are two IPX networks on this router — S0/0 and S0/1 — both of which are in an up/up state.

```
RouterB#show ipx interface brief
Interface             IPX Network Encapsulation Status                IPX State
Ethernet0/0           unassigned  not config'd  administratively down  n/a
Serial0/0             3           PPP           up                    [up]
Serial0/1             2           PPP           up                    [up]
```

The **show ipx route** command will display the routing table information for RouterB. We see that RouterB has two directly connected networks: Network 2 and Network 3. Networks 1, 4, and 5 have been learned via IPX RIP.

```
RouterB#show ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          2 (PPP),           Se0/1
C          3 (PPP),           Se0/0
R          1 [07/01] via        2.000a.000a.000a,   30s, Se0/1 ← RIP route
R          4 [07/01] via        3.000c.000c.000c,   31s, Se0/0 ← RIP route
R          5 [07/01] via        2.000a.000a.000a,   31s, Se0/ ← RIP route
```

Using the **IPX ping** command, verify that you can reach RouterC and RouterA:

```
RouterB#ping ipx 3.c.c.c ← ping RouterC

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 3.000c.000c.000c, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

```
RouterB#ping ipx 2.a.a.a ← ping RouterA

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 2.000a.000a.000a, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

Let's try to telnet to RouterC. Issue the **telnet** command and when prompted for a host, enter **3.c.c.c**. We see that RouterB issues an error message that it is unable to find a computer address. Telnet is an IP protocol application. IPX does not use telnet, nor does it have an equivalent. The only test tool that is available when running IPX on a network is the IPX ping. This is why it is important to always run the IP protocol on your network.

```
RouterB#telnet
Host: 3.c.c.c ← Try to telnet to an IPX address
Translating "3.c.c.c"...domain server (255.255.255.255)
% Unknown command or computer name, or unable to find computer address
      á
      Telnet can only be used with the IP protocol. IPX does not support telnet
```

Let's examine how the IPX RIP routing protocol works. Enable IPX RIP debugging with the **debug ipx routing activity** and **debug ipx routing events** commands. If you are not connected to the console port of the router, be sure to also issue the **terminal monitor** command to send all debug output to your terminal session.

```
RouterB#debug ipx routing activity
IPX routing debugging is on

RouterB#debug ipx routing events
IPX routing events debugging is on
```

IPX RIP has many similarities to IP RIP in terms of how updates are sent and received. Every 60 seconds a router will send an update to each directly connected neighbor. The update consists of all routes that the router can reach and the distance to those routes. With IP RIP the metric is hop count. With IPX RIP there are two metrics. The first metric is hop count. The second metric is delay. By default, the delay on a WAN interface is 6 and the delay on a LAN interface is 1. The following two **show interface** outputs for an Ethernet and a serial interface show that the delay is listed in the output of the command.

```
RouterA#show ipx int e 0/0
Ethernet0/0 is up, line protocol is up
  IPX address is 1.00e0.1e5b.2601, NOVELL-ETHER [up]

      A LAN interface has a default IPX delay of 1
                                        ↓
Delay of this IPX network, in ticks is 1 throughput 0 link delay 0

RouterA#show ipx int s 0/0
Serial0/0 is up, line protocol is up
  IPX address is 2.000a.000a.000a [up]

      A WAN interface has a default IPX delay of 6
                                      ↓
  Delay of this IPX network, in ticks is 6 throughput 0 link delay 0
```

IPX RIP will prefer a route with lower delay over a route with a lower hop count.

The **debug ipx routing** output below shows the updates sent out from and received on RouterB. The first update is sent from RouterB to RouterC. It informs RouterC that RouterB has a route to IPX Network 5, IPX Network 1, and IPX Network 2.

```
RouterB sends an update to RouterC
↓
```

```
IPXRIP: positing full update to 3.ffff.ffff.ffff via Serial0/0 (broadcast)
IPXRIP: src=3.000b.000b.000b, dst=3.ffff.ffff.ffff, packet sent
    network 5, hops 2,  delay 13
    network 1, hops 2,  delay 13
    network 2, hops 1,  delay 7
```

Next, RouterB receives an update from RouterC. RouterC informs RouterB that it has a route to IPX Network 4.

```
RouterB receives an update from RouterC
↓
IPXRIP: update from 3.000c.000c.000c
    4 in 1 hops, delay 7
```

RouterB then sends an update to RouterA. RouterB tells RouterA that it has a route to IPX Network 4 and IPX Network 3.

```
RouterB sends an update to RouterA
↓
IPXRIP: positing full update to 2.ffff.ffff.ffff via Serial0/1 (broadcast)
IPXRIP: src=2.000b.000b.000b, dst=2.ffff.ffff.ffff, packet sent
    network 4, hops 2,  delay 13
    network 3, hops 1,  delay 7
```

Finally, RouterB receives an update from RouterA. RouterA informs RouterB that it has a route to IPX Network 1 and IPX Network 5.

```
RouterB receives an update from RouterA
↓
IPXRIP: update from 2.000a.000a.000a
    1 in 1 hops, delay 7
    5 in 1 hops, delay 7
```

This process repeats itself every 60 seconds.

Lab #82: IPX EIGRP

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers. One of the routers must have two serial interfaces, and the other two routers must have one serial interface and one Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers
- A Cisco IOS image that supports the IPX protocol.

Configuration Overview

This lab will demonstrate IPX routing protocols. By default, IPX RIP is enabled on all interfaces. EIGRP can also be used as a routing protocol for IPX networks. EIGRP has several advantages over RIP, such as

- Faster convergence
- Less network traffic dedicated to routing updates (EIGRP only sends out periodic updates of its routing table)
- Lower CPU utilization

- Better scaling in large networks
- Automatic redistribution with IPX RIP

In this lab, we will be running IPX RIP on the LAN interfaces and EIGRP on all other interfaces. Since IPX RIP is enabled on all interfaces by default, we will be explicitly turning it off on those interfaces where we want to run EIGRP.

As shown in Figure 18−7, this lab defines five IPX networks. RouterA, RouterB, and RouterC are each given IPX node numbers a.a.a, b.b.b, and c.c.c, respectively.

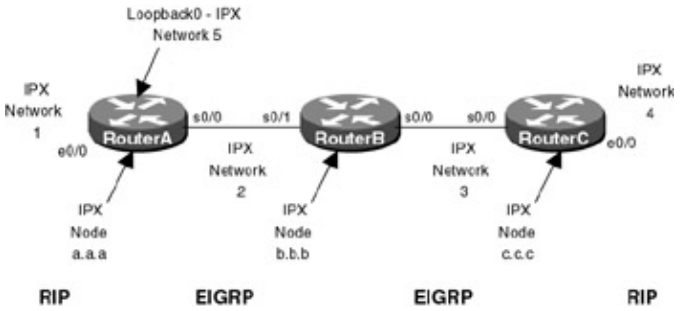


Figure 18−7: IPX EIGRP

The three routers are connected as shown in Figure 18−7. RouterB acts as a DCE and supplies clocking to RouterA and RouterC.

Router Configuration

The configurations for the three routers in this example are as follows (key IPX commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
ipx routing 000a.000a.000a ← Enable IPX routing. Define the IPX node to be
                              000a.000a.000a
!
interface Loopback0
 no ip address
 ipx network 5 ← Make this interface IPX Network 5
!
interface Ethernet0/0
 no ip address
 no keepalive
 ipx network 1
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 2 ← Make this interface IPX Network 2
 no fair-queue
!
no ip classless
!
!
ipx router eigrp 1 ← Enable IPX EIGRP autonomous system 1
```

```
 network 2 ← Include IPX Network 2 in EIGRP updates
!
!
ipx router rip ← Enable IPX RIP on this router
 no network 2 ← Do not advertise IPX Network 2 in RIP updates
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
ipx routing 000b.000b.000b ← Enable IPX routing. Define the IPX node to be
                              000b.000b.000b
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 3 ← Make this interface IPX Network 3
 no fair-queue
 clockrate 64000
!
interface Serial0/1
 no ip address
 encapsulation ppp
 ipx network 2 ← Make this interface IPX Network 2
 clockrate 64000
!
no ip classless
!
!
ipx router eigrp 1 ← Enable IPX EIGRP autonomous system 1
 network all ← Advertise all IPX networks on this router in EIGRP updates
!
!
no ipx router rip ← Do not enable IPX RIP on this router
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
```

```
!
hostname RouterC
!
!
ipx routing 000c.000c.000c ← Enable IPX routing. Define the IPX node to be
                              000c.000c.000c
!
interface Ethernet0/0
 no ip address
 no keepalive
 ipx network 4 ← Make this interface IPX Network 4
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 3 ← Make this interface IPX Network 3
 no fair-queue
!
no ip classless
!
!
ipx router eigrp 1 ← Enable IPX EIGRP autonomous system 1
 network 3 ← Include IPX Network 3 in EIGRP updates
!
!
ipx router rip ← Enable IPX RIP on this router
 no network 3 ← Do not advertise IPX Network 3 in RIP updates
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Notice that IPX SAP has to be turned off on those interfaces where we do not want it to run.

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Verify that all IPX interfaces are up and active with the **show ipx interface brief** command.

```
RouterA#show ipx interface brief
Interface            IPX Network Encapsulation Status                IPX State
Ethernet0/0          1           NOVELL-ETHER  up                    [up]
Serial0/0            2           PPP           up                    [up]
Loopback0            5           UNKNOWN       up                    [up]
```

The **show ipx route** command shows us that we have three directly connected networks (Network 1, Network 2, and Network 5). Two remote networks have been learned via EIGRP. These are Networks 3, and 4. Notice that there are no RIP learned routes in this routing table.

```
RouterA#show ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          1 (NOVELL-ETHER),  Et0/0
C          2 (PPP),           Se0/0
C          5 (UNKNOWN),       Lo0
```

```
EIGRP learned route
↓
E          3 [2681856/0] via        2.000b.000b.000b, age 02:08:01,

EIGRP learned route
↓
E          4 [2707456/1] via        2.000b.000b.000b, age 02:07:52,
                     385u, Se0/0
```

The **show ipx eigrp neighbor** command will display information on what neighboring EIGRP routers have been discovered.

```
RouterA#show ipx eigrp neigh

IPX EIGRP Neighbors for process 1
H   Address                 Interface   Hold Uptime   SRTT   RTO  Q  Seq
                                        (sec)         (ms)       Cnt Num
0   2.000b.000b.000b        Se0/0         13 02:10:19   53   318  0  38
      á
      RouterB, interface S0/1  is an EIGRP neighbor
```

The **show ipx eigrp interfaces** command will show what router interfaces are running EIGRP. Notice that only interface S0/0 of RouterA is an EIGRP interface. Interface E0/0 on RouterA is still running IPX RIP.

```
RouterA#show ipx eigrp interfaces

IPX EIGRP Interfaces for process 1

                    Xmit Queue   Mean   Pacing Time   Multicast     Pending
Interface    Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer    Routes
Se0/0          1        0/0        53       0/15          263       0
á
Only interface S0/0 is running EIGRP. Interface E0/0 is still running EIGRP
```

Now let's connect to RouterB. The **show ipx interface brief** command should show us that all interfaces are up and active.

```
RouterB#show ipx interface brief
Interface            IPX Network Encapsulation Status                IPX State
Ethernet0/0          unassigned  not config'd  administratively down  n/a
Serial0/0            3           PPP           up                     [up]
Serial0/1            2           PPP           up                     [up]
```

The **show ipx route** command should show EIGRP routes to three IPX networks: 1, 4, and 5.

```
RouterB#show ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          2 (PPP),           Se0/1
C          3 (PPP),           Se0/0
E          1 [2195456/1] via        2.000a.000a.000a, age 02:08:28,
                     392u, Se0/1
E          4 [2195456/1] via        3.000c.000c.000c, age 02:08:28,
                     3u, Se0/0
E          5 [2297856/1] via        2.000a.000a.000a, age 02:08:28,
                     1u, Se0/1
```

There should be two discovered EIGRP neighbors: IPX Network 2 and IPX Network 3. Verify this with the **show ipx eigrp neighbor** command.

```
RouterB#show ipx eigrp neigh

IPX EIGRP Neighbors for process 1
H   Address                 Interface   Hold Uptime   SRTT   RTO  Q  Seq
                                        (sec)         (ms)       Cnt Num

      EIGRP neighbor RouterA
      ↓
1   2.000a.000a.000a        Se0/1         10 02:11:10   22   200  0  20

      EIGRP neighbor RouterC
      ↓
0   3.000c.000c.000c        Se0/0         14 02:11:41   43   258  0  22
```

Verify with the **show ipx eigrp interfaces** command that both serial interfaces on RouterB are running EIGRP.

```
RouterB#show ipx eigrp interfaces

IPX EIGRP Interfaces for process 1

                    Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Se0/0          1        0/0        43       0/15          207          0
Se0/1          1        0/0        22       0/15          103          0
á
Both serial interfaces on RouterB are using EIGRP for their routing protocol
```

The **show ipx eigrp traffic** command is a useful command that shows how much EIGRP traffic has been sent and received on the router.

```
RouterB#show ipx eigrp traffic
IP−EIGRP Traffic Statistics for process 1
  Hellos sent/received: 3433/3430
  Updates sent/received: 11/11
  Queries sent/received: 10/7
  Replies sent/received: 7/10
  Acks sent/received: 37/33
  Input queue high water mark 2, 0 drops
```

Now connect to RouterC. Verify that all IPX interfaces are active with the **show ipx interface brief** command.

```
RouterC#show ipx interface brief
Interface            IPX Network Encapsulation Status                IPX State
Ethernet0/0          4           NOVELL−ETHER  up                    [up]
Serial0/0            3           PPP           up                    [up]
BRI0/0               unassigned  not config'd  administratively down  n/a
BRI0/0:1             unassigned  not config'd  administratively down  n/a
BRI0/0:2             unassigned  not config'd  administratively down  n/a
```

The **show ipx route** command should reveal that there are three networks that have been learned via EIGRP. These should be Networks 1, 2, and 5.

```
RouterC#show ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.
```

```
No default route known.

C          3 (PPP),           Se0/0
C          4 (NOVELL-ETHER),  Et0/0
E          1 [2707456/1] via        3.000b.000b.000b, age 02:09:47,
                     4u, Se0/0
E          2 [2681856/0] via        3.000b.000b.000b, age 02:09:47,
                     1u, Se0/0
E          5 [2809856/1] via        3.000b.000b.000b, age 02:09:47,
                     1u, Se0/0
```

The **show ipx eigrp interfaces** command should show that there is one interface on this router that is running EIGRP, Serial0/0.

```
RouterC#show ipx eigrp interfaces

IPX EIGRP Interfaces for process 1

                    Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Se0/0          1        0/0        20       0/15          95           0
```

Lab #83: Static SAP Entries andSAP Access Lists

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers. One of the routers must have two serial interfaces, and the other two routers must have one serial interface and one Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- Cisco IOS image that supports the IPX protocol.

Configuration Overview

This lab will demonstrate how SAP advertisements work on a Cisco router. We will define static SAPs on our routers and explore how these work. SAP updates can take up a lot of bandwidth on an IPX network. The Cisco IOS allows static SAP entries to be created. Finally, we will demonstrate how a Cisco router can filter SAP updates.

As shown in Figure 18−8, this lab defines five IPX networks. RouterA, RouterB, and RouterC are each given IPX node numbers a.a.a, b.b.b, and c.c.c, respectively.

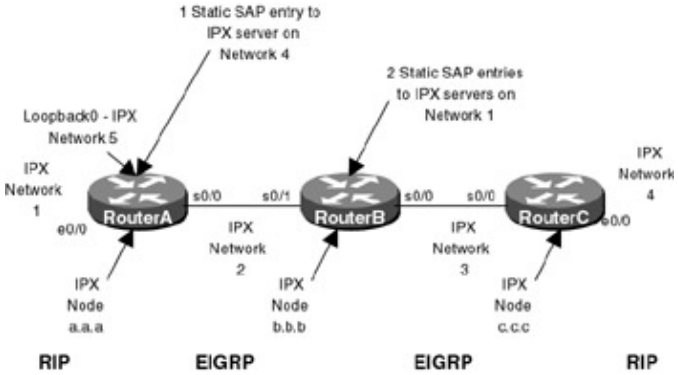


Figure 18−8: IPX SAP

The three routers are connected as shown in Figure 18−8. RouterB acts as a DCE and supplies clocking to RouterA and RouterC.

Note Even though we are not running IPX RIP/SAP on the wide area network, we will see that SAP updates are still propagated throughout the network.

Router Configuration

The configurations for the three routers in this example are as follows (key IPX commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
ipx routing 000a.000a.000a ← Enable IPX routing. Define the IPX node to be
                              000a.000a.000a
!
interface Loopback0
 no ip address
 ipx network 5 ← Make this interface IPX Network 5
!
interface Ethernet0/0
 no ip address
 no keepalive
 ipx network 1 ← Make this interface IPX Network 1
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 2 ← Make this interface IPX Network 2
 no fair-queue
!
no ip classless
!
!
ipx router eigrp 1 ← Enable IPX EIGRP autonomous system 1
 network 2 ← Include IPX Network 2 in EIGRP updates
!
!
ipx router rip ← Enable IPX RIP on this router
 no network 2 ← Do not advertise IPX Network 2 in RIP updates
!
```

```
!
ipx sap 4 Server4 4.00e0.1e5b.0a81 451 2 ← Define a static SAP entry on this
                                           router
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
ipx routing 000b.000b.000b ← Enable IPX routing. Define the IPX node to be
                             000b.000b.000b
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 3 ← Make this interface IPX Network 3
 no fair-queue
 clockrate 64000
!
interface Serial0/1
 no ip address
 encapsulation ppp
 ipx network 2
 clockrate 64000
!
no ip classless
!
!
ipx router eigrp 1 ← Enable IPX EIGRP autonomous system 1
 network all ← Include all IPX networks in EIGRP advertisements
!
!
no ipx router rip ← Do not enable IPX RIP on this router
!
ipx sap 4 Server1 1.00e0.1e5b.2601 451 1 ← Define a static SAP entry on this
                                           router
ipx sap 7 Server2 1.00e0.1e5b.2601 451 1 ← Define a static SAP entry on this
                                           router
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
```

```
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
ipx routing 000c.000c.000c ← Enable IPX routing. Define the IPX node to be
                              000c.000c.000c
!
interface Ethernet0/0
 no ip address
 no keepalive
 ipx network 4 ← Make this interface IPX Network 4
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 3 ← Make this interface IPX Network 3
 no fair-queue
!
no ip classless
!
!
ipx router eigrp 1 ← Enable IPX EIGRP autonomous system 1
 network 3 ← Include IPX Network 3 in EIGRP updates
!
!
ipx router rip ← Enable IPX RIP on this router
 no network 3 ← Do not advertise IPX Network 3 in RIP updates
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Looking at the configurations of our three routers we see that we have defined three static SAP entries:

1. RouterA has a static SAP entry to a server (Server4) that is located on IPX Network 4.
2. RouterB has a static SAP entry to a server (Server1) that is located on IPX Network 1.
3. RouterB has a second static SAP entry to a server (Server2) that is also located on IPX Network 1.

Let's connect to RouterA. We can view the known IPX servers with the **show ipx servers** command. Notice that RouterA only knows of one IPX server, Server4. This is the server that we have statically defined on RouterA. Why does RouterA not know about the two servers (Server1 and Server2) that we statically defined on RouterB? The answer requires an understanding of RIP/SAP split horizon. RIP/SAP split horizon says that a router will never advertise RIP routing or SAP server information out of the same interface that it learned the information from. In this case, the static SAP entry on RouterB that points to two servers on RouterA will never be broadcast to RouterA since RouterB treats the static entry as if it was learned from RouterA. Thus, RouterA should not have an entry for the two servers that were statically defined on RouterB.

```
RouterA#show ipx servers
Codes: S - Static, P - Periodic, E - EIGRP, N - NLSP, H - Holddown, + = detail
1 Total IPX Servers

Table ordering is based on routing and server info

   Type      Name              Net     Address    Port     Route Hops     Itf
S   4        Server4           4.00e0.1e5b.0a81:0451       2707456/01      2
```

```
Se0/0
```

Now connect to RouterB. Use the **show ipx servers** command to view all IPX servers known to RouterB. RouterB knows about two IPX servers. These are the two servers (Server1 and Server2) that we statically defined on RouterB. Why does RouterB not know about the IPX server (Server4) that is statically defined on RouterA ? Once again the answer is split horizon. The static SAP entry on RouterA points to IPX Network 4. The static SAP entry on RouterA is treated as if it were learned from RouterB since RouterB is the next hop towards IPX Network 4. Thus, RouterA will not send the static SAP entry to RouterB since it thinks that the entry came from RouterB in the first place.

```
RouterB#show ipx servers
Codes: S − Static, P − Periodic, E − EIGRP, N − NLSP, H − Holddown, + = detail
2 Total IPX Servers

Table ordering is based on routing and server info

   Type    Name              Net     Address    Port     Route Hops    Itf
S   4     Server1            1.00e0.1e5b.2601:0451       2195456/01    1 Se0/1
S   7     Server2            1.00e0.1e5b.2601:0451       2195456/01    1 Se0/1
```

Now let's connect to RouterC. The **show ipx servers** command shows us that RouterC knows about two IPX servers (Server1 and Server2). These are the two servers that were statically defined on RouterB. RouterB will advertise these server entries to RouterC because RouterB treats the static entries as if they were learned from RouterA. Thus, RouterB is allowed to send the static SAP entries to RouterC without violating the split horizon rule.

```
RouterC#show ipx servers
Codes: S − Static, P − Periodic, E − EIGRP, N − NLSP, H − Holddown, + = detail
2 Total IPX Servers

Table ordering is based on routing and server info

   Type    Name              Net     Address    Port     Route Hops    Itf
E     4    Server1            1.00e0.1e5b.2601:0451       2707456/01    2 Se0/0
E     7    Server2            1.00e0.1e5b.2601:0451       2707456/01    2 Se0/0
```

Let's turn on SAP debugging with the **debug ipx sap events** and **debug ipx sap activity** commands. Remember to also use the **term mon** command to direct the debug output to your terminal if you are not connected to the console port of the router.

```
RouterC#debug ipx sap activity
IPX service debugging is on

RouterC#debug ipx sap events
IPX service events debugging is on
```

The following output will be repeated every 60 seconds. We see that RouterC is sending a SAP update to IPX Network 4 telling it about two IPX servers (Server1 and Server2). Notice that we do not see any SAP updates coming into RouterC from RouterB. This is because we are running EIGRP on the WAN link between RouterC and RouterB, not RIP/SAP.

```
      RouterC broadcasts the SAP updates to the Ethernet LAN on Ethernet0/0
                                ↓
IPXSAP: positing update to 4.ffff.ffff.ffff via Ethernet0/0 (broadcast) (full)
IPXSAP: Update type 0x2 len 160 src:4.00e0.1e5b.0a81 dest:4.ffff.ffff.ffff(452)
 type 0x4, "Server1", 1.00e0.1e5b.2601(451), 2 hops ← RouterC advertises two
IPX servers to IPX Network 4
 type 0x7, "Server2", 1.00e0.1e5b.2601(451), 2 hops
```

Cisco supports extensive IPX filtering capabilities. One of the Cisco IPX features is the ability to filter outgoing or incoming SAP updates. This is frequently used for security purposes where you do not want

certain users or networks to know about specific servers. Let's change the configuration of RouterB so that RouterB only sends an IPX SAP server update to RouterC for Server1 and not Server2. Enter configuration mode with the **config term** command. Enter the global command **access−list 1000 deny −1 7 Server2** and **access−list 1000 permit −1**. Then go into interface configuration mode using the **int s 0/0** command and enter the command **ipx output−sap−filter 1000**. We have now configured an access list on RouterB that will not send out any updates for an IPX server named Server2 that is a SAP type 7.

```
RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#access−list 1000 deny −1 7 Server2
RouterB(config)#access−list 1000 permit −1
RouterB(config)#
RouterB(config)#int s 0/0
RouterB(config−if)#ipx output−sap−filter 1000
RouterB(config−if)#exit
RouterB(config)#exit
RouterB#
```

After entering the above access list commands on RouterB, quickly connect to RouterC. IPX SAP debugging should still be enabled on RouterC. The following debug output will be seen on RouterC. Notice how RouterC deletes the entry to Server2 by first declaring it unreachable (advertises it with a hop count of 16) and then no longer advertises it.

```
IPXEIGRP: Sending EIGRP SAP flash
IPXEIGRP: Received EIGRP SAP from 3.000b.000b.000b ← EIGRP update received
from RouterB

IPXSAP: positing update to 4.ffff.ffff.ffff via Ethernet0/0 (broadcast) (full)
IPXSAP: Update type 0x2 len 160 src:4.00e0.1e5b.0a81 dest:4.ffff.ffff.ffff(452)
 type 0x4, "Server1", 1.00e0.1e5b.2601(451), 2 hops
 type 0x7, "Server2", 1.00e0.1e5b.2601(451), 16 hops ← RouterC advertises
                                                       Server2 as being 16 hops
                                                       away. This means that it
                                                       is unreachable
IPXSAP: server type 7 named Server2 metric 255 being deleted
IPX: SAP queue−hash deleted for type 7, count 2

IPXSAP: positing update to 4.ffff.ffff.ffff via Ethernet0/0 (broadcast) (full)
IPXSAP: Update type 0x2 len 96 src:4.00e0.1e5b.0a81 dest:4.ffff.ffff.ffff(452)
 type 0x4, "Server1", 1.00e0.1e5b.2601(451), 2 hops ← RouterC no longer
                                                      advertises Server2

IPXSAP: positing update to 4.ffff.ffff.ffff via Ethernet0/0 (broadcast) (full)
IPXSAP: Update type 0x2 len 96 src:4.00e0.1e5b.0a81 dest:4.ffff.ffff.ffff(452)
 type 0x4, "Server1", 1.00e0.1e5b.2601(451), 2 hops ← RouterC no longer
                                                      advertises Server2
```

Turn off all debugging output with the **undebug all** command.

```
RouterC#undebug all
All possible debugging has been turned off
```

The **show ipx server** command should now only show one server, Server1.

```
RouterC#show ipx server
Codes: S − Static, P − Periodic, E − EIGRP, N − NLSP, H − Holddown, + = detail
1 Total IPX Servers

Table ordering is based on routing and server info

   Type  Name                  Net     Address    Port    Route Hops  Itf
E     4  Server1               1.00e0.1e5b.2601:0451      2707456/01  2  Se0/0
```

Let's reconnect to RouterB. Use the **show ipx server** command to display all known servers. We see that RouterB still knows about two servers — Server1 and Server2 — even though it is filtering any updates related to Server2 to RouterC.

```
RouterB#show ipx server
Codes: S − Static, P − Periodic, E − EIGRP, N − NLSP, H − Holddown, + = detail
2 Total IPX Servers

Table ordering is based on routing and server info

   Type  Name                    Net     Address    Port    Route Hops  Itf
S     4  Server1                 1.00e0.1e5b.2601:0451      2195456/01  1  Se0/1
S     7  Server2                 1.00e0.1e5b.2601:0451      2195456/01  1  Se0/1
```

The **show access−list** command can be used to verify that RouterB has an active access list.

```
RouterB#show access−list
IPX SAP access list 1000 ← Access list 1000
    deny FFFFFFFF 7 Server2 ← Do not sent any updates to any network regarding
                              IPX Server2 with a server type of 7
    permit FFFFFFFF ← Permit SAP updates to all other networks
```

Now let's remove the output−sap−filter from RouterB. Enter configuration mode and under interface s 0/0, type the command **no ipx output−sap−filter 1000**.

```
RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#int s 0/0
RouterB(config−if)#no ipx output−sap−filter 1000
RouterB(config−if)#exit
RouterB(config)#exit
```

Now connect to RouterC. After a few seconds, the entry for Server2 will reappear in the **show ipx server** output.

```
RouterC#show ipx server
Codes: S − Static, P − Periodic, E − EIGRP, N − NLSP, H − Holddown, + = detail
2 Total IPX Servers

Table ordering is based on routing and server info

   Type Name                   Net     Address    Port     Route Hops    Itf
E     4 Server1                1.00e0.1e5b.2601:0451       2707456/01    2 Se0/0
E     7 Server2                1.00e0.1e5b.2601:0451       2707456/01    2 Se0/0
             á
           The entry for Server2 will now be back in the IPX server list
```

Now we are going to add an input SAP filter on RouterC. An input SAP filter will filter out SAP updates that come into a router. Enter router configuration mode and enter the following **access−list** and **ipx input−sap−filter** statements.

```
RouterC#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterC(config)#access−list 1000 deny −1 4 Server1
RouterC(config)#access−list 1000 permit −1
RouterC(config)#exit
RouterC(config)#int s 0/0
RouterC(config−if)#ipx input−sap−filter 1000 ← Deny any incoming SAP
                                               advertisements that are for server type 4 and
                                               for a server named Server1
RouterC(config−if)#exit
RouterC#
```

Now view the IPX server list for RouterC with the **show ipx server** command. After a few minutes, the entry for Server1 will no longer be listed. RouterC is now filtering out these incoming SAP advertisements.

```
RouterC#sh ipx server
Codes: S − Static, P − Periodic, E − EIGRP, N − NLSP, H − Holddown, + = detail
1 Total IPX Servers

Table ordering is based on routing and server info

   Type  Name                  Net     Address    Port     Route Hops    Itf
E     7  Server2               1.00e0.1e5b.2601:0451       2707456/01    2 Se0/0
```

The Cisco IOS also provides extensive router filtering capabilities. Output route filters prevent routes to selected networks from being advertised to other routers. Input route filters prevent advertised routes from being entered into the IPX routing table. Let's start off with an output route filter. View the IPX routing table of RouterC with the **show ipx route** command. We see that RouterC has learned about IPX Networks 1, 2, and 5 via EIGRP.

```
RouterC#show ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          3 (PPP),           Se0/0
C          4 (NOVELL−ETHER),  Et0/0

Routes to Networks 1, 2, and 5 are learned via EIGRP
↓
E          1 [2707456/1] via        3.000b.000b.000b, age 00:03:23,
                            4u, Se0/0
E          2 [2681856/0] via        3.000b.000b.000b, age 00:03:24,
                            1u, Se0/0
E          5 [2809856/1] via        3.000b.000b.000b, age 00:03:24,
                            1u, Se0/0
```

Connect to RouterA and enter configuration mode. Enter the following **access−list** and **distribute−list** commands. A **distribute−list** command is used with EIGRP to filter routes. The access list will deny RouterA from advertising any information on IPX network 5.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#access−list 810 deny 5 ← Do not advertise IPX Network 5
RouterA(config)#access−list 810 permit −1 ← Advertise all other IPX networks
RouterA(config)#
RouterA(config)#router eigrp 1
RouterA(config−ipx−router)#distribute−list 810 out
RouterA(config−ipx−router)#exit
RouterA(config)#exit
```

Now connect to RouterC. After a short period, the **show ipx route** command will reveal that the entry for a route to IPX Network 5 is no longer in the routing table.

```
RouterC#sh ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

4 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.
```

```
No default route known.

C          3 (PPP),           Se0/0
C          4 (NOVELL-ETHER),  Et0/0
E          1 [2707456/1] via        3.000b.000b.000b, age 00:00:34,
                              2u, Se0/0
E          2 [2681856/0] via        3.000b.000b.000b, age 00:09:09,
                              1u, Se0/0
```

Now connect to RouterB. Use the **show ipx route** command to examine the routing table. Notice that the route to IPX Network 5 has also been deleted from RouterB's routing table. RouterA is no longer advertising IPX Network 5 to either RouterB or RouterC.

```
RouterB#sh ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses

4 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          2 (PPP),           Se0/1
C          3 (PPP),           Se0/0
E          1 [2195456/1] via        2.000a.000a.000a, age 00:01:52,
                     15u, Se0/1
E          4 [2195456/1] via        3.000c.000c.000c, age 00:01:53,
                     7u, Se0/0
```

Now we will add an input route filter. Enter router configuration mode on RouterC. Add the following **access-list** and **distribute-list** commands. This access list will filter any incoming advertisements for IPX Network 1 that come into RouterC.

```
RouterC#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterC(config)#access-list 820 deny 1 ← Filter out any routing updates for IPX Network 1
RouterC(config)#access-list 820 permit -1
RouterC(config)#
RouterC(config)#ipx router eigrp 1
RouterC(config-ipx-router)#distribute-list 820 in
RouterC(config-ipx-router)#exit
RouterC(config)#exit
```

Now take a look at the IPX routing table for RouterC with the **show ipx route** command. The routing entry to IPX Network 1 has been removed from the routing table.

```
RouterC#sh ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses

3 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          3 (PPP),           Se0/0
C          4 (NOVELL-ETHER),  Et0/0
E          2 [2681856/0] via        3.000b.000b.000b, age 00:00:08,
                     1u, Se0/0
```

Connect to RouterB and use the **show ipx route** command to view the routing table. We see that the route to IPX Network 1 is still in the routing table. This is because we are filtering this route as it comes into RouterC. The route is not filtered to RouterB.

```
RouterB#sh ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

4 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          2 (PPP),           Se0/1
C          3 (PPP),           Se0/0
E          1 [2195456/1] via        2.000a.000a.000a, age 00:03:40,
                     27u, Se0/1
E          4 [2195456/1] via        3.000c.000c.000c, age 00:00:23,
                     2u, Se0/0
```

Lab #84: IPX Configuration Over a Frame Relay Core

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers. Three of the routers must have one serial interface, and the other router must have three serial interfaces.
- Three Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports the IPX protocol.

Configuration Overview

This lab will demonstrate how to configure IPX to run over a Frame Relay network. Frame Relay is a NBMA (nonbroadcast multiple access) technology. Configuring IPX to run over a Frame Relay core requires special considerations, such as knowing how to configure split horizons.

As shown in Figure 18−9, RouterA, RouterB, and RouterC are each connected to a Frame Relay switch. The Frame Relay switch is a fourth router that is only configured for Frame Relay switching. Each of the three routers running IPX will be assigned an internal IPX loopback network number. We will see in this lab that we will be able to learn each of these internal networks over the Frame Relay core.

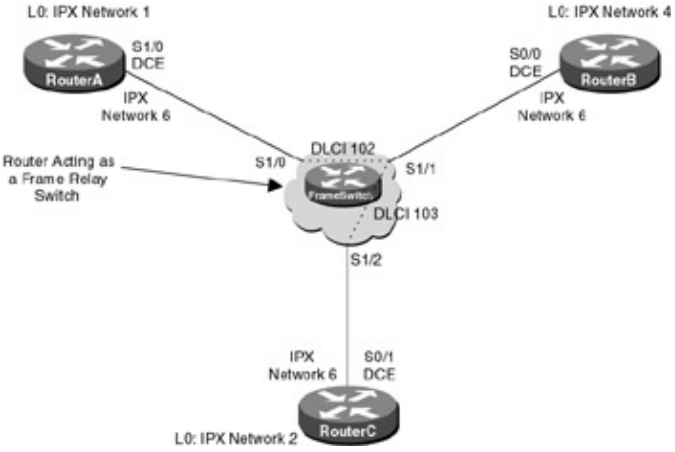


Figure 18−9: IPX over Frame Relay

Router Configuration

The configurations for the routers in this example are as follows (key IPX commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
ipx routing 0001.0001.0001
!
interface Loopback1
 no ip address
 ipx network 1
!
interface Serial1/0
 encapsulation frame-relay
 ipx network 6
 no fair-queue
 clockrate 800000
 frame-relay map ipx 6.0002.0002.0002 102 broadcast ← Frame Relay map
                                                       statements are used to
                                                       control which DLCIs will
                                                       carry traffic
 frame-relay map ipx 6.0004.0004.0004 102 broadcast
 no frame-relay inverse-arp ← Disable inverse ARP since we are using map
                              statements
 frame-relay lmi-type ansi
!
ipx router eigrp 100
 network 6
!
ipx router rip
 no network 6
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
```

```
!
end
```

RouterB

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
ipx routing 0004.0004.0004
!
interface Loopback1
 no ip address
 ipx network 4
!
interface Serial0/0
 encapsulation frame-relay
 ipx network 6
 no ipx split-horizon eigrp 100 ← RouterB is the hub router. EIGRP split
                                  horizon needs to be disabled on this router
 clockrate 800000
 frame-relay map ipx 6.0001.0001.0001 102 broadcast
 frame-relay map ipx 6.0002.0002.0002 103 broadcast
 no frame-relay inverse-arp
 frame-relay lmi-type ansi
!
ipx router eigrp 100
 network 6
!
ipx router rip
 no network 6
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
enable password cisco
!
ipx routing 0002.0002.0002
!
interface Loopback1
 no ip address
```

```
 ipx network 2
!
interface Serial0/0
 encapsulation ppp
 ipx network 7
!
interface Serial0/1
 encapsulation frame-relay
 ipx network 6
 clockrate 800000
 frame-relay map ipx 6.0001.0001.0001 103 broadcast
 frame-relay map ipx 6.0004.0004.0004 103 broadcast
 no frame-relay inverse-arp
 frame-relay lmi-type ansi
!
ipx router eigrp 100
 network 6
 network 7
!
ipx router rip
 no network 6
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

FrameSwitch

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
!
frame-relay switching
!
interface Serial1/0
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 102 interface Serial1/1 102
!
interface Serial1/1
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 102 interface Serial1/0 102
 frame-relay route 103 interface Serial1/2 103
!
interface Serial1/2
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 103 interface Serial1/1 103
!
no ip classless
!
```

```
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Use the **show ipx route** command to verify that all of the neighboring networks are being learned over the Frame Relay core. We see that RouterA is learning IPX Network 2 and IPX Network 4 via IPX EIGRP.

```
RouterA#show ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

4 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          1 (UNKNOWN),       Lo1
C          6 (FRAME−RELAY),   Se1/0
E          2 [2809856/1] via        6.0004.0004.0004, age 00:43:01,
                     1u, Se1/0
E          4 [2297856/1] via        6.0004.0004.0004, age 00:43:57,
                     1u, Se1/0
```

Now let's connect to RouterB. RouterB is the hub router in this configuration. Verify with the **show ipx route** command that RouterB is learning routes to the other networks in this configuration. We see that RouterB had learned routes to IPX Network 1 and IPX Network 2 via IPX EIGRP. These are the two loopback networks on RouterA and RouterC.

```
RouterB#show ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

4 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          4 (UNKNOWN),       Lo1
C          6 (FRAME−RELAY),   Se0/0
E          1 [2297856/1] via        6.0001.0001.0001, age 00:44:26,
                     6u, Se0/0
E          2 [2297856/1] via        6.0002.0002.0002, age 00:43:31,
                     1u, Se0/0
```

The **show ipx eigrp neighbor** command on RouterB shows us that RouterB has established EIGRP neighbors on RouterA (6.0001.0001.0001) and Router C (6.0002.0002.0002).

```
RouterB#show ipx eigrp neigh

IPX EIGRP Neighbors for process 100
H   Address                 Interface   Hold Uptime   SRTT   RTO  Q  Seq
                                        (sec)         (ms)       Cnt Num
1   6.0002.0002.0002        Se0/0        179 00:44:18    5   200  0  21
0   6.0001.0001.0001        Se0/0        175 00:44:39    5   200  0  17
```

Now connect to RouterC. Verify that routes are being learned to the loopback networks on RouterA and RouterB. We see below that routes are being learned to IPX Networks 1 and 4:

```
RouterC#sh ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

4 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          2 (UNKNOWN),       Lo1
C          6 (FRAME−RELAY),   Se0/1
E          1 [2809856/1] via        6.0004.0004.0004, age 00:10:55,
                     2u, Se0/1
E          4 [2297856/1] via        6.0004.0004.0004, age 00:10:55,
                     1u, Se0/1
```

Let's verify that we have end−to−end connectivity by trying to ping the IPX loopback interface on RouterA with the **ping ipx 1.1.1.1** command. We see below that the ping is successful:

```
RouterC#ping ipx 1.1.1.1

Type escape sequence to abort.
Sending 5, 100−byte IPX cisco Echoes to 1.0001.0001.0001, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 28/29/32 ms
```

Now connect to RouterB. Go into configuration mode and enable split horizon on interface S0/0.

```
RouterB(config)#interface Serial0/0
RouterB(config−if)#ipx split−horizon eigrp 100
```

Reconnect to RouterA. We see from the **show ipx route** command that RouterA is no longer learning any routes to the other networks due to split horizon being enabled.

```
RouterA#show ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses

2 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

No default route known.

C          1 (UNKNOWN),       Lo1
C          6 (FRAME−RELAY),   Se1/0
```

Lab #85: IPX Dial Backup

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers. Each router must have one serial interface and one BRI interface.
- One Cisco crossover cable. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.

- Two ISDN BRI cables.
- A Cisco rolled cable for console port connection to the routers.
- Cisco IOS image that supports the IPX protocol.
- Two ISDN BRI circuits.

Configuration Overview

This lab will demonstrate how to configure a router for IPX dial backup using an IPX floating static route. An IPX floating static route appears in the IPX routing table as a default route. It will not be used unless a route to a given destination does not exist.

The two routers are connected as shown in Figure 18−10. RouterA acts as a DCE and supplies clocking to RouterB.

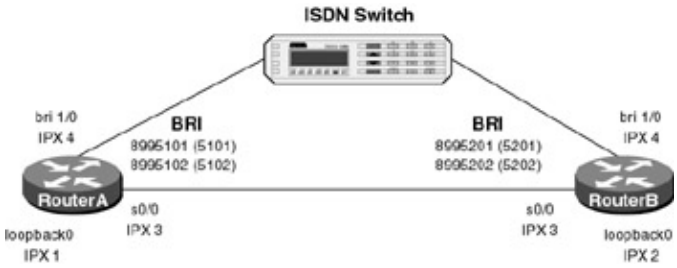


Figure 18−10: IPX dial backup

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. For this lab we used an Adtran Atlas 800. Information on configuring the Adtran Atlas 800 switch can be found in Chapter 3.

Router Configuration

The configurations for the routers in this example are as follows (key IPX dial backup commands are highlighted in bold).

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
!
username RouterB password 0 cisco
!
!
ip subnet-zero
!
lane client flush
ipx routing 000a.000a.000a
isdn switch-type basic-ni ← Set the ISDN switch type
cns event-service server
!
!
interface Loopback0
 no ip address
 ipx network 1
!
```

```
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 3
 no fair-queue
 clockrate 800000
!
interface BRI1/0
 no ip address
 encapsulation ppp
 dialer map ipx 4.000b.000b.000b name RouterB broadcast 8995201 ← Configure the
                                                                  dialer map
 dialer load-threshold 255 either ← Set the load threshold to the maximum limit
                                    so only one B channel will be used to make
                                    our calls
 dialer-group 1 ← Assign this interface to dialer group 1
 ipx network 4
 isdn switch-type basic-ni
 isdn spid1 5101 8995101 ← Set the SPID values for the ISDN circuit
 isdn spid2 5102 8995102
 ppp authentication chap
!
ip classless
no ip http server
!
access-list 900 permit any any cping ← Access list 900 defines interesting
                                       traffic
access-list 900 deny rip
access-list 900 permit any any
dialer-list 1 protocol ipx list 900 ← Define interesting traffic parameters
!
!
ipx route default 4.000b.000b.000b floating-static ← Configure the IPX floating
                                                     static route
!
ipx router eigrp 1
 network 3
!
!
ipx router rip
 no network 3
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
!
username RouterA password 0 cisco
!
!
ip subnet-zero
```

```
!
lane client flush
ipx routing 000b.000b.000b
isdn switch-type basic-ni
cns event-service server
!
!
interface Loopback0
 no ip address
 ipx network 2
!
interface Serial0/0
 no ip address
 encapsulation ppp
 ipx network 3
!
interface BRI1/0
 no ip address
 encapsulation ppp
 dialer map ipx 4.000a.000a.000a name RouterA broadcast
 dialer load-threshold 1 either
 dialer-group 1
 ipx network 4
 isdn switch-type basic-ni
 isdn spid1 5201 8995201
 isdn spid2 5202 8995202
 ppp authentication chap
!
ip classless
no ip http server
!
access-list 900 permit any any cping
access-list 900 deny rip
dialer-list 1 protocol ipx list 900
!
!
ipx router eigrp 1
 network 3
 network 2
!
!
ipx router rip
 no network 2
 no network 3
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. The **show isdn status** command indicates that the ISDN circuit has been properly configured. Notice that both SPIDs have been sent to the ISDN switch and validated.

```
RouterA#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI1/0  interface
        dsl 8, interface ISDN Switchtype = basic-ni
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
```

```
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 8(established)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 8(established)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 0
    The Free Channel Mask:  0x80000003
    Total Allocated ISDN CCBs = 0
```

We see from the **show ipx route** command that RouterA is learning about the loopback interface (IPX Network 2) on RouterB via IPX EIGRP. We also see that that IPX Network 2 is being learned via IPX Network 3 (the serial link between RouterA and RouterB). Notice that the IPX routing table also contains an entry for the floating static route that we defined. Since this is an IPX floating static route, it will not be installed in the routing table unless no other routes to a given destination exist.

```
RouterA#show ipx route
Codes: C − Connected primary network,    c − Connected secondary network
       S − Static, F − Floating static, L − Local (internal), W − IPXWAN
       R − RIP, E − EIGRP, N − NLSP, X − External, A − Aggregate
       s − seconds, u − uses, U − Per−user static

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

Current default route is:

F   FFFFFFFE via        4.000b.000b.000b,        BR1/0 ← Floating static route

C          1 (UNKNOWN),       Lo0
C          3 (PPP),           Se0/0
C          4 (PPP),           BR1/0
E          2 [1889792/0] via        3.000b.000b.000b, age 00:33:57,
                     1u, Se0/0
```

Now connect to RouterB. Use the **show isdn status** command to verify that the ISDN circuit is ready to receive a call. We see that both spids have been successfully sent to the ISDN switch.

```
RouterB#show isdn status
Global ISDN Switchtype = basic−ni
ISDN BRI1/0  interface
     dsl 8, interface ISDN Switchtype = basic−ni
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 0
    The Free Channel Mask:  0x80000003
    Total Allocated ISDN CCBs = 0
```

Now reconnect to RouterA. Enable PPP authentication and dialer debugging with the **debug ppp authentication** and **debug dialer** commands.

```
RouterA#debug ppp authentication
PPP authentication debugging is on

RouterA#debug dialer
Dial on demand events debugging is on
```

Now we will start an extended ping from RouterA to the loopback interface of RouterB. After the ping has started, the serial cable connecting RouterA to RouterB should be disconnected.

```
RouterA#ping
Protocol [ip]: ipx
Target IPX address: 2.b.b.b
Repeat count [5]: 1000
Datagram size [100]: 1500
Timeout in seconds [2]:
Verbose [n]:
Type escape sequence to abort.
Sending 10000, 1500-byte IPX Novell Echoes to 2.000b.000b.000b, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!..
          á
After the ping from RouterA to RouterB has begun, pull the serial cable connecting
RouterA to RouterB. When the cable is pulled, the ping will start to fail
```

After the serial cable is disconnected, the ping will begin to fail. If PPP authentication debugging is enabled, the following output will be seen, indicating that RouterA is placing an ISDN call to RouterB:

```
02:48:43: BR1/0 DDR: Dialing cause ipx (s=4.000a.000a.000a, d=2.000b.000b.000b)
02:48:43: BR1/0 DDR: Attempting to dial 8995201
02:48:43: %LINK-3-UPDOWN: Interface Serial0/0, changed state to down
02:48:43: %LINK-3-UPDOWN: Interface BRI1/0 :1, changed state to up
02:48:43: BR1/0:1 PPP: Treating connection as a callout
02:48:43: BR1/0:1 CHAP: O CHALLENGE id 9 len 28 from "RouterA"
02:48:43: BR1/0:1 CHAP: I CHALLENGE id 9 len 28 from "RouterB"
02:48:43: BR1/0:1 CHAP: O RESPONSE id 9 len 28 from "RouterA"
02:48:43: BR1/0:1 CHAP: I SUCCESS id 9 len 4
02:48:43: BR1/0:1 CHAP: I RESPONSE id 9 len 28 from "RouterB"
02:48:43: BR1/0:1 CHAP: O SUCCESS id 9 len 4
02:48:43: BR1/0:1 DDR: dialer protocol up
02:48:44: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to down
02:48:44: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to up
02:48:49: %ISDN-6-CONNECT: Interface BRI1/0 :1 is now connected to 8995201 RouterB
Once the ISDN circuit is established, the ping will start to pass once again
↓
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!.
Success rate is 99 percent (576/579), round-trip min/avg/max = 32/121/388 ms
```

When the ping has completed, check the routing table with the **show ipx route** command. We see below that RouterA is now learning about the loopback network (IPX Network 2) on RouterB via IPX RIP over the ISDN interface:

```
RouterA#show ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses, U - Per-user static

4 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

Current default route is:

F   FFFFFFFE via        4.000b.000b.000b,        BR1/0
```

```
C          1 (UNKNOWN),       Lo0
C          4 (PPP),           BR1/0
R          2 [07/01] via        4.000b.000b.000b,   32s, BR1/0
á
```
RouterA now learns about RouterB's loopback interface via IPX RIP over the ISDN interface

Now reconnect the serial cable between RouterA and RouterB. After the ISDN idle timer expires, the ISDN call will be disconnected.

```
02:52:45: %ISDN-6-DISCONNECT: Interface BRI1/0:1  disconnected from
8995201 RouterB, call lasted 119 seconds
02:52:45: %LINK-3-UPDOWN: Interface BRI1/0 :1, changed state to down
02:52:45: BR1/0 :1 DDR: disconnecting call
02:52:46: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0 :1, changed state to down
```

After the ISDN call is disconnected, check the routing table with the **show ipx route** command. We see that RouterA is once again learning about RouterB's loopback interface via the serial cable connecting RouterA and RouterB.

```
RouterA#show ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses, U - Per-user static

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.

Current default route is:

F   FFFFFFFE via        4.000b.000b.000b,        BR1/0

C          1 (UNKNOWN),       Lo0
C          3 (PPP),           Se0/0
C          4 (PPP),           BR1/0
E          2 [1889792/0] via        3.000b.000b.000b, age 00:01:56,
                     1291u, Se0/0
```

IPX Monitoring and Troubleshooting Commands

This section will discuss key IPX monitoring and troubleshooting commands.

{show ipx interface brief} The **show ipx interface brief** command can be used to get a quick snapshot of the state of all interfaces on a router that are running the IPX protocol.

```
RouterA#show ipx interface brief
Interface            IPX Network Encapsulation Status                IPX State
Ethernet0/0          1           NOVELL-ETHER  up                    [up]
Serial0/0            2           PPP           up                    [up]
Loopback0            5           UNKNOWN       up                    [up]
```

{show ipx route} Typing the **show ipx route** command displays the routing table for this router. This routing table shows us that three IPX networks are directly connected: Network 1 is on Ethernet0, Network 2 is on Serial 0, and Network 5 is on Loopback 0. RouterA has learned about two networks via the IPX RIP routing protocol. Network 3 is 1 hop and 7 ticks away, and Network 4 is 2 hops and 13 ticks away.

```
RouterA#show ipx route
Codes: C - Connected primary network,    c - Connected secondary network
       S - Static, F - Floating static, L - Local (internal), W - IPXWAN
       R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
       s - seconds, u - uses

5 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.
```

```
No default route known.

C          1 (NOVELL-ETHER),  Et0/0
C          2 (PPP),           Se0/0
C          5 (UNKNOWN),       Lo0

            Tick count         Next hop address
              ↓                 ↓
R          3 [07/01] via        2.000b.000b.000b,   49s, Se0/0
                 á
                 Hop count to destination network

            Tick count         Next hop address
              ↓                 ↓
R          4 [13/02] via         2.000b.000b.000b,   50s, Se0/0
                 á
                 Hop count to destination network
```

{show interface} When running IPX, there are two **show interface** commands that refer to the interface. The **show interface** command will show what link control protocols have been negotiated and opened. Traffic information and lead state status for the interface will also be displayed.

```
RouterA#show int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open
  Open: CDPCP, IPXCP ← No IP is enabled on this interface
  Last input 00:00:01, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     99 packets input, 3888 bytes, 0 no buffer ← Packet's input
     Received 99 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     100 packets output, 3902 bytes, 0 underruns ←  Packet's output
     0 output errors, 0 collisions, 16 interface resets
     0 output buffer failures, 0 output buffers swapped out
     31 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

{show ipx interface} Specific IPX information for an interface running the IPX protocol can be displayed with the **show ipx int s 0/0** command. This command shows the IPX address of the interface as well as IPX routing, filtering, and SAP information.

```
RouterA#show ipx int s 0/0
Serial0/0 is up, line protocol is up
  IPX address is 2.000a.000a.000a [up] ← IPX address

      A WAN interface has a default IPX delay of 6
                                          ↓
  Delay of this IPX network, in ticks is 6 throughput 0 link delay 0
  IPXWAN processing not enabled on this interface.
  IPX SAP update interval is 1 minute(s)
  IPX type 20 propagation packet forwarding is disabled
  Incoming access list is not set
  Outgoing access list is not set
  IPX helper access list is not set
  SAP GNS processing enabled, delay 0 ms, output filter list is not set
  SAP Input filter list is not set
  SAP Output filter list is not set
  SAP Router filter list is not set
  Input filter list is not set
```

```
  Output filter list is not set
  Router filter list is not set
  Netbios Input host access list is not set
  Netbios Input bytes access list is not set
  Netbios Output host access list is not set
  Netbios Output bytes access list is not set
  Updates each 60 seconds, aging multiples RIP: 3 SAP: 3
  SAP interpacket delay is 55 ms, maximum size is 480 bytes
  RIP interpacket delay is 55 ms, maximum size is 432 bytes
  Watchdog processing is disabled, SPX spoofing is disabled, idle time 60
  IPX accounting is disabled
  IPX fast switching is configured (enabled)
  RIP packets received 9, RIP packets sent 9 ← RIP is running on this interface
  SAP packets received 1, SAP packets sent 1 ← SAP is running on this interface
```

{ping ipx} IPX is limited in its diagnostic capabilities as compared to IP. With IPX, the only tool available to test network connectivity is the **ping ipx** command.

```
RouterA#ping ipx 2.b.b.b ← ping RouterB

Type escape sequence to abort.
Sending 5, 100-byte IPX cisco Echoes to 2.000b.000b.000b, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

{show ipx traffic} The **show ipx traffic** command displays IPX traffic information for all interfaces on the router. User traffic, routing protocols, and SAP statistics are displayed.

```
RouterC#show ipx traffic
System Traffic for 0.0000.0000.0001 System-Name: RouterC
Rcvd:   36 total, 0 format errors, 0 checksum errors, 0 bad hop count,
        0 packets pitched, 36 local destination, 0 multicast
Bcast:  16 received, 29 sent
Sent:   50 generated, 0 forwarded
        0 encapsulation failed, 0 no route
SAP:    1 SAP requests, 0 SAP replies, 0 servers
        0 SAP Nearest Name requests, 0 replies
        0 SAP General Name requests, 0 replies
        5 SAP advertisements received, 4 sent
        2 SAP flash updates sent, 0 SAP format errors
RIP:    1 RIP requests, 0 RIP replies, 5 routes
        9 RIP advertisements received, 18 sent
        2 RIP flash updates sent, 0 RIP format errors
Echo:   Rcvd 5 requests, 15 replies
        Sent 15 requests, 5 replies
        0 unknown: 0 no socket, 0 filtered, 0 no helper
        0 SAPs throttled, freed NDB len 0
Watchdog:
        0 packets received, 0 replies spoofed
Queue lengths:
        IPX input: 0, SAP 0, RIP 0, GNS 0
        SAP throttling length: 0/(no limit), 0 nets pending lost route reply
      Delayed process creation: 0
EIGRP:  Total received 0, sent 0
        Updates received 0, sent 0
        Queries received 0, sent 0
        Replies received 0, sent 0
        SAPs received 0, sent 0
NLSP:   Level-1 Hellos received 0, sent 0
        PTP Hello received 0, sent 0
        Level-1 LSPs received 0, sent 0
        LSP Retransmissions: 0
        LSP checksum errors received: 0
        LSP HT=0 checksum errors received: 0
        Level-1 CSNPs received 0, sent 0
        Level-1 PSNPs received 0, sent 0
```

```
        Level-1 DR Elections: 0
        Level-1 SPF Calculations: 0
        Level-1 Partial Route Calculations: 0
```

{show ipx eigrp neighbor} The **show ipx eigrp neighbor** command will display information on what neighboring EIGRP routers have been discovered.

```
RouterA#show ipx eigrp neigh

IPX EIGRP Neighbors for process 1
H   Address                 Interface   Hold Uptime   SRTT   RTO  Q  Seq
                                        (sec)         (ms)       Cnt Num
0   2.000b.000b.000b        Se0/0         13 02:10:19   53   318  0  38
```

{show ipx eigrp interfaces} The **show ipx eigrp interfaces** command will show what router interfaces are running EIGRP.

```
RouterA#show ipx eigrp interfaces

IPX EIGRP Interfaces for process 1

                   Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Se0/0          1        0/0        53       0/15         263           0
á
Interface S0/0 is running EIGRP
```

{show ipx eigrp traffic} The **show ipx eigrp traffic** command is a useful command that shows how much EIGRP traffic has been sent and received on the router.

```
RouterB#show ipx eigrp traffic
IP-EIGRP Traffic Statistics for process 1
  Hellos sent/received: 3433/3430
  Updates sent/received: 11/11
  Queries sent/received: 10/7
  Replies sent/received: 7/10
  Acks sent/received: 37/33
  Input queue high water mark 2, 0 drops
```

{show ipx servers} The **show ipx servers** command will display any servers that have either been statically defined on the router or learned via SAP updates.

```
RouterA#show ipx servers
Codes: S - Static, P - Periodic, E - EIGRP, N - NLSP, H - Holddown, + = detail
1 Total IPX Servers

Table ordering is based on routing and server info

   Type Name                  Net      Address    Port     Route Hops   Itf
S     4 Server4               4.00e0.1e5b.0a81:0451       2707456/01   2 Se0/0
```

{show access-list} The **show access-list** command is used to display information on access lists that have been defined on the router.

```
RouterB#show access-list
IPX SAP access list 1000 ← Access list 1000
    deny FFFFFFFF 7 Server2 ← Do not sent any updates to any network regarding
                              IPX Server2 with a server type of 7
    permit FFFFFFFF ← Permit SAP updates to all other networks
```

{debug ipx routing activity}

{debug ipx routing events} The **debug ipx routing activity** and d**ebug ipx routing events** commands display information on IPX RIP routing protocol activity.

```
RouterB#debug ipx routing activity
IPX routing debugging is on

RouterB#debug ipx routing events
IPX routing events debugging is on
```

{debug ipx sap activity}

{debug ipx sap events} The **debug ipx sap activity** and **debug ipx sap events** commands will display information SAP packets being sent or received on the router.

```
RouterC#debug ipx sap activity
IPX service debugging is on

RouterC#debug ipx sap events
IPX service events debugging is on
```

Conclusion

This chapter explored the Novell IPX networking protocol. Although it is declining in popularity, Novell IPX is still in widespread use. The hands−on labs in this chapter explored key Novell IPX topics such as

- Basic IPX configuration and monitoring
- IPX EIGRP configuration
- IPX static SAP entries
- IPX SAP and router filtering capabilities
- Configuring IPX over a Frame Relay core
- IPX dial backup with PPP callback

Chapter 19: AppleTalk

Overview

Topics Covered in This Chapter

- AppleTalk technology overview
- Cisco AppleTalk support
- AppleTalk EIGRP configuration
- AppleTalk GRE tunnels
- AppleTalk traffic filtering
- AppleTalk Zone filtering
- AppleTalk over Frame Relay
- AppleTalk dial backup with floating static routes
- Troubleshooting AppleTalk networks

Introduction

AppleTalk is a networking protocol developed by Apple Computer to provide networking services for its Macintosh computers. AppleTalk is the most automatic of all the desktop protocols, but it is also the chattiest. For example, the default routing protocol for AppleTalk is RTMP. RTMP sends routing updates every 10 seconds to all directly connected neighbors.

AppleTalk Terminology

An AppleTalk node can be any device that is connected to an AppleTalk network and is assigned an AppleTalk address. Nodes can be Macintosh computers, printers, or any other device that resides on the network and is addressable.

An AppleTalk network can be thought of as a physical LAN or WAN that contains one or more AppleTalk nodes.

An AppleTalk zone is a logical group of networks. A zone will usually consist of AppleTalk nodes that reside in different physical locations. Zones are very similar in concept to a virtual LAN. In Figure 19−1 we see an example of how AppleTalk zones can work. Figure 19−1 shows an AppleTalk network with three Ethernet segments. The Ethernet segments on RouterA and RouterB are both in zone Engineering. When a Macintosh user on the Ethernet LAN connected to RouterC wants to access resources in the Engineering zone, he or she is given access to the LAN on RouterA and RouterB. Zones allow you to functionally group network resources without any regard to their actual physical location.

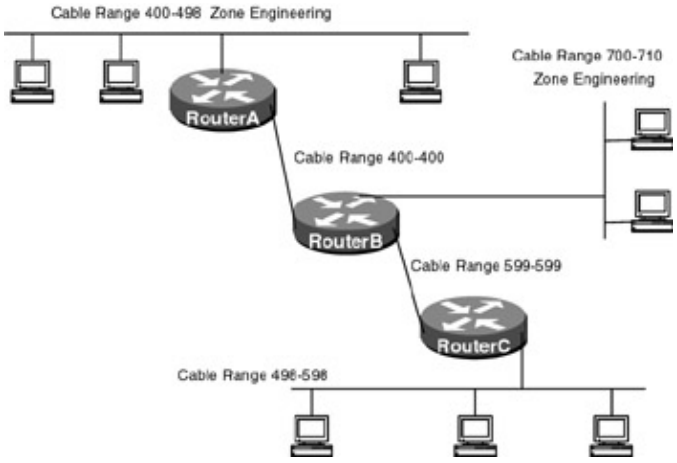


Figure 19−1: Improper AppleTalk address range

AppleTalk Addressing

Early AppleTalk networks were referred to as Phase I or nonextended networks. Phase I networks had a limited address space. Each LAN or WAN segment was allowed to contain up to 127 hosts and up to 127 servers. Each LAN or WAN segment could only be assigned a single AppleTalk network number.

AppleTalk Phase II networks are much more flexible in their network addressing. A Phase II network allows multiple network numbers to exist on each network segment. This means that a LAN can contain multiple AppleTalk networks. The range of network numbers that exist on a network segment is referred to as the *cable range* of the segment. The cable range must be unique and cannot overlap with other router interfaces. Figure 19−1 shows an example of an AppleTalk network with improperly assigned cable range numbers. In the case of Figure 19−1, there is an address conflict since network 498 has been assigned to both Ethernet LANs. Figure 19−2 shows a properly configured AppleTalk network — there are no address overlaps.

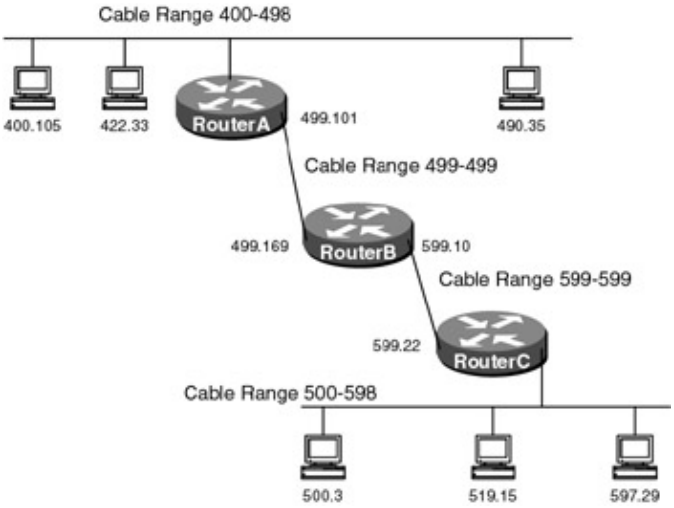


Figure 19−2: Proper AppleTalk address assignment

AppleTalk node address assignment is designed to minimize the amount of configuration needed on a Macintosh computer. When a Macintosh is first powered on, it sends a broadcast to any routers on the same network segment asking what the cable range of the network segment is. Once a router responds, the Macintosh chooses a network number within the cable range. The Macintosh then picks a node number. Before the AppleTalk node uses the network.node combination it has picked, it queries the network to see if the network.node combination is already in use. If the address is already used, it will continue to choose new addresses until an unused address has been found.

As shown in Figure 19−3, an AppleTalk address is 24 bits long. The address is written in a network.node format. The first 16 bits are the network number and the last eight bits are the node number. This means that all AppleTalk networks will be numbered less than 65,536 and all AppleTalk nodes will be numbered less than 256. Node numbers 0 and 255 are reserved (255 is used as a network broadcast address). An AppleTalk network can therefore have 254 nodes per network.

Figure 19−3: AppleTalk address structure

AppleTalk Protocol Stack

Figure 19−4 shows the AppleTalk stack and its relationship to the OSI stack.

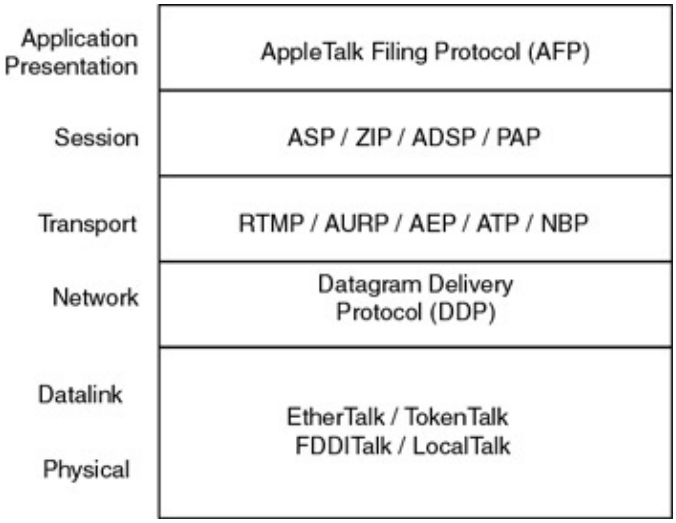


Figure 19−4: AppleTalk protocol stack

Physical and Datalink Layers

In addition to being supported on WAN links such as frame relay and ISDN, AppleTalk is supported on four major LAN platforms:

- **EtherTalk:** Apple's version of Ethernet.
- **TokenTalk:** Apple's version of token ring.
- **FDDITalk:** Apple's version of FDDI.
- **AppleTalk:** An Apple proprietary serial link that runs at 230Kbps.

Network Layer

AppleTalk uses the Datagram Delivery Protocol (DDP) at the network layer for routing packets in a network. AppleTalk is a routable protocol since it has a network layer address associated with each AppleTalk node. DDP is a connectionless network protocol. Figure 19−5 shows the DDP packet header in more detail.

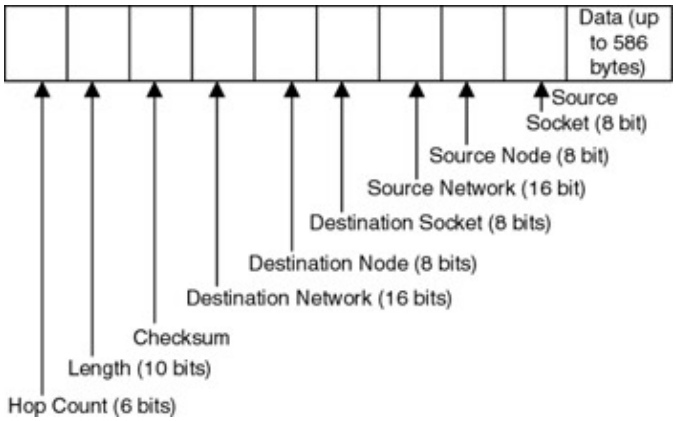


Figure 19−5: DDP packet

Transport Layer

Several protocols exist in the AppleTalk transport layer:

- **Routing Table Maintenance Protocol (RTMP):** A distance vector routing protocol that is similar to IP RIP. RTMP is very chatty. It sends out a routing update to all connected neighbors every 10 seconds.
- **AppleTalk Echo Protocol (AEP):** AEP is a simple protocol that generates packets that can be used to test the reachability of various network nodes.
- **AppleTalk Transaction Protocol (ATP):** ATP provides connection−based data transfer for AppleTalk traffic. It functions in a similar mode to TCP in an IP network. ATP provides for data

acknowledgment, retransmission, packet sequencing, and fragmentation and reassembly.
- **NBP:** The Name Binding Protocol associates an AppleTalk name with an address.

Session Layer

AppleTalk supports several upper−layer protocols:

- **AppleTalk Session Protocol (ASP):** ASP establishes and maintains sessions between an AppleTalk client and a server.
- **Zone Information Protocol:** The Zone Information Protocol maintains network number to zone name mappings in zone information tables. ZIP uses RTMP routing tables to keep up with network topology changes. When ZIP finds a routing table entry that is not in the ZIP, it creates a new ZIP entry.
- **AppleTalk Printer Access Protocol (PAP):** PAP is a connection−oriented protocol that establishes and maintains connections between clients and printers.

Application/Presentation Layer

The AppleTalk Filing Protocol (AFP) helps clients share server files across a network.

AppleTalk Routing Protocols

Cisco supports three routing protocols for AppleTalk networks:

- **RTMP:** The Routing Table Maintenance Protocol is enabled by default on an AppleTalk network. RTMP is a distance vector routing protocol that uses hop count as its metric. The update period for RTMP is every 10 seconds regardless of whether or not there was a change in the network. This frequent update has the effect of producing a large amount of routing traffic on an AppleTalk network.
- **AURP:** AppleTalk Update−Based Routing Protocol (AURP) is a routing protocol similar to RTMP in that it is a distance vector routing protocol with a maximum hop count of 15 hops. AURP differs from RTMP in that it only sends routing updates when a change has occurred in the network, RTMP sends updates every 10 seconds. AURP is also a tunneling protocol, which allows AppleTalk to be tunneled in TCP/IP, thus allowing two AppleTalk networks to be connected over a TCP/IP network. The TCP/IP connection is called a "tunnel" and is counted as one network hop. The router that connects an AppleTalk network to a tunnel is referred to as an exterior router.
- **EIGRP:** AppleTalk EIGRP is used mainly for WAN links in an AppleTalk network. AppleTalk EIGRP uses the same composite metric that IP and IPX EIGRP use. AppleTalk EIGRP also uses the same DUAL routing algorithm, only sending out routing updates when a change has occurred in the network. AppleTalk EIGRP differs from IP and IPX EIGRP in that the autonomous system number used to start the routing process must be unique for each router. AppleTalk EIGRP features automatic redistribution with the RTMP routing protocol.

AppleTalk Zones

An AppleTalk zone is a grouping of similar resources. It is very similar in concept to a Virtual LAN (VLAN). Each AppleTalk network must be defined to be a member of one or more zones. The AppleTalk Zone Information Protocol (ZIP) maintains a listing of all zone names and associated AppleTalk network numbers for the entire network. Members of a particular zone can be located anywhere in the entire network. Let's look at what will happen when an AppleTalk node such as an Apple Macintosh needs a service such as a printer:

1. The Macintosh chooser will send a request to the local router for a list of all zones.
2. The Macintosh looks in the list of zones for the appropriate service.
3. If the appropriate service is found, the Macintosh will send a request to each of the cable numbers in the selected zone.

4. The local router sends this request as a multicast to the selected zones.
5. The services in the selected zones will reply to the router that sent the multicast.
6. The router that sent the multicast will forward these replies to the originating Macintosh node.
7. The Macintosh node can now select the appropriate service.

Commands Discussed in This Chapter

- **access−list** *access−list−number* **[deny|permit] cable−range | zones | additional−zones | other−access**
- **appletalk access−group** *access−list−number*
- **appletalk cable−range** *cable−range [network.node]*
- **appletalk distribute list** *access−list−number* **out**
- **appletalk eigrp−splithorizon**
- **appletalk local−routing**
- **appletalk protocol [aurp|eigrp|rmp]**
- **appletalk route−redistribution**
- **appletalk routing [eigrp** *route−number]*
- **appletalk static network** *network−number* **to** *network.node* **[floating] zone** *zone−name*
- **appletalk zip−reply−filter** *access−list−number*
- **appletalk zone** *zone−name*
- **ebug apple zip**
- **ping appletalk** *[network.node]*
- **show appletalk access−lists**
- **show appletalk eigrp interfaces** *[type number]*
- **show appletalk eigrp neighbors** *[interface]*
- **show appletalk eigrp traffic**
- **show appletalk globals**
- **show appletalk interface [brief]** *[type number]*
- **show appletalk neighbors** *[neighbor−address]*
- **show appletalk route** *[network | type number]*
- **show appletalk traffic**
- **show appletalk zone** *[zone−name]*
- **tunnel destination**
- **tunnel source** *[ip−address | type number]*

Definitions

access−list: This global configuration command defines the actions that the router should take for various data, route, zone, and other AppleTalk access lists.

appletalk access−group: This interface configuration command assigns an access list to an interface.

appletalk cable−range: This interface configuration command defines an extended AppleTalk network.

appletalk distribute list: This interface configuration command is used to filter routing updates.

appletalk eigrp−splithorizon: This interface configuration command enables split horizon.

appletalk local−routing: This global configuration command is used when configuring AppleTalk to run over an NBMA network.

appletalk protocol: This interface configuration command specifies what routing protocol to use on a particular interface. The default AppleTalk routing protocol is RTMP.

appletalk route−redistribution: This global configuration command causes RTMP routes to be redistributed into EIGRP and EIGRP routes to be redistributed into RTMP.

appletalk routing: This global configuration command is used to enable AppleTalk routing on a router. The command can optionally enable the EIGRP routing protocol on the router.

appletalk static network: This global configuration command defines a static or floating static route.

appletalk zip−reply−filter: This interface command is used with an access list to limit the number of zones that are visible on an AppleTalk network.

appletalk zone: This interface command sets the zone name for an AppleTalk network.

debug apple zip: This debug command enables AppleTalk ZIP debug.

ping appletalk: This exec command is used to verify host reachability.

show appletalk access−lists: This exec command displays all AppleTalk access lists that are defined on the router.

show appletalk eigrp interfaces: This exec command displays information on router interfaces that are configured for the EIGRP routing protocol.

show appletalk eigrp neighbors: This exec command will display information on any EIGRP neighbor routers.

show appletalk globals: This exec command displays information on how AppleTalk is configured to operate on the router.

show appletalk interface: This exec command displays the status of interfaces that are running AppleTalk.

show appletalk neighbors: This exec command displays information on directly connected routers that are running AppleTalk.

show appletalk route: This exec command displays all entries in the AppleTalk routing table.

show appletalk traffic: This exec command shows information about the amount of AppleTalk traffic that is flowing through the router.

show appletalk zone: This exec command displays the contents of the AppleTalk ZIP table.

tunnel destination: This interface configuration command sets the source IP address for an AppleTalk tunnel.

tunnel source: This interface configuration command sets the source IP address for an AppleTalk tunnel.

IOS Requirements

These labs were done using IOS 11.1.

Lab #86: Basic AppleTalk Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers. One of the routers must have two serial interfaces, and the other two routers must have one serial interface and one Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers
- A Cisco IOS image that supports the AppleTalk protocol

Configuration Overview

This lab will demonstrate basic AppleTalk configuration and monitoring. A three−node AppleTalk network will be set up. The RTMP dynamic routing algorithm will be used to learn all routes in the network.

The three routers are connected as shown in Figure 19−6. RouterB acts as a DCE and supplies clocking to both RouterA and RouterC.

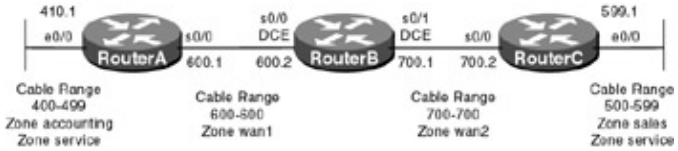


Figure 19−6: Basic AppleTalk connectivity

Note Making changes to AppleTalk routing parameters will sometimes require the router to be reloaded. Make sure to save the configuration before reloading the router.

Router Configuration

The configurations for the three routers in this example are as follows. AppleTalk commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.1
no service udp−small−servers
no service tcp−small−servers
!
hostname RouterA
!
appletalk routing ← Enable AppleTalk routing on this router
!
interface Ethernet0/0
 no ip address
 no keepalive
 appletalk cable−range 400−499 410.1 ← Define a cable range for this interface
                                       and an address of 410.1
 appletalk zone accounting ← Define the primary AppleTalk zone to be accounting
 appletalk zone service ← Define the secondary AppleTalk zone to be service
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable−range 600−600 600.1 ← Define a cable range for this interface
                                       and an address of 600.1
 appletalk zone wan1 ← Define the primary AppleTalk zone to be wan1
!
interface Serial0/1
 no ip address
```

```
 shutdown
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Serial1/0
 no ip address
 shutdown
!
no ip classless
logging buffered
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
appletalk routing ← Enable AppleTalk routing on this router
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable-range 600-600 600.2 ← Define a cable range for this interface
                                       and an address of 600.2
 appletalk zone wan1 ← Define the AppleTalk zone to be wan1
 no fair-queue
 clockrate 64000 ← Provide clocking to neighbor router
!
interface Serial0/1
 no ip address
 encapsulation ppp
 appletalk cable-range 700-700 700.1 ← Define a cable range for this interface
                                       and an address of 700.1
 appletalk zone wan2 ← Define the AppleTalk zone to be wan2
 clockrate 64000 ← Provide clocking to neighbor router
!
no ip classless
logging buffered
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
appletalk routing ← Enable AppleTalk routing on this router
!
interface Ethernet0/0
 no ip address
 no keepalive
 appletalk cable-range 500-599 599.1 ← Define a cable range for this interface
                                       and an address of 599.1
 appletalk zone sales ← Define the primary AppleTalk zone to be sales
 appletalk zone service ← Define the secondary AppleTalk zone to be service
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable-range 700-700 700.2 ← Define a cable range for this interface
                                       and an address of 700.2
 appletalk zone wan2 ← Define the primary AppleTalk zone to be wan2
 no fair-queue
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Serial1/0
 no ip address
 shutdown
!
no ip classless
logging buffered
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA and typing the **show appletalk route** command. This command will display the contents of the AppleTalk routing table. We see that RouterA has two directly connected AppleTalk networks, 400—499 (which is located on E0/0) and 600—600 (which is located on S0/0). Two networks have been learned via the AppleTalk RTMP routing protocol. These are networks 500—599 (which is located on E0/0 of RouterC) and 700—700 (which is the serial link between RouterB and RouterC). We see that the AppleTalk routing table shows the zones that are associated with each network.

```
RouterA#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
4 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 400-499 directly connected, Ethernet0/0, zone accounting
              Additional zones: 'service'
```

```
R Net 500-599 [2/G] via 600.2, 8 sec, Serial0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Serial0/0, zone wan1
R Net 700-700 [1/G] via 600.2, 8 sec, Serial0/0, zone wan2
```

The **show appletalk zone** command will display all zones that are on the network. Notice that the service zone exists on both RouterA and RouterC.

```
RouterA#show appletalk zone
Name                                Network(s)
wan1                                600-600
wan2                                700-700
accounting                          400-499
service                             500-599 400-499 ← This zone exists on two different AppleTa
sales                               500-599
Total of 5 zones
```

Another useful command is **show appletalk globals**. This command provides a summary of the entire AppleTalk network. We see from the output below that there are a total of four routes and five zones in our network. We also see that our RTMP routing protocol will send an update every 10 seconds, mark a route as bad after 20 seconds, and discard a route after 60 seconds.

```
RouterA#show appletalk globals
AppleTalk global information:
  Internet is incompatible with older, AT Phase1, routers.
  There are 4 routes in the internet.
  There are 5 zones defined.
  Logging of significant AppleTalk events is disabled.
  ZIP resends queries every 10 seconds.
  RTMP updates are sent every 10 seconds.
  RTMP entries are considered BAD after 20 seconds.
  RTMP entries are discarded after 60 seconds.
  AARP probe retransmit count: 10, interval: 200 msec.
  AARP request retransmit count: 5, interval: 1000 msec.
  DDP datagrams will be checksummed.
  RTMP datagrams will be strictly checked.
  RTMP routes may not be propagated without zones.
  Routes will not be distributed between routing protocols.
  Routing between local devices on an interface will not be performed.
  IPTalk uses the udp base port of 768 (Default).
  AppleTalk EIGRP is not enabled.
  Alternate node address format will not be displayed.
  Access control of any networks of a zone hides the zone.
```

The **show appletalk traffic** command shows all AppleTalk traffic that has been received or sent from the router. Traffic statistics are broken up into specific AppleTalk protocols such as routing, AppleTalk echo (similar to an IP ping), and the Zone Information Protocol (ZIP).

```
RouterA#show appletalk traffic
AppleTalk statistics:
  Rcvd:  74 total, 0 checksum errors, 0 bad hop count
         74 local destination, 0 access denied
         0 for MacIP, 0 bad MacIP, 0 no client
         7 port disabled, 0 no listener
         0 ignored, 0 martians
  Bcast: 0 received, 143 sent
  Sent:  145 generated, 0 forwarded, 0 fast forwarded, 0 loopback
         0 forwarded from MacIP, 0 MacIP failures
         0 encapsulation failed, 0 no route, 0 no source
  DDP:   74 long, 0 short, 0 macip, 0 bad size
  NBP:   15 received, 0 invalid, 0 proxies
         0 replies sent, 20 forwards, 15 lookups, 0 failures
  RTMP:  60 received, 0 requests, 0 invalid, 0 ignored
         127 sent, 0 replies
  AURP: 0 Open Requests, 0 Router Downs
```

```
        0 Routing Information sent, 0 Routing Information received
        0 Zone Information sent, 0 Zone Information received
        0 Get Zone Nets sent, 0 Get Zone Nets received
        0 Get Domain Zone List sent, 0 Get Domain Zone List received
        0 bad sequence
  ATP:   0 received
  ZIP:   9 received, 8 sent, 0 netinfo
AppleTalk statistics:
  Echo:  0 received, 0 discarded, 0 illegal
         0 generated, 0 replies sent
  Responder:  0 received, 0 illegal, 0 unknown
         0 replies sent, 0 failures
  AARP:  0 requests, 0 replies, 0 probes
         0 martians, 0 bad encapsulation, 0 unknown
         10 sent, 0 failures, 0 delays, 0 drops
  Lost: 0 no buffers
  Unknown: 0 packets
  Discarded: 0 wrong encapsulation, 0 bad SNAP discriminator
```

Notice that the **show interface e 0/0** command does not display any AppleTalk−specific information. It only shows the MAC address of the port and high−level input and output traffic information.

```
RouterA#show interface e 0/0 ← There is no AppleTalk−specific information shown
                                in this command's output
Ethernet0/0 is up, line protocol is up
  Hardware is AmdP2, address is 00e0.1e5b.0d21 (bia 00e0.1e5b.0d21)
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 164/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive not set
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:06, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     77 packets output, 7574 bytes, 0 underruns
     77 output errors, 0 collisions, 3 interface resets
     0 babbles, 0 late collision, 0 deferred
     77 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

To see AppleTalk information for a specific port, you need to use the **show appletalk interface** command. Type **show appletalk interface e 0/0** to display the AppleTalk information for the Ethernet 0/0 port on RouterA. The output of this command gives us important AppleTalk interface information such as the cable range of this interface, the interface address, and zone information.

```
RouterA#show appletalk interface e 0/0
Ethernet0/0 is up, line protocol is up
  AppleTalk cable range is 400−499 ← Network cable range information
  AppleTalk address is 410.1, Valid ← Interface address information
  AppleTalk primary zone is,"accounting" ← Primary zone
  AppleTalk additional zones: "service" ← Secondary zone
  AppleTalk address gleaning is disabled
  AppleTalk route cache is enabled
```

A serial interface running AppleTalk can also have port information displayed with two different commands. The **show interface s 0/0** command shows general interface information. The only indication that this interface is running AppleTalk is the atalkcp LCP that is indicated as open. This occurs as part of the PPP negotiation process and tells us that AppleTalk traffic can be carried across this serial link.

```
RouterA#show interface s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open

AppleTalk control protocol has been negotiated and open
          ↓
  Open: atalkcp, cdp
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations 0/1 (active/max active)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     185 packets input, 7207 bytes, 0 no buffer
     Received 185 broadcasts, 0 runts, 0 giants
     5 input errors, 0 CRC, 5 frame, 0 overrun, 0 ignored, 0 abort
     185 packets output, 6968 bytes, 0 underruns
     0 output errors, 0 collisions, 14 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Specific AppleTalk information can be displayed for the serial interface with the **show appletalk interface s 0/0** command. As with the Ethernet interface, this command will show us AppleTalk information for the serial interface of this router.

```
RouterA#show appletalk interface s 0/0
Serial0/0 is up, line protocol is up
  AppleTalk cable range is 600-600
  AppleTalk address is 600.1, Valid
  AppleTalk zone is "wan1"
  AppleTalk port configuration verified by 600.2
  AppleTalk address gleaning is not supported by hardware
  AppleTalk route cache is enabled
```

The **show appletalk neighbors** command can be used to verify that you are connected to the proper neighbors. The output of this command shows us that we are connected to a neighbor at AppleTalk address 600.2. This is the s0/0 interface of RouterB.

```
RouterA#show appletalk neighbors
AppleTalk neighbors:
  600.2         Serial0/0, uptime 00:08:10, 0 secs
            Neighbor is reachable as a RTMP peer
```

AppleTalk supports a ping command that can be used to test for network reachability. Let's make sure that the s0/0 interface of RouterC is active. Type the **ping appletalk 700.2** command. This will send an AppleTalk echo request to RouterC at AppleTalk address 700.2. The ping should be successful, as shown below:

```
RouterA#ping appletalk 700.2

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 700.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms
```

Make sure that the Ethernet interface of RouterC is also reachable. Use the **ping appletalk 599.1** command to verify that the interface is active.

```
RouterA#ping appletalk 599.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 599.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms
```

The only network connectivity/reachability aid that AppleTalk supports is the ping command. Try to telnet to RouterC at AppleTalk address 599.1. The following output shows what will happen. We see that the telnet was not successful. This is because telnet is a TCP/IP application. It is important to always run the TCP/IP protocol on your network. Network access and SNMP are vital to a successful network and TCP/IP is key to these functions.

```
RouterA#telnet 599.1
% Unknown command or computer name, or unable to find computer address
```

Now let's connect to RouterB and examine its AppleTalk status. Type **show appletalk route** to display the AppleTalk routing table. We see that RouterB has two directly connected AppleTalk networks, 600—600 (serial connection to RouterA) and 700—700 (serial connection to RouterC). Two networks are being learned via the AppleTalk RTMP routing protocol. These are 400—499 (Ethernet port on RouterA) and 500—599 (Ethernet port on RouterC).

```
RouterB#sh appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
4 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400-499 [1/G] via 600.1, 9 sec, Serial0/0, zone accounting
              Additional zones: 'service'
R Net 500-599 [1/G] via 700.2, 7 sec, Serial0/1, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Serial0/0, zone wan1
C Net 700-700 directly connected, Serial0/1, zone wan2
```

The **show appletalk zone** command reveals a zone table that is identical to the zone table of RouterA. Assuming that no zone filters are in effect, the zone table of all routers on a network should be identical.

```
RouterB#show appletalk zone
Name                                Network(s)
wan1                                600-600
wan2                                700-700
accounting                          400-499
service                             400-499 500-599
sales                               500-599
Total of 5 zones
```

Now let's connect to RouterC. Display the AppleTalk routing table with the **show appletalk route** command. We see that RouterC has two directly connected AppleTalk networks. The first directly connected network is 500—599 (Ethernet interface of RouterC) and the second directly connected network is 700—700 (serial connection between RouterC and RouterA).

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
4 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400-499 [2/G] via 700.1, 3 sec, Serial0/0, zone accounting
              Additional zones: 'service'
C Net 500-599 directly connected, Ethernet0/0, zone sales
```

```
              Additional zones: 'service'
R Net 600-600 [1/G] via 700.1, 3 sec, Serial0/0, zone wan1
C Net 700-700 directly connected, Serial0/0, zone wan2
```

As with the zone tables, the zone information on RouterC is identical to the zone information on RouterA.

```
RouterC#show appletalk zone
Name                                Network(s)
wan1                                600-600
wan2                                700-700
accounting                          400-499
service                             400-499 500-599
sales                               500-599
Total of 5 zones
```

Verify that RouterC has network connectivity with RouterA by using the ping command to verify that RouterC is active.

```
RouterC#ping appletalk 410.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 410.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms
```

Lab #87: AppleTalk EIGRP Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers. One of the routers must have two serial interfaces, and the other two routers must have one serial interface and one Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers
- A Cisco IOS image that supports the AppleTalk protocol

Configuration Overview

This lab will demonstrate AppleTalk EIGRP. Like IP EIGRP, AppleTalk EIGRP has many advantages over AppleTalk RTMP, such as less traffic overhead and faster convergence times. As shown in Figure 19-7, each of the three routers will be configured to run AppleTalk EIGRP on the wide area portion of our test network. The Ethernet interfaces on both RouterA and RouterC will still run RTMP. AppleTalk features automatic redistribution between RTMP and EIGRP.

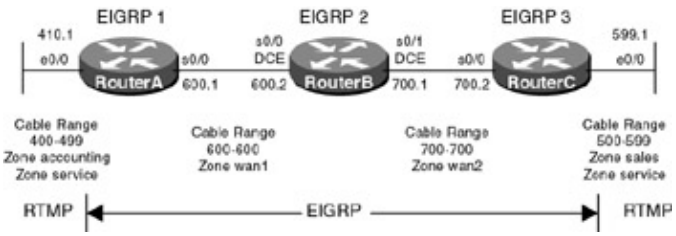


Figure 19-7: AppleTalk EIGRP

The three routers are connected as shown in Figure 19-7. RouterB acts as a DCE and supplies clocking to RouterA and RouterC.

Note Making changes to AppleTalk routing parameters will sometimes require the router to be reloaded. Make sure to save the configuration before reloading the router.

Note Every router that runs AppleTalk EIGRP must have a unique EIGRP process number. This is the opposite of IP EIGRP, where all routers must have the same EIGRP process number.

Router Configuration

The configurations for the three routers in this example are as follows. Key AppleTalk commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
appletalk routing eigrp 1 ← Enable EIGRP Routing. Each AppleTalk router must
                            have a different EIGRP process number
appletalk route-redistribution ← This command is automatically added when
                                 AppleTalk EIGRP is enabled
!
interface Ethernet0/0
 no ip address
 no keepalive
 appletalk cable-range 400-499 410.1 ← Define a cable range for this interface
                                       and an address of 410.1
 appletalk zone accounting ← Define the primary AppleTalk zone to be accounting
 appletalk zone service ← Define the secondary AppleTalk zone to be service
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable-range 600-600 600.1 ← Define a cable range for this interface
                                       and an address of 600.1

 appletalk zone wan1 ← Define the primary AppleTalk zone to be wan1
 appletalk protocol eigrp ← Enable EIGRP on this interface
 no appletalk protocol rtmp ← Disable RTMP on this interface
!
interface Serial0/1
 no ip address
 shutdown
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
```

```
!
enable password cisco
!
appletalk routing eigrp 2 ← Enable EIGRP Routing. Each AppleTalk router must
                            have a different EIGRP process number
appletalk route-redistribution ← This command is automatically added when
                                 AppleTalk EIGRP is enabled
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable-range 600-600 600.2 ← Define a cable range for this interface
                                       and an address of 600.1
 appletalk zone wan1 ← Define the primary AppleTalk zone to be wan1
 appletalk protocol eigrp ← Enable EIGRP on this interface
 no appletalk protocol rtmp ← Disable RTMP on this interface
 no fair-queue
 clockrate 64000 ← Provide clocking to neighbor router
!
interface Serial0/1
 no ip address
 encapsulation ppp
 appletalk cable-range 700-700 700.1 ← Define a cable range for this
interface and an address of 700.1
 appletalk zone wan2 ← Define the primary AppleTalk zone to be wan2
 appletalk protocol eigrp ← Enable EIGRP on this interface
 no appletalk protocol rtmp ← Disable RTMP on this interface
 clockrate 64000 ← Provide clocking to neighbor router
!
no ip classless
logging buffered
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
appletalk routing eigrp 3 ← Enable EIGRP Routing. Each AppleTalk router must
                            have a different EIGRP process number
appletalk route-redistribution ← This command is automatically added when
                                 AppleTalk EIGRP is enabled
!
interface Ethernet0/0
 no ip address
 no keepalive
 appletalk cable-range 500-599 599.1 ← Define a cable range for this interface
                                       and an address of 599.1

 appletalk zone sales ← Define the primary AppleTalk zone to be sales
```

```
 appletalk zone service ← Define the primary AppleTalk zone to be service
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable-range 700-700 700.2 ← Define a cable range for this interface
                                       and an address of 700.2

 appletalk zone wan2 ← Define the primary AppleTalk zone to be wan2
 appletalk protocol eigrp ← Enable EIGRP on this interface
 no appletalk protocol rtmp ← Disable RTMP on this interface
 no fair-queue
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Serial1/0
 no ip address
 shutdown
!
no ip classless
logging buffered
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Type the **show appletalk route** command to display the AppleTalk routing table. We see that we have two directly connected networks and two networks that are being learned via EIGRP. Recall from the previous lab that these EIGRP learned networks were previously RTMP learned networks.

```
RouterA#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
4 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 400-499 directly connected, Ethernet0/0, zone accounting
              Additional zones: 'service'
E Net 500-599 [2/G] via 600.2, 2153 sec, Serial0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Serial0/0, zone wan1
E Net 700-700 [1/G] via 600.2, 2200 sec, Serial0/0, zone wan2
```

Verify that you can ping the Ethernet interface of RouterC with the **ping appletalk 599.1** command. This command should be 100-percent successful, indicating that the entire network is up and active.

```
RouterA#ping appletalk 599.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 599.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/56 ms
```

There are several important EIGRP commands. Type the command **show appletalk eigrp interface** to display interfaces on RouterA that are running EIGRP. Notice that on RouterA only the S0/0 interface is

running EIGRP. The Ethernet interface (E0/0) is still running the AppleTalk RTMP routing protocol.

```
RouterA#show appletalk eigrp interface
AT/EIGRP Neighbors for process 1, router id 1

                    Xmit Queue     Mean   Pacing Time    Multicast     Pending
Interface    Peers  Un/Reliable    SRTT   Un/Reliable    Flow Timer    Routes
Se0/0          1        0/0          21       0/10           98            0
```

The **show appletalk eigrp neighbor** command will display active EIGRP neighbor routers. RouterB at AppleTalk address 600.2 is the only EIGRP neighbor of RouterA.

```
RouterA#show appletalk eigrp neighbor
AT/EIGRP Neighbors for process 1, router id 1
H   Address                 Interface   Hold Uptime    SRTT   RTO     Q    Seq
                                        (sec)          (ms)         Cnt   Num
0   600.2                   Se0/0         14 00:37:29   21    200     0     8
```

The **show appletalk eigrp traffic** command can be used to display EIGRP traffic that passes through a router. We see from this command output that RouterA is actively passing EIGRP hello messages.

```
RouterA#show appletalk eigrp traffic
AT-EIGRP Traffic Statistics
  Hellos sent/received: 499/488
  Updates sent/received: 6/4
  Queries sent/received: 0/2
  Replies sent/received: 2/0
  Acks sent/received: 5/6
  Input queue high water mark 1, 0 drops
```

Another way to verify that EIGRP is running on a particular interface is to use the **show appletalk interface** command. Type the command for the s0/0 interface. Notice from the command output below that the routing protocol for the interface is EIGRP.

```
RouterA#show appletalk interface s 0/0
Serial0/0 is up, line protocol is up
  AppleTalk cable range is 600-600
  AppleTalk address is 600.1, Valid
  AppleTalk zone is "wan1"
  Routing protocols enabled: EIGRP
  AppleTalk port configuration verified by 600.2
  AppleTalk address gleaning is not supported by hardware
  AppleTalk route cache is enabled
```

Now connect to RouterB. The routing table can be displayed with the **show appletalk route** command. RouterB has two directly connected networks and two networks that have been learned via EIGRP. Notice that there are no RTMP learned routes on RouterB.

```
RouterB#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
4 routes in internet

The first zone listed for each entry is its default (primary) zone.

E Net 400-499 [1/G] via 600.1, 2299 sec, Serial0/0, zone accounting
              Additional zones: 'service'
E Net 500-599 [1/G] via 700.2, 2240 sec, Serial0/1, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Serial0/0, zone wan1
C Net 700-700 directly connected, Serial0/1, zone wan2
```

The **show appletalk interface** command can be used to verify that EIGRP is running on both serial interfaces of RouterB.

```
RouterB#show appletalk interface s 0/0
Serial0/0 is up, line protocol is up
  AppleTalk cable range is 600-600
  AppleTalk address is 600.2, Valid
  AppleTalk zone is "wan1"
  Routing protocols enabled: EIGRP
  AppleTalk port configuration verified by 600.1
  AppleTalk address gleaning is not supported by hardware
  AppleTalk route cache is enabled


RouterB#show appletalk interface s 0/1
Serial0/1  is up, line protocol is up
  AppleTalk cable range is 700-700
  AppleTalk address is 700.1, Valid
  AppleTalk zone is "wan2"
  Routing protocols enabled: EIGRP
  AppleTalk port configuration verified by 700.2
  AppleTalk address gleaning is not supported by hardware
  AppleTalk route cache is enabled
```

The EIGRP routing status for the serial interfaces on RouterB can also be verified with the **show appletalk eigrp interface** command.

```
RouterB#show appletalk eigrp interface
AT/EIGRP Neighbors for process 1, router id 2

                    Xmit Queue    Mean    Pacing Time    Multicast     Pending
Interface    Peers  Un/Reliable   SRTT    Un/Reliable    Flow Timer    Routes
Se0/0          1        0/0       285        0/10          1418           0
Se0/1          1        0/0        28        0/10            50           0
```

RouterB's EIGRP neighbors can be displayed with the **show appletalk eigrp neighbor** command. Neighbor 700.2 is the serial interface of RouterC and neighbor 600.1 is the serial interface of RouterA.

```
RouterB#show appletalk eigrp neighbor
AT/EIGRP Neighbors for process 1, router id 2
H   Address                 Interface   Hold Uptime    SRTT   RTO     Q     Seq
                                          (sec)         (ms)        Cnt    Num
1   700.2                   Se0/1         14 00:37:41    28    200    0      2
0   600.1                   Se0/0         12 00:38:39   285   1710    0      8
```

Now connect to RouterC. Display the router's routing table with the **show appletalk route** command. We see that RouterC has learned two networks via the EIGRP routing protocol.

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
4 routes in internet

The first zone listed for each entry is its default (primary) zone.

E Net 400-499 [2/G] via 700.1, 2299 sec, Serial0/0, zone accounting
              Additional zones: 'service'
C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
E Net 600-600 [1/G] via 700.1, 2299 sec, Serial0/0, zone wan1
C Net 700-700 directly connected, Serial0/0, zone wan2
```

Display the interfaces on RouterC that are running EIGRP with the **show appletalk eigrp interface** command. We see that only the serial interface on the router is running EIGRP.

```
RouterC#show appletalk eigrp interface
AT/EIGRP Neighbors for process 1, router id 3

                    Xmit Queue   Mean    Pacing Time     Multicast   Pending
Interface   Peers   Un/Reliable  SRTT    Un/Reliable    Flow Timer    Routes
Se0/0         1        0/0        24        0/10            50          0
```

RouterC's EIGRP neighbors can be displayed with the **show appletalk eigrp neighbor** command. RouterB at AppleTalk address 700.1 is the only EIGRP neighbor to RouterC.

```
RouterC#show appletalk eigrp neighbor
AT/EIGRP Neighbors for process 1, router id 3
H   Address               Interface    Hold Uptime    SRTT   RTO    Q     Seq
                                       (sec)          (ms)          Cnt   Num
0   700.1                 Se0/0        13 00:38:46    24     200    0     7
```

Let's see what happens if two neighbor routers have the same AppleTalk EIGRP process ID. Connect to RouterB and enter router configuration mode. Enter the command **appletalk routing eigrp 3**.

```
RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#appletalk routing eigrp 3
RouterB(config)#exit
RouterB#
```

After this command is entered, both RouterB and RouterC will have the same EIGRP process ID. The following output will be seen. RouterB is telling us that RouterC (AppleTalk address 700.2) has the same router ID as it does.

```
%AT-5-COMPATERR4: AppleTalk EIGRP neighbor incompatibility; 700.2 has same route
r ID (3)
%AT-5-COMPATERR4: AppleTalk EIGRP neighbor incompatibility; 700.2 has same route
r ID (3)
%AT-5-COMPATERR4: AppleTalk EIGRP neighbor incompatibility; 700.2 has same route
r ID (3)
%AT-5-COMPATERR4: AppleTalk EIGRP neighbor incompatibility; 700.2 has same route
r ID (3)
%AT-5-COMPATERR4: AppleTalk EIGRP neighbor incompatibility; 700.2 has same route
r ID (3)

RouterB#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#appletalk routing eigrp 2
RouterB(config)#exit
```

Lab #88: AppleTalk GRE Tunnel

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers. One of the routers must have two serial interfaces, and the other two routers must have one serial interface and one Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers
- A Cisco IOS image that supports the AppleTalk protocol

Configuration Overview

This lab will show how to configure an AppleTalk GRE tunnel. This Cisco feature enables a network to run AppleTalk on the end router nodes. Routers that connect the AppleTalk end nodes run a GRE tunnel to tunnel any AppleTalk traffic in an IP packet.

Figure 19−8 shows the logical representation of this lab. As far as AppleTalk is concerned, RouterA and RouterC are directly connected via network 600. The tunnel makes the intermediate router (RouterB) invisible to the two AppleTalk routers (RouterA and RouterC).

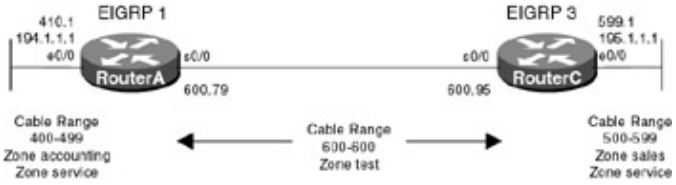


Figure 19−8: AppleTalk GRE/IP tunnel logical representation

The three routers are connected as shown in Figure 19−9. RouterA and RouterC are running both AppleTalk and IP on each of their interfaces. RouterB is only configured for IP. RouterB acts as a DCE and supplies clocking to RouterA and RouterC.

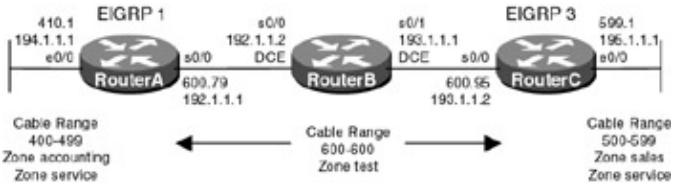


Figure 19−9: AppleTalk GRE/IP tunnel

Note Notice that RouterB is only running TCP/IP. There is no AppleTalk routing being run on RouterB. All AppleTalk traffic that is being generated on RouterA and RouterC is being encapsulated in TCP/IP.

Note Making changes to AppleTalk routing parameters will sometimes require the router to be reloaded. Make sure to save the configuration before reloading the router.

Router Configuration

The configurations for the three routers in this example are as follows. Key AppleTalk commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
appletalk routing eigrp 1 ← Enable EIGRP Routing. Each AppleTalk router must
                            have a different EIGRP process number
appletalk route-redistribution ← This command is automatically added when
                                 AppleTalk EIGRP is enabled
!
interface Tunnel1
 no ip address
 appletalk cable-range 600-600 600.79 ← Define a cable range for this interface
                                        and an address of 600.79

 appletalk zone test ← Define the primary AppleTalk zone to be test
 appletalk protocol eigrp ← Enable EIGRP on this interface
 tunnel source Ethernet0/0 ← Define a tunnel source interface
```

```
 tunnel destination 195.1.1.1 ← Define a tunnel destination address
!
interface Ethernet0/0
 ip address 194.1.1.1 255.255.255.0
 no keepalive
 appletalk cable-range 400-499 410.1 ← Define a cable range for this interface
                                       and an address of 410.1

 appletalk zone accounting ← Define the primary AppleTalk zone to be accounting
 appletalk zone service ← Define the secondary AppleTalk zone to be service
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation ppp
!
router rip
 network 192.1.1.0
 network 194.1.1.0
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation ppp
 no fair-queue
 clockrate 64000 ← Provide clocking to neighbor router
!
interface Serial0/1
 ip address 193.1.1.1 255.255.255.0
 encapsulation ppp
 clockrate 64000 ← Provide clocking to neighbor router
!
router rip
 network 193.1.1.0
 network 192.1.1.0
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.1
```

```
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
appletalk routing eigrp 3 ← Enable EIGRP Routing. Each AppleTalk router must
                            have a different EIGRP process number
appletalk route-redistribution ← This command is automatically added when
                                 AppleTalk EIGRP is enabled
!
interface Tunnel1
 no ip address
 appletalk cable-range 600-600 600.95 ← Define a cable range for this interface
                                        and an address of 600.95
 appletalk zone test ← Define the primary AppleTalk zone to be test
 appletalk protocol eigrp ← Enable EIGRP on this interface
 tunnel source Ethernet0/0 ← Define a tunnel source interface
 tunnel destination 194.1.1.1 ← Define a tunnel destination address
!
interface Ethernet0/0
 ip address 195.1.1.1 255.255.255.0
 no keepalive
 appletalk cable-range 500-599 599.1 ← Define a cable range for this interface
                                       and an address of 599.1
 appletalk zone sales ← Define the primary AppleTalk zone to be sales
 appletalk zone service ← Define the secondary AppleTalk zone to be service
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.0
 encapsulation ppp
 no fair-queue
!
router rip
 network 193.1.1.0
 network 195.1.1.0
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA and displaying the routing table with the **show appletalk route** command. Notice that there are a total of three routes in the AppleTalk network. Recall from the two previous labs that there were four AppleTalk routes before we added the tunnel interface. The tunnel between RouterA and RouterC is a single AppleTalk network. Note that the EIGRP routing protocol is running over the tunnel, and network 500—599 on RouterC is being learned via EIGRP.

```
RouterA#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 400-499 directly connected, Ethernet0/0, zone accounting
              Additional zones: 'service'
E Net 500-599 [1/G] via 600.95, 523 sec, Tunnel1, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

Display th e zones on the network with the **show appletalk zone** command. We see that the tunnel zone is displayed as zone test at network 600—600.

```
RouterA#show appletalk zone
Name                                Network(s)
test                                600-600
accounting                          400-499
service                             500-599 400-499
sales                               500-599
Total of 4 zones
```

Display the active EIGRP interfaces on RouterA with the **show appletalk eigrp interface** command. We see that the only EIGRP interface on RouterA is the tunnel interface.

```
RouterA#show appletalk eigrp interface
AT/EIGRP Neighbors for process 1, router id 1

                     Xmit Queue    Mean    Pacing Time    Multicast     Pending
Interface    Peers   Un/Reliable   SRTT    Un/Reliable    Flow Timer    Routes
Tul              1        0/0        436       58/1100       3212             0
```

We can verify end−to−end network connectivity with the **ping appletalk 599.1** command. This will send a ping command to the Ethernet interface of RouterC. The ping should be 100−percent successful.

```
RouterA#ping appletalk 599.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 599.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/72/72 ms
```

The tunnel interface status can be displayed with the **show interface tunnel1** command. This command will show us how much traffic has passed through the tunnel. It also shows us the tunnel source and destination, as well as the tunnel protocol and transport, which is GRE and IP.

```
RouterA#show interface tunnel1
Tunnel1 is up, line protocol is up
  Hardware is Tunnel
  MTU 1500 bytes, BW 9 Kbit, DLY 500000 usec, rely 255/255, load 1/255
  Encapsulation TUNNEL, loopback not set, keepalive set (10 sec)
  Tunnel source 194.1.1.1 (Ethernet0/0), destination 195.1.1.1
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Checksumming of packets disabled,  fast tunneling enabled
  Last input 00:00:00, output 00:00:04, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/0, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     250 packets input, 14254 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     5298 packets output, 284287 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Since we are running an IP tunnel between the Ethernet interfaces of RouterA and RouterC, we are not running any AppleTalk protocol on the serial interface of RouterA. This can be verified with the **show appletalk interface s 0/0** command. As can be seen below, there is no AppleTalk running on this interface:

```
RouterA#show appletalk interface s 0/0
Serial0/0 is up, line protocol is up
  AppleTalk protocol processing disabled
```

Use the **show appletalk interface e 0/0** command to verify that the Ethernet interface of RouterA is still running AppleTalk.

```
RouterA#show appletalk interface e 0/0
Ethernet0/0 is up, line protocol is up
  AppleTalk cable range is 400-499
  AppleTalk address is 410.1, Valid
  AppleTalk primary zone is "accounting"
  AppleTalk additional zones: "service"
  AppleTalk address gleaning is disabled
  AppleTalk route cache is enabled
```

Now let's connect to RouterB. RouterB is not running any AppleTalk protocols on any of its interfaces. The tunnel that runs from RouterA to RouterC is only using RouterB as a TCP/IP transit node. We can verify that we are not running any AppleTalk protocols on RouterB with the **show appletalk global**, **show appletalk route**, and **show appletalk zone** commands. Each of these command's output should indicate that no AppleTalk is running on RouterB.

```
RouterB#show appletalk global
% Appletalk not running


RouterB#show appletalk route
% Appletalk not running


RouterB#show appletalk zone
% Appletalk not running
```

Display the IP routing table on RouterB with the **show ip route** command. We see that RouterB has two directly connected networks and two networks that have been learned via the RIP routing protocol. IP is enabled and running on all routers in our network. The IP routing table shows us that the entire network is reachable from the Ethernet interface of RouterA (194.1.1.1) to the Ethernet interface of RouterC (195.1.1.1).

```
RouterB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

     192.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.1.1.0/24 is directly connected, Serial0/0
C       192.1.1.1/32 is directly connected, Serial0/0
     193.1.1.0/24  is variably subnetted, 2 subnets, 2 masks
C       193.1.1.0/24 is directly connected, Serial0/1
C       193.1.1.2/32 is directly connected, Serial0/1
R    194.1.1.0/24 [120/1] via 192.1.1.1, 00:00:24, Serial0/0
R    195.1.1.0/24 [120/1] via 193.1.1.2, 00:00:13, Serial0/1
```

Now let's connect to RouterC. Verify that you can reach RouterA with the **ping appletalk 410.1** command.

```
RouterC#ping appletalk 410.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 410.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/83/108 ms
```

Display the AppleTalk routing table with the **show appletalk route** command. As we saw on RouterA, there are only three routes in our AppleTalk network.

```
RouterC#show appletalk route
Codes: R − RTMP derived, E − EIGRP derived, C − connected, A − AURP
       S − static  P − proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

E Net 400−499 [1/G] via 600.79, 652 sec, Tunnel1, zone accounting
              Additional zones: 'service'
C Net 500−599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600−600 directly connected, Tunnel1, zone test
```

Verify that the output of the **show appletalk zone** command matches the **show appletalk zone** output on RouterA. Every router on an AppleTalk network should have an identical list of zones (assuming that there are no zone filters in effect).

```
RouterC#show appletalk zone
Name                                Network(s)
test                                600−600
accounting                          400−499
service                             400−499 500−599
sales                               500−599
Total of 4 zones
```

The **show appletalk interface s 0/0** command reveals that the serial interface on RouterC is not running any AppleTalk routing. The AppleTalk traffic is being tunneled in IP once it leaves the Ethernet interface of RouterC.

```
RouterC#show appletalk interface s 0/0
Serial0/0 is up, line protocol is up
  AppleTalk protocol processing disabled
```

The **show appletalk interface e 0/0** command will verify that the AppleTalk protocol is being run on the Ethernet interface of RouterC.

```
RouterC#show appletalk interface e 0/0
Ethernet0/0 is up, line protocol is up
  AppleTalk cable range is 500−599
  AppleTalk address is 599.1, Valid
  AppleTalk primary zone is "sales"
  AppleTalk additional zones: "service"
  AppleTalk address gleaning is disabled
  AppleTalk route cache is enabled
```

Display the status of the IP tunnel on RouterC with the **show interface tunnel1** command. The tunnel should be in an up/up state. Notice how the command displays the source and destination of the tunnel.

```
RouterC#show interface tunnel1
Tunnel1 is up, line protocol is up
  Hardware is Tunnel
  MTU 1500 bytes, BW 9 Kbit, DLY 500000 usec, rely 255/255, load 1/255
  Encapsulation TUNNEL, loopback not set, keepalive set (10 sec)
  Tunnel source 195.1.1.1 (Ethernet0/0), destination 194.1.1.1
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Checksumming of packets disabled,  fast tunneling enabled
  Last input 00:00:01, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/0, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     250 packets input, 14459 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
```

```
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     5386 packets output, 289355 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Lab #89: AppleTalk Trafficand Zone Filtering

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Three Cisco routers. One of the routers must have two serial interfaces, and the other two routers must have one serial interface and one Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers
- A Cisco IOS image that supports the Appletalk protocol

Configuration Overview

This lab will demonstrate several types of AppleTalk filters:

- **AppleTalk zone filters:** This type of filter will prevent selected zones from appearing in the zone list of a router.
- **AppleTalk data packet filters:** This type of filter will prevent traffic from reaching selected AppleTalk nodes.
- **AppleTalk routing filters:** This type of filter will prevent AppleTalk routing updates from being sent out of a router.

The three routers are connected as shown in Figure 19−10. RouterB acts as a DCE and supplies clocking to RouterA and RouterC.

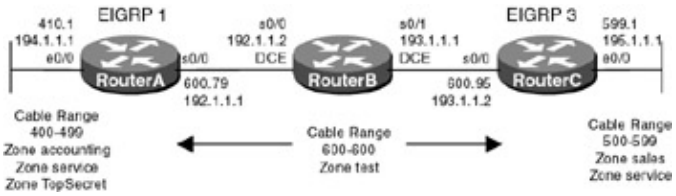


Figure 19−10: AppleTalk filtering

Router Configuration

The configurations for the three routers in this example are as follows. Key AppleTalk commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
appletalk routing eigrp 1 ← Enable EIGRP Routing. Each AppleTalk router must
                            have a different EIGRP process number
```

```
appletalk route-redistribution ← This command is automatically added when
                                 AppleTalk EIGRP is enabled
!
interface Tunnel1
 no ip address
 appletalk cable-range 600-600 600.79 ← Define a cable range for this interface
                                        and an address of 600.79
 appletalk zone test ← Define the primary AppleTalk zone to be test
 appletalk protocol eigrp ← Enable EIGRP on this interface
 tunnel source Ethernet0/0 ← Define a tunnel source interface
 tunnel destination 195.1.1.1 ← Define a tunnel destination address
!
interface Ethernet0/0
 ip address 194.1.1.1 255.255.255.0
 no keepalive
 appletalk cable-range 400-499 410.1 ← Define a cable range for this interface
                                       and an address of 410.1
 appletalk zone accounting ← Define the primary AppleTalk zone to be accounting
 appletalk zone service ← Define the secondary AppleTalk zone to be service
 appletalk zone TopSecret ← Define the secondary AppleTalk zone to be TopSecret
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation ppp
!
router rip
 network 192.1.1.0
 network 194.1.1.0
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation ppp
 no fair-queue
 clockrate 64000 ← Provide clocking to neighbor router
!
interface Serial0/1
 ip address 193.1.1.1 255.255.255.0
 encapsulation ppp
 clockrate 64000 ← Provide clocking to neighbor router
!
router rip
 network 193.1.1.0
 network 192.1.1.0
!
line con 0
line aux 0
line vty 0 4
 password cisco
```

```
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.1
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
!
appletalk routing eigrp 3 ← Enable EIGRP Routing. Each AppleTalk router must
                            have a different EIGRP process number
appletalk route-redistribution ← This command is automatically added when
                                 AppleTalk EIGRP is enabled
!
interface Tunnel1
 no ip address
 appletalk cable-range 600-600 600.95 ← Define a cable range for this interface
                                        and an address of 600.95
 appletalk zone test ← Define the primary AppleTalk zone to be test
 appletalk protocol eigrp ← Enable EIGRP on this interface
 tunnel source Ethernet0/0 ← Define a tunnel source interface
 tunnel destination 194.1.1.1 ← Define a tunnel destination address
!
interface Ethernet0/0
 ip address 195.1.1.1 255.255.255.0
 no keepalive
 appletalk cable-range 500-599 599.1 ← Define a cable range for this interface
                                       and an address of 599.1
 appletalk zone sales ← Define the primary AppleTalk zone to be sales
 appletalk zone service ← Define the secondary AppleTalk zone to be service
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.0
 encapsulation ppp
 no fair-queue
!
router rip
 network 193.1.1.0
 network 195.1.1.0
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA and making sure that our test network is up and active with the **show appletalk route** command. RouterA should have two directly connected networks and one network that is being learned via the EIGRP routing protocol.

```
RouterA#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.
```

```
C Net 400-499 directly connected, Ethernet0/0, zone accounting
              Additional zones: 'TopSecret','service'
E Net 500-599 [1/G] via 600.95, 1958 sec, Tunnel1, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

Verify that RouterA has all our network zones in its zone table with the **show appletalk zone** command.

```
RouterA#show appletalk zone
Name                                Network(s)
TopSecret                           400-499
test                                600-600
accounting                          400-499
service                             500-599 400-499
sales                               500-599
Total of 5 zones
```

Now connect to RouterC and display its route table using the **show appletalk route** command. Make sure that network 400—499 is in the route table.

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

E Net 400-499 [1/G] via 600.79, 2010 sec, Tunnel1, zone accounting
              Additional zones: 'service','TopSecret'
C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

Display the zone list on RouterC with the **show appletalk zone** command. The list of zones on RouterC should match the list of zones on RouterA. We see that these two zone lists do match — both RouterC and RouterA have five zones in their zone table.

```
RouterC#show appletalk zone
Name                                Network(s)
TopSecret                           400-499
test                                600-600
accounting                          400-499
service                             400-499 500-599
sales                               500-599
Total of 5 zones
```

If you have not already saved the configuration on RouterC, do so now with the **write mem** command.

Reconnect to RouterA. We are going to reload RouterC while we monitor the AppleTalk ZIP protocol on RouterA. We will see how RouterC requests a zone list from RouterA when RouterC reboots. Enable AppleTalk ZIP debug with the **debug apple zip** command. Make sure that you are connected to the console port of RouterA. If you are not connected to the console port, issue the **term mon** command to send all debug output to the screen.

```
RouterA#debug apple zip
AppleTalk ZIP Packets debugging is on
```

While still connected to RouterA, power off RouterC and then power it on again. After RouterC finishes booting and loading the IOS, you will see the following debug output on RouterA:

```
1  AT: Recvd ZIP cmd 5 from 600.95-6
2  AT: Answering ZIP GetNetInfo rcvd from 600.95 via Tunnel1
```

```
3  AT: Sent GetNetInfo reply to 600.95 via Tunnel1
4  AT: Recvd ZIP cmd 1 from 600.95-6
5  AT: 1 networks in ZIPquery pkt from 600.95
6  AT: Sent ZIP answer with 3 nets to 600.95
7  AT: NextNbrZipQuery: [500-599] zoneupdate 0 gw: 600.95 n: 600.95
8  AT: NextNbrZipQuery: r->rpath.gwptr: 606BA174, n: 606BA174
9  AT: maint_SendNeighborQueries, sending 1 queries to 600.95
10 AT: 1 query packet sent to neighbor 600.95
11 AT: Recvd ZIP cmd 2 from 600.95-6
12 AT: 2 zones in ZIPreply pkt, src 600.95
13 AT: net 500, zonelen 5, name sales
14 AT: net 500, zonelen 7, name service
15 AT: in CancelZoneRequest, cancelling req on 500-599 . . . succeeded
16 AT: atzip_GC() called.
```

The first line of the debug output shows RouterA receiving a ZIP request from RouterC (600.95 is the tunnel interface AppleTalk address of RouterC). Line 2 shows us that this request is a ZIP GetNetInfo request. The sixth line of the debug output ("AT: Sent ZIP answer with 3 nets to 600.95") shows RouterA sending RouterC zone information on three AppleTalk networks. Line 10 shows RouterA sending a ZIP query packet to RouterC at AppleTalk address 600.95. Line 12 shows RouterC sending a ZIP reply back to RouterA. Notice that RouterC sends RouterA information on two zones. Lines 13 and line 14 show the two zones that RouterC sends to RouterA. These are the sales and service zones.

Reconnect to RouterC and verify that the zone table is correct using the **show appletalk zone** command. We see that three networks (TopSecret, accounting, and service) are defined on the 400—499 network, which is located on RouterA. These are the three zones that RouterA sent to RouterC in the ZIP reply.

```
RouterC#show appletalk zone
Name                                Network(s)
TopSecret                           400-499
test                                600-600
accounting                          400-499
service                             500-599 400-499
sales                               500-599
Total of 5 zones
```

Now reconnect to RouterA. We are going to add a zone access list on RouterA. The access list will deny any zone reply packets to be sent from RouterA that refer to the TopSecret zone.

Enter router configuration mode with the **config term** command. Enter the global access-list commands shown below. Then enter interface configuration mode and enter the **appletalk zip-reply-filter 600** statement under the tunnel1 interface.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#access-list 600 deny zone TopSecret
RouterA(config)#access-list 600 permit additional-zones
RouterA(config)#access-list 600 permit other-access
RouterA(config)#int tunnel1
RouterA(config-if)#appletalk zip-reply-filter 600
RouterA(config-if)#exit
RouterA(config)#exit
RouterA#
```

The access list that was just entered on RouterA can be confirmed with **the show apple access 600** command.

```
RouterA#sh apple access 600
AppleTalk access list 600:
  deny zone TopSecret
  permit additional-zones
  permit other-access
```

RouterA now has an AppleTalk zone reply filter that will restrict any ZIP reply packets from being sent out of the tunnel1 interface that reference the zone TopSecret. Verify that AppleTalk ZIP debug is still enabled with the **show debug** command.

```
RouterA#sh debug
Appletalk:
  AppleTalk ZIP Packets debugging is on
```

Now that the ZIP access list is on RouterA, power off RouterC and power it back on again. While RouterC is reloading, monitor RouterA to see what debug activity takes place. The following debug output should be seen:

```
RouterA#

1  AT: Recvd ZIP cmd 5 from 600.95-6
2  AT: Answering ZIP GetNetInfo rcvd from 600.95 via Tunnel1
3  AT: Sent GetNetInfo reply to 600.95 via Tunnel1
4  AT: Recvd ZIP cmd 1 from 600.95-6
5  AT: 1 networks in ZIPquery pkt from 600.95
6  AT: Sent ZIP answer with 2 nets to 600.95
7  AT: NextNbrZipQuery: [500-599] zoneupdate 0 gw: 600.95 n: 600.95
8  AT: NextNbrZipQuery: r->rpath.gwptr: 606BA174, n: 606BA174
9  AT: maint_SendNeighborQueries, sending 1 queries to 600.95
10 AT: 1 query packet sent to neighbor 600.95
11 AT: Recvd ZIP cmd 2 from 600.95-6
12 AT: 2 zones in ZIPreply pkt, src 600.95
13 AT: net 500, zonelen 5, name sales
14 AT: net 500, zonelen 7, name service
15 AT: in CancelZoneRequest, cancelling req on 500-599 . . . succeeded
16 AT: atzip_GC() called
```

Notice in lines 1 and 2 that RouterC sends a ZIP GetNetInfo query to RouterA. In line 6, RouterA responds with only two networks. Recall from our previous trace that RouterA responded with three networks when we did not have a ZIP reply filter.

Verify that all zones are visible on RouterA with the **show appletalk zone** command.

```
RouterA#show appletalk zone
Name                                Network(s)
TopSecret                           400-499
test                                600-600
accounting                          400-499
service                             500-599 400-499
sales                               500-599
Total of 5 zones
```

Now connect to RouterC and display the zone listings with the **show appletalk zone** command. We see that while RouterA knows about five zones, RouterC only knows about four zones. Compare the zone listing of RouterA with the zone listing of RouterC. Notice that RouterC does not have an entry for the TopSecret zone. This is the effect of the ZIP reply filter that we added on RouterA. RouterA is no longer telling RouterC about the TopSecret zone.

```
RouterC#show appletalk zone
Name                                Network(s)
test                                600-600
accounting                          400-499
service                             500-599 400-499
sales                               500-599
Total of 4 zones
```

From RouterC, try to ping the Ethernet interface of RouterA at AppleTalk address 410.1. The ping should be successful. The 400—499 AppleTalk network is still accessible from RouterC even though we have an active

zone filter and the zone corresponding to network 410 is not in the zone list.

```
RouterC#ping apple 410.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 410.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/72/72 ms
```

Now we will add an AppleTalk data packet filter to RouterA. The data packet filter will deny any traffic that is destined for the network associated with the zone TopSecret. As shown below from the configuration of RouterA, we still have access-list 600 defined.

```
access-list 600 deny zone TopSecret
access-list 600 permit additional-zones
access-list 600 permit other-access
```

Enter router configuration mode with the **config term** command. Enter interface configuration mode and type the **appletalk access-group 600** command. This command will take the existing access-list 600 and apply it to a data packet filter on RouterA. Any traffic that is sent out of the tunnel1 interface of RouterA that has a source AppleTalk address associated with the zone TopSecret will be discarded.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#
RouterA(config)#interface tunnel1
RouterA(config-if)#appletalk access-group 600
RouterA(config-if)#exit
RouterA(config)#exit
```

Now connect to RouterC. Try to ping RouterA at AppleTalk address 410.1. The ping will not be successful. This is due to the data packet filter that we have added to RouterA.

```
RouterC#ping 410.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 410.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Reconnect to RouterA and remove the data packet filter with the **no appletalk access-group 600** command entered in router interface configuration mode.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#int tunnel 1
RouterA(config-if)#no appletalk access-group 600
RouterA(config-if)#exit
RouterA(config)#exit
```

Now connect to RouterC and try to ping RouterA at AppleTalk address 410.1. The ping should now be successful.

```
RouterC#ping 410.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echoes to 410.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/72/72 ms
```

Now we are going to add an outgoing routing table update filter on RouterA. A routing table update filter will control what routes RouterA advertises to RouterC. We will change our routing protocol between RouterA

and RouterC from EIGRP to RTMP. RTMP sends 10−second updates and that makes it easier to demonstrate how this lab should work.

Let's start out by adding the access list and disabling EIGRP on RouterA. We will not activate the access list at this time.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#access−list 601 deny cable−range 400−499
RouterA(config)#access−list 601 deny other−access
RouterC(config)#int tunnel1
RouterC(config−if)#no appletalk protocol eigrp
RouterC(config−if)#exit
RouterA(config)#exit
```

The access list that was just entered on RouterA can be confirmed with **the show apple access 601** command.

```
RouterA#sh apple access 601
AppleTalk access list 601:
  deny cable−range 400−499
  deny other−access
```

Connect to RouterC, enter configuration mode, and remove the EIGRP routing protocol from the tunnel1 interface.

```
RouterC#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterC(config)#int tunnel1
RouterC(config−if)#no appletalk protocol eigrp
RouterC(config−if)#exit
RouterC(config)#exit
```

Reconnect to RouterA and display the routing table with the **show appletalk route** command. Notice that the 500−599 network is now being learned via RTMP instead of EIGRP. RTMP updates are sent every 10 seconds. The screen print below shows us that the last RTMP update from RouterC was received nine seconds ago:

```
RouterA#sh appletalk route
Codes: R − RTMP derived, E − EIGRP derived, C − connected, A − AURP
       S − static  P − proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 400−499 directly connected, Ethernet0/0, zone accounting
              Additional zones: 'TopSecret','service'
R Net 500−599 [1/G] via 600.95, 9 sec, Tunnel1, zone sales
              Additional zones: 'service'
C Net 600−600 directly connected, Tunnel1, zone test
```

Type the **show appletalk route** command several more times. Watch how the RTMP 10−second counter resets itself to 0 seconds every 10 seconds. This proves that RouterA is actively receiving RTMP updates from RouterC.

```
RouterA#sh appletalk route
Codes: R − RTMP derived, E − EIGRP derived, C − connected, A − AURP
       S − static  P − proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 400−499 directly connected, Ethernet0/0, zone accounting
              Additional zones: 'TopSecret','service'
```

```
R Net 500-599 [1/G] via 600.95, 2 sec, Tunnel1, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

Now connect to RouterC. Let's verify that RouterC is learning about RouterA via RTMP and that the RTMP updates are being received every 10 seconds. Type the **show appletalk route** command to view the AppleTalk routing table. The screen print below shows us that an RTMP update was received from RouterC nine seconds ago.

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400-499 [1/G] via 600.79, 9 sec, Tunnel1, zone accounting
              Additional zones: 'service','TopSecret'
C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

Type the **show appletalk route** command several more times. Watch how the RTMP 10-second counter resets itself to 0 seconds every 10 seconds. This shows us that RouterC is actively receiving RTMP updates from RouterA.

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400-499 [1/G] via 600.79, 0 sec, Tunnel1, zone accounting
              Additional zones: 'service','TopSecret'
C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

Now connect to RouterA. We will now activate the routing table update access list. Enter configuration mode with the **config term** command. Under interface configuration mode, enter the command **appletalk distribute-list 601 out** under the tunnel1 interface.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#int tunnel1
RouterA(config-if)#appletalk distribute-list 601 out
RouterA(config-if)#exit
RouterA(config)#exit
```

Now connect to RouterC. Enter the **show appletalk route** command a few times. You will shortly notice that RTMP updates are no longer being sent from RouterA to RouterC. The screen print below shows us that the last RTMP update was received 15 seconds ago from RouterA.

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400-499 [1/G] via 600.79, 15 sec, Tunnel1, zone accounting
              Additional zones: 'service','TopSecret'
```

```
C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

As seen in the screen print below, the last RTMP update was received from RouterA 76 seconds ago:

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400-499 [31/B] via 600.79, 76 sec, zone accounting
              Additional zones: 'service','TopSecret'
C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

The RTMP route to RouterA will eventually be deleted from the routing table. We see below that we no longer have the route in our table. The only entries in the routing table are the two networks that are directly connected to RouterC.

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
2 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

Now let's reconnect to RouterA and delete the routing table update access list. Enter router configuration mode with the **config term** command and in interface configuration mode enter the command **no appletalk distribute-list 601 out.**

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#int tunnel1
RouterA(config-if)#no appletalk distribute-list 601 out
RouterA(config-if)#exit
RouterA(config)#exit
```

Reconnect to RouterC and display the AppleTalk routing table with the **show appletalk route** command. We see that the RouterA is once again visible via an RTMP learned route.

```
RouterC#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400-499 [1/G] via 600.79, 9 sec, Tunnel1, zone accounting
              Additional zones: 'service','TopSecret'
C Net 500-599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Tunnel1, zone test
```

After several seconds, type the **show appletalk route** command again. We see that the counters are once again resetting themselves every 10 seconds.

```
RouterC#show appletalk route
Codes: R − RTMP derived, E − EIGRP derived, C − connected, A − AURP
       S − static  P − proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 400−499 [1/G] via 600.79, 0 sec, Tunnel1, zone accounting
              Additional zones: 'service','TopSecret'
C Net 500−599 directly connected, Ethernet0/0, zone sales
              Additional zones: 'service'
C Net 600−600 directly connected, Tunnel1, zone test
```

Lab #90: AppleTalk Configuration Over a Frame Relay Core

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Four Cisco routers. Two of the routers must have one serial interface, one router must have a serial interface and an Ethernet interface, and the other router must have three serial interfaces.
- Three Cisco crossover cables. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers
- A Cisco IOS image that supports the AppleTalk protocol

Configuration Overview

This lab will demonstrate how to configure AppleTalk to run over a Frame Relay network. Frame Relay is a NBMA (nonbroadcast multiple access) technology. Configuring AppleTalk to run over a frame relay core requires special considerations, such as knowing how to configure split horizons. In this lab we are running AppleTalk EIGRP as our routing protocol. We will need to disable EIGRP split horizon on the hub router (RouterB).

As shown in Figure 19−11, RouterA, RouterB, and RouterC are each connected to a Frame Relay switch. The Frame Relay switch is a fourth router that is only configured for Frame Relay switching.

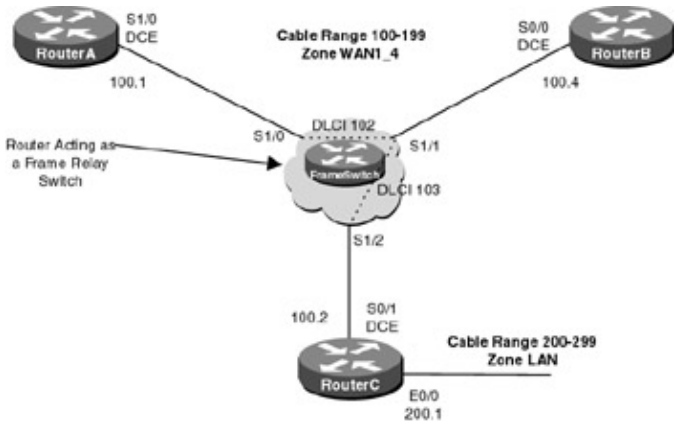


Figure 19−11: AppleTalk over frame relay

Note AppleTalk does not allow cable range addresses to be assigned to a loopback interface.

Note When configuring AppleTalk over frame relay, the appletalk local−routing global command must be used on each spoke router.

Router Configuration

The configurations for the routers in this example are as follows. Key AppleTalk frame relay commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
appletalk routing eigrp 1
appletalk route-redistribution
!
interface Serial1/0
 encapsulation frame-relay
 appletalk cable-range 100-199 100.1
 appletalk zone WAN1_4
 appletalk protocol eigrp
 no appletalk protocol rtmp
 no fair-queue
 clockrate 800000
 frame-relay map appletalk 100.2 102 broadcast ← Provide mappings to RouterC
 frame-relay map appletalk 100.4 102 broadcast ← Provide mappings to RouterB
 no frame-relay inverse-arp
 frame-relay lmi-type ansi
!
ip classless
appletalk local-routing ← This command needs to be used on each spoke router
!
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
appletalk routing eigrp 4
appletalk route-redistribution
```

```
!
interface Serial0/0
 encapsulation frame-relay
 appletalk cable-range 100-199 100.4
 appletalk zone WAN1_4
 appletalk protocol eigrp
 no appletalk protocol rtmp
 no appletalk eigrp-splithorizon ← The hub router must have EIGRP split horizon
                                   disabled
 clockrate 800000
 frame-relay map appletalk 100.1 102 broadcast
 frame-relay map appletalk 100.2 103 broadcast
 no frame-relay inverse-arp
 frame-relay lmi-type ansi
!
ip classless
appletalk local-routing
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
enable password cisco
!
appletalk routing eigrp 2
appletalk route-redistribution
!
interface Ethernet0/0
 no ip address
 no keepalive
 appletalk cable-range 200-299 200.1
 appletalk zone LAN
!
interface Serial0/1
 encapsulation frame-relay
 appletalk cable-range 100-199 100.2
 appletalk zone WAN1_4
 appletalk protocol eigrp
 no appletalk protocol rtmp
 clockrate 800000
 frame-relay map appletalk 100.1 103 broadcast
 frame-relay map appletalk 100.4 103 broadcast
 no frame-relay inverse-arp
 frame-relay lmi-type ansi
!
no ip classless
appletalk local-routing
!
!
line con 0
line aux 0
line vty 0 4
```

```
 password cisco
 login
!
end
```

FrameSwitch

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname FrameSwitch
!
!
frame-relay switching
!
interface Serial1/0
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 102 interface Serial1/1  102
!
interface Serial1/1
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 102 interface Serial1/0  102
 frame-relay route 103 interface Serial1/2  103
!
interface Serial1/2
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 103 interface Serial1/1  103
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Use the **show apple route** command to verify that we are learning about cable range 200—299 over the frame relay core.

```
RouterA#show apple route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
2 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 100-199 directly connected, Serial1/0, zone WAN1_4
E Net 200-299 [2/G] via 100.4, 172 sec, Serial1/0, zone LAN
```

The **show apple zone** command will verify that we are learning about the LAN zone over the frame relay core.

```
RouterA#show apple zone
Name                                Network(s)
WAN1_4                              100-199
LAN                                 200-299
Total of 2 zones
```

Use the **ping apple** command to verify that we can reach the E0/0 interface on RouterC.

```
RouterC#ping apple 200.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echos to 200.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/20/24 ms
```

Now let's connect to RouterB. Go into configuration mode and enable split horizon on interface S0/0.

```
RouterB(config)#interface Serial0/0
RouterB(config-if)#appletalk eigrp-splithorizon
```

Now connect to RouterA. Display the AppleTalk routing table with the **show appletalk route** command. Notice that RouterA is no longer learning a route to the 200-299 network due to split horizon being enabled.

```
RouterA#show apple route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
1 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 100-199 directly connected, Serial1/0, zone WAN1_4
```

Lab #91: AppleTalk Dial Backup with Floating Static Routes

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers. Each router must have one serial interface and one BRI interface. One of the two routers will also need to have an Ethernet interface.
- One Cisco crossover cable. If a Cisco crossover cable is not available, you can use a Cisco DTE cable connected to a Cisco DCE cable.
- Two ISDN BRI cables
- A Cisco rolled cable for console port connection to the routers
- A Cisco IOS image that supports the AppleTalk protocol
- Two ISDN BRI circuits

Configuration Overview

This lab will demonstrate how to configure a router for AppleTalk dial backup using an AppleTalk floating static route. AppleTalk floating static routes have to be configured for each individual network that is being backed up.

The two routers are connected as shown in Figure 19−12. RouterA acts as a DCE and supplies clocking to RouterB.

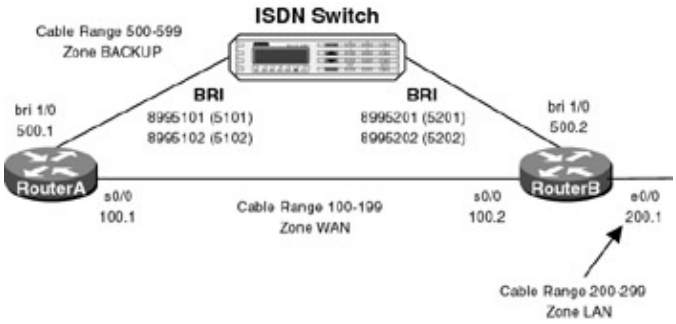


Figure 19−12: AppleTalk dial backup

ISDN Switch Setup

If you do not have access to actual ISDN circuits, you can use an ISDN desktop switch. For this lab we used an Adtran Atlas 800. Information on configuring the Adtran Atlas 800 switch can be found in Chapter 3.

Router Configuration

The configurations for the routers in this example are as follows. Key AppleTalk dial backup commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
!
username RouterB password 0 cisco
!
!
ip subnet-zero
!
appletalk routing eigrp 1
appletalk route-redistribution
lane client flush
isdn switch-type basic-ni
cns event-service server
!
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable-range 100-199 100.1
 appletalk zone WAN
 appletalk protocol eigrp
 no fair-queue
 clockrate 800000
!
interface BRI1/0
 no ip address
 encapsulation ppp
 dialer map appletalk 500.2 name RouterB broadcast 8995201 ← Configure a dialer
                                                              map to the far-end
                                                              interface
```

```
 dialer load-threshold 255 either ← Set the load threshold high so that only
                                    one B channel will be used
 dialer-group 1 ← Corresponds to dialer list 1
 appletalk cable-range 500-599 500.1
 appletalk zone BACKUP
 isdn switch-type basic-ni
 isdn spid1 5101 8995101
 isdn spid2 5102 8995102
 ppp authentication chap
!
ip classless
no ip http server
!
access-list 600 deny other-nbps ← Access list 600 defines our interesting
                                  traffic
access-list 600 permit other-access broadcast-deny
dialer-list 1 protocol appletalk list 600
appletalk static cable-range 200-299 to 500.2 floating zone LAN ← Define our
                                                                  AppleTalk
                                                                  floating
                                                                  static route
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
!
username RouterA password 0 cisco
!
!
ip subnet-zero
!
appletalk routing eigrp 2
appletalk route-redistribution
lane client flush
isdn switch-type basic-ni
cns event-service server
!
!
interface Ethernet0/0
 no ip address
 no keepalive
 appletalk cable-range 200-299 200.1
 appletalk zone LAN
!
interface Serial0/0
 no ip address
 encapsulation ppp
 appletalk cable-range 100-199 100.2
 appletalk zone WAN
!
```

```
interface BRI1/0
 no ip address
 encapsulation ppp
 dialer map appletalk 500.1 name RouterA broadcast
 dialer load-threshold 1 either
 dialer-group 1
 appletalk cable-range 500-599 500.2
 appletalk zone BACKUP
 isdn switch-type basic-ni
 isdn spid1 5201 8995201
 isdn spid2 5202 8995202
 ppp authentication chap
!
interface TokenRing1/0
 no ip address
 shutdown
 ring-speed 16
!
ip classless
no ip http server
!
access-list 600 permit other-access broadcast-deny
dialer-list 1 protocol appletalk list 600
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Make sure that the ISDN circuit is functioning properly with the **show isdn status** command. We see that both SPIDs have been successfully sent to the switch.

```
RouterA#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI1/0 interface
     dsl 8, interface ISDN Switchtype = basic-ni
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 8(established)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 8(established)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 0
    The Free Channel Mask:  0x80000003
    Total Allocated ISDN CCBs = 0
```

Connect to RouterB and verify that its ISDN circuit is functioning properly. We see below that RouterB is ready to accept a call:

```
RouterB#show isdn status
Global ISDN Switchtype = basic-ni
ISDN BRI1/0  interface
```

```
     dsl 8, interface ISDN Switchtype = basic-ni
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, Ces = 2, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 0
    The Free Channel Mask:  0x80000003
    Total Allocated ISDN CCBs = 0
```

Now reconnect to RouterA. Display the routing table on RouterA with the **show appletalk route** command. We see that RouterA has learned a route to the 200—299 network via its serial interface.

```
RouterA#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 100-199 directly connected, Serial0/0, zone WAN
R Net 200-299 [1/G] via 100.2, 9 sec, Serial0/0, zone LAN
C Net 500-599 directly connected, BRI1/0, zone BACKUP
```

The **show apple static** command indicates that RouterA has a floating static route that points to network 200—299.

```
RouterA#show apple static

               AppleTalk     Static      Entries:
               ----------------------------------

   Network      NextIR        Zone                            Status

 200  - 299     500.2         LAN                               A(Floating)
```

Verify that RouterA can reach the e0/0 interface of RouterB with the **ping apple 200.1** command.

```
RouterA#ping apple 200.1

Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echos to 200.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

Now we will start a ping from RouterA to the LAN (e0/0) interface of RouterB.

```
RouterA#ping
Protocol [ip]: apple
Target AppleTalk address: 200.1
Repeat count [5]: 20000
Datagram size [100]: 1500
% A decimal number between 23 and 599.
Datagram size [100]:
Timeout in seconds [2]:
Verbose [n]:
Sweep range of sizes [n]:
```

```
Type escape sequence to abort.
Sending 20000, 100-byte AppleTalk Echos to 200.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!.
            á
```

While the ping is running, pull the serial cable between RouterA and RouterB

After the serial cable is disconnected, the S0/0 interface will go down
↓

```
00:58:57: %LINK-3-UPDOWN: Interface Serial0/0, changed state to down
00:58:58: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to down
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEERRRRRRRRRRRRRRRRRRRRRRRR
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
```

The ISDN interface will be activated. An ISDN call will be placed from RouterA to RouterB
↓

```
01:01:03: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
01:01:03: BR1/0:1 PPP: Treating connection as a callout
01:01:03: BR1/0:1 CHAP: O CHALLENGE id 4 len 28 from "RouterA"
01:01:03: BR1/0:1 CHAP: I CHALLENGE id 4 len 28 from "RouterB"
01:01:03: BR1/0:1 CHAP: O RESPONSE id 4 len 28 from "RouterA"
01:01:03: BR1/0:1 CHAP: I SUCCESS id 4 len 4
01:01:03: BR1/0:1 CHAP: I RESPONSE id 4 len 28 from "RouterB"
01:01:03: BR1/0:1 CHAP: O SUCCESS id 4 len 4E
```

After the ISDN circuit has been activated, display the routing table with the **show apple route** command. Notice that the route to the 200—299 network is now being learned via the BRI interface.

```
RouterA#show apple route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
2 routes in internet

The first zone listed for each entry is its default (primary) zone.

R Net 200-299 [1/G] via 500.2, 9 sec, BRI1/0, zone LAN
C Net 500-599 directly connected, BRI1/0, zone BACKUP
```

The **show dialer** command verifies that the reason that RouterA dialed was because of traffic destined for AppleTalk address 200.1.

```
RouterA#show dialer

BRI1/0 - dialer type = ISDN

Dial String      Successes   Failures    Last DNIS   Last status
8995201                  4          0    00:01:46       successful
0 incoming call(s) have been screened.
0 incoming call(s) rejected for callback.

BRI1/0:1 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is data link layer up
Dial reason: appletalk (s=500.1, d=200.1)
Time until disconnect 22 secs
Connected to 8995201 (RouterB)

BRI1/0:2 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
```

```
Dialer state is idle
```

Reconnect the serial cable between RouterA and RouterB. As shown below, the s0/0 interface will become active a short time after connecting the cable:

```
01:03:18: %LINK-3-UPDOWN: Interface Serial0/0, changed state to up
01:03:18: Se0/0  PPP: Treating connection as a dedicated line
01:03:19: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
01:04:35: %AT-6-CONFIGOK: Serial0/0: AppleTalk interface enabled; verified by 100.2
```

Once the idle timer expires, the ISDN call will be disconnected.

```
01:04:54: %ISDN-6-DISCONNECT: Interface BRI1/0:1
disconnected from 8995201 RouterB, call lasted 231 seconds
01:04:54: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to down
01:04:55: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to down
```

After the ISDN call has been disconnected, display the routing table with the **show apple route** command. We see that RouterA is now learning about network 200—299 over the serial link connecting RouterA to RouterB.

```
RouterA#show apple route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
3 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 100-199 directly connected, Serial0/0, zone WAN
R Net 200-299 [1/G] via 100.2, 4 sec, Serial0/0, zone LAN
C Net 500-599 directly connected, BRI1/0, zone BACKUP
```

AppleTalk Monitoring and Troubleshooting Commands

This section will discuss key AppleTalk monitoring and troubleshooting commands.

{show appletalk route} The **show appletalk route** command will display the contents of the AppleTalk routing table. In the example below we see that RouterA has two directly connected AppleTalk networks 400—499 (which is located on E0/0) and 600−600 (which is located on S0/0). Two networks have been learned via the AppleTalk RTMP routing protocol. These are networks 500—599 and 700—700. We see that the AppleTalk routing table shows the zones that are associated with each network.

```
RouterA#show appletalk route
Codes: R - RTMP derived, E - EIGRP derived, C - connected, A - AURP
       S - static  P - proxy
4 routes in internet

The first zone listed for each entry is its default (primary) zone.

C Net 400-499 directly connected, Ethernet0/0, zone accounting
              Additional zones: 'service'
R Net 500-599 [2/G] via 600.2, 8 sec, Serial0/0, zone sales
              Additional zones: 'service'
C Net 600-600 directly connected, Serial0/0, zone wan1
R Net 700-700 [1/G] via 600.2, 8 sec, Serial0/0, zone wan2
```

{show appletalk zone} The **show appletalk zone** command will display all zones that are on the network. Notice that the service zone exists on two different AppleTalk networks.

```
RouterA#show appletalk zone
Name                            Network(s)
wan1                            600-600
```

```
wan2                                700-700
accounting                          400-499
service                             500-599 400-499 ← This zone exists on two
                                                      different AppleTalk
                                                      networks
sales                               500-599
Total of 5 zones
```

{show appletalk globals} Another useful command is **show appletalk globals**. This command provides a summary of the entire AppleTalk network. We see that there are a total of four routes and five zones in this example network. We also see that the RTMP routing protocol will send an update every 10 seconds, mark a route as bad after 20 seconds, and discard a route after 60 seconds.

```
RouterA#show appletalk globals
AppleTalk global information:
  Internet is incompatible with older, AT Phase1, routers.
  There are 4 routes in the internet.
  There are 5 zones defined.
  Logging of significant AppleTalk events is disabled.
  ZIP resends queries every 10 seconds.
  RTMP updates are sent every 10 seconds.
  RTMP entries are considered BAD after 20 seconds.
  RTMP entries are discarded after 60 seconds.
  AARP probe retransmit count: 10, interval: 200 msec.
  AARP request retransmit count: 5, interval: 1000 msec.
  DDP datagrams will be checksummed.
  RTMP datagrams will be strictly checked.
  RTMP routes may not be propagated without zones.
  Routes will not be distributed between routing protocols.
  Routing between local devices on an interface will not be performed.
  IPTalk uses the udp base port of 768 (Default).
  AppleTalk EIGRP is not enabled.
  Alternate node address format will not be displayed.
  Access control of any networks of a zone hides the zone.
```

{show apple access} The **show appletalk access** command shows the AppleTalk access lists that have been defined on the router. If the command includes a specific access-list number, only that numbered access list will be displayed.

```
RouterA#sh appletalk access
AppleTalk access list 600:
  deny zone TopSecret
  permit additional-zones
  permit other-access
AppleTalk access list 601:
  deny cable-range 400-499
  deny other-access

RouterA#sh apple access 600
AppleTalk access list 600:
  deny zone TopSecret
  permit additional-zones
  permit other-access
```

{show appletalk traffic} The **show appletalk traffic** command shows all AppleTalk traffic that has been received or sent from the router. Traffic statistics are broken up into specific AppleTalk protocols such as routing, echo, and Zone Information Protocol (ZIP).

```
RouterA#show appletalk traffic
AppleTalk statistics:
  Rcvd:  74 total, 0 checksum errors, 0 bad hop count
         74 local destination, 0 access denied
         0 for MacIP, 0 bad MacIP, 0 no client
         7 port disabled, 0 no listener
```

```
        0 ignored, 0 martians
  Bcast: 0 received, 143 sent
  Sent:  145 generated, 0 forwarded, 0 fast forwarded, 0 loopback
         0 forwarded from MacIP, 0 MacIP failures
         0 encapsulation failed, 0 no route, 0 no source
  DDP:   74 long, 0 short, 0 macip, 0 bad size
  NBP:   15 received, 0 invalid, 0 proxies
         0 replies sent, 20 forwards, 15 lookups, 0 failures
  RTMP:  60 received, 0 requests, 0 invalid, 0 ignored
         127 sent, 0 replies
  AURP: 0 Open Requests, 0 Router Downs
        0 Routing Information sent, 0 Routing Information received
        0 Zone Information sent, 0 Zone Information received
        0 Get Zone Nets sent, 0 Get Zone Nets received
        0 Get Domain Zone List sent, 0 Get Domain Zone List received
        0 bad sequence
  ATP:   0 received
  ZIP:   9 received, 8 sent, 0 netinfo
AppleTalk statistics:
  Echo:  0 received, 0 discarded, 0 illegal
         0 generated, 0 replies sent
  Responder:  0 received, 0 illegal, 0 unknown
         0 replies sent, 0 failures
  AARP:  0 requests, 0 replies, 0 probes
         0 martians, 0 bad encapsulation, 0 unknown
         10 sent, 0 failures, 0 delays, 0 drops
  Lost: 0 no buffers
  Unknown: 0 packets
  Discarded: 0 wrong encapsulation, 0 bad SNAP discriminator
```

{show interface} The standard **show interface** commands do not display any AppleTalk−specific information. In the example below, the **show interface e 0/0** command only shows the MAC address of the port and high−level input and output traffic information.

```
RouterA#show interface e 0/0 ← There is no AppleTalk−specific information shown
                                in this commands output
Ethernet0/0 is up, line protocol is up
  Hardware is AmdP2, address is 00e0.1e5b.0d21 (bia 00e0.1e5b.0d21)
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 164/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive not set
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:06, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     77 packets output, 7574 bytes, 0 underruns
     77 output errors, 0 collisions, 3 interface resets
     0 babbles, 0 late collision, 0 deferred
     77 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

{show appletalk interface} AppleTalk information for a specific port can be displayed with the **show appletalk interface** command. The output of this command gives us important AppleTalk interface information such as the cable range of this interface, the interface address, and zone information.

```
RouterA#show appletalk interface e 0/0
Ethernet0/0 is up, line protocol is up
  AppleTalk cable range is 400−499 ← Network cable range information
  AppleTalk address is 410.1, Valid ← Interface address information
  AppleTalk primary zone is, "accounting" ← Primary zone
```

```
  AppleTalk additional zones: "service" ← Secondary zone
  AppleTalk address gleaning is disabled
  AppleTalk route cache is enabled
```

A serial interface running AppleTalk can also have port information displayed with two different commands. The **show interface** command shows general interface information. The only indication that this interface is running AppleTalk is the **atalkcp LCP** that is indicated as open. This occurs as part of the PPP negotiation process and tells us that AppleTalk traffic can be carried across this serial link.

```
RouterA#show interface s 0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open

AppleTalk control protocol has been negotiated and open
          ↓
  Open: atalkcp, cdp
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/1 (active/max active)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     185 packets input, 7207 bytes, 0 no buffer
     Received 185 broadcasts, 0 runts, 0 giants
     5 input errors, 0 CRC, 5 frame, 0 overrun, 0 ignored, 0 abort
     185 packets output, 6968 bytes, 0 underruns
     0 output errors, 0 collisions, 14 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Specific AppleTalk information can be displayed for the serial interface with the **show appletalk interface** command. As with the Ethernet interface, this command will show us AppleTalk information for the serial interface of this router.

```
RouterA#show appletalk interface s 0/0
Serial0/0 is up, line protocol is up
  AppleTalk cable range is 600-600
  AppleTalk address is 600.1, Valid
  AppleTalk zone is, "wan1"
  AppleTalk port configuration verified by 600.2
  AppleTalk address gleaning is not supported by hardware
  AppleTalk route cache is enabled
```

{show appletalk neighbors} The **show appletalk neighbors** command can be used to verify that you are connected to the proper neighbors. The output of this command shows us that we are connected to a neighbor at AppleTalk address 600.2.

```
RouterA#show appletalk neighbors
AppleTalk neighbors:
  600.2         Serial0/0, uptime 00:08:10, 0 secs
            Neighbor is reachable as a RTMP peer
```

AppleTalk supports a ping command that can be used to test for proper network connectivity.

{show appletalk eigrp interface} When running AppleTalk EIGRP, there are several important commands used to monitor the network. The **show appletalk eigrp interface** command is used to display interfaces on

routers that are running EIGRP.

```
RouterA#show appletalk eigrp interface
AT/EIGRP Neighbors for process 1, router id 1

                   Xmit Queue    Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable   SRTT   Un/Reliable   Flow Timer   Routes
Se0/0          1        0/0         21       0/10          98           0
```

{show appletalk eigrp neighbor} The **show appletalk eigrp neighbor** command will display active EIGRP neighbor routers. In the example below, RouterA has an EIGRP neighbor at AppleTalk address 600.2:

```
RouterA#show appletalk eigrp neighbor
AT/EIGRP Neighbors for process 1, router id 1
H   Address                 Interface   Hold Uptime    SRTT   RTO     Q    Seq
                                          (sec)         (ms)         Cnt   Num
0   600.2                   Se0/0           14 00:37:29   21    200   0    8
```

{show appletalk eigrp traffic} The **show appletalk eigrp traffic** command can be used to display EIGRP traffic that passes through a router. We see from this command output that RouterA is actively passing EIGRP hello messages.

```
RouterA#show appletalk eigrp traffic
AT-EIGRP Traffic Statistics
  Hellos sent/received: 499/488
  Updates sent/received: 6/4
  Queries sent/received: 0/2
  Replies sent/received: 2/0
  Acks sent/received: 5/6
  Input queue high water mark 1, 0 drops
```

Another way to verify that EIGRP is running on a particular interface is to use the **show appletalk interface** command. Notice from the command output below that the routing protocol for the interface is EIGRP:

```
RouterA#show appletalk interface s 0/0
Serial0/0 is up, line protocol is up
  AppleTalk cable range is 600-600
  AppleTalk address is 600.1, Valid
  AppleTalk zone is "wan1"
  Routing protocols enabled: EIGRP
  AppleTalk port configuration verified by 600.2
  AppleTalk address gleaning is not supported by hardware
  AppleTalk route cache is enabled
```

{show interface tunnel#} An AppleTalk GRE/IP tunnel interface can have its status displayed with the **show interface tunnel#** command. This command will show us how much traffic has passed through the tunnel. It also shows us the tunnel source and destination as well as the tunnel protocol and transport, which is GRE and IP.

```
RouterA#show interface tunnel1
Tunnel1 is up, line protocol is up
  Hardware is Tunnel
  MTU 1500 bytes, BW 9 Kbit, DLY 500000 usec, rely 255/255, load 1/255
  Encapsulation TUNNEL, loopback not set, keepalive set (10 sec)
  Tunnel source 194.1.1.1 (Ethernet0/0), destination 195.1.1.1
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Checksumming of packets disabled,  fast tunneling enabled
  Last input 00:00:00, output 00:00:04, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/0, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
```

```
250 packets input, 14254 bytes, 0 no buffer
Received 0 broadcasts, 0 runts, 0 giants
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
5298 packets output, 284287 bytes, 0 underruns
0 output errors, 0 collisions, 0 interface resets
0 output buffer failures, 0 output buffers swapped out
```

Conclusion

This chapter explored the AppleTalk networking protocol. AppleTalk requires the least amount of configuration on end systems but has a drawback in that it is the chattiest of all desktop protocols. We explored several ways to reduce the amount of overhead on an AppleTalk network such as running EIGRP on the WAN or using a GRE tunnel. We also demonstrated the Cisco IOS support for AppleTalk routing access lists, data access lists, and ZIP access lists.

Chapter 20: Catalyst 5000 Switches

Overview

Topics Covered in This Chapter

- LAN switching overview
- Catalyst product line
- VLAN configuration on the Catalyst switch
- Configuring Catalyst switches for port security
- Configuring an ISL truck on a Catalyst switch
- Routing between VLANs
- Troubleshooting Catalyst switches

Introduction

This chapter will discuss Cisco Systems LAN switching products — specifically, the Catalyst 5000 family of switches. The Catalyst is much more than just a high−speed switching hub, supporting multiple VLANs, advanced router functionality, and ATM LAN emulation.

Catalyst 5000 Series Overview

The Catalyst 5000 series of switches consists of four different switch models. The different models can be summarized as follows:

Switch	Slots	Overview
Catalyst 5002	2	• Supports a single supervisor module • Can accommodate a single switching module
Catalyst 5000	5	• Supports a single supervisor module • Can accommodate up to four switching modules • Supports the route switch module
Catalyst 5505	5	• Supports a single supervisor module • Can accommodate a redundant supervisor module • Can accommodate up to four additional switching modules • Supports the route switch module
Catalyst 5500	13	• Supports a single supervisor module • Can accommodate a redundant supervisor module • Can accommodate up to 12 additional switching modules • Supports Lightstream ATM modules • Supports the route switch module

Catalyst 5500 Product Overview

The Catalyst 5500 is a 13−slot high−performance switch. Highlights of the 5500 include

- A 3.6−Gbps switch fabric
- Up to 528 switched 10−Mbps Ethernet ports

- Up to 264 switched 100−Mbps Ethernet ports
- Up to 132 switched 100−Mbps fiber Ethernet ports
- Up to 8 ATM OC−12 ports
- Up to 32 ATM OC−3 ports
- Up to 32 DS3 ATM interfaces
- Up to 96 25−Mbps ATM ports
- Up to 7 route switch modules
- Up to 7 ATM LANE modules
- Up to 11 FDDI modules
- Capability for dual redundant supervisor engines
- Hot−swappable modules
- Hot−swappable power supplies
- Hot−swappable fan assemblies

Catalyst Components

Figure 20−1 shows the types of cards that can populate a Catalyst 5000 series switch.

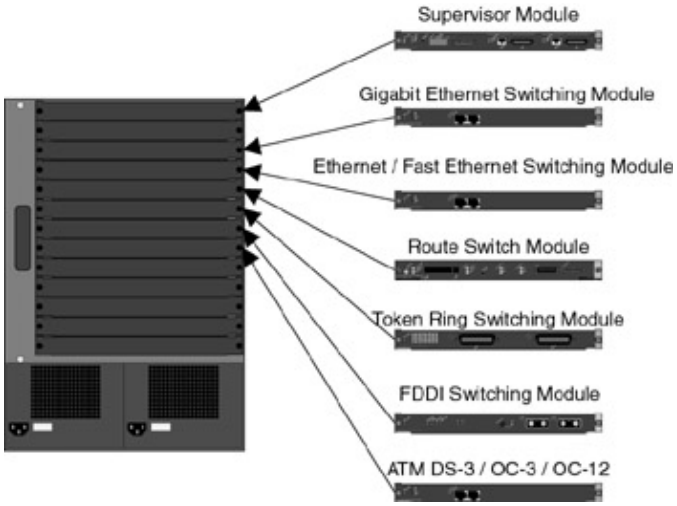


Figure 20−1: Catalyst switch components

- **Supervisor Engine** The Supervisor Engine is the main processor for the Catalyst switch. The Catalyst 5500 can accommodate up to two Supervisor Engines. If one Supervisor Engine fails, the other will take over for the failed unit. The Supervisor II only supports 1.2 Gbps of backplane bandwidth. The Supervisor III supports 3.6 Gbps of backplane bandwidth as well as fast EtherChannel links up to 400 Mbps.
- **Route switch module** This module provides routing functionality to the Catalyst switch. The RSM runs the traditional Cisco router IOS and is comparable in performance to a Cisco 7500 router. The RSM does not have any physical interfaces. It uses the concept of logical interfaces to route traffic between different VLANs.
- **Ethernet/token ring/FDDI switching modules** The Catalyst supports a variety of LAN switching modules. In addition, the Catalyst supports Fast EtherChannel links at speeds up to 800−Mbps full duplex using multiple 100−Mbps Ethernet links grouped into a single logical link.

VLANs

In order to fully understand the concept of a VLAN, we must first review the various ways of connecting together hosts on a LAN.

Figure 20−2 shows the traditional way of connecting six workstations to a nonswitched Ethernet network. Each of the six workstations connects to a basic Ethernet hub. The hub effectively connects all six workstations together onto the same Ethernet cable. The entire hub constitutes a single collision domain (only one workstation can transmit at a time) and a single broadcast domain (all workstations will receive all traffic

that is sent by any other workstation). All six workstations reside in the same collision and broadcast domain.

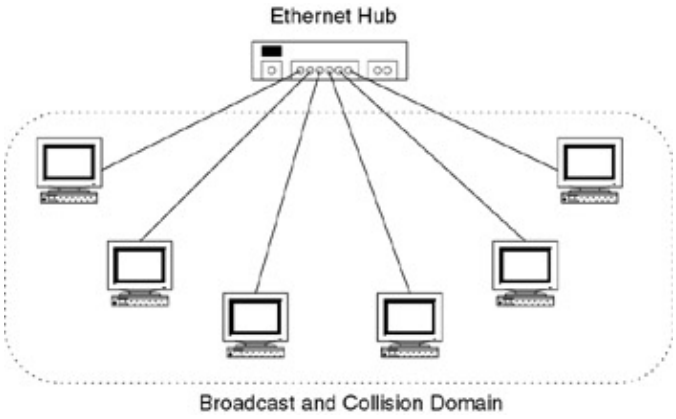


Figure 20−2: Basic Ethernet hub

Figure 20−3 shows a bridge device. Three workstations reside on two different LANs. The two LANs are connected together by the bridge device. Each LAN connected to the bridge is a separate collision domain, but all six workstations reside on a single broadcast domain.

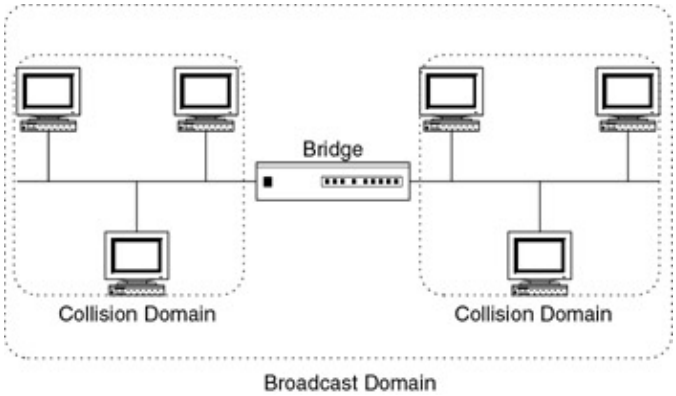


Figure 20−3: Bridge example

Figure 20−4 shows a router device. Three workstations reside on two different LANs. The two LANs are connected together by the router. Each LAN resides in its own collision domain and its own broadcast domain.

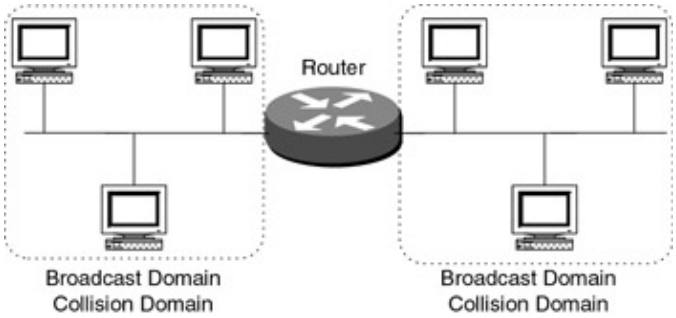


Figure 20−4: Router example

Figure 20−5 shows a LAN switch that supports virtual LANs. All six workstations are connected to the same LAN switch.

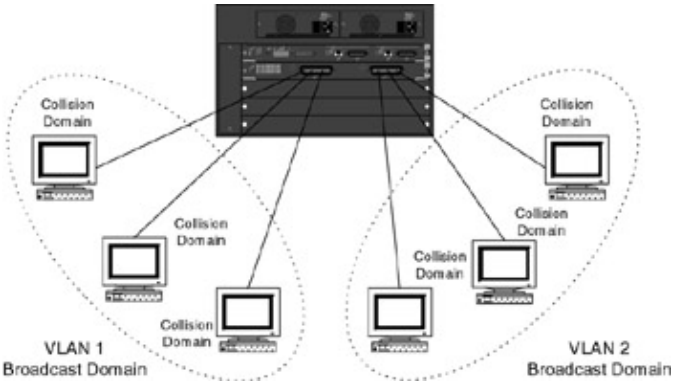


Figure 20−5: LAN switch example

A virtual LAN (VLAN) is an administratively defined broadcast domain. All end stations that reside in a common VLAN will receive broadcast packets that are sent by other end stations that reside on the VLAN. A VLAN may sound very similar to a traditional LAN switch, but the key difference is that in a VLAN, the end stations do not need to be in the same physical location.

The three workstations that reside in each VLAN are all in a single broadcast domain. Each of the six workstations is in its own collision domain.

Routing Between VLANs

Two routers that reside in separate VLANs encounter the same issue that two routers residing on two different LANs have. How do you route between the VLANs? The Catalyst switch can accomplish this in one of two ways:

1. The Catalyst has the ability to connect to a router via a 100−Mbps Ethernet link using Interswitch Link (ISL) encapsulation. The router that is connected to the Catalyst uses subinterfaces to route between the VLANs. Each VLAN is assigned to a separate subinterface on the router. This concept is shown in Figure 20−6.

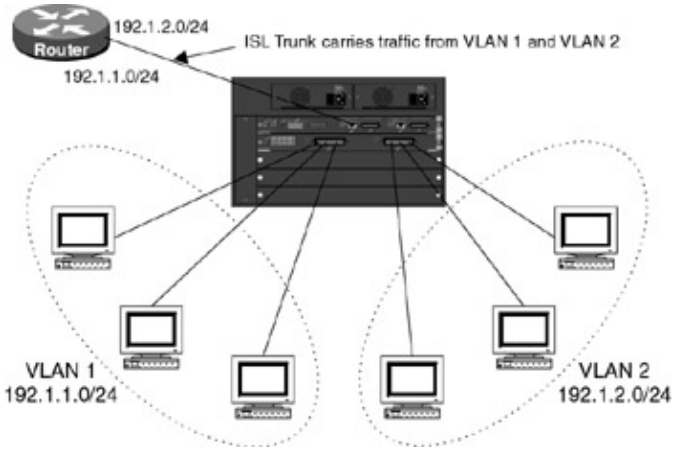


Figure 20−6: Routing between VLANs

2. Use a route switch module (RSM) — The Catalyst RSM is a Cisco 7500 class router that is packaged in a Catalyst card form factor. It does not have any physical interfaces. Instead, it uses virtual interfaces to route between VLANs.

Accessing the Catalyst

Every Catalyst 5000 family switch has an internal logical interface, referred to as the SC0 interface. The SC0 interface is used to provide an active IP address that can be used to telnet into the Catalyst for configuration and monitoring. The SC0 interface is usually in VLAN 1, but can be moved to any VLAN. Without an active SC0 interface, the Catalyst switch would need to be accessed via the console or Aux port on the Supervisor Engine. The Catalyst also supports SLIP connections. The Catalyst SLIP IP address is configured by defining the SL0 interface on the switch.

Catalyst Trunks

Catalyst user ports can also be defined as trunks. These trunks can be used to connect a Catalyst switch to other Catalyst switches or to a router.

Catalyst Configuration

Configuring a Cisco Catalyst switch is different than configuring a Cisco router in several ways:

- The router has a separate configuration mode. With the Catalyst, you type the configuration commands at the enable prompt command line. On both the Catalyst and router, changes take effect immediately.
- A Cisco router has several modes of operation such as exec, debug, configuration, and so forth. The Catalyst switch only has normal and privileged modes.
- The router has two types of configuration memory: running and startup. The running configuration is the configuration that is currently active on the router. The startup configuration is the configuration that is stored in NVRAM. When you make a configuration change on the router, the running configuration changes but the startup configuration does not change. With the Catalyst, there is only one configuration memory and it gets changed as soon as a configuration change is made.
- The **show run** command on the router will display the currently running configuration. The Cisco router configuration is usually very short, only showing commands that have been entered and that are not the default commands. The Catalyst switch configuration is very long. It shows every parameter for the switch, whether or not it has been configured by the user. Below are some lines from a Catalyst switch configuration:

```
#module 5 : 12-port 10/100BaseTX Ethernet
set module name    5
set module enable  5
set vlan 1    5/1-10
set vlan 2    5/11-12
set port channel 5/1-12 off
set port channel 5/1-12 auto
set port enable     5/1-12
set port level      5/1-12  normal
set port speed      5/1-12  auto
set port trap       5/1-12  disable
set port name       5/11 RouterB
set port name       5/12 RouterA
set port name       5/1-10
set port security   5/1-12  disable
set port broadcast  5/1-12  0
set port membership 5/1-12  static
set cdp enable   5/1-12
set cdp interval 5/1-12 60
set trunk 5/1  auto 1-1005
set trunk 5/2  auto 1-1005
set trunk 5/3  auto 1-1005
set trunk 5/4  auto 1-1005
set trunk 5/5  auto 1-1005
set trunk 5/6  auto 1-1005
set trunk 5/7  auto 1-1005
set trunk 5/8  auto 1-1005
set trunk 5/9  auto 1-1005
set trunk 5/10 auto 1-1005
set trunk 5/11 auto 1-1005
set trunk 5/12 off 1-1005
set spantree portfast    5/1-12 disable
set spantree portcost    5/1  100
set spantree portcost    5/2  100
set spantree portcost    5/3  100
set spantree portcost    5/4  100
set spantree portcost    5/5  100
```

```
set spantree portcost    5/6  100
set spantree portcost    5/7  100
set spantree portcost    5/8  100
set spantree portcost    5/9  100
set spantree portcost    5/10 100
set spantree portcost    5/11 100
set spantree portcost    5/12 100
set spantree portpri     5/1-12 32
set spantree portvlanpri 5/1  0
set spantree portvlanpri 5/2  0
set spantree portvlanpri 5/3  0
set spantree portvlanpri 5/4  0
set spantree portvlanpri 5/5  0
set spantree portvlanpri 5/6  0
set spantree portvlanpri 5/7  0
set spantree portvlanpri 5/8  0
set spantree portvlanpri 5/9  0
set spantree portvlanpri 5/10 0
set spantree portvlanpri 5/11 0
set spantree portvlanpri 5/12 0
set spantree portvlancost 5/1  cost 99
set spantree portvlancost 5/2  cost 99
set spantree portvlancost 5/3  cost 99
set spantree portvlancost 5/4  cost 99
set spantree portvlancost 5/5  cost 99
set spantree portvlancost 5/6  cost 99
set spantree portvlancost 5/7  cost 99
set spantree portvlancost 5/8  cost 99
set spantree portvlancost 5/9  cost 99
set spantree portvlancost 5/10 cost 99
set spantree portvlancost 5/11 cost 99
set spantree portvlancost 5/12 cost 99
```

Commands Discussed in This Chapter

- **clear config all**
- **ping** *host [packet_size] [packet_count]*
- **set interface sc0** *[ip_addr [netmask [broadcast]]]*
- **set ip permit {enable | disable} / set ip permit** *ip_addr [mask]*
- **set port name** *mod_num/port_num [name_string]*
- **set port security** *mod_num/port_num* **{enable | disable}** *[mac_addr]*
- **set trunk** *mod_num/port_num* **[on | off | desirable | auto]** *[vlan_range]*
- **set vlan** *vlan_num mod_num/port_num*
- **set vtp domain** *name*
- **show cam dynamic**
- **show interface**
- **show ip permit**
- **show mac** *[mod_num/[port_num]]*
- **show module** *mod_num*
- **show port** *[mod_num/port_num]*
- **show system**
- **show trunk** *[mod_num[/port_num]]*
- **show version**
- **show vlan** *[vlan]*
- **show vtp domain**

Definitions

clear config all: This privileged command clears the Catalyst configuration and resets the switch.

ping: This normal mode command sends ICMP echo request packets to the selected node.

set interface: This privileged command sets the sc0 interface for inband telnet and SNMP access. It can also be used to set the sl0 interface for SLIP telnet and SNMP access.

set ip permit: This privileged command enables or disables the IP permit list and creates an entry in the IP permit list.

set port name: This privileged command sets the name of a Catalyst switch port.

set port security: This privileged command enables or disables MAC level port security on the switch.

set trunk: This privileged command configures a Catalyst port to become a trunk.

set vlan: This privileged command configures VLAN options on the switch.

set vtp domain: This privileged command sets the VTP domain name.

show cam dynamic: This normal command shows the contents of the CAM table.

show interface: This normal command displays information about the Catalyst switch interfaces.

show ip permit: This normal command displays information on IP permit lists that are defined on the switch.

show mac: This normal command displays information on MAC level statistics on the switch.

show module: This normal command displays module information for the switch.

show port: This normal command displays port level statistics for the switch.

show system: This normal command displays system information for the switch.

show trunk: This normal command shows trunking information for the switch.

show version: This normal command show hardware and software version information for the switch.

show vlan: This normal command displays VLAN information for the switch.

show vtp domain: This normal command displays VTP domain information for the switch.

IOS Requirements

These labs were done using Cisco IOS 11.2. ISL trunks are supported in IOS 11.2 and higher. The Catalyst switch was running version 3.1.

Lab #92: Basic Catalyst Configuration, VLANs, and Port Security

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with Ethernet interfaces
- A Catalyst switch with 10−Mbps or 10/100−Mbps Ethernet ports
- Two Ethernet cables
- A Cisco rolled cable for console port connection to the routers
- A straight−through cable for console port connection to the Catalyst switch

Configuration Overview

This lab will demonstrate how to configure a Catalyst 5500 for basic LAN switching. Two routers, RouterA and RouterB, will be connected to a Catalyst switch as shown in Figure 20−7. The two routers will both reside in the same VLAN. Two Catalyst security features will also be demonstrated, IP permit and MAC filtering:

Figure 20−7: Catalyst configuration with port security

- **IP permit** This feature allows up to 10 IP addresses to be entered into the Catalyst. When IP permit is enabled, the Catalyst will only accept telnet and SNMP traffic from the 10 predefined IP addresses. If an unauthorized address attempts to send telnet or SNMP traffic to the switch, the traffic is rejected and the Catalyst records the source address of the rejected traffic.
- **MAC filtering** The Catalyst can be configured to reject incoming traffic on a port that does not have a source MAC address that matches a predefined MAC address that has been entered into the switch.

Note Cisco makes many models of LAN switches. Although this lab was done using a Catalyst 5500 switch, there are other LAN switches in the Cisco product line that could be used. For example, the Catalyst 1924 Enterprise Edition switch is a low−cost switch that is capable of doing VLANs and can also have a 100Mbps ISL trunk.

Note The Catalyst does not use the same IOS as a Cisco router. You will notice that the command set is very different. Many items that are taken for granted on the router, such as being able to use the tab key to complete a command, are not available on the Catalyst switch. Catalyst ports are referred to by slot and port number. For example, in this lab we are connected to the 11th and 12th port of Card 5. The Catalyst will refer to these ports as 5/11 and 5/12, respectively.

Router Configuration

The configurations for the two routers in this example are as follows.

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Ethernet0/0
```

```
 ip address 192.1.1.1 255.255.255.0 ← Define the IP address for the interface
                                       connected to the Catalyst switch
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 exec-timeout 30 0
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
interface Ethernet0/0
 ip address 192.1.1.2 255.255.255.0 ← Define the IP address for the interface
                                       connected to the Catalyst switch
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 exec-timeout 30 0
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to the Catalyst 5500. We will clear the entire configuration on the Catalyst so we are sure that we are starting with a known configuration. Use the command **clear config all** to set the Catalyst back to its factory default state.

```
Console> (enable) clear config all
This command will clear all configuration in NVRAM.
This command will cause ifIndex to be reassigned on the next system startup.
Do you want to continue (y/n) [n]? y
.......
.........................
.............
.........................
.........................
.................

System configuration cleared.
```

After the Catalyst has been reset, all ports are defined to be on a single VLAN, VLAN1. The Catalyst acts as a large multiport LAN switch. The Catalyst will automatically sense that an active LAN is connected to one of its ports and set the corresponding port parameters correctly. We see from the **show port** output below that ports 5/11 and 5/12 have been automatically configured. Their status is connected; they are both in VLAN 1; and they are both running 10−Mbps half−duplex Ethernet. Keep in mind that we did not have to configure ports 5/11 and 5/12 after we reset the Catalyst to factory default state.

```
Console> (enable) sh port
Port    Name    Status      Vlan     Level     Duplex     Speed    Type
----    ----    ---------   ----     ------    ------     -----    -------------
5/11            connected   1        normal    a-half     a-10     10/100 BaseTX
5/12            connected   1        normal    a-half     a-10     10/100 BaseTX
```

More detailed port status is available by adding the port number after the **show port** command. Type **show port 5/11** to view the status for port 5/11 . We see that additional data such as MAC−level security information and Ethernet collision and error statistics are listed.

```
Console> (enable) sh port 5/11
Port    Name    Status      Vlan     Level     Duplex     Speed    Type
----    ----    ---------   ----     ------    ------     -----    -------------
5/11            connected   1        normal    a-half     a-10     10/100 BaseTX

Port    Security     Secure-Src-Addr     Last-Src-Addr     Shutdown     Trap
----    --------     ---------------     -------------     --------     --------
5/11    disabled                                           No           disabled

Port   Broadcast-Limit         Broadcast-Drop
----   ---------------         --------------
5/11   -                       0
Port   Status      Channel     Channel     Neighbor            Neighbor
                   mode        status      device              port
-----  ---------   -------     ----------  --------            --------
5/11   connected   auto        not channel

Port   Align-Err    FCS-Err     Xmit-Err      Rcv-Err      UnderSize
-----  ---------    -------     --------      -------      ---------
5/11   0            0           0             0            0

Port   Single-Col  Multi-Coll  Late-Coll  Excess-Col  Carri-Sen   Runts   Giants
-----  ----------  ----------  ---------- ----------  ---------   -----   ------
5/11   0           0           0          0           0           0       0

Last-Time-Cleared
-------------------------
Sun May 16 1999, 02:25:04
```

Catalyst ports can be given names to make them easier to identify. Use the **set port name** command to give names to ports 5/11 and 5/12.

```
Console> (enable) set port name 5/11 RouterB
Port 5/11 name set.
Console> (enable) set port name 5/12 RouterA
Port 5/12 name set.
```

We see from the **show port 5/12** command that the port name has been set to RouterA.

```
Console> (enable) sh port 5/12
Port  Name      Status     Vlan     Level     Duplex     Speed   Type
----- ------    ---------  ----     ------    ------     -----    -------------
5/12  RouterA   connected  1        normal    a-half     a-10    10/100 BaseTX

Port  Security     Secure-Src-Addr     Last-Src-Addr     Shutdown    Trap
----- --------     ---------------     -------------     --------    --------
5/12  disabled                                           No          disabled

Port  Broadcast-Limit     Broadcast-Drop
----- ---------------     --------------
5/12  -                   0
Port  Status      Channel     Channel       Neighbor     Neighbor
                  mode        status        device       port
----- ---------   -------     -----------   --------     --------
5/12  connected   auto        not channel
```

```
Port  Align-Err   FCS-Err     Xmit-Err        Rcv-Err          UnderSize
----- ---------   -------     --------        -------          ---------
5/12  0           0           0               0                0

Port  Single-Col  Multi-Coll   Late-Coll   Excess-Col  Carri-Sen  Runts   Giants
----  ----------  ----------   ---------   ----------  ---------  -----   ------
5/12  0           0            0           0           0          0       0

Last-Time-Cleared
--------------------------
Sun May 16 1999, 02:25:04
```

Now connect to RouterB. Verify that you can ping RouterA at IP address 192.1.1.1. Remember that both RouterA and RouterB were automatically put into VLAN1 when we reset the Catalyst switch.

```
RouterB#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

The Catalyst switch can be assigned an internal IP address that is used for SNMP and telnet access. The IP address can be verified with the **show interface** command. We see below that there are no IP addresses set for the switch.

```
Console> (enable) sh interface
sl0: flags=51<UP,POINTOPOINT,RUNNING>
        slip 0.0.0.0 dest 128.73.35.160
sc0: flags=63<UP,BROADCAST,RUNNING>
        vlan 1 inet 0.0.0.0 netmask 0.0.0.0 broadcast 0.0.0.0
```

The IP address for inband access can be entered into the switch with the **set interface sc0** command. Enter an sc0 IP address of 192.1.1.3 as shown below. Notice that this address is on the same network as the IP addresses of RouterA (192.1.1.1) and RouterB (192.1.1.2).

```
Console> (enable) set interface sc0 192.1.1.3
Interface sc0 IP address set.
```

The **show interface** command will now indicate that the sc0 IP address has been set to 192.1.1.3.

```
Console> (enable) sh interface
sl0: flags=51<UP,POINTOPOINT,RUNNING>
        slip 0.0.0.0 dest 128.73.35.160
sc0: flags=63<UP,BROADCAST,RUNNING>
        vlan 1 inet 192.1.1.3 netmask 255.255.255.0 broadcast 192.1.1.255
```

Once the sc0 address has been set, verify that it is active by pinging the sc0 address.

```
Console> (enable) ping 192.1.1.3
192.1.1.3 is alive
```

We will also be able to ping RouterA and RouterB.

```
Console> (enable) ping 192.1.1.1
192.1.1.1 is alive


Console> (enable) ping 192.1.1.2
192.1.1.2 is alive
```

Both RouterA and RouterB should be able to ping the sc0 interface of the Catalyst switch. We see below that RouterA is able to ping the Catalyst.

```
RouterA#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
```

IP Permit Lists

The Catalyst switch has powerful security features. One such feature is the IP permit capability of the switch. The IP permit feature of the Catalyst allows the user to define up to 10 IP addresses that are allowed inbound SNMP and telnet access to the switch. The permit list can be displayed with the **show ip permit** command. We see below that there are no IP addresses in the permit list of the switch.

```
Console> (enable) show ip permit
IP permit list feature disabled.
Permit List      Mask
-----------      ----

Denied IP Address      Last Accessed Time      Type
-----------------      ------------------      ----
```

Let's add an IP address to the permit list of the switch with the **set ip permit 192.1.1.1** command. This will allow RouterA inbound SNMP and telnet access to the Catalyst switch.

```
Console> (enable) set ip permit 192.1.1.1
192.1.1.1 added to IP permit list.
```

The **show ip permit** command will now indicate that 192.1.1.1 is on the permit list. Notice that the IP permit list feature has been disabled. This is the default state of the IP permit list.

```
Console> (enable) show ip permit
IP permit list feature disabled.
Permit List      Mask
-----------      ----
192.1.1.1

Denied IP Address      Last Accessed Time      Type
-----------------      ------------------      ----
```

After the IP permit list has been defined, it must be enabled with the **set ip permit enable** command.

```
Console> (enable) set ip permit enable
IP permit list enabled.
```

Now let's connect to RouterB. The IP address of RouterB's Ethernet interface that is connected to the Catalyst switch is 192.1.1.2. This address is not on the IP permit list of the Catalyst switch. Let's try to ping the sc0 interface of the Catalyst switch from RouterB. We see that the ping is successful. Remember that the IP permit list only denies inbound SNMP and telnet access to the switch.

```
RouterB#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

Let's try to telnet to the Catalyst switch at IP address 192.1.1.3. We see that the Catalyst switch rejects the telnet session since RouterB's address of 192.1.1.2 is not on the IP permit list.

```
RouterB#telnet 192.1.1.3
Trying 192.1.1.3 ... Open
Access not permitted. Closing connection...

[Connection to 192.1.1.3 closed by foreign host]
```

Now connect to the Catalyst switch and display the IP permit list with the **show ip permit** command. We see that the denied IP address list now has an entry for the telnet session that we just tried to initiate from RouterB.

```
Console> (enable) show ip permit
IP permit list feature enabled.
Permit List      Mask
-----------      ----
192.1.1.1

Denied IP Address      Last Accessed Time      Type
-----------------      ------------------      ------
192.1.1.2              05/25/99,14:25:50       Telnet
```

Disable the IP permit list with the **set ip permit disable** command.

```
Console> (enable) set ip permit disable
IP permit list disabled.
```

Now reconnect to RouterB and try to telnet to the Catalyst switch. We see that the telnet is now successful since the IP permit list has been disabled.

```
RouterB#telnet 192.1.1.3
Trying 192.1.1.3 ... Open

Cisco Systems Console

Enter password:
Console> ena
Enter password:
Console> (enable)
Console> (enable) exit

[Connection to 192.1.1.3 closed by foreign host]
```

Secure Port Filtering

The Catalyst switch can be configured to only allow inbound traffic on a switch port that contains a MAC address that has been entered into the Catalyst switch. This feature is called secure port filtering. We see from the output below of the **show port 5/12** command that there are no entries under the MAC Source Address fields on the interface.

```
Console> (enable) sh port 5/12
Port   Name        Status      Vlan    Level    Duplex    Speed    Type
----   -------     ---------   ----    ------   ------    -----    -------------
5/12   RouterA     connected   1       normal   a-half    a-10     10/100 BaseTX

Port   Security    Secure-Src-Addr     Last-Src-Addr     Shutdown    Trap
----   --------    ---------------     -------------     --------    --------
5/12   disabled                                          No          disabled

Port   Broadcast-Limit     Broadcast-Drop
----   ---------------     --------------
5/12   -                   0
```

```
Port   Status      Channel     Channel      Neighbor   Neighbor
                   mode        status       device     port
----   ---------   -------     -----------  --------   --------
5/12   connected   auto        not channel

Port   Align-Err   FCS-Err     Xmit-Err   Rcv-Err      UnderSize
----   ---------   -------     --------   -------      ---------
5/12   0           0           0          0            0

Port   Single-Col  Multi-Coll  Late-Coll  Excess-Col   Carri-Sen  Runts   Giants
----   ----------  ----------  ---------  ----------   ---------  -----   ------
5/12   0           0           0          0            0          0       0

Last-Time-Cleared
--------------------------
Sun May 16 1999, 02:25:04
```

We will now configure the Catalyst to only allow inbound Ethernet packets on port 5/12 that contain a specific source MAC address. In order for us to configure Secure Port Filtering on the Catalyst, we will need to know the MAC address of the host that is connected to port 5/12. RouterA's E0/0 interface is connected to port 5/12 on the Catalyst switch. Connect to RouterA and use the **show interface e0/0** command to view the MAC address for the Ethernet interface of the router. We see that the MAC address for this interface is 00e0.1e5b.2761.

```
RouterA#sh int e 0/0
Ethernet0/0 is up, line protocol is up
  Hardware is AmdP2, address is 00e0.1e5b.2761 (bia 00e0.1e5b.2761)
  Internet address is 192.1.1.1/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:22, output 00:00:07, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     18672 packets input, 17647218 bytes, 0 no buffer
     Received 3662 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     24112 packets output, 18236637 bytes, 0 underruns
     118 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 1 deferred
     118 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Now connect to the Catalyst switch. Use the **set port security** command shown below to define what MAC address will be accepted when traffic comes into the Catalyst switch.

```
Console> (enable) set port security 5/12 enable 00-e0-1e-5b-27-62
Port 5/12 port security enabled with 00-e0-1e-5b-27-62 as the secure mac address
Trunking disabled for Port 5/12 due to Security Mode
```

Reconnect to RouterA. Try to ping the sc0 interface of the Catalyst at IP address 192.1.1.3. We see that the ping fails.

```
RouterA#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Connect to the Catalyst switch. We see that the status of the port is shutdown. The reason for the port being shut down is shown under the Secure−Src−Addr and Last−Src−Addr columns. These two columns show what MAC address will be allowed into the switch port and what the last MAC address sent to the port was. Notice that the Last−Src−Addr does not match the Secure−Src−Addr.

```
Console> (enable) show port 5/12
Port   Name      Status     Vlan      Level      Duplex    Speed  Type
----   -------   --------   ----      ------     ------    -----  -------------
5/12   RouterA   shutdown   1         normal     a-half    a-10   10/100 BaseTX

Port   Security    Secure-Src-Addr       Last-Src-Addr       Shutdown   Trap
----   --------    -----------------     -----------------   --------   --------
5/12   enabled     00-e0-1e-5b-27-62     00-e0-1e-5b-27-61   Yes        disabled

Port   Broadcast-Limit      Broadcast-Drop
----   ---------------      --------------
5/12   -                    0
Port   Status      Channel     Channel     Neighbor    Neighbor
                   mode        status      device      port
----   --------    -------     ----------- --------    --------
5/12   shutdown    auto        not channel

Port   Align-Err   FCS-Err     Xmit-Err    Rcv-Err     UnderSize
----   ---------   -------     --------    -------     ---------
5/12   0           0           0           0           0

Port   Single-Col  Multi-Coll  Late-Coll   Excess-Col  Carri-Sen   Runts  Giants
----   ----------  ----------  ---------   ----------  ---------   -----  ------
5/12   0           0           0           0           0           0      0

Last-Time-Cleared
--------------------------
Sun May 16 1999, 02:25:04
```

Disable port security on port 5/12 with the **set port security 5/12 disable** command.

```
Console> (enable) set port security 5/12 disable
Port 5/12 port security disabled.
```

Use the **show port 5/12** command to view the port status. We see that the status is now connected.

```
Console> (enable) sh port 5/12
Port   Name       Status       Vlan  Level     Duplex     Speed    Type
----   -------    ---------    ----  ------    ------     -----    -------------
5/12   RouterA    connected    1     normal    a-half     a-10     10/100 BaseTX

Port   Security    Secure-Src-Addr      Last-Src-Addr      Shutdown     Trap
-----  --------    ---------------      -------------      --------     --------
5/12   disabled                                           No           disabled

Port   Broadcast-Limit      Broadcast-Drop
----   ---------------      --------------
5/12   -                    0
Port   Status      Channel     Channel     Neighbor    Neighbor
                   mode        status      device      port
----   ---------   -------     ----------- --------    ---------
5/12   connected   auto        not channel

Port   Align-Err   FCS-Err     Xmit-Err    Rcv-Err     UnderSize
----   ---------   -------     --------    -------     ---------
5/12   0           0           0           0           0


Port   Single-Col  Multi-Coll  Late-Coll   Excess-Col  Carri-Sen   Runts  Giants
----   ----------  ----------  ---------   ----------  ---------   -----  ------
```

```
5/12   0           0           0            0          0           0      0

Last-Time-Cleared
--------------------------
Sun May 16 1999, 02:25:04
```

Connect to RouterA. You should once again be able ping RouterB at IP address 192.1.1.2.

```
RouterA#ping 192.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/7/8 ms
```

Now we are going to move both RouterA and RouterB to VLAN 2. Remember, when we reset the Catalyst we said that the switch resets in a state where all ports are in VLAN 1. The Catalyst switch must have a domain name before it can use VLAN numbers other than 1. We see in the **show vtp domain** output that the domain name has not been set yet on this switch.

```
Console> (enable) sh vtp domain
Domain Name       Domain Index       VTP Version       Local Mode       Password
-----------       ------------       -----------       ----------       --------
                  1                  2                 server           -

Vlan-count      Max-vlan-storage      Config Revision      Notifications
----------      ----------------      ---------------      -------------
5               1023                  0                    disabled

Last Updated      V2 Mode      Pruning      PruneEligible on Vlans
------------      --------     --------     ----------------------
0.0.0.0           disabled     disabled     2-1000
```

Set the VTP domain name with the command **set vtp domain CCIE_STUDY_GUIDE**

```
Console> (enable) set vtp domain CCIE_STUDY_GUIDE
VTP domain CCIE_STUDY_GUIDE modified

Console> (enable) show vtp domain
Domain Name           Domain Index      VTP Version      Local Mode      Password
----------------      ------------      -----------      ----------      --------
CCIE_STUDY_GUIDE      1                 2                server          -

Vlan-count      Max-vlan-storage      Config Revision      Notifications
----------      ----------------      ---------------      -------------
5               1023                  0                    disabled

Last Updater        V2 Mode      Pruning      PruneEligible on Vlans
------------        -------      -------      ----------------------
0.0.0.0             disabled     disabled     2-1000
```

Use the **set vlan 2 5/11** command to move port 5/11 to VLAN 2. Notice that the switch automatically modifies VLAN 1 and removes port 5/11 from VLAN 1.

```
Console> (enable) set vlan 2 5/11
Vlan 2 configuration successful
VLAN 2 modified.
VLAN 1 modified.
VLAN  Mod/Ports
----  ------------------------
2     5/11
```

Use the **set vlan 2 5/12** command to move port 5/12 to VLAN 2.

```
Console> (enable) set vlan 2 5/12
VLAN 2 modified.
VLAN 1 modified.
VLAN  Mod/Ports
----  ------------------------
2     5/11-12
```

Activate the VLAN with the command **set vlan 2**.

```
Console> (enable) set vlan 2
Vlan 2 configuration successful
```

The **show vlan 2** command will now indicate that VLAN2 is active and contains two ports: 5/11 and 5/12.

```
Console> (enable) sh vlan 2
VLAN   Name                             Status        Mod/Ports, Vlans
----   --------                         ------        ----------------
2      VLAN0002                         active        5/11-12


VLAN  Type  SAID    MTU  Parent   RingNo  BrdgNo   Stp  BrdgMode  Trans1  Trans2
----  ----  ------  ----  ------  ------  ------   ---  --------  ------  ------
2     enet  100002  1500  -       -       -        -    -         0       0


VLAN      AREHops      STEHops      Backup CRF
----      -------      -------      ----------
```

The VLAN status can also be displayed using the **show vlan** command. We see that all of the other Ethernet ports still reside in the default VLAN 1.

```
Console> (enable) sh vlan
VLAN   Name                             Status        Mod/Ports, Vlans
----   ------------------------         ------        ----------------
1      default                          active        2/1-2
                                                      3/1-24
                                                      5/1-10
                                                      7/1-24
                                                      10/1-24
2      VLAN0002                         active        5/11-12
       1002 fddi-default                active
       1003 token-ring-default          active        12/1-16
       1004 fddinet-default             active
       1005 trnet-default               active


VLAN  Type   SAID    MTU   Parent  RingNo  BrdgNo  Stp  BrdgMode  Trans1  Trans2
----  ----   ----    ----  ------  ------  ------  ---  --------  ------  ------
1     enet   100001  1500  -       -       -       -    -         0       0
2     enet   100002  1500  -       -       -       -    -         0       0
1002  fddi   101002  1500  -       0x0     -       -    -         0       0
1003  trcrf  101003  1500  0       0x0     -       -    -         0       0
1004  fdnet  101004  1500  -       -       0x0     ieee -         0       0
1005  trbrf  101005  1500  -       -       0x0     ibm  -         0       0


VLAN      AREHops      STEHops      Backup CRF
----      -------      -------      ----------
1003      7            7            off
```

We can verify that VLAN 2 is active by connecting to RouterA and trying to ping RouterB at IP address 192.1.1.2. We see from the output below that the ping was successful. RouterA and RouterB are now both on VLAN 2.

```
RouterA#ping 192.1.1.2
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/7/8 ms
```

Lab #93: ISL Trunk with Routing Between VLANs

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with Ethernet interfaces
- One Cisco router with a 100−Mbps Ethernet interface
- A Catalyst switch
- Three Ethernet cables
- A Cisco rolled cable for console port connection to the routers
- A straight−through cable for console port connection to the Catalyst switch

Configuration Overview

This lab will demonstrate how to route between two VLANs. As shown in Figure 20−8, RouterA will reside in VLAN 1 and RouterB will reside in VLAN 2. Both VLAN 1 and VLAN 2 reside in different IP networks. Since the Catalyst is a layer 2 switch, it is unable to route between the two VLANs. A layer 3 router is needed to perform this function. The solution is to define a high−speed trunk between the Catalyst switch and a router. This trunk is referred to as an Interswitch Link (ISL) and runs over a 100−Mbps Ethernet interface.

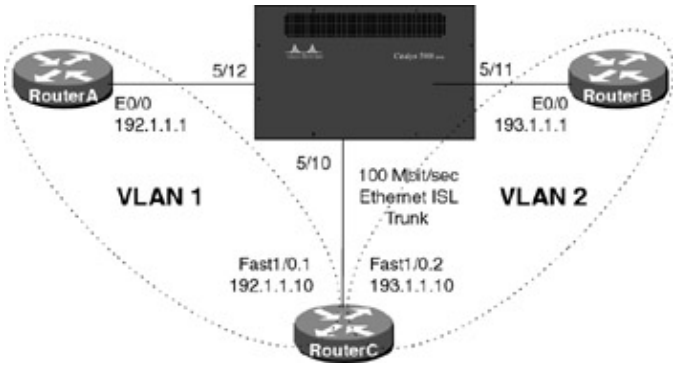


Figure 20−8: Routing between two VLANs

Note Cisco makes many models of LAN switches. Although this lab was done using a Catalyst 5500 switch, there are other LAN switches in the Cisco product line that could be used. For example, the Catalyst 1924 Enterprise Edition is a low−cost switch that is capable of doing VLANs and can also have a 100−Mbps ISL trunk.

Note The Catalyst does not use the same IOS as a Cisco router. You will notice that the command set is very different. Many items that are taken for granted on the router, such as being able to use the tab key to complete a command, are not available on the Catalyst switch.

Note Catalyst ports are referred to by slot and port number. For example, in this lab we are connected to the 11th and 12th port of Card 5. The Catalyst will refer to these ports as 5/11 and 5/12, respectively.

Router Configuration

The configurations for the three routers in this example are as follows.

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
interface Ethernet0/0
 ip address 192.1.1.1 255.255.255.0 ← Define the IP address for the interface
                                      connected to the Catalyst switch
!
router rip
 network 192.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 exec-timeout 30 0
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
!
interface Ethernet0/0
 ip address 193.1.1.1 255.255.255.0 ← Define the IP address for the interface
                                      connected to the Catalyst switch
!
router rip
 network 193.1.1.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 exec-timeout 30 0
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
```

```
no service tcp-small-servers
!
hostname RouterC
!
interface FastEthernet1/0 ← This 100Mbps interface connects to the Catalyst
                            trunk port
 no ip address
 no logging event subif-link-status
!
interface FastEthernet1/0.1 ← This subinterface accepts traffic from VLAN 1
 encapsulation isl 1 ← Define ISL encapsulation and accept traffic from VLAN 1
 ip address 192.1.1.10 255.255.255.0 ← IP address for this subinterface
 no ip redirects
!
interface FastEthernet1/0.2 ← This subinterface accepts traffic from VLAN 2
 encapsulation isl 2 ← Define ISL encapsulation and accept traffic from VLAN 2
 ip address 193.1.1.10 255.255.255.0 ← IP address for this subinterface
 no ip redirects
!
router rip ← We need to dynamically route between VLAN 1 and VLAN 2. Our routes
             will be learned via RIP
 network 192.1.1.0 ← Propagate RIP for the network on VLAN 1
 network 193.1.1.0 ← Propagate RIP for the network on VLAN 2
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by setting the Catalyst 5500 to its factory default setting with the **clear config all** command. Remember from the previous chapter that after the Catalyst has been reset, all of the Ethernet ports will be assigned to VLAN 1.

```
Console> (enable) clear config all
This command will clear all configuration in NVRAM.
This command will cause ifIndex to be reassigned on the next system startup.
Do you want to continue (y/n) [n]? y
.......
.........................
.............
.........................
.........................
.................
```

System configuration cleared.

Since we will be assigning Catalyst ports to multiple VLANs, we must set the VTP domain name of the switch with the **set vtp domain** command.

```
Console> (enable) set vtp domain CCIE_LAB
VTP domain CCIE_LAB modified
```

Port 5/12 is in VLAN 1 for this lab. We do not need to enter any commands to place port 5/12 into VLAN 1 since this is the default state of the Catalyst switch. Port 5/11 will be assigned to VLAN 2 for this lab. To assign port 5/11 to VLAN 2, we use the **set vlan 2 5/11** command.

```
Console> (enable) set vlan 2 5/11
```

```
Vlan 2 configuration successful
VLAN 2 modified.
VLAN 1 modified.
VLAN  Mod/Ports
----  ------------------------
2      5/10-11
```

Enable VLAN 2 with the **set vlan 2** command.

```
Console> (enable) set vlan 2
Vlan 2 configuration successful
```

Port 5/10 will be the trunk port for this lab. Port 5/10 will connect to our Cisco router. We will see shortly that port 5/10 will transmit all VLAN traffic to the Cisco router. The Cisco router will then be able to route between our two VLANs. We need to set port 5/10 to trunk mode with the **set trunk 5/10 on** command.

```
Console> (enable) set trunk 5/10 on
Port(s) 5/10 trunk mode set to on.
```

The status of port 5/10 can be viewed with the **show port 5/10** command. We see that the port is active and is now defined as a trunk port. Notice that the port is running at 100-Mbps full duplex. (The a- before the full duplex and 100 Mb indicates that these settings were autosensed by the Catalyst switch.)

```
Console> (enable) sh port 5/10
Port  Name      Status      Vlan      Level     Duplex    Speed    Type
----  ----      ---------   -----     ------    ------    -----    -------------
5/10            connected   trunk     normal    a-full    a-100    10/100 BaseTX

Port  Security    Secure-Src-Addr     Last-Src-Addr     Shutdown     Trap
----  --------    ---------------     -------------     --------     --------
5/10  disabled                                          No           disabled

Port  Broadcast-Limit        Broadcast-Drop
----  ---------------        -------------
5/10  -                      -
Port  Status      Channel      Channel      Neighbor   Neighbor
                  mode         status       device     port
----  ---------   -------      -----------  --------   --------
5/10  connected   auto         not channel

Port  Align-Err   FCS-Err      Xmit-Err   Rcv-Err      UnderSize
----  ---------   -------      --------   -------      ---------
5/10  0           0            0          0            0

Port  Single-Col  Multi-Coll   Late-Coll  Excess-Col   Carri-Sen  Runts   Giants
----  ----------  ----------   ---------  ----------   ---------  -----   ------
5/10  0            0           0          0            0          0       -

Last-Time-Cleared
-------------------------
Sun May 16 1999, 02:25:04
```

Verify that the ports connected to RouterA and RouterB (5/12 and 5/11) are connected. Notice that port 5/11 (RouterB) is in VLAN 2, while port 5/12 (RouterA) is in VLAN 1.

```
Console> (enable) sh port 5/11
Port     Name     Status     Vlan    Level     Duplex     Speed     Type
----     ----     ---------  ----    ------    ------     -----     ------------
5/11              connected  2       normal    a-half     a-10      10/100BaseTX

Console> (enable) sh port 5/12
Port     Name     Status     Vlan    Level     Duplex     Speed     Type
----     ----     ---------  ----    ------    ------     -----     ------------
5/12              connected  1       normal    a-half     a-10      10/100BaseTX
```

The **show trunk** command gives us specific information on our trunk, showing us what VLANs are allowed on the trunk (by default, all VLAN's are allowed on a trunk) and what VLANs are active on the trunk. We see that in our case, all traffic from all VLANs is allowed on trunk 5/10.

```
Console> (enable) sh trunk
Port      Mode      Status
----      ----      --------
5/10      on        trunking

Port      Vlans allowed on trunk
----      ---------------------------------------------------------------------
5/10      1-1005

Port      Vlans allowed and active in management domain
----      ---------------------------------------------------------------------
5/10      1-2,1003,1005

Port      Vlans in spanning tree forwarding state and not pruned
----      ---------------------------------------------------------------------
5/10      1-2,1003,1005
```

Now let's connect to RouterA and view the routing table with the **show ip route** command. We see that we are learning a route to the 193.1.1.0 network. The 193.1.1.0 network connects RouterB to the Catalyst switch on VLAN 2. The routing table on RouterA tells us that RouterC is working properly and is routing between two VLANs.

```
RouterA#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1- OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1- OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1- IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

C    192.1.1.0/24 is directly connected, Ethernet0/0
R    193.1.1.0/24 [120/1] via 192.1.1.10, 00:00:26, Ethernet0/0
```

Make sure that we have end-to-end connectivity by trying to ping RouterA at IP address 193.1.1.1. The ping should be successful.

```
RouterA#ping 193.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 193.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

Now let's connect to RouterB. View the routing table on RouterB with the **show ip router** command. We see that RouterB has learned a route to RouterA via RIP.

```
RouterB#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1- OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1- OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1- IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

R    192.1.1.0/24 [120/1] via 193.1.1.10, 00:00:10, Ethernet0/0
C    193.1.1.0/24 is directly connected, Ethernet0/0
```

Make sure that we can ping RouterA at IP address 192.1.1.1.

```
RouterB#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/6/8 ms
```

Now connect to RouterC and view its routing table with the **show ip route** command. We see that RouterC has two directly connected networks. Each of these networks is coming into RouterC on the same physical 100−Mbps Ethernet circuit. The Ethernet circuit has defined two subinterfaces, VLAN 1 is associated with subinterface FastEthernet 1/0.1 and VLAN 2 is assigned to subinterface FastEthernet 1/0.2.

```
RouterC#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1- OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1- OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1- IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

C    192.1.1.0/24 is directly connected, FastEthernet1/0.1
C    193.1.1.0/24 is directly connected, FastEthernet1/0.2
```

From RouterC, ping RouterA and RouterB to verify that the circuit is active.

```
RouterC#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms


RouterC#ping 193.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 193.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
```

Troubleshooting

{show version} The **show version** command displays important system−level information, including the version of system firmware, firmware level, and serial number for each card installed in the switch, system memory, and uptime statistics.

```
Console> (enable) show ver
WS-C5500 Software, Version McpSW: 3.1(1) NmpSW: 3.1
Copyright (c) 1995-1997 by Cisco Systems
NMP S/W compiled on Dec 31 1997, 18:36:38
MCP S/W compiled on Dec 31 1997, 18:33:15

System Bootstrap Version: 3.1(2)

Hardware Version: 1.3  Model: WS-C5500  Serial #: 069028115

Module     Ports     Model         Serial #     Hw     Fw         Fw1       Sw
------     -----     --------      ---------    ---    -------    ------    ---
2          2         WS-X5530      008167898    1.8    3.1        4.1       3.1
3          24        WS-X5224      008161402    1.3    3.1                  3.1
```

```
5          12        WS-X5203     008451509    1.1    3.1                3.1
7          24        WS-X5224     008161009    1.3    3.1                3.1
10         24        WS-X5224     008161288    1.3    3.1                3.1
12         16        WS-X5030     007380744    1.0    1.0(117   2.2(4)   3.1

           DRAM                   FLASH                    NVRAM
Module     Total   Used    Free   Total    Used     Free    Total   Used    Free
------     ------  ------  ------ -----    -----    -----   -----   ----    ----
2          32640K  11854K  20786K 8192K    3224K    4968K   512K    106K    406K

Uptime is 5 days, 20 hours, 14 minutes
```

{show module} The **show module** command shows what type of card is inserted into each slot of the Catalyst switch. Burned−in MAC address information is also displayed for each card.

```
Console> (enable) show module
Mod Module-Name   Ports  Module-Type              Model      Serial-Num  Status
--------------    -----  ----------------------   --------  ---------------
2                 2      10/100 BaseTX Supervis   WS-X5530   008167898   ok
3                 24     10/100 BaseTX Ethernet   WS-X5224   008161402   ok
5                 12     10/100 BaseTX Ethernet   WS-X5203   008451509   ok
7                 24     10/100 BaseTX Etherne    WS-X5224   008161009   ok
10                24     10/100 BaseTX Ethernet   WS-X5224   008161288   ok
12                16     Token Ring               WS-X5030   007380744   ok

Mod   MAC-Address(es)                            Hw       Fw         Sw
---   ----------------------------------------   ---      ---        ------
2     00-90-f2-a7-c1-00 thru 00-90-f2-a7-c4-ff   1.8      3.1(2)     3.1(1)
3     00-10-7b-2e-ca-e8 thru 00-10-7b-2e-ca-ff   1.3      3.1(1)     3.1(1)
5     00-10-7b-09-9a-50 thru 00-10-7b-09-9a-5b   1.1      3.1(1)     3.1(1)
7     00-10-7b-3d-be-f0 thru 00-10-7b-3d-bf-07   1.3      3.1(1)     3.1(1)
10    00-10-7b-3d-be-c0 thru 00-10-7b-3d-be-d7   1.3      3.1(1)     3.1(1)
12    00:05:77:05:86:42 thru 00:05:77:05:86:52   1.0      1.0(117    3.1(1)

Mod       Sub-Type     Sub-Model    Sub-Serial      Sub-Hw
---       --------     ---------    ----------      ------
2         EARL 1+      WS-F5520     0008157389      1.1
2         uplink       WS-U5531     0008577601      1.1
```

{show mac} The **show mac** command displays detailed statistics on traffic passing through the Catalyst switch. The following output has been truncated to just show the statistics for three ports on a Catalyst switch. Notice the detailed reporting statistics for each port, including total received and transmitted frames; multicast, unicast, and broadcast statistics; error statistics; and total octets transmitted and received.

```
Console> (enable) show mac
MAC    Rcv-Frms     Xmit-Frms   Rcv-Multi    Xmit-Multi   Rcv-Broad   Xmit-Broad
---    --------     ---------   ---------    ----------   ---------   ----------
5/10   30948        251858      14649        251758       08              0
5/11   44490        166061       4953        145105       96           5774
5/12   43857        166409       4438        145408       15           5823


MAC    Dely-Exced   MTU-Exced   In-Discard   Lrn-Discrd   In-Lost     Out-Lost
----   ----------   ---------   ----------   ----------   -------     --------
5/10   0            0           38           0            0           0
5/11   0            0           61           0            0           0
5/12   2            0           73           0            0           0


Port   Rcv-Unicast    Rcv-Multicast    Rcv-Broadcast
----   -----------    ------ ------    -------------
5/10   16192          14649            108
5/11   39441           4953             96
5/12   39405           4438             15
```

```
Port   Xmit-Unicast   Xmit-Multicast   Xmit-Broadcast
----   ------------   --------------   --------------
5/10     100          251764              0
5/11   15182          145107           5774
5/12   15178          145410           5823


Port   Rcv-Octet      Xmit-Octet
----   ---------      ----------
5/10    3183207       23975586
5/11   20334264       27851660
5/12   20290059       27865755


Last-Time-Cleared
-------------------------
Sun May 16 1999, 02:25:04
```

{clear config all} The **clear config all** command causes the switch to be reset to its factory default state. In this state, all ports reside in VLAN 1 and the Catalyst acts as a large switching hub.

```
Console> (enable) clear config all
This command will clear all configuration in NVRAM.
This command will cause ifIndex to be reassigned on the next system startup.
Do you want to continue (y/n) [n]? y
.......
.........................
.............
.........................
.........................
................

System configuration cleared.
```

{show port} The **show port** command displays statistics on port-level configuration on the Catalyst switch. The Catalyst can automatically sense speed and duplex on each port of the switch. For example, we see in the output below that ports 5/11 and 5/12 have been automatically configured. Their status is connected, they are both in VLAN 1, and they are both running 10-Mbps half-duplex Ethernet.

```
Console> (enable) sh port
Port   Name          Status      Vlan    Level  Duplex   Speed     Type
----   ------------  ----------  ----   ------  ------   -----     ------------
5/1                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/2                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/3                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/4                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/5                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/6                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/7                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/8                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/9                  notconnect  1      normal  auto     auto      10/100 BaseTX
5/10                 notconnect  1      normal  auto     auto      10/100 BaseTX
5/11                 connected   1      normal  a-half   a-10      10/100 BaseTX
5/12                 connected   1      normal  a-half   a-10      10/100 BaseTX
```

{show port *slot/port*} More detailed port status is available by adding the port number after the **show port** command. In the example below, we see that additional data such as MAC-level security information and Ethernet collision and error statistics are listed for the specified port.

```
Console> (enable) sh port 5/11
Port   Name       Status      Vlan     Level    Duplex    Speed    Type
-----  ----       ---------   ----     ------   ------    -----    ------------
5/11              connected   1        normal   a-half    a-10     10/100 BaseTX

Port  Security    Secure-Src-Addr      Last-Src-Addr      Shutdown    Trap
```

```
----  --------    ---------------      -------------      --------    --------
5/11  disabled                                           No          disabled

Port  Broadcast-Limit         Broadcast-Drop
----  ---------------         --------------
5/11  -                       0
Port  Status       Channel     Channel      Neighbor    Neighbor
                   mode        status       device      port
----  ---------    -------     -----------  --------    --------
5/11  connected    auto        not channel

Port  Align-Err    FCS-Err     Xmit-Err    Rcv-Err     UnderSize
----- ---------    -------     --------    -------     ---------
5/11  0            0           0           0           0

Port  Single-Col   Multi-Coll  Late-Coll   Excess-Col  Carri-Sen  Runts   Giants
----  -----------  ----------  ---------   ----------  ---------  -----   ------
5/11  0            0           0           0           0          0       0

Last-Time-Cleared
-------------------------
Sun May 16 1999, 02:25:04
```

{show cam dynamic} The **show cam dynamic** command displays connected host MAC addresses that have been learned by the switch.

```
Console> (enable) show cam dynamic
VLAN     Dest MAC/Route Des      Destination Ports or VCs
----     ------------------      -----------------------------------------------
2        00-e0-1e-9c-8e-b0       5/10
1        00-e0-1e-9c-8e-b0       5/10
2        00-10-7b-06-c2-c1       5/11
1        00-e0-1e-5b-27-61       5/12
1        00-00-ff-ff-ff-fb       1/4
Total Matching CAM Entries Displayed = 5
```

{show system} The **show system** command displays system contacts, current and peak traffic utilization, uptime, and thermal information.

```
Console> (enable) show system
PS1-Status PS2-Status  Fan-Status  Temp-Alarm Sys-Status  Uptime d,h:m:s  Logout
---------- ----------  ----------  ----------  ---------  --------------  ------
ok         none        ok          off        ok          5,20:14:10      20 min

PS1-Type   PS2-Type   Modem    Baud   Traffic   Peak   Peak-Time
--------   --------   ------   ----   -------   ----   -------------------------
WS-C5508   none       disable  9600   0%        0%     Sun May 16 1999, 02:25:04

System Name               System Location             System Contact
-----------------------   -------------------------   ----------------------
```

{set interface} The **set interface** command is used to set the IP address for inband access to the switch.

```
Console> (enable) set interface sc0 192.1.1.3
Interface sc0 IP address set.
```

{show interface} The **show interface** command is used to display the internal Catalyst IP addresses for inband access and SLIP access.

```
Console> (enable) sh interface
sl0: flags=51<UP,POINTOPOINT,RUNNING>
        slip 0.0.0.0 dest 128.73.35.160
sc0: flags=63<UP,BROADCAST,RUNNING>
        vlan 1 inet 192.1.1.3 netmask 255.255.255.0 broadcast 192.1.1.255
```

{set ip permit ip−address} The **set ip permit** command creates an IP permit list that the Catalyst uses to allow inband telnet and SNMP access to the switch. Up to 10 IP addresses can be defined.

```
Console> (enable) set ip permit 192.1.1.1
192.1.1.1 added to IP permit list.
```

{show ip permit} The **show ip permit** command is used to display the IP permit lists for the switch and to see if any invalid IP addresses have tried to access the switch for telnet or SNMP access. The IP permit list must be enabled with the **set ip permit enable** command. You can turn off the IP permit list with the **set ip permit disable** command.

```
Console> (enable) show ip permit
IP permit list feature enabled.
Permit List         Mask
−−−−−−−−−−−−−−−−    −−−−−−−−−−−−−−−
192.1.1.1

Denied IP Address   Last Accessed Time       Type
−−−−−−−−−−−−−−−−−   −−−−−−−−−−−−−−−−−−       −−−−−
192.1.1.2           05/25/99,14:25:50        Telnet
```

{set port security} The **set port security** command is used to define what MAC addresses are allowed to send traffic into the switch on a per−port basis. The command shown below will cause the switch to only allow inbound traffic on port 5/12 from a host with a MAC address of 00−e0−1e−5b−27−62. Port security can be disabled with the **set port security 5/12 disable** command.

```
Console> (enable) set port security 5/12 enable 00−e0−1e−5b−27−62
Port 5/12 port security enabled with 00−e0−1e−5b−27−62 as the secure mac address
Trunking disabled for Port 5/12 due to Security Mode
```

{show vtp domain} The **show vtp domain** shows key domain information for the switch. The Catalyst switch must have a domain name set before it can use VLAN numbers other than VLAN 1. The VTP domain name is set with the **set vtp domain** command.

```
Console> (enable) sh vtp domain
Domain Name             Domain Index     VTP Version     Local Mode     Password
−−−−−−−−−−−−−−−−−−−     −−−−−−−−−−−−     −−−−−−−−−−−     −−−−−−−−−−     −−−−−−−−
                        1                2               server         −

Vlan−count      Max−vlan−storage      Config Revision      Notifications
−−−−−−−−−−      −−−−−−−−−−−−−−−      −−−−−−−−−−−−−−−      −−−−−−−−−−−−−
5               1023                  0                    disabled

Last Updated      V2 Mode      Pruning      PruneEligible on Vlans
−−−−−−−−−−−−      −−−−−−−      −−−−−−−      −−−−−−−−−−−−−−−−−−−−−−
0.0.0.0           disabled     disabled     2−1000
```

{set vlan *vlan_number slot_port***}** The **set vlan** command is used to place a specific port in a VLAN. The example below assigns port 5/12 to VLAN 2. The VLAN is activated with the **set vlan** command.

```
Console> (enable) set vlan 2 5/12
VLAN 2 modified.
VLAN 1 modified.
VLAN  Mod/Ports
−−−−  −−−−−−−−−
2     5/11−12
```

{show vlan} The **show vlan** command displays information on all of the VLANs defined on the Catalyst switch.

```
Console> (enable) sh vlan
VLAN    Name                                Status      Mod/Ports, Vlans
```

```
----    ----------------------------      ------      ---------------
1       default                           active      2/1-2
                                                      3/1-24
                                                      5/1-10
                                                      7/1-24
                                                      10/1-24
2       VLAN0002                          active      5/11-12
1002    fddi-default                      active
1003    token-ring-default                active      12/1-16
1004    fddinet-default                   active
1005    trnet-default                     active


VLAN  Type   SAID    MTU   Paren  RingNo  BrdgNo  Stp   BrdgMode  Trans1  Trans2
---- -----  ---------- ----- ------ ------ ------ ----  --------  ------  ------
1     enet   100001  1500  -      -       -       -     -         0       0
2     enet   100002  1500  -      -       -       -     -         0       0
1002  fddi   101002  1500  -      0x0     -       -     -         0       0
1003  trcrf  101003  1500  0      0x0     -       -     -         0       0
1004  fdnet  101004  1500  -      -       0x0     ieee  -         0       0
1005  trbrf  101005  1500  -      -       0x0     ibm   -         0       0


VLAN     AREHops     STEHops     Backup CRF
----     -------     -------     ----------
1003     7           7           off
```

{show vlan *vlan_number***}** When supplied with a specific VLAN number, the **show vlan** command displays information on the specified VLAN. We see below that the VLAN name, status, and member ports are some of the statistics that are displayed.

```
Console> (enable) sh vlan 2
VLAN     Name                          Status      Mod/Ports, Vlans
----     --------------------------    ------      ---------------------------
2        VLAN0002                      active      5/11-12


VLAN  Type  SAID    MTU   Parent  RingNo  BrdgNo  Stp  BrdgMode  Trans1   Trans2
----  ----  ----    ---   ------  ------  ------  ---  --------  ------   ------
2     enet  100002  1500  -       -       -       -    -         0        0


VLAN   AREHops   STEHops    Backup CRF
----   -------   -------    ----------
```

{set trunk} The **set trunk** command configures a Catalyst port as a trunk port.

```
Console> (enable) set trunk 5/10 on
Port(s) 5/10 trunk mode set to on.
```

{show trunk} The **show trunk** command displays specific information on Catalyst trunks, such as what VLANs are allowed on the trunk and what VLANs are active on the trunk. We see that in the following output that all traffic from all VLANs is allowed on trunk 5/10.

```
Console> (enable) sh trunk
Port      Mode      Status
----      ----      --------
5/10      on        trunking

Port      Vlans allowed on trunk
----      ---------------------------------------------------------------------
5/10      1-1005

Port      Vlans allowed and active in management domain
----      ---------------------------------------------------------------------
```

```
5/10      1-2,1003,1005

Port      Vlans in spanning tree forwarding state and not pruned
----      --------------------------------------------------------------------
5/10      1-2,1003,1005
```

Conclusion

This chapter has explored the operations and configuration of the Catalyst 5500, one of a family of a broad range of LAN switches sold by Cisco. We have seen that the Catalyst switch combines the capabilities of a switching hub with VLAN capabilities. The Catalyst can accept a router module in the form of a route switch module (RSM), making it into a layer 2 switch and a layer 3 router in a single unit.

Several Catalyst capabilities were demonstrated in the labs, including MAC port security, IP permit lists, routing between multiple VLANS, and ISL trunking.

Chapter 21: Loading the IOS Image on a Router

Overview

Topics Covered in This Chapter

- Cisco code load overview
- TFTP server configuration
- Cisco IOS naming conventions
- Loading IOS on a run from RAM router
- Loading IOS on a run from Flash router
- Loading IOS from a TFTP server
- Loading an IOS image from another router
- Troubleshooting TFTP transfers on a Cisco router

Introduction

This chapter will explain how to load an IOS image on to a Cisco router. We will examine the two types of memory platforms on Cisco routers, run from RAM, and run from Flash systems. Finally, we will show how to make a Cisco router into a TFTP server so that IOS code can be loaded directly from another router in your network.

Code Load Overview

All Cisco routers store their operating system, referred to as their Internetwork Operating System or IOS, in flash memory located on the router. Anytime a new version of the IOS needs to be loaded on the router, the flash memory will need to be upgraded with the new code. Cisco's primary method of loading code on the router is to load it via TFTP. TFTP is an anonymous (no password required) file transfer protocol that uses UDP for its transport layer. The router that needs the new code requests it from a TFTP server. A TFTP server is usually a PC or workstation running a TFTP daemon.

The TFTP server software used in this chapter is Exceed by Hummingbird Communications. Exceed includes many powerful TCP/IP programs, such as a TFTP server and an FTP server. Exceed is configured by first enabling the TFTP server service as shown in Figure 21−1.

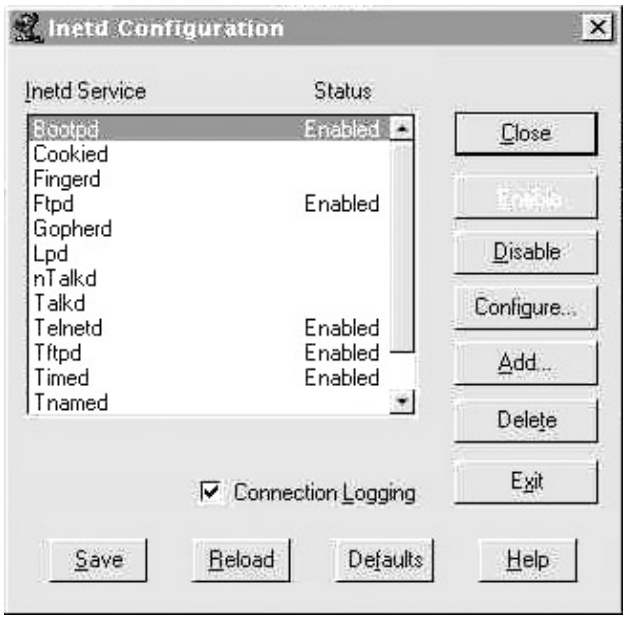


Figure 21−1: Enabling the TFTP server service
The TFTP download and upload directories are then defined. As shown in Figure 21−2, TFTP read and write operations will be done from a directory called download. Notice from Figure 21−2 that TFTP uses UDP port 69. Our PC has now been configured to act as a TFTP daemon.

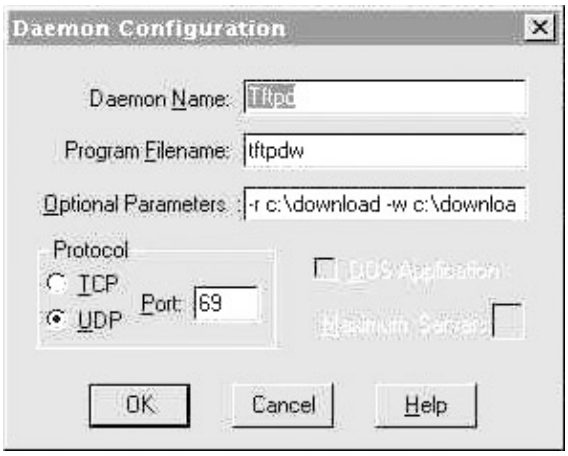


Figure 21−2: TFTP uses UDP port 69
As shown in Figure 21−3, there are four IOS images in the download directory of our workstation. During the labs in this chapter, our Cisco routers will be loading IOS images from this directory using TFTP.

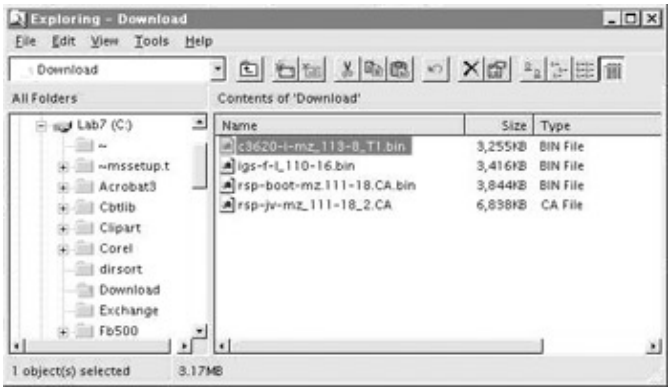


Figure 21−3: IOS images in the download directory
A Cisco router also has the ability to act as a TFTP server. This feature eliminates the need for a PC or workstation on your network that runs a TFTP server program.

Code Load Naming Conventions

Cisco IOS images adhere to a well−defined naming convention. Cisco maintains an online document on their Web site titled, "Software Naming Conventions for IOS." The naming conventions let you interpret the meaning of the characters in the filename of an IOS image. As an example, let's look at the IOS filenames for two of the IOS images we will be using during this chapter.

The IOS code filename for the Cisco 3620 is: c3620−i−mz_113−8_T1.bin. This filename can be interpreted as follows:

```
Hardware Platform is a Cisco 3620
↓       IP Subset Version
↓        ↓      Run from RAM                Platform Specific
↓        ↓      ↓                           ↓
c3620  − i−     m  z_           113−8_      T1.bin
                   á              á
                Zipped            IOS 11.3(8)
```

We see that this file is an IOS image for a Cisco 3620 router. It is an IP subset code load that is compressed and is designed to run from RAM. The IOS version is 11.3(8).

The IOS code filename for the Cisco 2500 is: igs−g−L_111−24.bin. This filename can be interpreted as follows:

```
Hardware Platform is a Cisco 2500 Series Router
↓          ISDN Subset Version
↓          ↓      Relocatable Code
↓          ↓      ↓
igs       −g−     L_           111−24.bin
                                 á
                                IOS 11.1(24)
```

We see that this file is an IOS image for a Cisco 2500 router. It is an ISDN and IP code load that is relocatable. The IOS version is 11.1(24).

Following are some more detailed descriptions of the IOS naming conventions:

- An IOS image name has three parts, each part is separated by dashes: e.g., aaaa−bbbb−cc, where:
    - ♦ aaaa = Platform
    - ♦ bbbbb = Feature sets
    - ♦ cc = Where the IOS image executes from and if the IOS image is compressed

Platform

The first part of the image name specifies what platform it runs on.

```
as5200     5200
c1600      1600
c2500      25xx, 3xxx, 5100, AP (11.2 and later only)
c25FX      Fixed Frad platform
c3620      3620
c3640      3640
c3800      3800
c4000      4000  (11.2 and later only)
c4500      4500, 4700
c7000      7000, 7010  (11.2 and later only)
c7200      7200
igs        IGS, 25xx, 3xxx, 5100, AP
```

Feature Sets

The following capabilities are defined.

```
    a − APPN
    a2 − ATM
    b − Appletalk
    boot − used for boot images
    c − Comm−server/Remote Access Server (RAS) subset (SNMP, IP, Bridging,
  IPX, Atalk, Decnet, FR, HDLC, PPP, X,25, ARAP, tn3270, PT,
  XRemote, LAT) (non−CiscoPro)
    c − CommServer lite (CiscoPro)
 c2 − Comm−server/Remote Access Server (RAS) subset (SNMP, IP, Bridging,
  IPX, Atalk, Decnet, FR, HDLC, PPP, X,25, ARAP, tn3270, PT,
  XRemote, LAT) (CiscoPro)
    d − Desktop subset (SNMP, IP, Bridging, WAN, Remote Node, Terminal
        Services, IPX, Atalk, ARAP)
  (11.2 − Decnet)
    d2 − reduced Desktop subset(SNMP, IP, IPX, ATALK, ARAP)
    diag − IOS based diagnostics images
    e − IPeXchange (no longer used in 11.3 and later)
      − StarPipes DB2 Access − Enables Cisco IOS to act as a "Gateway" to
        all IBM DB2 products for downstream clients/servers in 11.3T
```

eboot − ethernet boot image for mc3810 platform
f − FRAD subset (SNMP, FR, PPP, SDLLC, STUN)
f2 − modified FRAD subset, EIGRP, Pcbus, Lan Mgr removed, OSPF added
g − ISDN subset (SNMP, IP, Bridging, ISDN, PPP, IPX, Atalk)
g2 − gatekeeper proxy, voice and video
h − For Malibu(2910), 8021D, switch functions, IP Host
hdiag − Diagnostics image for Malibu(2910)
i − IP subset (SNMP, IP, Bridging, WAN, Remote Node, Terminal Services)
i2 − subset similar to IP subset for system controller image (3600)
i3 − reduced IP subset with BGP/MIB, EGP/MIB, NHRP, DIRRESP removed.
j − enterprise subset (formerly bpx, includes protocol translation)
******* not used until 10.3 ***
k − kitchen sink (enterprise for high−end) (Not used after 10.3)
k2 − high−end enterprise w/CIP2 ucode (Not used after 10.3)
k1 − Baseline Privacy key encryption (On 11.3 and up)
k2 − Triple DES (On 11.3 and up)
k3 − Reserved for future encryption capabilities (On 11.3 and up)
k4 − Reserved for future encryption capabilities (On 11.3 and up)
k5 − Reserved for future encryption capabilities (On 11.3 and up)
k6 − Reserved for future encryption capabilities (On 11.3 and up)
k7 − Reserved for future encryption capabilities (On 11.3 and up)
k8 − Reserved for future encryption capabilities (On 11.3 and up)
k9 − Reserved for future encryption capabilities (On 11.3 and up)
l − IPeXchange IPX, static routing, gateway
m − RMON (11.1 only)
n − IPX
o − Firewall (formerly IPeXchange Net Management)
p − Service Provider (IP RIP/IGRP/EIGRP/OSPF/BGP, CLNS ISIS/IGRP)
p2 − Service Provider w/CIP2 ucode
p3 − as5200 service provider
p4 − 5800 (Nitro) service provider
q − Async
q2 − IPeXchange Async
r − IBM base option (SRB, SDLLC, STUN, DLSW, QLLC) − used with **i,** in, d (See note below.)
r2 − IBM variant for 1600 images
r3 − IBM variant for Ardent images (3810)
r4 − reduced IBM subset with BSC/MIB, BSTUN/MIB, ASPP/MIB, RSRB/MIB removed.
s − source route switch (SNMP, IP, Bridging, SRB) (10.2 and following)
s − (11.2 only) additions to the basic subset:
c1000 − (OSPF, PIM, SMRP, NLSP, ATIP, ATAURP, FRSVC, RSVP, NAT)
c1005 − (X.25, full WAN, OSPF, PIM, NLSP, SMRP, ATIP, ATAURP, FRSVC, RSVP, NAT)
c1600 − (OSPF, IPMULTICAST, NHRP, NTP, NAT, RSVP, FRAME_RELAY_SVC)
AT "s" images also have: (SMRP,ATIP,AURP)
IPX "s" images also have: (NLSP,NHRP)
c2500 − (NAT, RMON, IBM, MMP, VPDN/L2F)
c2600 − (NAT, IBM, MMP, VPDN/L2F, VOIP and ATM)
c3620 − (NAT, IBM, MMP, VPDN/L2F) In 11.3T added VOIP
c3640 − (NAT, IBM, MMP, VPDN/L2F) In 11.3T added VOIP
c4000 − (NAT, IBM, MMP, VPDN/L2F)
c4500 − (NAT, ISL, LANE, IBM, MMP, VPDN/L2F)
c5200 − (PT, v.120, managed modems, RMON, MMP, VPDN/L2F)
c5300 − (MMP, VPDN, NAT, Modem Management, RMON, IBM)
c5rsm − (NAT, LANE and VLANS)
c7000 − (ISL, LANE, IBM, MMP, VPDN/L2F)
c7200 − (NAT, ISL, IBM, MMP, VPDN/L2F)
rsp − (NAT, ISL, LANE, IBM, MMP, VPDN/L2F)
t − (11.2) AIP w/ modified Ucode to connect to Teralink 1000 Data
u − IP with VLAN RIP (Network Layer 3 Switching Software, rsrb, srt, srb, sr/tlb)
v − VIP and dual RSP (HSA) support
v2 − Voice V2D
w − Reserved for WBU (remaining characters are specific to WBU)
i − IISP
l − LANE & PVC
p − PNNI
v − PVC traffic shaping

```
w2 − Reserved for CiscoAdvantage ED train (remaining characters are
specific to CiscoAdvantage)
       a − IPX, static routing, gateway
       b − Net Management
       c − FR/X.25
       y − Async
w3 − Reserved for Distributed Director
x − X.25 in 11.1 and earlier releases.  FR/X.25 in 11.2 (IPeXchange)
     H.323 Gatekeeper/Proxy in 11.3 releases for 2500, 3620, 3640
y − reduced IP (SNMP, IP RIP/IGRP/EIGRP, Bridging, ISDN, PPP) (C1003/4 )
  − reduced IP (SNMP, IP RIP/IGRP/EIGRP, Bridging, WAN − X.25) (C1005)
(11.2 − includes X.25) (c1005)
y − IP variant (no Kerberos, Radius, NTP, OSPF, PIM, SMRP, NHRP...)
(c1600)
y2 − IP variant (SNMP, IP RIP/IGRP/EIGRP, WAN − X.25, OSPF, PIM)
(C1005)
y2 − IP Plus variant (no Kerberos, Radius, NTP,...) (c1600)
y3 − IP/X.31
y4 − reduced IP variant (Cable, Mibs, DHCP, EZHTTP)
z − managed modems
40 − 40−bit encryption
56 − 56−bit encryption
56i − 56−bit encryption with IPSEC
```

Where the IOS Image Runs From

```
f − flash
m − RAM
r − ROM
l − relocatable
```

The following may be added if the image has been 'zip' compressed:

```
z − zip compressed (note lowercase)
```

Run from RAM and Run from Flash Routers

A Cisco router executes its IOS from either RAM or flash memory. Executing from flash memory is slower.

Run from flash routers are units such as the Cisco 2500 series and some of the Cisco 1600 series routers. The entire IOS is loaded into the flash memory in an uncompressed format. The Cisco IOS runs from the flash memory. Upgrading the IOS becomes an issue. How can you load new code into flash memory that is currently executing the IOS? Cisco addresses this problem by having a special IOS located in a ROM on the router. A boot helper program reloads the router from the boot ROM. The flash can then be upgraded and the new IOS image can be run from flash. Most run from flash routers are able to have dual banks of flash, which will permit an IOS file to be downloaded into one bank of flash at the same time that an IOS image is running out of the second bank of flash.

Run from RAM routers are units such as the Cisco 3600, 4000, 7000, and 7500 series. These routers store a compressed IOS image in flash. When booting, the router copies the IOS from flash into RAM and executes the IOS out of RAM. These run from RAM routers have their IOS upgraded by copying a new file to flash. Since flash is not being used to execute the IOS image, you can simply TFTP the new IOS image to the router's flash.

Commands Discussed in This Chapter

- **copy tftp flash**
- **debug tftp**

- **show flash [all | chips | detailed | err | partition** *number* **[all | chips | detailed | err] | summary]**
- **show version**
- **tftp server flash [***partition−number*:] *filename*

Definitions

copy tftp flash: This exec command copies a file from a TFTP server to the contents of flash memory on the router.

debug tftp: This debug command provides output showing any TFTP transactions that occur on the router.

show flash: This exec command displays the contents of flash memory.

show version: This exec command displays router information such as system configuration, IOS level, and the names and sources of configuration files.

tftp server: This global command specifies that the router should act as a TFTP server for the file specified in the command.

IOS Requirements

The copy TFTP command has been available since IOS 10.0. Other features, such as the capability to make the Cisco router into a TFTP server, have only been available since IOS 11.0.

Lab #94: Loading an IOS Image from a TFTP Server to a Run from RAM Router

Equipment Needed

The following equipment is needed to perform this lab exercise

- One Cisco router with an Ethernet interface.
- A PC running a TFTP server program with an Ethernet card. The PC should also have a terminal emulation program such as Procomm or Hyperterm.
- A Cisco rolled cable for console port connection to the router.
- An Ethernet hub with two Ethernet cables.

Configuration Overview

This lab will take a Cisco router that is running IOS 11.2(7) and upgrade it to IOS 11.3(8). This configuration will demonstrate how to load an IOS image on a Cisco router that utilizes a run from RAM architecture. Examples of run from RAM routers are the Cisco 3600 series, the Cisco 4000 series, and the Cisco 7000 series.

A PC running TFTP server software will be connected to the same LAN as a Cisco router. The software used in this lab is Exceed from Hummingbird Communications. The Exceed software package contains many TCP/IP programs, such as a TFTP server, an FTP server, and an X Window server. The new version of the IOS image will reside on the PC and will be transferred to the Cisco router using the TFTP transfer protocol. The PC will be acting as the TFTP server, and the Cisco router will be the TFTP client.

RouterA and the PC are connected as shown in Figure 21−4.

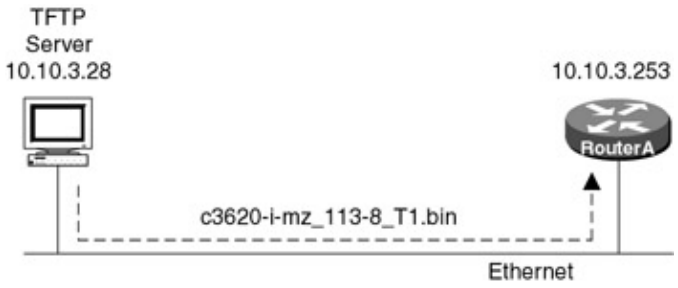


Figure 21−4: Connection between RouterA and the TFTP Server

Router Configuration

The configuration for the router in this example is as follows.

RouterA

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
!
!
interface Ethernet0/0
 ip address 10.10.3.253 255.255.255.0 ← The Ethernet interface is on the same
                                        network as the TFTP server
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterA will be loading a new IOS image from a TFTP server. RouterA's configuration does not need any special commands to load the IOS image. The only item that needs to be configured on RouterA is the Ethernet interface.

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. Use the **show version** command to find out what version of IOS the router is currently running. We see that the router is running a version of 11.2. The **show version** command also tells us other key information about the router's software image and memory capabilities. We see that the router has 16MB of DRAM. The DRAM is used to run the IOS on a run from RAM routers, such as the Cisco 3620 that we are using in this lab. We also see that this router has 16MB of flash memory. The flash memory stores one or more IOS images. The **show version** output also tells us that the currently running IOS was loaded from flash memory. Finally, we see that our router platform is a 3620 router.

```
RouterA#show version                Router is running IOS version 11.2(7a)P
Cisco Internetwork Operating System Software  ↓
IOS (tm) 3600 Software (C3620-I-M), Version 11.2(7a)P, SHARED
PLATFORM, RELEASE
SOFTWARE (fc1)
Copyright (c) 1986-1997 by cisco Systems, Inc.
Compiled Wed 02-Jul-97 08:25 by ccai
Image text-base: 0x600088E0, data-base: 0x60440000
```

```
ROM: System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY DEPLOYMENT
RELEASE SOFTWARE (fc2)

RouterA uptime is 54 minutes     The IOS was loaded from flash memory
System restarted by reload       ↓
System image file is "flash:c3620−i−mz.112−7a.P", booted via flash

This router is a Cisco 3620
↓
cisco 3620 (R4700) processor (revision 0x81) with 12288K/4096K
bytes of memory.
Processor board ID 05706232                          á
R4700 processor, Implementation 33, Revision 1.0   The router has 16MB of DRAM.
                                                   The DRAM is broken up into
                                                   12MB of main memory, used for
                                                    processing, and 4MB of shared
                                                    memory user for I/O
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
Basic Rate ISDN software, Version 1.0.
1 Ethernet/IEEE 802.3 interface(s)
1 Serial network interface(s)
1 ISDN Basic Rate interface(s)
DRAM configuration is 32 bits wide with parity disabled.
29K bytes of non−volatile configuration memory.
16384K bytes of processor board System flash (Read/Write) ← The router has 16MB
                                                             of flash memory

Configuration register is 0x2102
```

Typeconnecting to RouterA. Use the the **show flash** command to view the contents of the flash memory on the router. We see that the flash memory contains a single file, c3620−i−mz.112−7a.P. The size of the file is 2259976 bytes. The flash memory is 16MB in size.

```
RouterA#show flash

System flash directory:
File  Length   Name/status
  1   2259976  c3620−i−mz.112−7a.P ← There is only a single file in flash
                                     memory
[2260040 bytes used, 14517176 available, 16777216 total]
16384K bytes of processor board System flash (Read/Write)
á
16MB of flash memory on this router
```

Let's make sure that we can reach our TFTP server at IP address 10.10.3.28 by using a ping command.

```
RouterA#ping 10.10.3.28

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 10.10.3.28, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 1/3/8 ms
```

Once connecting to RouterA. Use thewe are sure we can reach the TFTP server, we can start loading the new IOS image to the router. Use the **copy tftp flash** command to start a TFTP transfer from the PC to the flash memory of RouterA. Notice that we will specify not to erase the current file that resides in the flash memory of the router.

```
RouterA#copy tftp flash

System flash directory:
File  Length   Name/status
  1   2259976  c3620−i−mz.112−7a.P
[2260040 bytes used, 14517176 available, 16777216 total]
```

```
Address or name of remote host [10.10.3.28]? ← Address of TFTP server
Source file name? c3620-i-mz_113-8_T1.bin ← Name of IOS image we want to load
Destination file name [c3620-i-mz_113-8_T1.bin]?
Accessing file 'c3620-i-mz_113-8_T1.bin' on 10.10.3.28...
Loading c3620-i-mz_113-8_T1.bin from 10.10.3.28 (via Ethernet0/0): ! [OK]

Erase flash device before writing? [confirm]n ← Do not erase the current file
                                                 in the router's flash memory
Copy 'c3620-i-mz_113-8_T1.bin' from server
  as 'c3620-i-mz_113-8_T1.bin' into Flash WITHOUT erase? [yes/no]y
Loading c3620-i-mz_113-8_T1.bin from 10.10.3.28 (via Ethernet0/0): !!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                  á
              An O means that a TFTP packet was received out of order

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 3332232/14517176  bytes]

Verifying checksum . . .   OK (0x1837)
Flash device copy took 00:00:35 [hh:mm:ss]
```

After the file download is complete, check the contents of the router's flash memory with the **show flash** command. We see that there are now two files in the flash memory of the router.

```
RouterA#show flash

System flash directory:
File  Length   Name/status
  1   2259976  c3620-i-mz.112-7a.P
  2   3332232  c3620-i-mz_113-8_T1.bin ← New file that we just loaded
[5592336 bytes used, 11184880 available, 16777216 total]
16384K bytes of processor board System flash (Read/Write)
```

Sinceconnecting to RouterA. Use the there are two files in the flash memory, we need to tell the router which file to load during its power on sequence. Enter router configuration mode with the **config term** command. Enter the **boot system flash** command shown next.

```
RouterA#config term
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#boot system flash c3620-i-mz_113-8_T1.bin
RouterA(config)#exit
```

You can verify that this command has been properly entered with the **show run** command.

```
RouterA#show run
Building configuration...

Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
boot system flash c3620-i-mz_113-8_T1.bin ← The router will load this file from
                                            flash memory during its power on
                                            sequence
```

```
!
interface Ethernet0/0
 ip address 10.10.3.253 255.255.255.0
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

The connecting to RouterA. Use theconfiguration changes must be written with a **write mem** command, since we have to reload the router.

```
RouterA#write mem
Building configuration...
[OK]

RouterA#reload
Proceed with reload? [confirm]
```

After the router reloads, it will be running IOS version 11.3(8)T1. We see that this file has been loaded from router flash.

```
RouterA#show ver
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-I-M), Version 11.3(8)T1,
RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Thu 11-Feb-99 17:22 by ccai
Image text-base: 0x60008918, data-base: 0x605B8000

ROM: System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY
DEPLOYMENT RELEASE SOFTWARE (fc2)

RouterA uptime is 5 minutes
System restarted by reload
System image file is "flash:c3620-i-mz_113-8_T1.bin",
booted via flash

cisco 3620 (R4700) processor (revision 0x81) with 12288K/4096K
bytes of memory.
Processor board ID 05706232
R4700 processor, Implementation 33, Revision 1.0
Bridging software.
X.25 software, Version 3.0.0.
Basic Rate ISDN software, Version 1.1.
1 Ethernet/IEEE 802.3 interface(s)
1 Serial network interface(s)
1 ISDN Basic Rate interface(s)
DRAM configuration is 32 bits wide with parity disabled.
29K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)

Configuration register is 0x2102
```

As an alternative, you can also load an IOS image to the router and erase the contents of the router's flash memory. We see an example here where there are two files in the flash memoryconnecting to RouterA. Use the of the router.

```
RouterA#show flash

System flash directory:
```

```
File  Length   Name/status
  1   2259976  c3620-i-mz.112-7a.P
  2   3332232  c3620-i-mz_113-8_T1.bin
[5592336 bytes used, 11184880 available, 16777216 total]
16384K bytes of processor board System flash (Read/Write)
```

If you want to load a new IOS image without keeping the old image, use the **copy tftp flash** command and allow the flash device to be erased before writing.

```
RouterA#copy tftp flash

System flash directory:
File  Length   Name/status
  1   2259976  c3620-i-mz.112-7a.P
  2   3332232  c3620-i-mz_113-8_T1.bin
[5592336 bytes used, 11184880 available, 16777216 total]
Address or name of remote host [10.10.3.28]? 10.10.3.28
Source file name? c3620-i-mz_113-8_T1.bin
Destination file name [c3620-i-mz_113-8_T1.bin]?
Accessing file 'c3620-i-mz_113-8_T1.bin' on 10.10.3.28 . . .
Loading c3620-i-mz_113-8_T1.bin from 10.10.3.28 (via Ethernet0/0): ! [OK]

Erase flash device before writing? [confirm] ← Pressing enter at this prompt
                                                will cause the flash to be erased before
                                                writing a new file
Flash contains files. Are you sure you want to erase? [confirm]

Copy 'c3620-i-mz_113-8_T1.bin' from server
  as 'c3620-i-mz_113-8_T1.bin' into Flash WITH erase? [yes/no]y
Erasing device ...  eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeee ... erased
á
The flash is being erased

Loading c3620-i-mz_113-8_T1.bin from 10.10.3.28 (via Ethernet0/0):
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!

                             á
                      An O means that a TFTP packet was received out of order
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!O!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!
O!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!
!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!
!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!O!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!O!!
[OK - 3332232/16777216  bytes]

Verifying checksum... OK (0x1837)
Flash device copy took 00:00:34 [hh:mm:ss]
```

After connecting to RouterA. Use thethe IOS download is complete, we see that there is only one file in the flash device, since we allowed the router to erase the flash before starting the download.

```
RouterA#sh flash

System flash directory:
File  Length   Name/status
  1   3332232  c3620-i-mz_113-8_T1.bin
[3332296 bytes used, 13444920 available, 16777216 total]
16384K bytes of processor board System flash (Read/Write)
```

Lab #95: Loading an IOS Image from a TFTP Server to a Run from Flash Router

Equipment Needed

Theconnecting to RouterA. Use the following equipment is needed to perform this lab exercise:

- One Cisco router with an Ethernet interface.
- A PC running a TFTP server program with an Ethernet card. The PC should also have a terminal emulation program such as Procomm or Hyperterm.
- A Cisco rolled cable for console port connection to the router.
- An Ethernet Hub with two Ethernet cables.

Configuration Overview

This configuration will demonstrate how to load an IOS image on a Cisco router that utilizes a run from flash architecture. Examples of run from flash routers are the Cisco 2500 series and some of the Cisco 1600 series.

A PC running TFTP server software will be connected to the same LAN as a Cisco router. The software used in this lab is Exceed from Hummingbird Communications. The Exceed software package contains many TCP/IP programs such as a TFTP server, an FTP server, and an X Windows server. The new version of the IOS image will reside on the PC and will be transferred to the Cisco router using the TFTP transfer protocol. The PC will be acting as the TFTP server, and the Cisco router will be the TFTP client.

RouterC and the PC are connected as shown in Figure 21−5.

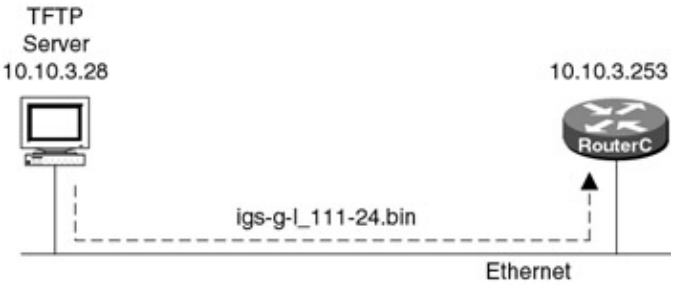


Figure 21−5: Connection between RouterC and the TFTP Server

Router Configuration

The configuration for the router in this example is as follows.

RouterC

```
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RouterC
!
!
interface Ethernet0
ip address 10.10.3.253 255.255.255.0 ← The Ethernet interface is on the same
                                       network as the TFTP server
!
no ip classless
!
line con 0
```

```
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start byconnecting to RouterC. Use the **show version** command to find out what version of IOS the router is currently running. We see that the router is running a version of 11.1. The **show version** command also tells us other key information about the router's software image and memory capabilities. We see that the router has 2MB of DRAM. We also see that this router has 8MB of flash memory. The flash memory stores one or more IOS images. The **show version** output also tells us that the currently running IOS was loaded from flash memory. Finally, we see that our router platform is a 2524 router.

```
RouterC#sh ver                          Router is running IOS version 11.1(4)
Cisco Internetwork Operating System Software  ↓
IOS (tm) 3000 Software (IGS-I-L), Version 11.1(4), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1996 by cisco Systems, Inc.
Compiled Mon 17-Jun-96 15:45 by mkamson
Image text-base: 0x0301F2B4, data-base: 0x00001000

ROM: System Bootstrap, Version 11.0(5), SOFTWARE
ROM: 3000 Bootstrap Software (IGS-BOOT-R), Version 11.0(5), RELEASE SOFTWARE (fc1)

RouterC uptime is 8 minutes       The IOS was loaded from flash memory
System restarted by reload        ↓
System image file is "flash:igs-i-l.111-4", booted via flash

This router is a Cisco 2524
  ↓
cisco 2524 (68030) processor (revision B) with 1024K/1024K bytes of memory.
                                                 á
                  The router has 2MB of DRAM. 1MB is used for processor memory and
1MB is used for I/O memory

Processor board ID 03879418, with hardware revision 00000000
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
Basic Rate ISDN software, Version 1.0.
1 Ethernet/IEEE 802.3 interface.
2 Serial network interfaces.
1 ISDN Basic Rate interface.
5-in-1 module for Serial Interface 0
56k 4-wire CSU/DSU for Serial Interface 1
Integrated NT1 for ISDN Basic Rate interface
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY) ← The router has 8MB of
                                                           flash memory


Configuration register is 0x2102
```

Display the contents of the router's flash memory using the **show flash** command. We see that the flash contains a single file.

```
RouterC#show flash

System flash directory:
File  Length   Name/status
  1   3747048  igs-i-l.111-4
[3747112 bytes used, 4641496 available, 8388608 total]
8192K bytes of processor board System flash (Read ONLY)
```

Let's make sure that we can reach our TFTP server at IP address 10.10.3.28 by using a ping command.

```
RouterA#ping 10.10.3.28

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.3.28, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/8 ms
```

Once we verify that we can ping our TFTP server, we can start to download the new IOS image to the router. The Cisco 2524 is a run from flash router. This means that the router's IOS image executes out of the same flash memory that the IOS image resides in. Loading a new IOS to the router is a bit more complex than loading a new IOS image on a router that runs the IOS from RAM. The router will reload itself and load a small IOS image out of its ROM memory. It will then load the new IOS image into flash memory. After the new IOS image is loaded, the router will reload the new image out of the flash memory.

```
RouterC#copy tftp flash
                        ****  NOTICE  ****
Flash load helper v1.0
This process will accept the copy options and then terminate
the current system image to use the ROM based image for the copy. ← The router
                                                                    will load a
                                                                    special ROM
                                                                    based IOS
                                                                    image which
                                                                    will write
                                                                    the new IOS
                                                                    to flash
                                                                    memory
Routing functionality will not be available during that time.
If you are logged in via telnet, this connection will terminate.
Users with console access can see the results of the copy operation.
                       ____ ******** ____
Proceed? [confirm]

System flash directory:
File  Length   Name/status
  1   3747048  igs-i-l.111-4
[3747112 bytes used, 4641496 available, 8388608 total]
Address or name of remote host [255.255.255.255]? 10.10.3.28 ← TFTP server
                                                               address
Source file name? igs-g-l_111-24.bin
Destination file name [igs-g-l_111-24.bin]?
Accessing file 'igs-g-l_111-24.bin' on 10.10.3.28 . . .
Loading igs-g-l_111-24.bin from 10.10.3.28 (via Ethernet0): ! [OK]

Erase flash device before writing? [confirm]
Flash contains files. Are you sure you want to erase? [confirm] ← Erase the
                                                                   current flash
                                                                   contents

System configuration has been modified. Save? [yes/no]: y
Building configuration . . .
[OK]

Copy 'igs-g-l_111-24.bin' from server
  as 'igs-g-l_111-24.bin' into Flash WITH erase? [yes/no]y

%SYS-5-RELOAD: Reload requested
SERVICE_MODULE(1): self test finished: Passed
%SYS-4-CONFIG_NEWER: Configurations from version 11.1 may not be correctly understood.
%FLH: igs-g-l_111-24.bin from 10.10.3.28 to flash  . . .

System flash directory:
File  Length   Name/status
  1   3747048  igs-i-l.111-4
```

```
[3747112 bytes used, 4641496 available, 8388608 total]
Accessing file 'igs-g-l_111-24.bin' on 10.10.3.28...
Loading igs-g-l_111-24.bin .from 10.10.3.28 (via Ethernet0): ! [OK]

Erasing device... eeeeeeeeeeeeeeeeeeeeeeeeeeeeee ...erased ← The router is
                                                              erasing the
                                                              flash memory
Loading igs-g-l_111-24.bin from 10.10.3.28 (via Ethernet0): !!!!!!!!!!O!!!!!!
!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!O!!!
                                    á
                     An O means that a TFTP packet was received out of order

!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!O!!!!!!!!!!!!!!!!!O!!!!!!!!
O!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!
O!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!O!!!!!!!!!!!!!!
!!!!!!!!!!O!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!O!!!!!!!!!!!!!
!!!!!!!!!!O!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!O!!!!!!!!!!!!!
!!!!!!!!!!!O!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!
!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!O!!!!!!!!!!!!!!!!!O!!!!!!!!!!!O!
!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!O!
!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!O!!!!!!!!!!!!!!!!!
!!!!!!!!O!!!!!!!!!!!!O!!!!!!!!!!!!!!!!!!!!!!!!!!O!!!!!!!!!!!!O!!!!!!!!!!!!!
[OK - 3735976/8388608  bytes]

Verifying checksum... OK (0xB84C)
Flash copy took 0:02:11 [hh:mm:ss]
%FLH: Re-booting system after download ← The router will reload itself before
                                          booting the new IOS image

             Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

           cisco Systems, Inc.
           170 West Tasman Drive
           San Jose, California 95134-1706



Cisco Internetwork Operating System Software
IOS (tm) 3000 Software (IGS-G-L), Version 11.1(24), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Mon 04-Jan-99 19:14 by richv
Image text-base: 0x0301F310, data-base: 0x00001000

cisco 2524 (68030) processor (revision B) with 1024K/1024K bytes of memory.
Processor board ID 03879418, with hardware revision 00000000
Bridging software.
Basic Rate ISDN software, Version 1.0.
1 Ethernet/IEEE 802.3 interface.
1 ISDN Basic Rate interface.
Integrated NT1 for ISDN Basic Rate interface
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)
```

After the IOS reloads, use the **show version** command to verify that the router is running the new system image. We see that the router is now running IOS 11.1(24).

```
RouterC#show version
Cisco Internetwork Operating System Software
IOS (tm) 3000 Software (IGS-G-L), Version 11.1(24), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
```

```
Compiled Mon 04-Jan-99 19:14 by richv
Image text-base: 0x0301F310, data-base: 0x00001000

ROM: System Bootstrap, Version 11.0(5), SOFTWARE
ROM: 3000 Bootstrap Software (IGS-BOOT-R), Version 11.0(5), RELEASE SOFTWARE (fc1)

RouterC uptime is 0 minutes
System restarted by reload
System image file is "flash:igs-g-l_111-24.bin", booted via flash

cisco 2524 (68030) processor (revision B) with 1024K/1024K bytes of memory.
Processor board ID 03879418, with hardware revision 00000000
Bridging software.
Basic Rate ISDN software, Version 1.0.
1 Ethernet/IEEE 802.3 interface.
1 ISDN Basic Rate interface.
Integrated NT1 for ISDN Basic Rate interface
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)

Configuration register is 0x2102
```

Use the **show flash** command to verify that we have the correct file in our router's flash.

```
RouterC#show flash

System flash directory:
File  Length   Name/status
  1   3735976  igs-g-l_111-24.bin
[3736040 bytes used, 4652568 available, 8388608 total]
8192K bytes of processor board System flash (Read ONLY)
```

Lab #96: Loading an IOS Image from Another Router

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each with a single serial interface.
- A Cisco V.35 crossover cable. If no crossover cable is available, you can use a Cisco DCE cable connected to a Cisco DTE cable.
- A Cisco rolled cable for console port connection to the router.

Configuration Overview

This configuration will demonstrate how a Cisco router can act as a TFTP server. This is a powerful capability of the router. Recall from the two previous labs that we needed to have a TFTP server software package running on a PC in order to load an IOS image on the router. With the TFTP server capability built into the router, we can load an IOS image from any router in our network from which we have IP connectivity.

RouterA and RouterB will be connected as shown in Figure 21-6. RouterB will act as a DCE, supplying clock to RouterA.

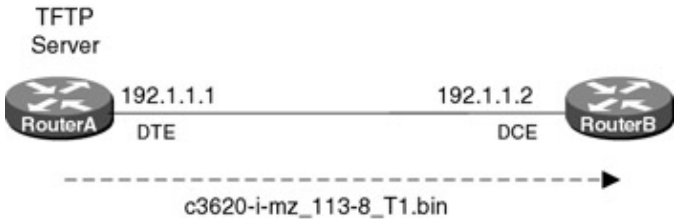


Figure 21−6: Connection between RouterA and RouterB
RouterA will be configured to be a TFTP server. RouterB will be the TFTP client. RouterB will request the file c3620−i−mz_113−8_T1.bin from RouterA.

Router Configuration

The configurations for the two routers in this example are as follows.

RouterA (TFTP Server)

```
Current configuration:
!
version 11.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
boot system flash c3620-i-mz_113-8_T1.bin
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
 encapsulation ppp
!
no ip classless
!
tftp-server flash c3620-i-mz_113-8_T1.bin ← RouterA is acting as a TFTP server.
                                            It will only accept requests for
                                            the file c3620-i-mz_113-8_T1.bin
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB (TFTP Client)

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 encapsulation ppp
 no fair-queue
```

```
 clockrate 64000 ← RouterB acts as a DCE supplying a clock to RouterA
!
no ip classless
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

The **tftp−server flash c3620−i−mz_113−8_T1.bin** in the configuration of RouterA defines RouterA to be a TFTP server. The command will allow requests for the file **c3620−i−mz_113−8_T1.bin** (IOS version 11.3(8)) to be retrieved from the flash memory of RouterA. Let's check the contents of the flash on RouterA to make sure that the correct file is there. Use the **show flash** command to view the contents of RouterA's flash memory. We see that the file is in the flash memory of RouterA.

```
RouterA#show flash

System flash directory:
File  Length   Name/status
  1   3332232  c3620−i−mz_113−8_T1.bin ← RouterA is configured so that only
                                         this file can be requested via TFTP
                                         out of its flash memory
[3332296 bytes used, 13444920 available, 16777216 total]
16384K bytes of processor board System flash (Read/Write)
```

Now let's connect to RouterB. Verify that we can reach RouterA by pinging RouterA at IP address 192.1.1.1.

```
RouterB#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round−trip min/avg/max = 28/30 /32 ms
```

Let's show the contents of the flash memory on RouterB. We see that RouterB has three IOS images in its flash memory, 11.2(7), 11.3(3), and 11.2(16).

```
RouterB#show flash

System flash directory:
File  Length   Name/status
  1   2259976  c3620−i−mz.112−7a.P ← 11.2(7)
  2   4568036  c3620−is−mz_113−3a_T.bin ← 11.3(3)
  3   2972356  c3620−d−mz_112−16_p.bin ← 11.2(16)
[9800560 bytes used, 6976656 available, 16777216 total]
16384K bytes of processor board System flash (Read/Write)
```

Now let's copy an IOS image from RouterA to RouterB. We will use the same command that we used in the previous two labs. The only difference here is that a Cisco router instead of a PC is acting as a TFTP server. Type the **copy tftp flash** command.

```
RouterB#copy tftp flash

System flash directory:
File  Length   Name/status
  1   2259976  c3620−i−mz.112−7a.P
  2   4568036  c3620−is−mz_113−3a_T.bin
  3   2972356  c3620−d−mz_112−16_p.bin
```

```
[9800560 bytes used, 6976656 available, 16777216 total]
Address or name of remote host [192.1.1.1]? 192.1.1.1 ← Address of RouterA (our
                                                        TFTP server)
Source file name? c3620-i-mz_113-8_T1.bin ← The IOS file that we want to load
Destination file name [c3620-i-mz_113-8_T1.bin]?
Accessing file 'c3620-i-mz_113-8_T1.bin' on 192.1.1.1...
Loading c3620-i-mz_113-8_T1.bin from 192.1.1.1 (via Serial0/0): ! [OK]

Erase flash device before writing? [confirm] ← We will erase the flash before
                                               loading our new image
Flash contains files. Are you sure you want to erase? [confirm]

Copy 'c3620-i-mz_113-8_T1.bin' from server
  as 'c3620-i-mz_113-8_T1.bin' into Flash WITH erase? [yes/no]y

Erasing device...             Flash memory is being erased
                              ↓
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee ...erased
Loading c3620-i-mz_113-8_T1.bin from 192.1.1.1 (via Serial0/0): !!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 3332232/16777216  bytes]

Verifying checksum.... OK (0x1837)
Flash device copy took 00:08:38 [hh:mm:ss]
```

After the TFTP code load is complete, use the **show flash** command to view the contents of the flash memory on RouterB. We see that the three original files have been erased and an IOS image of 11.3(8) is now in the router.

```
RouterB#show flash

System flash directory:
File  Length   Name/status
  1   3332232  c3620-i-mz_113-8_T1.bin
[3332296 bytes used, 13444920 available, 16777216 total]
16384K bytes of processor board System flash (Read/Write)
```

Reload the router with the **reload** command.

```
RouterB#reload
Proceed with reload? [confirm]
```

The router will reload with IOS version 11.3(8).

```
                                   The router will reload with IOS 11.3(8)
Cisco Internetwork Operating System Software   ↓
IOS (tm) 3600 Software (C3620-I-M), Version 11.3(8)T1,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Thu 11-Feb-99 17:22 by ccai
Image text-base: 0x60008918, data-base: 0x605B8000

cisco 3620 (R4700) processor (revision 0x81) with 12288K/4096K bytes of memory.
Processor board ID 05706229
R4700 processor, Implementation 33, Revision 1.0
Bridging software.
X.25 software, Version 3.0.0.
1 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
```

```
DRAM configuration is 32 bits wide with parity disabled.
29K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)
```

Troubleshooting TFTP Transferson a Cisco Router

{debug tftp} The Cisco IOS provides a command, **debug tftp**, that shows the status of TFTP transfers. The output that follows shows how TFTP sends an acknowledgment packet for every block of traffic sent.

```
RouterA#debug tftp
TFTP Packets debugging is on
RouterA#
02:25:06: TFTP: Sending block 216 (retry 0), socket_id 0x60A3F8E4 ← Block sent
02:25:06: TFTP: Received ACK for block 216, socket_id 0x60A3F8E4 ← Block sent
02:25:06: TFTP: Sending block 217 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 217, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 218 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 218, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 219 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 219, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 220 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 220, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 221 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 221, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 222 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 222, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 223 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 223, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 224 (retry 0), socket_id 0x60A3F8E4
02:25:06: TFTP: Received ACK for block 224, socket_id 0x60A3F8E4
02:25:06: TFTP: Sending block 225 (retry 0), socket_id 0x60A3F8E4
02:25:07: TFTP: Received ACK for block 225, socket_id 0x60A3F8E4
02:25:07: TFTP: Sending block 226 (retry 0), socket_id 0x60A3F8E4
02:25:07: TFTP: Received ACK for block 226, socket_id 0x60A3F8E4
```

{show flash} The show flash command displays all IOS images that are loaded in the flash memory of the router. We see that an image of IOS 11.3(8) is loaded in flash. The show flash command also displays how much total flash and available flash there is on the router.

```
RouterB#show flash

System flash directory:
File  Length   Name/status
  1   3332232  c3620-i-mz_113-8_T1.bin ← Single IOS image in
the router's flash
[3332296 bytes used, 13444920 available, 16777216 total] ← 16MB flash total,
                                                           3.3MB used, 13.4MB
                                                           available
16384K bytes of processor board System flash (Read/Write)
```

{show version} The **show version** command displays key information about the router's software image and memory capabilities.

```
RouterC#sh ver                          Router is running IOS version 11.1(4)
Cisco Internetwork Operating System Software  ↓
IOS (tm) 3000 Software (IGS-I-L), Version 11.1(4), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1996 by cisco Systems, Inc.
Compiled Mon 17-Jun-96 15:45 by mkamson
Image text-base: 0x0301F2B4, data-base: 0x00001000

ROM: System Bootstrap, Version 11.0(5), SOFTWARE
ROM: 3000 Bootstrap Software (IGS-BOOT-R), Version 11.0(5), RELEASE SOFTWARE (fc1)

RouterC uptime is 8 minutes       The IOS was loaded from flash memory
```

```
System restarted by reload        ↓
System image file is "flash:igs-i-l.111-4", booted via flash

This router is a Cisco 2524
  ↓
cisco 2524 (68030) processor (revision B) with 1024K/1024K bytes of memory.
                                                          á
      The router has 2MB of DRAM. 1MB is used for processor memory and 1MB is used
for I/O memory
```

Conclusion

The ability to load IOS images to a Cisco router via TFTP is a powerful feature. As we saw in this section, a Cisco router can also act as a TFTP server for a network of routers.

Chapter 22: Cisco Password Recovery

Overview

Topics Covered in This Chapter

- Cisco password recovery overview
- Understanding the Cisco configuration register
- Password recovery on a Cisco 3600
- Password recovery on a Cisco 2500
- Password recovery on a Catalyst Switch

Introduction

This chapter provides detailed information on recovering lost or unknown passwords on Cisco routers and Catalyst switches. Password recovery instructions will be given for the Cisco 2500, Cisco 3600, and Catalyst 5000 family of routers and switches.

Password Recovery Overview

A Cisco router goes through a predefined startup sequence. After power−on tests and loading of the IOS image, the router looks to NVRAM for its configuration instructions. These configuration instructions not only contain information on routing protocols and addressing, but they also contain information on the login passwords of the router.

Password recovery involves telling the router to ignore the contents of the NVRAM when the router goes through its startup sequence. This is done by modifying the router's configuration register, a 16−bit register located in the router's NVRAM. This causes the router to

load a blank configuration containing no login passwords. After logging into the router without any passwords, the user can then view the passwords in the NVRAM configuration and either use them, delete them, or change them. The router is then rebooted with known passwords.

Password recovery techniques vary by router family, but in general most observe the following format:

1. Connect a terminal to the console port of the router.
2. If the router is powered on, power it off and back on again. If the router is powered off, power it on.
3. While the router is booting, you must break into the boot sequence and put the router in monitor mode.
4. While in monitor mode, configure the router to boot up without reading the contents of NVRAM.
5. Reload the router.
6. After the router reloads without reading NVRAM, there will not be any privileged or nonprivileged passwords. Get into privileged mode and either view, change, or delete the NVRAM passwords.
7. Go into configuration mode and set the router to boot from NVRAM.
8. Reload the router. The passwords are now known.

Configuration Register

A Cisco router has a 16−bit register known as the virtual configuration register. This register resides in NVRAM. The register is used to set several basic features on the router, such as:

- Causing the router to ignore the contents of NVRAM

- Setting the console baud rate
- Setting the format of IP broadcast packets
- Causing the router to boot from ROM or Flash, or to use the startup configuration to determine the correct location for the router's IOS image

It is the first item in the previous list — causing the router to ignore the contents of NVRAM — that is used for password recovery.

The 16−bit value of the configuration register is always expressed in hexadecimal format. It is always written as 0xVALUE where VALUE is the register settings. We will see, for example, that a typical configuration register value is 0x2102.

Figure 22−1 shows the meaning of each bit position in the virtual configuration register for a Cisco 3600 router.

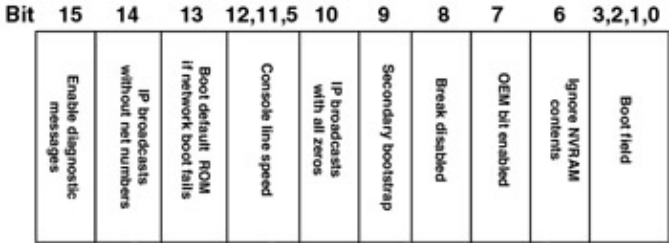


Figure 22−1: Cisco 3600 configuration register
Let's look at some of the key fields of the virtual configuration register and examine their possible values:

Bits 0−3 — Boot field

These four bits determine if the router will reload into ROM monitor mode, boot from the first image located in flash, or get its image loading instructions from the configuration located in NVRAM.

Bit 6 — NVRAM ignore

When bit 6 is set to a 1, the router will ignore the contents of NVRAM when it boots. This is the bit that we set when doing password recovery.

Bit 8 — Break disable

Setting this bit to a 1 causes the router to ignore the BREAK key.

Bits 5, 11, 12 — Console speed

These three bits determine the speed of the routers console. The 3600 console port defaults to 9,600 bps but can operate at speeds from 1,200 to 115,200 bps.

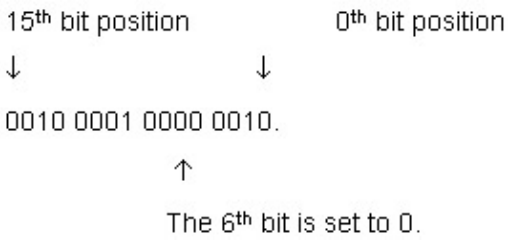
Interpreting the Configuration Register

Let's look at a typical configuration register value of 0x2102 and review how to convert this hexadecimal value to a binary value. Figure 22−2 contains a hexadecimal−to−binary conversion chart.

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Figure 22−2: Binary−to−hexadecimal conversion chart

Conversion from the hexadecimal value of 0x2102 to a binary value is a simple exercise. Each digit of the hexadecimal register value gets converted to four binary bits. The 0x2102 value should be converted one hexadecimal digit at a time. The first hexadecimal digit is a 2 and gets converted to a 0010. The second hexadecimal digit is a 1 and gets converted to 0001. The third hexadecimal digit is a 0 and gets converted to 0000. The last hexadecimal digit is a 2 and gets converted to 0010. After converting each individual hexadecimal digit, a 16−bit value can be created. The 16−bit value would be:



The bit numbers are counted so that the rightmost bit is the 0^{th} bit and the leftmost bit is the 15^{th} bit.

We see from this example that the 6^{th} bit is set to zero. This means that the contents of NVRAM will not be ignored when the router reboots.

Breaking the Normal Router Startup Sequence

The key to successfully recovering a lost or unknown password is being able to interrupt the normal startup sequence of the router and gain access to monitor mode. This is accomplished by issuing a break signal from your terminal emulator while the router is booting. The break sequence varies on different terminal emulators. The two most popular terminal emulators are Windows 95 Hyperterm and ProComm. The break sequence for ProComm is generated by pressing the ALT+B keys at the same time. In Windows 95 Hyperterm, the break sequence is generated by pressing the CTRL+BREAK keys at the same time.

Commands Discussed in This Chapter

- **show version**
- **show running−config**
- **show startup−config**
- **confreg**

- **reset**
- **config−register**
- **i**
- **o/r**
- **enable**
- **config term**
- **copy startup−config running config**
- **write erase**
- **reload**
- **set pass**
- **set enablepass**

Definitions

show version: An exec command that is used to show the system hardware, IOS version, configuration file, boot image, and contents of configuration register.

show running−config: An exec command that displays the contents of the currently executing configuration.

show startup−config: An exec command that shows the contents of the saved configuration stored in NVRAM.

confreg: A ROM monitor command used to view and change the contents of the configuration register.

reset: A ROM monitor command used to reload the router after changing the contents of the configuration register. This command is specific to certain Cisco models such as the 3600 series.

config−register: A global configuration command used to change the contents of the 16−bit configuration register.

i: A ROM monitor command used to reload a router after changing the contents of the configuration register. This command is specific to certain Cisco models such as the 2500 series.

o/r: A ROM monitor command used to change the contents of the configuration register. This command is specific to certain Cisco models such as the 2500 series.

enable: An exec command used to place a Cisco router or Catalyst switch into enabled mode.

config term: An exec command used to enter router configuration mode.

copy startup−config running config: An exec command used to copy the configuration stored in NVRAM to the currently running configuration.

write erase: An exec command that causes the configuration stored in NVRAM to be erased.

reload: An exec command which causes the IOS to reload.

set pass: A Catalyst switch command used to set the nonenabled password.

set enablepass: A Catalyst switch command used to set the enabled password.

IOS Requirements

These password recovery procedures apply to all IOS versions 10.0 and later.

Lab #97: Cisco 3600 Password Recovery

Equipment Needed

The following equipment is needed to perform this lab exercise:

- A Cisco 3600 series router
- A PC running a terminal emulation program, preferably ProComm or Windows Hyperterm
- A Cisco rolled cable connecting the router to the PC

Configuration Overview

This section will provide detailed instructions on recovering an unknown password on a Cisco 3600 series router, as shown in Figure 22−3.

Figure 22−3: Cisco 3600 password recovery

Note Pressing the break sequence too soon after powering on the router can cause the router to lock up. In this case, simply power cycle the router again. It's a good idea to wait to press the break sequence until the router prints a message describing its processor type and main memory configuration.

Note Keep in mind that terminal emulation programs use different key combinations to generate the break sequence. The two most popular terminal emulators are Windows 95 Hyperterm and ProComm. The break sequence for ProComm is generated by pressing the ALT−B keys at the same time. In Windows 95 Hyperterm, the break sequence is generated by pressing the CTRL−BREAK keys at the same time.

Note Password recovery can only be performed with a terminal attached to the console port of the router. These procedures will not work on the aux port of the router.

Password Recovery Procedures

Before beginning, the router should have an enable password and a login password set. The following configuration shows an example of the enable and login password both set to "cisco".

Router

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname Cisco3620
!
enable password cisco ← Enable Password
!
no ip classless
!
line con 0
 password cisco ← Login Password
 login
line aux 0
line vty 0 4
 login
!
```

```
end
```

The following **show version** command reveals that the configuration register of the router is set to a value of 0x2102. As described in the previous section, this value will cause the router to use the NVRAM configuration file during the boot process. It is this register value that will be changed during the password recovery process, causing the router to ignore the contents of the NVRAM configuration file during the boot process.

```
Cisco3620#sh ver
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620−I−M), Version 11.2(8)P, RELEASE SOFTWARE (fc1)
Copyright (c) 1986−1997 by cisco Systems, Inc.
Compiled Mon 11−Aug−97 19:50 by ccai
Image text−base: 0x600088E0, data−base: 0x6044A000

ROM: System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY DEPLOYMENT RELEASE
SOFTWARE (fc2)

Cisco3620 uptime is 1 minute
System restarted by reload
System image file is "flash:c3620−i−mz.112−8.P", booted via flash

cisco 3620 (R4700) processor (revision 0x81) with 12288K/4096K bytes of memory.
Processor board ID 05706480
R4700 processor, Implementation 33, Revision 1.0
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
DRAM configuration is 32 bits wide with parity disabled.
29K bytes of non−volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)
8192K bytes of processor board PCMCIA Slot0 flash (Read/Write)
Configuration register is 0x2102
```

The first step in the password recovery process is to power cycle the router, turning it off and back on again. If the router is already off, turn it on. During the first few seconds of the boot process, you will see the following displayed:

```
System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY DEPLOYMENT RELEASE
SOFTWARE (fc2)
Copyright (c) 1994−1996 by cisco Systems, Inc.
C3600 processor with 16384 Kbytes of main memory
Main memory is configured to 32 bit mode with parity disabled ← Press the break
                                                                sequence here
```

After these messages are displayed, press the proper break sequence. Remember that every terminal emulation program has its own key combinations to force a break. The break sequence for ProComm is generated by pressing the ALT+B keys at the same time. In Windows 95 Hyperterm, the break sequence is generated by pressing the CTRL+BREAK keys at the same time. When the proper break sequence is pressed, the router will go into monitor mode:

```
monitor: command "boot" aborted due to user interrupt
```

At the rommon prompt type the command **confreg**.

```
rommon 1 >
rommon 1 > confreg
```

A current configuration summary will be displayed. You will be asked a series of questions. The proper yes and no responses should be entered for each question. Answer yes to the questions "do you wish to change the configuration ?", "ignore system config info ?", and "change the boot characteristics ?". Answer no to the

other questions. Finally, type a **2** to cause the system to boot the full IOS image upon startup.

```
    Configuration Summary
enabled are:
load rom after netboot fails
console baud: 9600
boot: image specified by the boot system commands
      or default to: cisco2−C3600
do you wish to change the configuration? y/n  [n]:  y
enable  "diagnostic mode"? y/n  [n]:  n
enable  "use net in IP bcast address"? y/n  [n]:  n
disable "load rom after netboot fails"? y/n  [n]:  n
enable  "use all zero broadcast"? y/n  [n]:  n
enable  "break/abort has effect"? y/n  [n]:  n
enable  "ignore system config info"? y/n  [n]:  y
change console baud rate? y/n  [n]:  n
change the boot characteristics? y/n  [n]:  y
enter to boot:
 0 = ROM Monitor
 1 = the boot helper image
 2−15 = boot system
    [2]:  2
```

A configuration summary will be printed out showing that the router will now ignore the system configuration information (NVRAM) upon startup.

```
    Configuration Summary
enabled are:
load rom after netboot fails
ignore system config info
console baud: 9600
boot: image specified by the boot system commands
      or default to: cisco2−C3600
```

You will again be asked if you want to change the configuration; this time, answer no:

```
do you wish to change the configuration? y/n  [n]:  n
```

The router will display a reminder that it must be reset in order for the changes to take effect. Type the **reset** command at the monitor prompt:

```
You must reset or power cycle for new config to take effect
rommon 2 >
rommon 2 > reset
```

The router will reload. This time it will ignore the configuration information contained in the NVRAM. When the router finishes booting, there will not be a console password or an enable password. Press ENTER to get into user mode and type **ena** to get into privileged mode:

```
Press RETURN to get started!


Router>
Router>ena
Router#
```

Use the **show running−configuration** command to view the router's current configuration. You will notice that the configuration contains no passwords of any kind. This is the configuration that gets created when the router ignores the contents of NVRAM:

```
Current configuration:
!
version 11.2
```

```
no service udp-small-servers
no service tcp-small-servers
!
hostname Router
!
no ip classless
!
line con 0 ← No passwords
line aux 0
line vty 0 4
 login
!
end
```

The key to password recovery is being able to view, change, or delete the passwords contained in the routers normal startup configuration stored in NVRAM, as detailed below:

Viewing: If you wish to learn the current password and continue to use it, type the **show startup-configuration** command. From the following configuration, you will notice that both the console and enable passwords are set to "cisco". The viewing option will only work if the password is not encrypted. If the password is encrypted, you will either have to change it or delete it.

```
Router#sh start
Using 355 out of 30712 bytes
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname Cisco3620
!
enable password cisco ← Enable password
!
no ip classless
!
line con 0
 password cisco ← Login password
 login
line aux 0
line vty 0 4
 login
!
end
```

Changing: To change the current passwords you will need to copy the NVRAM configuration to the running configuration. This is done via the **copy startup-configuration running-configuration**. Enter configuration mode using the **config term** command. Type the new passwords. Press CTRL+Z to exit configuration mode when you are done. Type **write mem** to save the new passwords to NVRAM.

Delete: You can erase the passwords by using the **write erase** command.

The final step before reloading is to change the router's configuration register to a value that will cause the router to load its initial configuration from NVRAM. The current configuration register value can be viewed using the **show version** command. The value appears at the end of the command's output. In this case, the value of 0x2142 causes the router to ignore the contents of NVRAM.

```
Router#sh ver
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-I-M), Version 11.2(8)P, RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1997 by cisco Systems, Inc.
Compiled Mon 11-Aug-97 19:50 by ccai
Image text-base: 0x600088E0, data-base: 0x6044A000
```

```
ROM: System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY DEPLOYMENT RELEASE
SOFTWARE (fc2)

Router uptime is 0 minutes
System restarted by power-on
System image file is "flash:c3620-i-mz.112-8.P", booted via flash

cisco 3620 (R4700) processor (revision 0x81) with 12288K/4096K bytes of memory.
Processor board ID 05706480
R4700 processor, Implementation 33, Revision 1.0
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
DRAM configuration is 32 bits wide with parity disabled.
29K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)
8192K bytes of processor board PCMCIA Slot0 flash (Read/Write)
Configuration register is 0x2142
```

The router's configuration register value is changed by typing **configur term** at the command prompt. Use the **config-register** command to enter the new register value; in this case, the new register value is 0x2102.

```
Router#config term
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#config-register 0x2102
Router(config)#exit
```

The **show version** command will now reflect the new setting. Notice that this new setting will not take effect until the router is reloaded.

```
Router#sh ver
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-I-M), Version 11.2(8)P, RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1997 by cisco Systems, Inc.
Compiled Mon 11-Aug-97 19:50 by ccai
Image text-base: 0x600088E0, data-base: 0x6044A000

ROM: System Bootstrap, Version 11.1(7)AX [kuong (7)AX], EARLY DEPLOYMENT
RELEASE SOFTWARE (fc2)

Router uptime is 1 minute
System restarted by power-on
System image file is "flash:c3620-i-mz.112-8.P", booted via flash

cisco 3620 (R4700) processor (revision 0x81) with 12288K/4096K bytes of memory.
Processor board ID 05706480
R4700 processor, Implementation 33, Revision 1.0
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
DRAM configuration is 32 bits wide with parity disabled.
29K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)
8192K bytes of processor board PCMCIA Slot0 flash (Read/Write)
Configuration register is 0x2142 (will be 0x2102 at next reload)
```

Type the **reload** command at the router prompt to reload the router and cause the new configuration register values to take effect. You do not need to save configuration register changes.

```
Router#reload

System configuration has been modified. Save? [yes/no]: n
Proceed with reload? [confirm]
```

The router will now reload. It will obtain its initial configuration from the contents of NVRAM. The passwords of the router are now known and can be used to access the privileged mode of the router.

Lab #98: Cisco 2500 Password Recovery

Equipment Needed

The following equipment is needed to perform this lab exercise:

- A Cisco 2500 series router
- A PC running a terminal emulation program, preferably ProComm or Windows Hyperterm
- A Cisco rolled cable connecting the router to the PC

Configuration Overview

This section will provide detailed instructions on recovering an unknown password on a Cisco 2500 series router as shown in Figure 22−4.

Figure 22−4: Cisco 2500 password recovery

Note Pressing the break sequence too soon after powering on the router can cause the router to lock up. In this case, simply power cycle the router again. It's a good idea to wait to press the break sequence until the router prints a message related to the size of the router's main memory.

Note Keep in mind that terminal emulation programs use different key combinations to generate the break sequence. The two most popular terminal emulators are Windows 95 Hyperterm and ProComm. The break sequence for ProComm is generated by pressing the ALT−B keys at the same time. In Windows 95 Hyperterm, the break sequence is generated by pressing the CTRL−BREAK keys at the same time.

Note Password recovery can only be performed with a terminal attached to the console port of the router. These procedures will not work on the aux port of the router.

Password Recovery Procedures

Before beginning, the router should have an enable password and a login password set. The following configuration shows an example of the enable and login passwords, both set to "cisco":

```
Current configuration:
!
version 11.0
service udp-small-servers
service tcp-small-servers
!
hostname Cisco2500
!
enable password cisco ← Enable password
!
line con 0
 password cisco ← Login password
 login
line aux 0
 transport input all
line vty 0 4
 login
```

```
!
end
```

The following **show version** command reveals that the configuration register of the router is set to a value of 0x2102. As described in the previous section, this value will cause the router to use the NVRAM configuration file during the boot process. It is this register value that will be changed during the password recovery process, causing the router to ignore the contents of the NVRAM configuration file during the boot process.

```
Cisco2500#sh ver
Cisco Internetwork Operating System Software
IOS (tm) 3000 Software (IGS-D-L), Version 11.0(10), RELEASE SOFTWARE (fc3)
Copyright (c) 1986-1996 by cisco Systems, Inc.
Compiled Mon 05-Aug-96 18:19 by loreilly
Image text-base: 0x03025A14, data-base: 0x00001000

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
ROM: 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE SOFTWARE (fc1)

Cisco2500 uptime is 2 minutes
System restarted by reload
System image file is "flash:igs-d-l.110-10", booted via flash

cisco 2500 (68030) processor (revision A) with 4096K/2048K bytes of memory.
Processor board ID 01412484, with hardware revision 00000000
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface.
2 Serial network interfaces.
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)

Configuration register is 0x2102
```

The first step in the password recovery process is to power cycle the router, turning it off and back on again. If the router is already off, turn it on. During the first few seconds of the boot process, you will see the following displayed:

```
System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
Copyright (c) 1986-1995 by cisco Systems
2500 processor with 4096 Kbytes of main memory ← Press the break sequence here
```

After these messages are displayed, press the proper break sequence. Remember that every terminal emulation program has its own key combinations to force a break. The break sequence for ProComm is generated by pressing the ALT+B keys at the same time. In Windows 95 Hyperterm, the break sequence is generated by pressing the CTRL+BREAK keys at the same time. When the proper break sequence is pressed, the router will go into monitor mode:

```
Abort at 0x10EA838 (PC)
>
```

At the > prompt, type the command **o/r 0x42.**

```
>o/r 0x42
>
```

Type an **i** to reboot the router.

```
>
>i
```

The router will reload. This time, it will ignore the configuration information contained in the NVRAM. When the router finishes booting, there will not be a console password or an enable password. Press ENTER to get into user mode and type **ena** to get into privileged mode:

```
Press RETURN to get started!


Router>
Router>ena
Router#
```

Use the **show running−configuration** command to view the router's current configuration. You will notice that the configuration contains no passwords of any kind. This is the configuration that gets created when the router ignores the contents of NVRAM:

```
Router#sh run


Building configuration...

Current configuration:
!
version 11.0
service udp−small−servers
service tcp−small−servers
!
hostname Router
!
line con 0 ← No passwords
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

The key to password recovery is being able to view, change, or delete the passwords contained in the router's normal startup configuration stored in NVRAM, as detailed below:

Viewing: If you wish to learn the current password and continue to use it, type the **show startup−configuration** command. From the following configuration, you will notice that both the console and enable passwords are set to "cisco". The viewing option will only work if the password is not encrypted. If the password is encrypted, you will either have to change it or delete it.

```
Router#sh start
Using 348 out of 32762 bytes
!
version 11.0
service udp−small−servers
service tcp−small−servers
!
hostname Cisco2500
!
enable password cisco ← Enable password
!
line con 0
 password cisco ← Login password
 login
line aux 0
 transport input all
line vty 0 4
 login
!
end
```

Changing: To change the current passwords, you will need to copy the NVRAM configuration to the running configuration. This is done via the **copy startup−configuration running−configuration** command. Enter configuration mode using the **config term** command. Type the new passwords. Press CTRL+Z to exit configuration mode when you are done. Type **write mem** to save the new passwords to NVRAM.

Erase: You can erase the passwords by using the **write erase** command.

The final step before reloading is to change the router's configuration register to a value that will cause the router to load its initial configuration from NVRAM. The current configuration register value can be viewed using the **show version** command. The value appears at the end of the command's output. In this case, the value of 0x2142 causes the router to ignore the contents of NVRAM.

```
Router#sh ver
Cisco Internetwork Operating System Software
IOS (tm) 3000 Software (IGS−D−L), Version 11.0(10), RELEASE SOFTWARE (fc3)
Copyright (c) 1986−1996 by cisco Systems, Inc.
Compiled Mon 05−Aug−96 18:19 by loreilly
Image text−base: 0x03025A14, data−base: 0x00001000

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
ROM: 3000 Bootstrap Software (IGS−RXBOOT), Version 10.2(8a), RELEASE SOFTWARE (fc1)

Router uptime is 1 minute
System restarted by power−on
System image file is "flash:igs−d−l.110−10", booted via flash

cisco 2500 (68030) processor (revision A) with 4096K/2048K bytes of memory.
Processor board ID 01412484, with hardware revision 00000000
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface.
2 Serial network interfaces.
32K bytes of non−volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)

Configuration register is 0x42
```

The router's configuration register value is changed by typing **configure term** at the command prompt. Use the **config−register** command to enter the new register value; in this case, the new register value is 0x2102.

```
Router#config term
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#conf
Router(config)#config−register 0x2102
Router(config)#exit
```

The **show version** command will now reflect the new setting. Notice that this new setting will not take effect until the router is reloaded.

```
Router#sh ver
Cisco Internetwork Operating System Software
IOS (tm) 3000 Software (IGS−D−L), Version 11.0(10), RELEASE SOFTWARE (fc3)
Copyright (c) 1986−1996 by cisco Systems, Inc.
Compiled Mon 05−Aug−96 18:19 by loreilly
Image text−base: 0x03025A14, data−base: 0x00001000

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
ROM: 3000 Bootstrap Software (IGS−RXBOOT), Version 10.2(8a), RELEASE SOFTWARE (fc1)

Router uptime is 1 minute
System restarted by power−on
System image file is "flash:igs−d−l.110−10", booted via flash

cisco 2500 (68030) processor (revision A) with 4096K/2048K bytes of memory.
```

```
Processor board ID 01412484, with hardware revision 00000000
Bridging software.
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface.
2 Serial network interfaces.
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)

Configuration register is 0x42 (will be 0x2102 at next reload)
```

Type the **reload** command at the router prompt to reload the router and cause the new configuration register values to take effect. You do not need to save configuration register changes.

```
Router#reload

System configuration has been modified. Save? [yes/no]: n
Proceed with reload? [confirm]
```

The router will now reload. It will obtain its initial configuration from the contents of NVRAM. The passwords of the router are now known and can be used to access the privileged mode of the router.

Lab #99: Cisco Catalyst 5000 Password Recovery

Equipment Needed

The following equipment is needed to perform this lab exercise:

- A Cisco Catalyst 5000 series switch
- A PC running a terminal emulation program, preferably ProComm or Windows Hyperterm
- A Cisco rolled cable connecting the switch to the PC

Configuration Overview

This section will provide detailed instructions on recovering an unknown password on a Cisco Catalyst 5000 series switch, as shown in Figure 22−5.

Figure 22−5: Catalyst 5000 password recovery

Note Password recovery can only be performed with a terminal attached to the console port of the switch. The Supervisor III module on the Catalyst 5500 has both a console port and an aux port. Password recovery will only work on the console port of the switch.

Password Recovery Procedures

The Catalyst switch follows a different password recovery scheme than the router. The passwords on the Catalyst switch are not valid for the first 30 seconds after the switch powers up and sends the initial system login prompt. This allows you to gain access to the switch by simply pressing ENTER at the password prompts. The drawback to this is that you only have 30 seconds to gain entry to the switch and change the passwords. Changing both system passwords in only 30 seconds can be difficult.

One way around this is to type the password change sequence into a text editor and copy it to the clipboard.

When the Catalyst switch reboots, you can paste this sequence to the screen and send the password change commands to the Catalyst switch all at once.

The first step in the password recovery process is to power cycle the switch, turning it off and back on again. As with a router, you must be connected to the console port. If the switch is already off, turn it on. During the first few seconds of the boot process, you will see the following displayed:

```
System Bootstrap, Version 3.1(2)
Copyright (c) 1994−1997 by cisco Systems, Inc.
Processor with 32768 Kbytes of main memory

Autoboot executing command: "boot bootflash:"
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Uncompressing file:
############################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
#########################################################################


System Power On Diagnostics
NVRAM Size .....................512KB
ID Prom Test ...................Passed
DPRAM Size .....................16KB
DPRAM Data 0x55 Test ...........Passed
DPRAM Data 0xaa Test ...........Passed
DPRAM Address Test .............Passed
Clearing DPRAM .................Done
System DRAM Memory Size ........32MB
DRAM Data 0x55 Test ............Passed
DRAM Data 0xaa Test ............Passed
DRAM Address Test ..............Passed
Clearing DRAM ..................Done
EARL++ .........................Present
EARL RAM Test ..................Passed
EARL Serial Prom Test ..........Passed
Level2 Cache ...................Present
Level2 Cache test ..............Passed

Boot image: bootflash:cat5000−sup3.3−1−1.bin

Running System Diagnostics from this Supervisor (Module 2)
This may take up to 2 minutes....please wait
```

When the following console message appears, press the ENTER key to log in to the switch. Remember that for the next 30 seconds the switch passwords will not be set, and pressing the ENTER key at a password prompt will allow access.

```
Cisco Systems Console ← Press the enter key when you see this message
```

Pressing the enter key the first time will give you access to the switches non−priviledged mode:

```
Console>
```

The following sequence can either be typed within the 30−second null password period or can be pasted to the terminal emulation screen. If you wish to paste it to the screen, you must first create the proper password change script in a text editor. The string would be as follows:

```
Ena ← ena is the Catalyst command for entering privileged mode. Follow this by
      pressing enter twice
```

```
set pass ← set pass will cause the Catalyst to change the non−privileged mode
           password. Follow this by pressing the enter key 4 times
set enablepass ← set enablepass will cause the Catalyst to change the
                 privileged mode password. Follow this by pressing the enter
                 key 4 times
```

After creating this string in the text editor, do a Select All and copy it to the clipboard. When the Catalyst prompt appears, copy this text sequence to the terminal:

```
Console> ena
Enter password:
Console> (enable) set pass
Enter old password:
Enter new password:
Retype new password:
Password changed.
Console> (enable) set enablepass
Enter old password:
Enter new password:
Retype new password:
Password changed.
Console> (enable)
```

The password on the Catalyst had now been set to the ENTER key for both the privileged and nonprivileged modes. The Catalyst passwords can now be set to any value.

Conclusion

This chapter provided detailed information on recovering lost or unknown passwords on Cisco routers and Catalyst switches. We saw that recovering the password on a Cisco router involves gaining console access to the router and changing the configuration register so that the router ignores the contents of NVRAM when it powers up.

Changing the password on the Catalyst 5000 family of switches involves gaining console access to the Catalyst and being able to change the system passwords within 30 seconds of first gaining console access after a power−on of the switch.

Chapter 23: HTTP Access with a Cisco Router

Overview

Topics Covered in This Chapter

- HTTP overview
- Configuring a Cisco router as an HTTP server
- Using access lists to restrict HTTP access to a router
- Troubleshooting HTTP transfers

Introduction

This Cisco IOS feature allows management access to a router via a standard Web browser such as Netscape Navigator or Microsoft Internet Explorer. This feature is beneficial to those users who are more comfortable using a Web browser interface than the Cisco command line.

HTTP Overview

HTTP is a client/server application. It uses TCP as its transport protocol. A client runs a Web browser application such as Netscape Navigator or Microsoft Internet Explorer. The Web client makes requests to an HTTP server running an HTTPD daemon program. These requests from the Web browser to the HTTPD server usually occur over TCP port 80. Figure 23−1 shows a simplified HTTP client and server configuration.

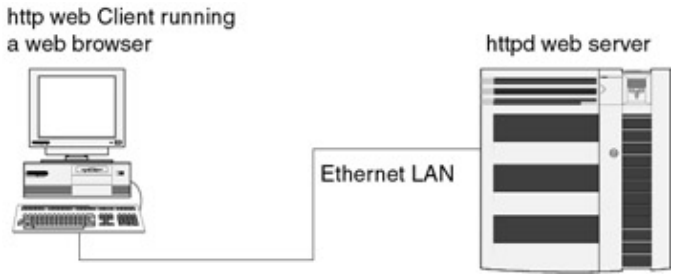


Figure 23−1: Basic HTTP client/server setup

Commands Discussed in This Chapter

- **ip http server**
- **ip http port [*number*]**
- **ip http access class [*access−list−number* | *name*]**
- **debug ip http url**
- **debug ip http tokens**
- **debug ip transactions**

Definitions

IP http server: This command is a global command that enables the router to respond to HTTP requests from a Web browser.

IP http port: This command is used to change the TCP port number that the router uses to listen to HTTP requests. By default, HTTP uses TCP port 80.

IP http access class: This command is used to control which Web browser hosts can access the router via HTTP requests.

IOS Requirements

The HTTP feature was first introduced in IOS version 11.0(6).

Lab #100: Basic Configuration Without an Access List

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having at least one serial port. One of the two routers must have an Ethernet port.
- A PC or workstation that is Ethernet connected to one of the two routers
- One Cisco DTE/DCE crossover cable. If no crossover cables are available, you can make a crossover cable by connecting a standard Cisco DTE cable to a standard Cisco DCE cable.
- Either an Ethernet crossover cable or two Ethernet cables and an Ethernet hub

Configuration Overview

The objective of this lab is to be able to access and manage both Cisco1 and Cisco2 from the Web browser running on the workstation at 10.10.3.77.

Figure 23−2 shows the router connectivity for the examples used in this chapter. A PC/Mac/UNIX workstation running a Web browser is Ethernet attached to a gateway Cisco router, Cisco1. Router Cisco1 is serially attached to a second router, Cisco2. IP addresses are as shown in the diagram.

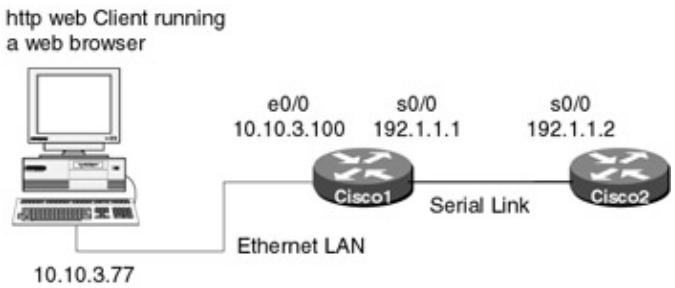


Figure 23−2: Cisco HTTP Web server setup

In this lab, the Cisco router will be running an HTTPD and will be acting as a Web server for a Web browser client application.

Router Configuration

The following are the configurations that should be entered into routers Cisco1 and Cisco2.

Cisco1

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname Cisco1
!
enable password cisco
!
```

```
no ip domain-lookup
!
interface Ethernet0/0
 ip address 10.10.3.100 255.255.255.0
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
!
router rip
 network 10.0.0.0
 network 192.1.1.0
!
ip http server ← Enable this router to act as an HTTP server
ip classless
!
line con 0
 exec-timeout 120 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
end
```

Cisco2

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname Cisco2
!
enable password cisco
!
no ip domain-lookup
!
interface Serial0/0
 ip address 192.1.1.2 255.255.255.0
 clockrate 800000
!
router rip
 network 192.1.1.0
!
ip http server ← Enable this router to act as an HTTP server
ip classless
!
line con 0
 password cisco
 login
line aux 0
 password cisco
 login
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
end
```

The **ip http server** command in each of the configurations enables the router to respond to HTTP requests from a Web client.

Monitoring and Testing the Configuration

After entering the proper configurations into Cisco1 and Cisco2, verify connectivity between the workstation and the two routers. This can be done by performing a ping on each router from the workstation.

```
C:\TEMP>ping 10.10.3.100

Pinging 10.10.3.100 with 32 bytes of data:

Reply from 10.10.3.100: bytes=32 time=7ms TTL=19
Reply from 10.10.3.100: bytes=32 time=7ms TTL=20
Reply from 10.10.3.100: bytes=32 time=7ms TTL=20
Reply from 10.10.3.100: bytes=32 time=7ms TTL=20
```

Verify connectivity to Cisco2 at address 192.1.1.2 via the same method.

Now start up the Web browser. As shown in Figure 23−3, enter the address of Cisco2 in the URL field and press RETURN. You should be presented with the initial login screen, as shown in Figure 23−3.

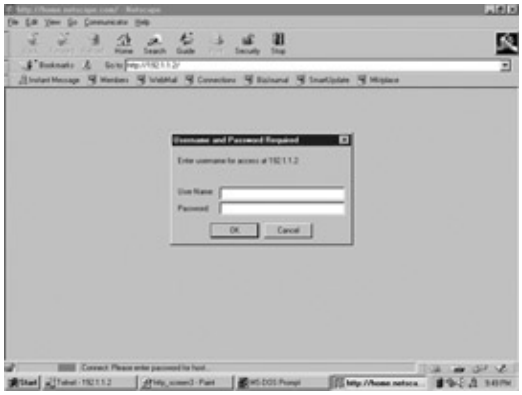


Figure 23−3: Web browser login screen
Enter a username of **cisco** and a password of **cisco**, then click OK.

You should now see the initial screen, as shown in Figure 23−4. Click on the "Show interfaces" option to view the status of the interfaces on Cisco2. Figure 23−5 depicts the output of this command.

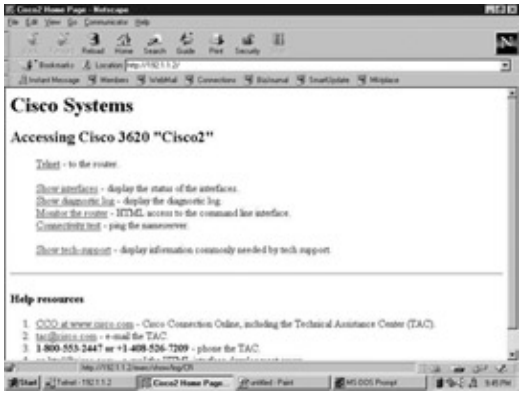


Figure 23−4: Router home screen

Figure 23−5: "Show interfaces" screen

Lab #101: Advanced Configurationwith an Access List

Configuration Overview

We will now add an **ip http access class** statement to the configuration we had in Lab #100. This command provides enhanced security, only allowing browsers at allowed IP addresses to access the router via HTTP.

Router Configuration

The configurations for the two routers in this example are as follows.

Cisco1

```
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname Cisco1
!
enable password cisco
!
no ip domain-lookup
!
interface Ethernet0/0
 ip address 10.10.3.100 255.255.255.0
!
interface Serial0/0
 ip address 192.1.1.1 255.255.255.0
!
router rip
 network 10.0.0.0
 network 192.1.1.0
!
ip http server ← Enable this router to act as an HTTP server
ip http access-class 1 ← Only allow hosts defined by access-list 1 HTTP access
                         to this router
ip classless
access-list 1 deny   10.10.3.77 ← Define an access-list for allowed HTTP hosts
access-list 1 permit any
!
line con 0
 exec-timeout 120 0
 password cisco
 login
line aux 0
 password cisco
 login
```

```
line vty 0 4
 exec-timeout 120 0
 password cisco
 login
!
end
```

The configuration for Cisco2 remains the same.

Monitoring and Testing the Configuration

Try to access router Cisco1 from the Web browser. You will now notice that the Web browser at 10.10.3.77 can no longer access Cisco1. The workstation can still access Cisco2 via the Web browser.

The **ip http access-class 1** command specifies that access list 1 should be applied to any HTTP requests. The **access-list 1 deny 10.10.3.77** command specifies that no traffic from 10.10.3.77 will be allowed. The **access-list 1 permit any** statement allows all other traffic.

The **ip http port** command can also be used in this configuration. This command allows the router to listen for HTTP requests on a port other than the default port of 80.

Troubleshooting HTTP

Several debug commands are available to assist in troubleshooting and monitoring HTTP connections to a router network:

{Debug ip http url} From the Cisco command line, enter the **debug ip http url** command:

```
Cisco2#debug ip http url
HTTP URL debugging is on
```

This command will show URL information. In the following example, we see that the Web browser at 10.10.3.77 has accessed the "Show interfaces" screen on the router:

```
HTTP: processing URL '/exec/show/interfaces/CR' from host 10.10.3.77
```

{Debug ip http tokens} From the Cisco command line, enter the **debug ip http tokens** command:

```
Cisco2#debug ip http tokens
HTTP tokens debugging is on
```

When an HTTP request is made to the router, this command will show individual token information parsed by the router.

```
HTTP: token len 3: 'GET'
HTTP: token len 1: ' '
HTTP: token len 1: '/'
HTTP: token len 1: ' '
HTTP: token len 4: 'HTTP'
HTTP: token len 1: '/'
HTTP: token len 1: '1'
HTTP: token len 1: '.'
HTTP: token len 1: '0'
HTTP: token len 2: '\15\12'
HTTP: token len 10: 'Connection'
HTTP: token len 1: ':'
HTTP: token len 1: ' '
HTTP: token len 4: 'Keep'
HTTP: token len 1: '-'
HTTP: token len 5: 'Alive'
HTTP: token len 2: '\15\12'
```

```
HTTP: token len 4: 'User'
HTTP: token len 1: '-'
HTTP: token len 5: 'Agent'
HTTP: token len 1: ':'
HTTP: token len 1: ' '
HTTP: token len 7: 'Mozilla'
HTTP: token len 1: '/'
HTTP: token len 1: '4'
HTTP: token len 1: '.'
```

{Debug ip http transactions} From the Cisco command line, enter the **debug ip http transactions** command:

```
Cisco2#debug ip http transactions
HTTP transactions debugging is on
```

When an HTTP request is made to the router, this command shows the high−level transactions between the Web browser and the router.

```
HTTP: parsed uri '/exec/show/tech-support/cr'
HTTP: client version 1.0
HTTP: parsed extension Referer
HTTP: parsed line  http://192.1.1.2/
HTTP: parsed extension Connection
HTTP: parsed line  Keep-Alive
HTTP: parsed extension User-Agent
HTTP: parsed line  Mozilla/4.05 [en] (Win95; I)
HTTP: parsed extension Host
HTTP: parsed line  192.1.1.2
HTTP: parsed extension Accept
HTTP: parsed line  image/gif, image/x-xbitmap, image/jpeg, image/
HTTP: parsed extension Accept-Language
HTTP: parsed line  en
HTTP: parsed extension Accept-Charset
HTTP: parsed line  iso-8859-1,*,utf-8
HTTP: parsed extension Authorization
HTTP: parsed authorization type Basic
HTTP: received GET 'exec'
```

Conclusion

This chapter demonstrates the HTTPD server capabilities of the Cisco router. Accessing devices via a Web interface is becoming an increasingly popular method of managing a network.

Chapter 24: Bridging and DLSW

Overview

Topics Covered in This Chapter

- DLSW overview
- Bridging with ISDN dial backup
- DLSW full mesh
- DLSW border peers
- DLSW backup peers
- Access expressions

Introduction

Bridging and DLSW are two technologies that are used to transport traffic that is not routable. The best example of nonroutable traffic is NetBEUI traffic. This chapter will present five labs on the subject of bridging and DLSW.

DLSW Overview

Data Link Switching (DLSW) is a method for integrating and forwarding nonroutable traffic over wide area networks by encapsulating the data in TCP/IP packets.

DLSW, introduced by IBM in 1992, is used for transporting nonroutable protocols across an IP network. Prior to DLSW being introduced, Source Route Bridging (SRB) was the primary method used to transport nonroutable protocols across an IP network. The principal difference between SRB and DLSW lies in how the two protocols treat local termination. Nonroutable protocols such as SNA and NetBEUI rely on link−layer acknowledgments and

keep−alive messages to ensure the integrity of connections and the delivery of data. SRB, as shown in Figure 24−1, handles acknowledgments on an end−to−end basis. As shown in Figure 24−2, DLSW terminates the data−link control. Therefore, link−layer acknowledgments and keep−alive messages do not have to traverse a WAN. SRB also has a restrictive network diameter of seven hops.

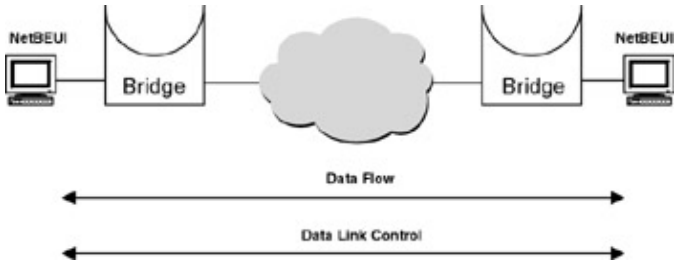


Figure 24−1: SRB control and data flow

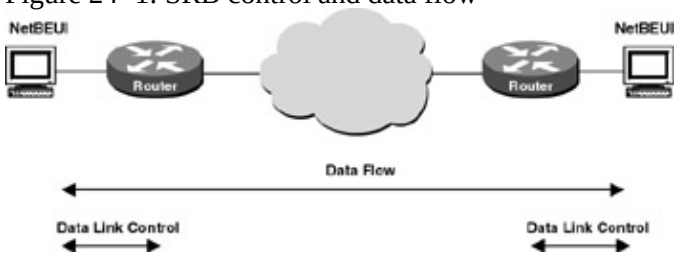


Figure 24−2: DLSW control and data flow

The Switch to Switch (SSP) protocol is used between DLSW routers to establish connections, locate

resources, forward data, and handle flow control and error recovery. Routing the data packets across the intermediate IP network is not handled by SSP. That task is left to the routing protocol that is being run on the network. SSP also encapsulates packets in TCP/IP for transport over IP−based networks and uses TCP as a means of reliable transport between DLSW nodes.

DLSW routers establish two TCP connections with each other. TCP connections are the basis for DLSW communication. TCP ensures reliable and guaranteed delivery of IP traffic. After a connection is established, the DLSW partners exchange a list of supported capabilities. Next, the DLSW partners establish circuits between SNA or NetBIOS end systems, and information frames can flow over the circuit.

Commands Discussed in This Chapter

- **access−expression {in | out}** *expression*
- **bridge group** *bridge group*
- **bridge** *bridge−group* **protocol {ieee | dec}**
- **debug dlsw reach**
- **debug dlsw peer**
- **debug dlsw local**
- **dlsw bridge−group** *group−number*
- d**lsw local−peer [peer−id** *ip−address***] [group** *group***] [border] [**cost *cost***] [lf** *size***] [keepalive** *seconds*] **[passive] [promiscuous] [init−pacing−window** *size*] [**max−pacing−window** *size***][biu−segment]**
- **dlsw remote−peer** *list−number* **tcp** *ip−address* **[backup−peer [***ip−address* **| frame−relay interface serial** *number dlci−number* **| interface** *name***]] [bytes−netbios−out** *bytes−list−name***] [cost *cost*] [dest−mac** *mac−address***] [dmac−output−list** *access−list−number***] [dynamic] [host−netbios−out** *host−list−name***] [keepalive** *seconds***] [lf** *size***] [linger** *minutes***] [lsap−output−list** *list***] [no−llc** *minutes***] [priority] [tcp−queue−max** *size***] [timeout** *seconds***]**
- **netbios access−list host** *name* **{permit | deny}** *pattern*
- **show access−expression [begin | exclude | include]**
- **show bridge [***bridge−group***] [***interface***] [***address mask***] [**verbose**]**
- **show bridge group [verbose]**
- **show dlsw capabilities**
- **show dlsw peers**
- **show dlsw reachability**
- **source−bridge ring−group** *ring−group* **[***virtual−mac−address***]**

Definitions

access−expression: This interface configuration command is used to define a per−interface access expression.

bridge group: This interface configuration command is used to assign a network interface to a bridge group.

bridge protocol: This global configuration command is used to assign a bridge group number and define a spanning−tree protocol as either IEEE 802.1D standard or Digital.

debug dlsw reachability: This debug command displays DLSW reachability event information.

debug dlsw peer: This debug command displays DLSW peer event information.

dlsw bridge−group: This global configuration command is used to link DLSW to the bridge group of an Ethernet LAN.

dlsw local−peer: This global configuration command is used to define the parameters of the DLSW local peer.

dlsw remote−peer: This global configuration command is used to identify the IP address of a peer with which to exchange traffic using TCP.

netbios access−list: This global configuration command is used to assign the name of an access list to a station or set of stations on the network.

show access−expression: This privileged exec mode command is used to display the defined input and output access list expressions.

show bridge: This privileged exec command is used to display classes of entries in the bridge forwarding database.

show bridge group: This privileged exec command is used to display information about configured bridge groups.

show dlsw capabilities: This privileged EXEC command is used to display the configuration of a specific peer or all peers.

show dlsw peers: This privileged EXEC command is used to display DLSW peer information.

show dlsw reachability: This privileged EXEC command is used to display DLSW reachability information.

source−bridge ring−group: This global configuration command is used to define a ring group identifier.

IOS Requirements

DLSW was introduced in IOS 10.3. All labs in this chapter were done using IOS 11.2 and higher.

Lab #102: Bridging with ISDN Dial Backup

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers. Each router must have 1 Serial, 1 BRI, and 1 Ethernet interface.
- Two workstations running NetBEUI.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- Two Ethernet crossover cables.
- A Cisco rolled cable for console port connection to the routers.
- An ISDN BRI circuit (see Chapter 3 for information on obtaining ISDN circuits).
- A Cisco IOS image that supports bridging.

Configuration Overview

This lab will demonstrate bridging with ISDN dial backup. RouterA and RouterB will have all IP routing functionality disabled. Each router will have a workstation attached to it that is only running NetBEUI. Windows networking will be used to verify that the configuration is properly working. When the primary link fails between the two routers, RouterA will initiate an ISDN call to RouterB, ensuring that the two

workstations will still be able to communicate.

The routers are connected as shown in Figure 24−3. RouterB acts as a DCE and supplies clocking to RouterA.

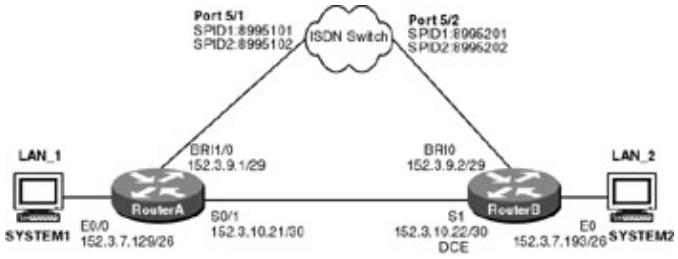


Figure 24−3: Bridging with ISDN backup

Note For instructions on how to configure a workstation to run NetBEUI, see the section at the end of this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. Bridging commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
username RouterB password 0 cisco
isdn switch-type basic-ni1
!
interface Ethernet0/0
 ip address 152.3.7.129 255.255.255.192
 no keepalive
 bridge-group 1 ← Assign interface to bridge group 1
!
interface Serial0/1
 backup delay 5 10 ← Backup interface will be activated 5 seconds after primary
                     goes down and de-activated 10 seconds after primary goes
                     back up
 backup interface BRI1/0 ← Specifies that interface BRI1/0 will be activated
                           when the S0/1  interface goes into a down state
 ip address 152.3.10.21 255.255.255.252
 encapsulation ppp
 bridge-group 1 ← Assign interface to bridge group 1
!
interface  BRI1/0
 ip address 152.3.9.1 255.255.255.248
 encapsulation ppp
 isdn spid1 5101 8995101
 isdn spid2 5102 8995102
 dialer map bridge name RouterB broadcast 8995201
 dialer-group 1
 ppp authentication chap
 bridge-group 1 ← Assign interface to bridge group 1
!
router ospf 64
 network 152.0.0.0 0.255.255.255 area 0
!
no ip classless
access-list 201 permit 0x0000 0xFFFF ← Permit all bridged traffic to act as
```

```
                                        interesting traffic
!
dialer-list 1 protocol bridge list 201
bridge 1 protocol ieee ← Define our spanning tree protocol
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
!
username RouterA password 0 cisco
isdn switch-type basic-ni1
!
interface Ethernet0
 ip address 152.3.7.193 255.255.255.192
 no keepalive
 bridge-group 1
!
interface  Serial1
 ip address 152.3.10.22 255.255.255.252
 encapsulation ppp
 clockrate 800000
 bridge-group 1
!
interface BRI0
 ip address 152.3.9.2 255.255.255.248
 encapsulation ppp
 isdn spid1 5201 8995201
 isdn spid2 5202 8995202
 dialer map bridge name RouterA broadcast
 dialer-group 1
 ppp authentication chap
 bridge-group 1
!
router ospf 64
 network 152.0.0.0 0.255.255.255 area 0
!
no ip classless
access-list 201 permit 0x0000 0xFFFF
dialer-list 1 protocol bridge list 201
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. The output of the **show bridge 1 verbose** command is shown next. Notice that ports E0/0 and S0/1 are in a forwarding state. Also notice that the MAC addresses of both System1 and System2 have been learned by the bridge.

```
RouterA#show bridge 1 verbose

Total of 300 station blocks, 297 free
Codes: P − permanent, S − self

BG Hash      Address      Action   Interface     VC    Age   RX count   TX count
 1 A7/0  0000.8635.46e1   forward  Ethernet0/0   −     4     123         57
 1 FB/0  0000.8635.4ab1   forward  Serial0/1     −     0     120         60

Flood ports        RX count    TX count
Ethernet0/0              63          63
Serial0/1                63          63
BRI1/0                    0           0
```

Use the **show bridge group** command to display the active bridge groups on RouterA. Bridge group 1 is the bridge group that we just defined. Notice that there are two ports in bridge group 1, interface E0/0 and interface S0/1. We see that both of these ports are in a forwarding state.

```
RouterA#show bridge group

Bridge Group 1 is running the IEEE compatible Spanning Tree protocol

   Port 4 (Ethernet0/0) of bridge group 1 is forwarding
   Port 3 (Serial0/1) of bridge group 1 is forwarding
```

Now connect to RouterB. The **show bridge 1 verbose** command indicates that both of the bridge group 1 interfaces on RouterB (S1 and E0) are in a forwarding state.

```
RouterB#show bridge 1 verbose

Total of 300 station blocks, 298 free
Codes: P − permanent, S − self

BG Hash      Address      Action   Interface     VC    Age   RX count   TX count
 1 A7/0   0000.8635.46e1  forward  Serial1       −     2          57         57
 1 FB/0   0000.8635.4ab1  forward  Ethernet0      −    0         122         53

Flood ports        RX count    TX count
BRI0                     0          69
Ethernet0               65           4
Serial1                  4          65
```

The **show bridge group** command verifies that we have three interfaces on RouterB that are part of bridge group 1 (BRI0, E0, and S1).

```
RouterB#show bridge group

Bridge Group 1 is running the IEEE compatible Spanning Tree protocol

   Port 3 (BRI0) of bridge group 1 is forwarding
   Port 2 (Ethernet0) of bridge group 1 is forwarding
   Port 7 (Serial1) of bridge group 1 is forwarding
```

At this point, the reader can verify that the bridging configuration is working properly by making sure that System1 on LAN_1 can see System2 on LAN_2 via Windows networking.

Now reconnect to RouterA. Type the **show isdn status** command. Notice that the router has not yet activated the ISDN circuit. This is due to the fact that we are using a backup interface. The router will not activate the ISDN circuit until it needs to activate the backup interface (BRI1/0). When using a backup interface, it is important to make sure that the SPIDs entered for the BRI circuit are correct. If not, you will not find out that the SPIDs are incorrect until it is time to activate the circuit.

```
RouterA#sh isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI1/0 interface
    Layer 1 Status:
     DEACTIVATED
    Layer 2 Status:
     Layer 2 NOT Activated
    Spid Status:
     TEI Not Assigned, ces = 1, state = 1(terminal down)
         spid1 configured, spid1 NOT sent, spid1 NOT valid
     TEI Not Assigned, ces = 2, state = 1(terminal down)
         spid2 configured, spid2 NOT sent, spid2 NOT valid
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 0
    Total Allocated ISDN CCBs = 0
```

The **show interface bri 1/0** command indicates that the interface is in a standby mode.

```
RouterA#show interface bri 1/0
BRI1/0 is standby mode, line protocol is down
  Hardware is QUICC BRI with U interface
  Internet address is 152.3.9.1/29
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

The **show interface S 0/1** command shows us that the interface is in an up/up state. We see that interface BRI 1/0 is listed as the backup interface for S 0/1. Notice that our **backup delay 5 10** command has resulted in the failure delay being set to 5 seconds and the secondary disable delay being set to 10 seconds.

```
RouterA#show interface s 0/1
Serial0/1 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 152.3.10.21/30
  Backup interface BRI1/0, kickin load not set, kickout load not set
      failure delay 5 sec, secondary disable delay 10 sec
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Open
  Open: IPCP, CDPCP, BRIDGECP
  Last input 00:00:01, output 00:00:04, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
```

```
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     8764 packets input, 386615 bytes, 0 no buffer
     Received 8707 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     3780 packets output, 189359 bytes, 0 underruns
     0 output errors, 0 collisions, 24 interface resets
     0 output buffer failures, 0 output buffers swapped out
     11 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Since we are still running over our primary circuit, the **show bridge 1 verbose** command indicates that interfaces E0/0 and S0/1 are still forwarding traffic. We see that the receive and transmit counts on interface BRI 1/0 are still 0.

```
RouterA#show bridge 1 verbose

Total of 300 station blocks, 298 free
Codes: P − permanent, S − self

BG Hash      Address      Action  Interface      VC    Age   RX count   TX count
 1 A7/0   0000.8635.46e1 forward  Ethernet0/0     −     3         123         57
 1 FB/0   0000.8635.4ab1 forward  Serial0/1       −     3          29         24

Flood ports        RX count    TX count
Ethernet0/0              89         110
Serial0/1               110          89
BRI1/0                    0           0
```

The **show bridge group** command tells us that interface BRI1/0 is part of bridge group 1, but that it is currently in a down state.

```
RouterA#sh bridge group

Bridge Group 1 is running the IEEE compatible Spanning Tree protocol

   Port 4 (Ethernet0/0) of bridge group 1 is forwarding
   Port 3 (Serial0/1) of bridge group 1 is forwarding
   Port 5 (BRI1/0) of bridge group 1 is down
```

Now let's fail the primary link and see how the ISDN backup circuit gets established. First, we will enable some key debug prompts so that we can see the backup process occur. Enable ppp authentication, ppp negotiation, and ISDN layer 3 (q931) and ISDN layer 2 (q921) debugging on RouterA.

```
RouterA#debug ppp authentication
PPP authentication debugging is on
RouterA#debug ppp negotiation
PPP protocol negotiation debugging is on
RouterA#debug isdn q931
ISDN Q931 packets debugging is on
RouterA#debug isdn q921
ISDN Q921 packets debugging is on
```

With the previous debug statements enabled, pull the S0/1 cable on RouterA. The following debug output will appear.

```
%LINEPROTO−5−UPDOWN: Line protocol on Interface Serial0/1, changed state to
down
%LINK−3−UPDOWN: Interface Serial0/1, changed state to down ← Interface S0/1
                                                              will go down when
```

```
                                                              the interface
                                                              cable is pulled

Se0/1 IPCP: State is Closed ← Once the interface goes down, all Layer 2 and
                              Layer 3 protocols that were opened will be closed
Se0/1 CDPCP: State is Closed
Se0/1 BNCP: State is Closed
Se0/1 PPP: Phase is TERMINATING
Se0/1 LCP: State is Closed
Se0/1 PPP: Phase is DOWN
Se0/1 PPP: Unsupported or un-negotiated protocol. Link cdp


Se0/1 IPCP: Remove route to 152.3.10.22 ← RouterA will remove its route to its
                                          directly connected neighbor
%LINK-3-UPDOWN: Interface BRI1/0:1, changed state to down
BR1/0:1 LCP: State is Closed
BR1/0:1 PPP: Phase is DOWN
%LINK-3-UPDOWN: Interface BRI1/0:2, changed state to down
BR1/0:2 LCP: State is Closed
BR1/0:2 PPP: Phase is DOWN
```

Notice that once the S0/1 interface is down, the BRI 1/0 interface will begin to activate itself.

```
ISDN BR1/0: TX ->  IDREQ  ri = 86  ai = 127 ← The ISDN interface will send an
                                              ID request to the ISDN switch for
                                              the first B channel

ISDN BR1/0: RX <-  IDASSN  ri = 86  ai = 64 ← The ISDN switch will reply with
                                              an ID assignment for the first B
                                              channel
ISDN BR1/0: TX ->  SABMEp sapi = 0  tei = 64
%LINK-3-UPDOWN: Interface BRI1/0, changed state to up ← The BRI 1/0 interface
                                                        will be declared up

ISDN BR1/0: RX <-  UAf sapi = 0  tei = 64
%ISDN-6-LAYER2UP: Layer 2 for Interface BR1/0, TEI 64 changed to up
ISDN BR1/0: TX ->  INFOc sapi = 0  tei = 64  ns = 0  nr = 0  i = 0x08007B3A0435313031
    INFORMATION pd = 8  callref = (null)
        SPID Information i = '5101' ← The router will send the SPID1 value to
                                      the ISDN switch
ISDN BR1/0: RX <-  RRr sapi = 0  tei = 64  nr = 1
ISDN BR1/0: RX <-  INFOc sapi = 0  tei = 64  ns = 0  nr = 1  i = 0x08007B3B02F081
    INFORMATION pd = 8  callref = (null)
        ENDPOINT IDent i = 0xF081
ISDN BR1/0: TX ->  RRr sapi = 0  tei = 64  nr = 1
ISDN BR1/0: Received EndPoint ID
ISDN BR1/0: TX ->  IDREQ  ri = 1463  ai = 127 ← The ISDN interface will send an
                                                ID request to the ISDN switch
                                                for the second B channel

ISDN BR1/0: RX <-  IDASSN  ri = 1463  ai = 65 ← The ISDN switch will reply with
                                                an ID assignment for the second
                                                B channel
ISDN BR1/0: TX ->  SABMEp sapi = 0  tei = 65
ISDN BR1/0: RX <-  UAf sapi = 0  tei = 65
%ISDN-6-LAYER2UP: Layer 2 for Interface BR1/0, TEI 65 changed to up
ISDN BR1/0: TX ->  INFOc sapi = 0  tei = 65  ns = 0  nr = 0  i = 0x08007B3A0435313032
    INFORMATION pd = 8  callref = (null)
        SPID Information i = '5102' ← The router will send the SPID2 value to
                                      the ISDN switch

ISDN BR1/0: RX <-  RRr sapi = 0  tei = 65  nr = 1
ISDN BR1/0: RX <-  INFOc sapi = 0  tei = 65  ns = 0  nr = 1  i = 0x08007B3B02F082
    INFORMATION pd = 8  callref = (null)
        ENDPOINT IDent i = 0xF082
ISDN BR1/0: TX ->  RRr sapi = 0  tei = 65  nr = 1
```

```
ISDN BR1/0: Received EndPoint ID
ISDN BR1/0: RX <-  RRp sapi = 0  tei = 64 nr = 1
ISDN BR1/0: TX ->  RRf sapi = 0  tei = 64  nr = 1
```

Now let's view the status of the ISDN circuit with the **show isdn status** command. We see that the ISDN circuit is now in an active state. Also, we see that a TEI has been assigned for each of the two B channels on the ISDN circuit. We also see that the router has successfully sent both SPIDs to the switch. Notice that there are no active ISDN calls.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI1/0 interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     0 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 1
     CCB: callid=0x0, sapi=0, ces=1, B-chan=0
    Total Allocated ISDN CCBs = 1
```

We just saw from the output of the **show isdn status** command that RouterA has still not placed a call to RouterB. We have seen that the ISDN circuit on RouterA will be activated as soon as the BRI 1/0 interface goes into backup mode. But, RouterA will not actually place a call to RouterB until some interesting traffic triggers the call. You can generate interesting traffic by going on to System1 and trying to view a file on System2 via network neighborhood. The following trace shows what happens when interesting traffic triggers an ISDN call from RouterA to RouterB.

```
ISDN BR1/0: TX ->  INFOc sapi = 0  tei = 64  ns = 1  nr = 1  i =
0x08010105040288901801832C0738393935323031
    SETUP pd = 8  callref = 0x01
        Bearer Capability i = 0x8890
        Channel ID i = 0x83
        Keypad Facility i = '8995201' ← RouterA places a call to 899-5201. This
                                        is the ISDN circuit connected to
                                        RouterB

ISDN BR1/0: RX <-  RRr sapi = 0  tei = 64  nr = 2
ISDN BR1/0: RX <-  INFOc sapi = 0  tei = 64  ns = 1  nr = 2  i = 0x08018102180189
    CALL_PROC pd = 8  callref = 0x81
        Channel ID i = 0x89
ISDN BR1/0: TX ->  RRr sapi = 0  tei = 64  nr = 2
ISDN BR1/0: RX <-  INFOc sapi = 0  tei = 64  ns = 2  nr = 2  i = 0x08018107180189
    CONNECT pd = 8  callref = 0x81 ← RouterA receives a connect message from
                                     the ISDN switch
        Channel ID i = 0x89

ISDN BR1/0: TX ->  INFOc sapi = 0  tei = 64  ns = 2  nr = 3  i = 0x0801010F
    CONNECT_ACK pd = 8  callref = 0x01 ← RouterA sends a connect acknowledge
                                         message to the ISDN switch

ISDN BR1/0: TX ->  RRr sapi = 0  tei = 64  nr = 3
%LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up ← The ISDN Call on
                                                          B-channel 1 is now
                                                          active
BR1/0:1 PPP: Treating connection as a callout
```

```
BR1/0:1 PPP: Phase is ESTABLISHING, Active Open
BR1/0:1 LCP: O CONFREQ [Closed] id 1 len 15 ← RouterA initiates an LCP
                                              configuration request

BR1/0:1 LCP:   AuthProto CHAP (0x0305C22305) ← RouterA is requesting CHAP
                                               authentication


BR1/0:1 LCP:   MagicNumber 0xE0B91E1E (0x0506E0B91E1E)
BR1/0:1 LCP: I CONFREQ [REQsent] id 1 len 15 ← RouterA receives an LCP
                                               negotiation request from RouterB
BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
BR1/0:1 LCP:    MagicNumber 0x00A6C01B (0x050600A6C01B)

BR1/0:1 LCP: O CONFACK [REQsent] id 1 len 15 ← RouterB sends a configuration
                                               acknowledgment to RouterA

BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
BR1/0:1 LCP:    MagicNumber 0x00A6C01B (0x050600A6C01B)
BR1/0:1 LCP: I CONFACK [ACKsent] id 1 len 15 ← RouterB receives a configuration
                                               acknowledgment from RouterA
BR1/0:1 LCP:    AuthProto CHAP (0x0305C22305)
BR1/0:1 LCP:    MagicNumber 0xE0B91E1E (0x0506E0B91E1E)
BR1/0:1 LCP: State is Open


BR1/0:1 PPP: Phase is AUTHENTICATING, by both ← Once the LCP negotiation is
                                                completed, the routers begin to
                                                authenticate to each other

BR1/0:1 CHAP: O CHALLENGE id 1 len 23 from "RouterA" ← RouterA sends a CHAP
                                                       challenge to RouterB

BR1/0:1 CHAP: I CHALLENGE id 1 len 23 from "RouterB" ← RouterA receives a CHAP
                                                       challenge from RouterB

BR1/0:1 CHAP: O RESPONSE id 1 len 23 from "RouterA"
BR1/0:1 CHAP: I SUCCESS id 1 len 4 ← RouterB sends RouterA a CHAP success
                                     message
BR1/0:1 CHAP: I RESPONSE id 1 len 23 from "RouterB"
BR1/0:1 CHAP: O SUCCESS id 1 len 4 ← RouterA sends RouterB a CHAP success
                                     message

BR1/0:1 PPP: Phase is UP
BR1/0:1 IPCP: O CONFREQ [Closed] id 1 len 10 ← Once authentication has
                                               completed, network layer
                                               attributes are negotiated
BR1/0:1 IPCP:    Address 152.3.9.1 (0x030698030901)
BR1/0:1 CDPCP: O CONFREQ [Closed] id 1 len 4
BR1/0:1 BNCP: O CONFREQ [Closed] id 1 len 4
BR1/0:1 IPCP: I CONFREQ [REQsent] id 1 len 10
BR1/0:1 IPCP:    Address 152.3.9.2 (0x030698030902)
BR1/0:1 IPCP: O CONFACK [REQsent] id 1 len 10
BR1/0:1 IPCP:    Address 152.3.9.2 (0x030698030902)
BR1/0:1 CDPCP: I CONFREQ [REQsent] id 1 len 4
BR1/0:1 CDPCP: O CONFACK [REQsent] id 1 len 4
BR1/0:1 BNCP: I CONFREQ [REQsent] id 1 len 4
BR1/0:1 BNCP: O CONFACK [REQsent] id 1 len 4
BR1/0:1 IPCP: I CONFACK [ACKsent] id 1 len 10
BR1/0:1 IPCP:    Address 152.3.9.1 (0x030698030901)
BR1/0:1 IPCP: State is Open
BR1/0:1 CDPCP: I CONFACK [ACKsent] id 1 len 4
BR1/0:1 CDPCP: State is Open
BR1/0:1 BNCP: I CONFACK [ACKsent] id 1 len 4
BR1/0:1 BNCP: State is Open
BR1/0 IPCP: Install route to 152.3.9.2 ← RouterA installs a route to its
                                         directly connected neighbor interface
                                         on RouterB
```

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to up
%ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 8995201 RouterB
                                á The call has been connected
```

We see in what follows that the **show isdn status** command now indicates that we have an active ISDN call.

```
RouterA#show isdn status
The current ISDN Switchtype = basic-ni1
ISDN BRI1/0 interface
    Layer 1 Status:
     ACTIVE
    Layer 2 Status:
     TEI = 64, State = MULTIPLE_FRAME_ESTABLISHED
     TEI = 65, State = MULTIPLE_FRAME_ESTABLISHED
    Spid Status:
     TEI 64, ces = 1, state = 5(init)
         spid1 configured, spid1 sent, spid1 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 1
     TEI 65, ces = 2, state = 5(init)
         spid2 configured, spid2 sent, spid2 valid
         Endpoint ID Info: epsf = 0, usid = 70, tid = 2
    Layer 3 Status:
     1 Active Layer 3 Call(s)
    Activated dsl 8 CCBs = 2
     CCB: callid=0x0, sapi=0, ces=1, B-chan=0
     CCB: callid=0x8001, sapi=0, ces=1, B-chan=1
    Total Allocated ISDN CCBs = 2
```

The **show dialer interface bri 1/0** will show dial specific information for the ISDN interface on RouterA. Notice that the reason the call was placed was that bridged traffic qualified as interesting traffic (Dial reason: bridge (0xF0F0)). We see that the current call will remain up for an additional 82 seconds. Any interesting traffic that is sent over the link will reset this interesting traffic counter.

```
RouterA#sh dialer int bri 1/0

BRI1/0 - dialer type = ISDN

Dial String      Successes   Failures    Last called   Last status
8995201                  1          0    00:00:56       successful
0 incoming call(s) have been screened.

BRI1/0:1 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is data link layer up
Dial reason: bridge (0xF0F0)
Time until disconnect 82 secs
Connected to 8995201 (RouterB)

BRI1/0:2 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle
```

The **show bridge group** command indicates that interface S0/1 is still part of Bridge Group1 but that the interface is now down. Interface BRI 1/0 is now in a forwarding state.

```
RouterA#show bridge group

Bridge Group 1 is running the IEEE compatible Spanning Tree protocol

   Port 4 (Ethernet0/0) of bridge group 1 is forwarding
   Port 3 (Serial0/1) of bridge group 1 is down ← S0/1 is still part of Bridge
                                                  Group 1 but it is down
```

```
   Port 5 (BRI1/0) of bridge group 1 is forwarding ← BRI 1/0 is now in a
                                                     forwarding state
```

The **show bridge 1 verbose** command shows us that the BRI interface is now transmitting bridged traffic over the ISDN link.

```
RouterA#show bridge 1 verbose

Total of 300 station blocks, 297 free
Codes: P − permanent, S − self

BG Hash      Address      Action   Interface     VC    Age   RX count   TX count
 1 A7/0   0000.8635.46e1  forward  Ethernet0/0   −     0           12          0
 1 FB/0   0000.8635.4ab1  forward  BRI1/0        −     0            6          6

Flood ports        RX count    TX count
Ethernet0/0              96         116
Serial0/1               110          90
BRI1/0                    6           6 ← Bridge traffic is now flowing over
                                          the ISDN circuit
```

Now let's connect to RouterB. The **show bridge 1 verbose** command indicates that interface BRI0 is now sending and receiving bridged traffic.

```
RouterB#show bridge 1 verbose

Total of 300 station blocks, 300 free
Codes: P − permanent, S − self


Flood ports        RX count    TX count
BRI0                     12         125
Ethernet0               119           6
Serial1                   6         110
```

The **show bridge group** command shows us that interface BRI 0 is now a part of the bridge group, while interface S1 is still a part of the bridge group but is down.

```
RouterB#show bridge group

Bridge Group 1 is running the IEEE compatible Spanning Tree protocol

   Port 3 (BRI0) of bridge group 1 is forwarding
   Port 2 (Ethernet0) of bridge group 1 is forwarding
   Port 7 (Serial1) of bridge group 1 is down
```

After the interesting traffic timeout expires, the ISDN call will be dropped.

Lab #103: DLSW Full Mesh

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with 2 serial interfaces and 1 Ethernet interface.
- Two Cisco routers with 1 serial interface and 1 Ethernet interface.
- Three Cisco crossover cables. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.

- Four workstations running NetBEUI.
- Four Ethernet crossover cables.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports DLSW.

Configuration Overview

This lab will configure any to any connectivity across four workstations. Each of these four workstations will be running NetBEUI and will not be running the IP protocol.

The four routers are physically connected as shown in Figure 24−4. Figure 24−5 shows the logical connectivity for this lab exercise. We see that DLSW requires that we have a full mesh of DLSW peers between all of our routers. This means that we need a DLSW peer between every router and each of its neighbors.

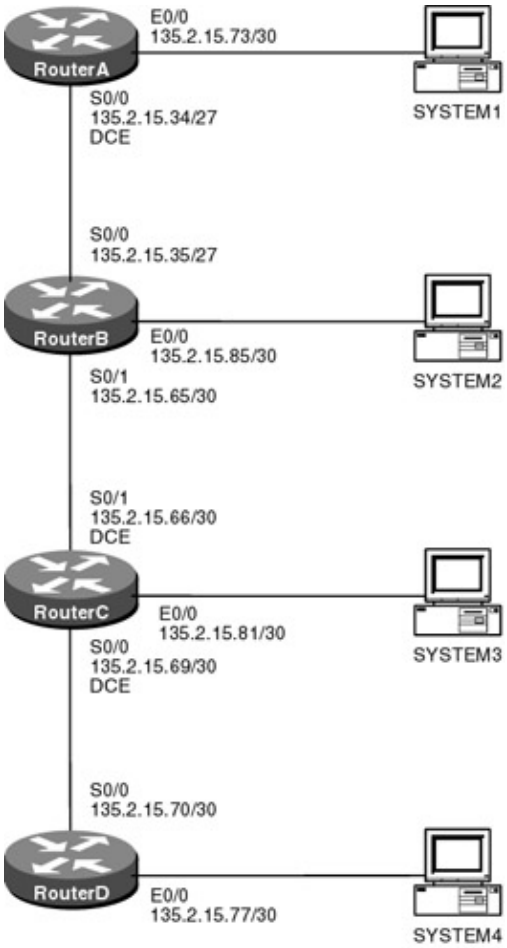


Figure 24−4: DLSW full mesh

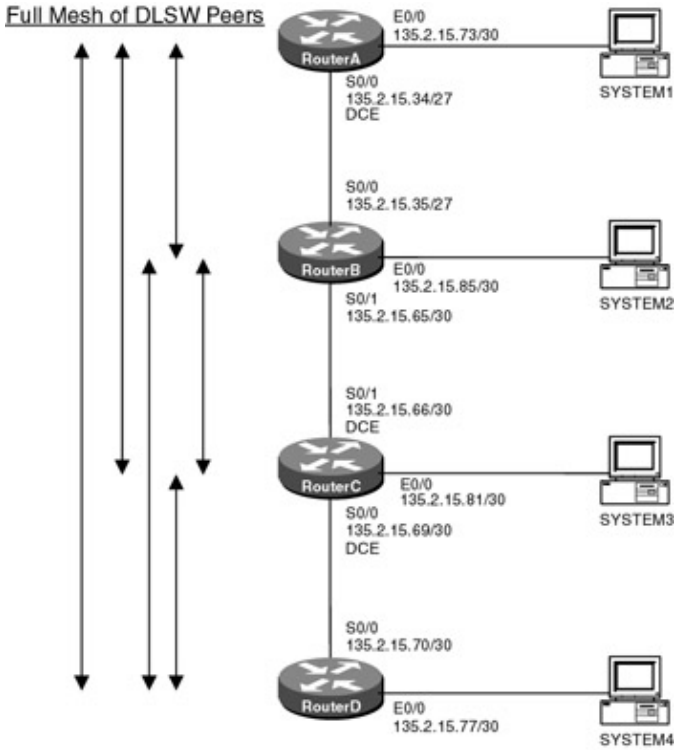


Figure 24−5: DLSW full mesh logical connectivity

Note For instructions on how to configure a workstation to run NetBEUI, see the section at the end of this chapter.

Router Configuration

The configurations for the four routers in this example are as follows. Key DLSW commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.73 ← Configure this router as the local peer
dlsw remote-peer 0 tcp 135.2.15.77 ← Configure each of the three other routers
                                     as remote peers
dlsw remote-peer 0 tcp 135.2.15.81 ← Configure each of the three other routers
                                     as remote peers
dlsw remote-peer 0 tcp 135.2.15.85 ← Configure each of the three other routers
                                     as remote peers
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.10.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.73 255.255.255.252
 bridge-group 1 ← Assign interface E0/0 to bridge group 1
!
interface Serial0/0
 ip address 135.2.15.34 255.255.255.224
```

```
 encapsulation ppp
 clockrate 800000
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee ← Define our spanning tree protocol
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.85
dlsw remote-peer 0 tcp 135.2.15.73
dlsw remote-peer 0 tcp 135.2.15.81
dlsw remote-peer 0 tcp 135.2.15.77
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.12.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.85 255.255.255.252
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.35 255.255.255.224
 encapsulation ppp
!
interface Serial0/1
 ip address 135.2.15.65 255.255.255.252
 encapsulation ppp
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
enable password cisco
!
no ip domain-lookup
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.81
dlsw remote-peer 0 tcp 135.2.15.77
dlsw remote-peer 0 tcp 135.2.15.73
dlsw remote-peer 0 tcp 135.2.15.85
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.9.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.81 255.255.255.252
 no cdp enable
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.69 255.255.255.252
 encapsulation ppp
 no fair-queue
 clockrate 800000
 no cdp enable
!
interface Serial0/1
 ip address 135.2.15.66 255.255.255.252
 encapsulation ppp
 clockrate 800000
 no cdp enable
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterD

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
```

```
hostname RouterD
!
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.77
dlsw remote-peer 0 tcp 135.2.15.81
dlsw remote-peer 0 tcp 135.2.15.73
dlsw remote-peer 0 tcp 135.2.15.85
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.13.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.77 255.255.255.252
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.70 255.255.255.252
 encapsulation ppp
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. We see from the **show dlsw peer** command that RouterA has three connected DLSW peers. Notice that the peer type is listed as configured (conf) for each of the three peers. This tells us that RouterA's peers have been explicitly defined in the router configuration.

```
RouterA# show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.77     CONNECT       3669      3505  conf      0    1   0    1d02h
 TCP 135.2.15.81     CONNECT       5099      3448  conf      0    1   0    1d02h
 TCP 135.2.15.85     CONNECT        453       537  conf      0    1   0 01:44:40
```

The **show dlsw capabilities** command verifies the status and capabilities of each remote peer.

```
RouterA# show dlsw capabilities
DLSw: Capabilities for peer 135.2.15.77(2065) ← Remote peer 135.2.15.77
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
```

```
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  local-ack configured   : yes
  priority configured    : no
  peer type              : conf
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
DLSw: Capabilities for peer 135.2.15.81(2065) ← Remote peer 135.2.15.81
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  local-ack configured   : yes
  priority configured    : no
  peer type              : conf
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
DLSw: Capabilities for peer 135.2.15.85(2065) ← Remote peer 135.2.15.85
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  local-ack configured   : yes
  priority configured    : no
  peer type              : conf
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
```

The **show dlsw capabilities** local command can similarly be used to verify the DLSW capabilities of the local peer.

```
RouterA# show dlsw capabilities local
```

```
DLSw: Capabilities for local peer
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  current border peer    : none
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
```

Now connect to RouterB. Use the **show dlsw peer** command to verify that RouterB has established a DLSW peer to RouterA, RouterB, and RouterC.

```
RouterB# show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.73     CONNECT        537       453  conf      0    1   0 01:44:53
 TCP 135.2.15.81     CONNECT        453       330  conf      0    1   0 01:43:47
 TCP 135.2.15.77     CONNECT        454       418  conf      0    1   0 01:43:18
```

The **show dlsw reach** command will display the NetBIOS names that have been locally cached. Notice that RouterB has seen System1, System3, and System4. These are the three workstations that are connected to RouterA, RouterC, and RouterD.

```
RouterB# show dlsw reach
DLSw Local MAC address reachability cache list
Mac Addr        status     Loc.    port                rif

DLSw Remote MAC address reachability cache list
Mac Addr        status     Loc.    Peer
0008.d296.2e0c  FOUND      REMOTE  135.2.15.73(2065)
000a.204b.a17b  FOUND      REMOTE  135.2.15.77(2065)
000a.204b.e39e  FOUND      REMOTE  135.2.15.81(2065)
000b.0396.4663  FOUND      REMOTE  135.2.15.81(2065)

DLSw Local NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    port                rif

DLSw Remote NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    Peer
System1         FOUND      REMOTE  135.2.15.73(2065)
System3         FOUND      REMOTE  135.2.15.81(2065)
System4         FOUND      REMOTE  135.2.15.77(2065)
```

RouterC and RouterD should also be checked to verify that they each have three DLSW peers. The output of the **show dlsw peer** command for RouterC and RouterD is shown here:

```
RouterC# show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.77     CONNECT       4918      6879  conf      0    0   0    1d02h
 TCP 135.2.15.73     CONNECT       3449      5101  conf      0    1   0    1d02h
 TCP 135.2.15.85     CONNECT        330       453  conf      0    1   0 01:44:06
```

```
RouterD# show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.81     CONNECT        6880      4919  conf      0    0   0    1d02h
 TCP 135.2.15.73     CONNECT        3507      3671  conf      0    1   0    1d02h
 TCP 135.2.15.85     CONNECT         419       455  conf      0    1   0 01:43:50
```

The configuration can also be verified by viewing the entire network neighborhood on any of the four workstations. Each workstation should see the other three workstations.

Lab #104: DLSW Border Peers

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers with 2 serial interfaces and 1 Ethernet interface.
- Two Cisco routers with 1 serial interface and 1 Ethernet interface.
- Three Cisco crossover cables. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- Four workstations running NetBEUI.
- Four Ethernet crossover cables.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports DLSW.

Configuration Overview

This lab will demonstrate DLSW border peers. Border peers are a method of reducing the full mesh requirements of DLSW. The four routers are physically connected as shown in Figure 24−6. The four routers are logically connected as shown in Figure 24−7. Notice the difference between the logical connectivity of this lab and the logical connectivity of our previous lab (DLSW full mesh). The border peer capability allows us to reduce our DLSW peering requirements to the following:

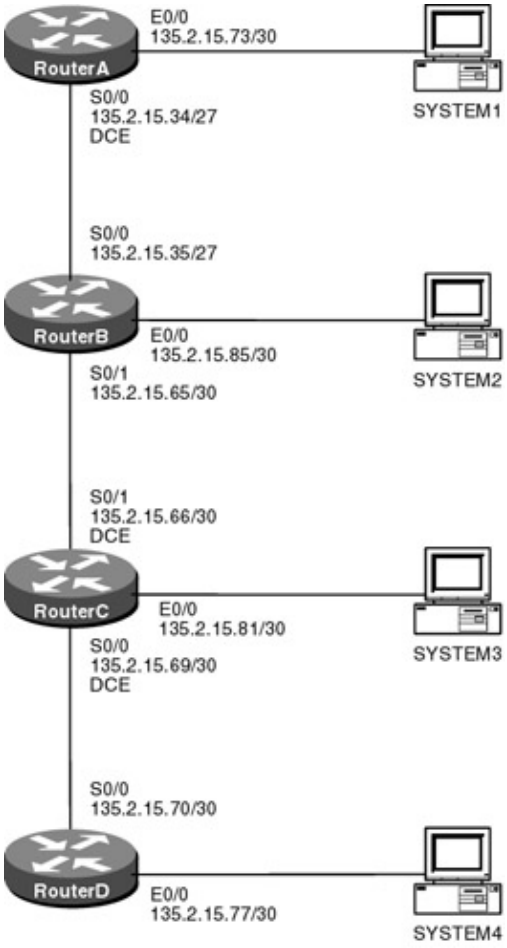


Figure 24−6: DLSW border peers

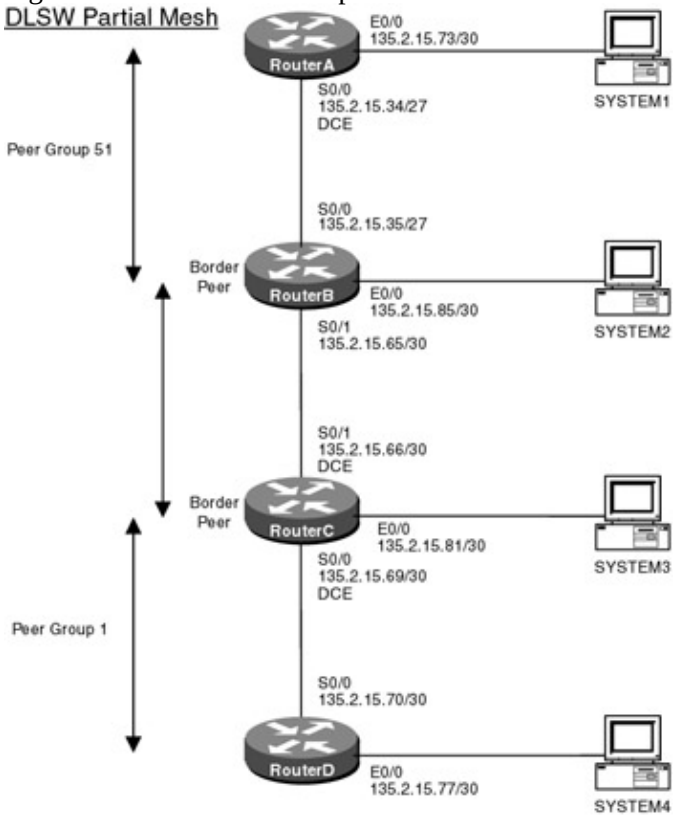


Figure 24−7: DLSW logical connectivity

- Each router within a peer group peers to its DLSW border peer router.
- Each border peer router peers to every other border peer router.

In this lab, RouterA and RouterB are in peer group 51, with RouterB being the border peer. RouterC and RouterD are in peer group 1, with RouterC being the border peer. Our peer requirements are as follows:

- RouterA must peer with RouterB (A router peers with its border peer)
- RouterB must peer with RouterC (Border peer routers must peer)
- RouterC must peer with RouterB (Border peer routers must peer)
- RouterD must peer with RouterC (A router peers with its border peer)

Thus, we need a total of 4 remote peer statements. This contrasts with 12 remote peer statements required in the previous lab.

Note For instructions on how to configure a workstation to run NetBEUI, see the section at the end of this chapter.

Router Configuration

The configurations for the four routers in this example are as follows. Key border peer commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.73 group 51 promiscuous ← Accept a peer
                                                           connection from any
                                                           neighbor
dlsw remote-peer 0 tcp 135.2.15.85 ← Peer with our Border Peer (RouterB)
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.10.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.73 255.255.255.252
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.34 255.255.255.224
 encapsulation ppp
 clockrate 800000
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
```

```
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.85 group 51 border promiscuous ← Define this
                                                                  router as a
                                                                  Border Peer
                                                                  for Peer
                                                                  Group 51
dlsw remote-peer 0 tcp 135.2.15.81 ← Peer with the other Border Peer (RouterC)
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.12.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.85 255.255.255.252
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.35 255.255.255.224
 encapsulation ppp
!
interface Serial0/1
 ip address 135.2.15.65 255.255.255.252
 encapsulation ppp
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterC
!
```

```
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.81 group 1 border promiscuous ← Define this
                                                                 router as a
                                                                 Border Peer
                                                                 for Peer Group
                                                                 1
dlsw remote-peer 0 tcp 135.2.15.85 ← Peer with the other Border Peer (RouterB)
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.9.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.81 255.255.255.252
 no cdp enable
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.69 255.255.255.252
 encapsulation ppp
 no fair-queue
 clockrate 800000
 no cdp enable
!
interface Serial0/1
 ip address 135.2.15.66 255.255.255.252
 encapsulation ppp
 clockrate 800000
 no cdp enable
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterD

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterD
!
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.77 group 1 promiscuous ← Accept a peer
                                                          connection from any
                                                          neighbor
dlsw remote-peer 0 tcp 135.2.15.81 ← Peer with our border peer (RouterC)
dlsw bridge-group 1
!
```

```
interface Loopback0
 ip address 135.2.13.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.77 255.255.255.252
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.70 255.255.255.252
 encapsulation ppp
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA. We see from the **show dlsw peer** command that RouterA has two active peers. Notice from the peer type that one of the peers is conf and one is pod. Conf refers to a peer that is explicitly configured. In the case of RouterA, 135.2.15.85 has been named as a remote peer and is shown as a conf peer type. Pod refers to a Peer on Demand. 135.2.15.77 (RouterD) is shown as a peer on demand. This is because RouterD is not explicitly configured as a peer in RouterA's configuration. The peer relationship is set up as necessary. The promiscuous statement in RouterA's local peer command allows this to happen.

```
RouterA#  show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.77     CONNECT         35        33   pod      0    1   0 00:09:37
 TCP 135.2.15.85     CONNECT      80721     68834  conf      0    1   0 00:10:37
```

Connecting to RouterC also verifies that only the peer to RouterB (135.2.15.85) has been explicitly configured.

```
RouterC#  show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.77     CONNECT         46        59  prom      0    1   0 00:11:13
 TCP 135.2.15.85     CONNECT      51528     56845  conf      0    0   0 00:10:38
```

The **show dlsw peer** command on RouterD verifies that only the peer to RouterC (135.2.15.81) has been explicitly configured.

```
RouterD#  show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.81     CONNECT      71953     37658  conf      0    0   0 00:12:01
 TCP 135.2.15.73     CONNECT         37        41   pod      0    0   0 00:10:13
 TCP 135.2.15.85     CONNECT         43        48   pod      0    1   0 00:10:09
```

Now connect to RouterB. We see from the **show dlsw peer** command that only the peer to RouterC is a configured type peer. The other two peers (to Router D and RouterA) are configured as necessary.

```
RouterB#  show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.81     CONNECT      56850     51538  conf      0    0   0 00:11:13
```

```
 TCP 135.2.15.77     CONNECT         44        41   pod      0    1   0 00:09:56
 TCP 135.2.15.73     CONNECT         38        53  prom      0    1   0 00:11:00
```

The **show dlsw reach** command indicates that RouterB has learned about the workstations on each of the other LANs in our network. We see that SYSTEM1, SYSTEM3, and SYSTEM4 have been cached.

```
RouterB#  show dlsw reach
DLSw Local MAC address reachability cache list
Mac Addr        status     Loc.    port                rif
000a.204b.cbb1  FOUND      LOCAL   TBridge-001    --no rif--

DLSw Remote MAC address reachability cache list
Mac Addr        status     Loc.    Peer
0008.d296.2e0c  FOUND      REMOTE  135.2.15.73(2065)
000a.204b.a17b  FOUND      REMOTE  135.2.15.77(2065)
000a.204b.e39e  FOUND      REMOTE  135.2.15.81(2065)
000b.0396.4663  FOUND      REMOTE  135.2.15.81(2065)
DLSw Local NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    port                rif
SYSTEM2         FOUND      LOCAL   TBridge-001    --no rif--

DLSw Remote NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    Peer
SYSTEM1         FOUND      REMOTE  135.2.15.73(2065)
SYSTEM3         FOUND      REMOTE  135.2.15.81(2065)
SYSTEM4         FOUND      REMOTE  135.2.15.77(2065)
```

This lab can be verified by making sure that each of the four workstations can see all of the other workstations via Windows networking.

Lab #105: DLSW Backup Peers

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with 1 Token Ring interface and 2 serial interfaces.
- Two Cisco routers with 1 serial interface and 1 Ethernet interface.
- Two Cisco crossover cables. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- Three Ethernet cables and an Ethernet hub.
- Two Token Ring cables and a Token Ring MAU.
- Two workstations running NetBEUI.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports bridging.

Configuration Overview

This lab will demonstrate DLSW backup peers. Backup peers allow a DLSW peering session to stay active when a remote peer fails. This is accomplished by having a backup peer reestablish the DLSW session.

Configuring a DLSW backup peer is a bit tricky. Referring to Figure 24−8, note that RouterA will have a primary peer and a backup peer. We will first need to configure RouterA's primary remote peer (152.3.7.1) with the **DLSW remote−peer** statement. Next, we will configure RouterA's backup peer (152.3.7.2). The three routers are connected as shown in Figure 24−8.

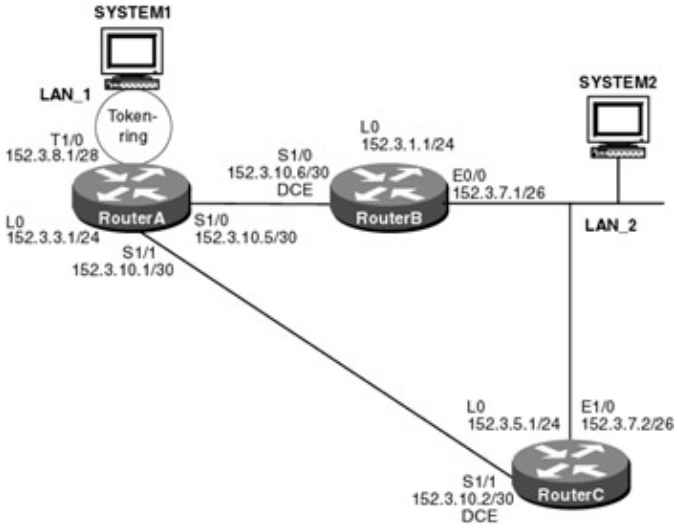


Figure 24−8: DLSW backup peers

Note For instructions on how to configure a workstation to run NetBEUI, see the section at the end of this chapter.

Router Configuration

The configurations for the three routers in this example are as follows. Key DLSW commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
source-bridge ring-group 169 ← Define a virtual ring for our DLSW configuration
dlsw local-peer peer-id 152.3.8.1 ← Define our Token-Ring interface (T1/0) as
                                    our local DLSW peer ID
dlsw remote-peer 0 tcp 152.3.7.1 ← Define our remote peer
dlsw remote-peer 0 tcp 152.3.7.2 backup-peer 152.3.7.1 ← Define our backup peer
!
interface Loopback0
 ip address 152.3.3.1 255.255.255.0
!
interface Serial1/0
 ip address 152.3.10.5 255.255.255.252
 encapsulation ppp
 no fair-queue
!
interface TokenRing1/0
 ip address 152.3.8.1 255.255.255.240
 ring-speed 4
 source-bridge 1 2 169
 source-bridge spanning
!
interface Serial1/1
 ip address 152.3.10.1 255.255.255.252
 encapsulation ppp
!
router ospf 64
 network 152.0.0.0 0.255.255.255 area 0
!
no ip classless
```

```
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterB
!
source-bridge ring-group 169
dlsw local-peer peer-id 152.3.7.1 promiscuous ← Only configure a DLSW local
                                                peer ID. The promiscuous
                                                statement will allow RouterB to
                                                accept a DLSW peer from any
                                                remote peer

dlsw bridge-group 2 ← Associate bridge-group 2 with the DLSW virtual ring 169
!
interface Loopback0
 ip address 152.3.1.1 255.255.255.0
!
interface Ethernet0/0
 ip address 152.3.7.1 255.255.255.192
 bridge-group 2
!
interface Serial1/0
 ip address 152.3.10.6 255.255.255.252
 encapsulation ppp
 clockrate 800000
!
router ospf 64
 network 152.0.0.0 0.255.255.255 area 0
!
ip classless
!
!
!
bridge 2 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
```

```
no service tcp-small-servers
!
hostname RouterC
!
source-bridge ring-group 169
dlsw local-peer peer-id 152.3.7.2 promiscuous
dlsw bridge-group 2
!
interface Loopback0
 ip address 152.3.5.1 255.255.255.0
!
interface Ethernet1/0
 ip address 152.3.7.2 255.255.255.192
 bridge-group 2
!
interface Serial1/1
 ip address 152.3.10.2 255.255.255.252
 encapsulation ppp
 clockrate 800000
!
router ospf 64
 network 152.0.0.0 0.255.255.255 area 0
!
no ip classless
!
bridge 2 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterA and typing the **show dlsw peer** command. Notice that peer 152.3.7.1 is in a connected state. A DLSW-connected peer indicates that the peer is active and functional. We see that there is transmit and receive traffic coming into and out of our peer.

```
RouterA#show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 152.3.7.1       CONNECT       1017       714  conf      0    0   0 05:01:47
 TCP 152.3.7.2       DISCONN          0         0  conf      0    0   -        -
```

The **show dlsw capabilities** command will display information on the remote peer at address 152.3.7.1. Notice that the peer type of the remote peer is listed as being prom. This indicates that 152.3.7.1 is a promiscuous peer. Recall that we did configure our local peer on RouterB as a promiscuous peer.

```
RouterA#show dlsw capabilities
DLSw: Capabilities for peer 152.3.7.1(2065)
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
```

```
  peer cost              : 3
  biu-segment configured : no
  local-ack configured   : yes
  priority configured    : no
  peer type              : prom
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
```

The **show dlsw capabilities local** command is used to query key information about the local DLSW peer.

```
RouterA#show dlsw capabilities  local
DLSw: Capabilities for local peer
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  current border peer    : none
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
```

The **show dlsw reachability** command will display a list of cached NetBIOS names and MAC addresses. Notice that our local workstations' NetBIOS name (SYSTEM1) and our remote workstations' NetBIOS name (SYSTEM2) have been learned and cached by RouterA.

```
RouterA#show dlsw reachability
DLSw Local MAC address reachability cache list
Mac Addr        status     Loc.    port                rif
00a0.24fd.c6d0  FOUND      LOCAL   TokenRing1/0     06B0.0012.0A90

DLSw Remote MAC address reachability cache list
Mac Addr        status     Loc.    Peer
0000.612c.9f32  FOUND      REMOTE  152.3.7.1(2065)
0007.78da.b08e  FOUND      REMOTE  152.3.7.1(2065)

DLSw Local NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    port                rif
SYSTEM1         FOUND      LOCAL   TokenRing1/0     06B0.0012.0A90

DLSw Remote NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    Peer
SYSTEM2         FOUND      REMOTE  152.3.7.1(2065) max-lf(17800)
```

Now switch to RouterB. The **show dlsw peer** command indicates that we have a connected peer at address 152.3.8.1. This is RouterA. Notice that the peer type is promiscuous. This is because we did not configure a remote peer on RouterB. We only configured RouterB with a local peer ID and the promiscuous option.

```
RouterB#show dlsw peer
```

```
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 152.3.8.1       CONNECT        765      1071  prom      0    1   0 05:05:18
```

The **show dlsw reachability** command on RouterB shows us that RouterB has learned and locally cached both of our workstations' NetBIOS names.

```
RouterB#show dlsw reachability
DLSw Local MAC address reachability cache list
Mac Addr        status     Loc.    port                rif
0000.612c.9f32  FOUND      LOCAL   TBridge-002    --no rif--

DLSw Remote MAC address reachability cache list
Mac Addr        status     Loc.    Peer
00a0.24fd.c6d0  FOUND      REMOTE  152.3.8.1(2065) max-lf(1500)

DLSw Local NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    port                rif
SYSTEM2         FOUND      LOCAL   TBridge-002    --no rif--

DLSw Remote NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    Peer
SYSTEM1         FOUND      REMOTE  152.3.8.1(2065) max-lf(1500)
```

The **show dlsw capabilities** command shows the peer capabilities of our remote peer at IP address 152.3.8.1. Notice that our remote peer is a configured peer and not a promiscuous peer. This is because we configured this peer with a remote peer statement.

```
RouterB#show dlsw capabilities
DLSw: Capabilities for peer 152.3.8.1(2065)
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  local-ack configured   : yes
  priority configured    : no
  peer type              : conf
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
```

Now connect to RouterC. The **show dlsw peer** command indicates that there are no active DLSW peers. This is due to the fact that RouterC has been configured as a backup peer. Since the primary DLSW peer between RouterA and RouterB is still active, the backup peer has not been activated.

```
RouterC#show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
```

The **show dlsw capabilities** command confirms that there are no active remote peers.

```
RouterC#show dlsw capabilities
```

```
DLSw: No remote peer capabilities known at this time
```

The **show dlsw capabilities local** command indicates that we have configured our router as a DLSW local peer.

```
RouterC#show dlsw capabilities  local
DLSw: Capabilities for local peer
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  current border peer    : none
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
```

Now reconnect to RouterA. We are going to fail the peer between RouterB and RouterA and monitor how RouterA automatically repeers to RouterC. Enable DLSW debugging with the **debug dlsw reach**, **debug dlsw peer**, and **debug dlsw local** commands.

```
RouterA#debug dlsw reach
DLSw reachability debugging is on at event level for all protocol traffic
RouterA#debug dlsw peer
DLSw peer debugging is on
RouterA#debug dlsw loc
DLSw local circuit debugging is on
```

The **show debug** command indicates that we have enabled debugging for both ends of our DLSW session (152.3.7.1 and 152.3.7.2).

```
RouterA#sh debug
DLSw:
  DLSw Peer debugging is on
DLSw reachability debugging is on at event level for all protocol traffic
  DLSw basic debugging for peer 152.3.7.1(2065) is on
  DLSw basic debugging for peer 152.3.7.2(2065) is on
  DLSw Local Circuit debugging is on
```

The screen print that follows shows what the debug output will look like when the DLSW peer is up and active. Notice that the router sends periodic DLSW keepalive requests to our DLSW neighbor at IP address 152.3.7.1. We see that we receive a response for each keepalive request that is sent.

```
DLSw: Keepalive Request sent to peer 152.3.7.1(2065)) ← Keepalive Request
DLSw: Keepalive Response from peer 152.3.7.1(2065) ← Keepalive Response
DLSw: Keepalive Request sent to peer 152.3.7.1(2065)) ← Keepalive Request
DLSw: Keepalive Response from peer 152.3.7.1(2065) ← Keepalive Response
CSM: Received CLSI Msg: UDATA_STN.Ind   dlen: 215 from TokenRing1/0
CSM:   smac 80a0.24fd.c6d0, dmac c000.0000.0080, ssap F0, dsap F0
CSM: Received frame type NETBIOS DATAGRAM from 00a0.24fd.c6d0, To1/0
CSM: Received CLSI Msg: UDATA_STN.Ind   dlen: 84 from TokenRing1/0
```

```
CSM:   smac 80a0.24fd.c6d0, dmac c000.0000.0080, ssap F0, dsap F0
CSM: Received frame type NETBIOS NAME_QUERY from 00a0.24fd.c6d0, To1/0
CSM: Received CLSI Msg: UDATA_STN.Ind   dlen: 84 from TokenRing1/0
CSM:   smac 80a0.24fd.c6d0, dmac c000.0000.0080, ssap F0, dsap F0
CSM: Received frame type NETBIOS NAME_QUERY from 00a0.24fd.c6d0, To1/0
CSM: Received CLSI Msg: UDATA_STN.Ind   dlen: 84 from TokenRing1/0
CSM:   smac 80a0.24fd.c6d0, dmac c000.0000.0080, ssap F0, dsap F0
CSM: Received frame type NETBIOS NAME_QUERY from 00a0.24fd.c6d0, To1/0
```

Now let's disconnect the cable connected to interface E0/0 on RouterB. The following debug output will be seen. Notice that RouterA sends repeated keepalive responses to the remote peer at IP address 152.3.7.1.

```
DLSw: Keepalive Response sent to peer 152.3.7.1(2065)) ← Repeated Keepalive
                                                         Responses
DLSw: Keepalive Response sent to peer 152.3.7.1(2065))
DLSw: Keepalive Response sent to peer 152.3.7.1(2065))
DLSw: Keepalive Response sent to peer 152.3.7.1(2065))
```

After a period of time, RouterA closes the peer connection to our remote peer at IP address 152.3.7.1.

```
DLSw: dlsw_tcpd_fini() for peer 152.3.7.1(2065)
DLSw: tcp fini closing connection for peer 152.3.7.1(2065) ← TCP connection to
                                                             remote peer is
                                                             closed
DLSw: action_d(): for peer 152.3.7.1(2065)
DLSw: peer 152.3.7.1(2065), old state CONNECT, new state DISCONN
```

RouterA then tries to establish a connection to its backup peer at IP address 152.3.7.2.

```
DLSw: action_a() attempting to connect peer 152.3.7.2(2065) ← Establishing a
                                                              new connection to
                                                              backup peer
DLSw: action_a(): Write pipe opened for peer 152.3.7.2(2065)
DLSw: peer 152.3.7.2(2065), old state DISCONN, new state WAIT_RD
DLSw: passive open 152.3.7.2(11000) -> 2065
DLSw: action_c(): for peer 152.3.7.2(2065)
DLSw: peer 152.3.7.2(2065), old state WAIT_RD, new state CAP_EXG
DLSw: CapExId Msg sent to peer 152.3.7.2(2065)
DLSw: Recv CapExPosRsp Msg from peer 152.3.7.2(2065)
DLSw: action_e(): for peer 152.3.7.2(2065)
DLSw: Recv CapExId Msg from peer 152.3.7.2(2065)
DLSw: Pos CapExResp sent to peer 152.3.7.2(2065)
DLSw: action_e(): for peer 152.3.7.2(2065)
```

After peer negotiation has been completed, the new peer goes into connect state.

```
DLSw: peer 152.3.7.2(2065), old state CAP_EXG, new state CONNECT ← Backup peer
                                                                   is now
                                                                   connected
DLSw: peer_act_on_capabilities() for peer 152.3.7.2(2065)
DLSw: action_f(): for peer 152.3.7.2(2065)
DLSw: closing read pipe tcp connection for peer 152.3.7.2(2065)
```

Notice that RouterA continues to try to establish a connection to its primary peer on RouterB at IP address 152.3.7.1.

```
DLSw: action_a() attempting to connect peer 152.3.7.1(2065)
DLSw: Keepalive Response sent to peer 152.3.7.2(2065))
DLSw: CONN: peer 152.3.7.1 open failed, timed out [10]
DLSw: action_a(): CONN failed - retries 1
DLSw: peer 152.3.7.1(2065), old state DISCONN, new state DISCONN
DLSw: action_a() attempting to connect peer 152.3.7.1(2065)
DLSw: Keepalive Response sent to peer 152.3.7.2(2065))
DLSw: Keepalive Response sent to peer 152.3.7.2(2065))
```

```
DLSw: CONN: peer 152.3.7.1 open failed, timed out [10]
DLSw: action_a(): CONN failed – retries 2
DLSw: peer 152.3.7.1(2065), old state DISCONN, new state DISCONN
DLSw: Keepalive Response sent to peer 152.3.7.2(2065))
DLSw: action_a() attempting to connect peer 152.3.7.1(2065)
DLSw: Keepalive Response sent to peer 152.3.7.2(2065))
```

The **show dlsw peer** command on RouterA shows us that our backup peer is now connected and our primary peer is now disconnected.

```
RouterA#show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 152.3.7.2       CONNECT          9         7  conf      0    0   0 00:02:43
 TCP 152.3.7.1       DISCONN       1139       815  conf      0    0   –        –
```

The **show dlsw reach** command indicates that our NetBIOS names are still being cached. We are now able to reach our remote workstation System2 via our new DLSW peer at IP address 152.3.7.2.

```
RouterA#show dlsw reach
DLSw Local MAC address reachability cache list
Mac Addr        status     Loc.    port                rif
00a0.24fd.c6d0  FOUND      LOCAL   TokenRing1/0     06B0.0012.0A90

DLSw Remote MAC address reachability cache list
Mac Addr        status     Loc.    Peer
0000.612c.9f32  FOUND      REMOTE  152.3.7.2(2065)

DLSw Local NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    port                rif
SYSTEM1         FOUND      LOCAL   TokenRing1/0     06B0.0012.0A90

DLSw Remote NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    Peer
SYSTEM2         FOUND      REMOTE  152.3.7.2(2065) ← SYSTEM2 is now being
                                                     learned via our backup
                                                     remote peer at IP address
                                                     152.3.7.2
```

Now let's reconnect to RouterB. The **show dlsw peer** command indicates that we no longer have any active DLSW peers.

```
RouterB#show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
```

The **show dlsw reach** shows us that we are not caching any remote NetBIOS peer names.

```
RouterB#show dlsw reach
DLSw Local MAC address reachability cache list
Mac Addr        status     Loc.    port                rif
0000.612c.9f32  FOUND      LOCAL   TBridge–002    –-no rif–-
0007.78da.b08e  FOUND      LOCAL   TBridge–002    –-no rif–-

DLSw Remote MAC address reachability cache list
Mac Addr        status     Loc.    peer

DLSw Local NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    port                rif
SYSTEM2         FOUND      LOCAL   TBridge–002    –-no rif–-

DLSw Remote NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    Peer
```

Now connect to RouterC. The **show dlsw peer** command indicates that we are connected to our remote peer on RouterA at IP address 152.3.8.1. Notice that the connection type is promiscuous, since we have not

configured any remote DLSW peers on RouterC.

```
RouterC#show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 152.3.8.1       CONNECT          9        11  prom      0    0   0 00:03:56
```

The **show dlsw reach** command indicates that RouterC has learned and cached the NetBIOS names of the workstations on our two LANs.

```
RouterC#show dlsw reach
DLSw Local MAC address reachability cache list
Mac Addr        status     Loc.    port                rif
0000.612c.9f32  FOUND      LOCAL   TBridge-002    --no rif--
0007.78da.6480  FOUND      LOCAL   TBridge-002    --no rif--
00a0.24fd.c6d0  FOUND      LOCAL   TBridge-002    --no rif--

DLSw Remote MAC address reachability cache list
Mac Addr        status     Loc.    peer

DLSw Local NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    port                rif
SYSTEM2         FOUND      LOCAL   TBridge-002    --no rif--
SYSTEM1         FOUND      LOCAL   TBridge-002    --no rif--

DLSw Remote NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    Peer
```

The **show dlsw capabilities local** and **show dlsw capabilities remote** commands show that our peer is properly configured.

```
RouterC#show dlsw capabilities local
DLSw: Capabilities for local peer
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  current border peer    : none
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
RouterC#show dlsw capabilities remote
DLSw: Capabilities for peer 152.3.8.1(2065)
  vendor id (OUI)        : '00C' (cisco)
  version number         : 1
  release number         : 0
  init pacing window     : 20
  unsupported saps       : none
  num of tcp sessions    : 1
  loop prevent support   : no
  icanreach mac-exclusive: no
  icanreach netbios-excl.: no
  reachable mac addresses: none
```

```
  reachable netbios names: none
  cisco version number   : 1
  peer group number      : 0
  border peer capable    : no
  peer cost              : 3
  biu-segment configured : no
  local-ack configured   : yes
  priority configured    : no
  peer type              : conf
  version string         :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3620-JS-M), Version 11.2(20)P,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Mon 11-Oct-99 21:14 by jaturner
```

Lab #106: Access Expressions

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers. Each router must have 1 serial interface and 1 Ethernet interface.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- One Ethernet crossover cable.
- Four workstations running NetBEUI.
- Four Ethernet cables and one Ethernet hub.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports DLSW.

Configuration Overview

This lab will demonstrate NetBIOS access lists and access expressions. An access expression is a logical combination of multiple access lists. This lab creates an access expression that only allows access to certain NetBEUI attached workstations on a LAN. SYSTEM1 (which is connected to RouterA) will only be allowed to see SYSTEM2 and SYSTEM3 via Windows networking. SYSTEM4 will not be visible to SYSTEM1 due to our configured access expression.

The three routers are connected as shown in Figure 24-9. RouterA acts as a DCE and supplies clocking to RouterB.

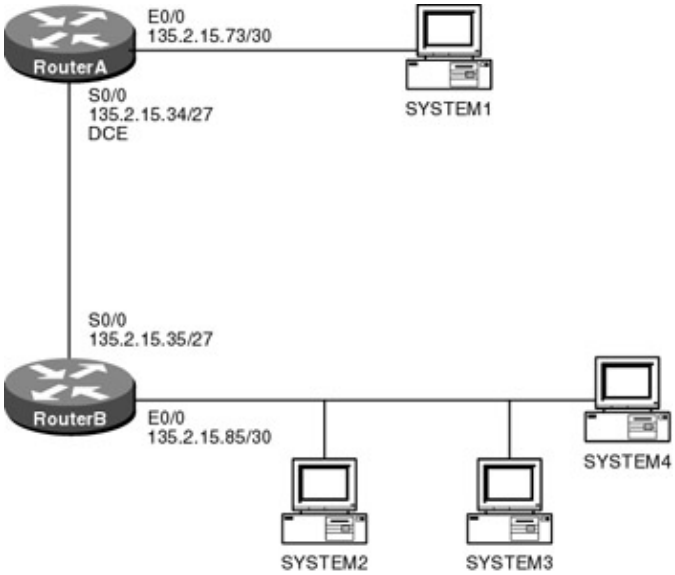


Figure 24−9: Access expressions

Note For instructions on how to configure a workstation to run NetBEUI, see the section at the end of this chapter.

Router Configuration

The configurations for the two routers in this example are as follows. Key access expression commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 11.2
no service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname RouterA
!
netbios access-list host test permit SYSTEM2 ← Configure an access list to only
                                                permit SYSTEM2
netbios access-list host test deny *
netbios access-list host rest permit SYSTEM3 ← Configure an access list to only
                                                permit SYSTEM3
netbios access-list host rest deny *
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.73
dlsw remote-peer 0 tcp 135.2.15.85
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.10.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.73 255.255.255.252
 access-expression input (netbios-host(rest) | netbios-host(test)) ← Apply the
                                                                     access
                                                                     list to
                                                                     E0/0
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.34 255.255.255.224
 encapsulation ppp
```

```
 clockrate 800000
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

RouterB

```
Current configuration:
!
version 11.2
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname RouterB
!
enable password cisco
!
source-bridge ring-group 100
dlsw local-peer peer-id 135.2.15.85
dlsw remote-peer 0 tcp 135.2.15.73
dlsw bridge-group 1
!
interface Loopback0
 ip address 135.2.12.1 255.255.255.0
!
interface Ethernet0/0
 ip address 135.2.15.85 255.255.255.252
 bridge-group 1
!
interface Serial0/0
 ip address 135.2.15.35 255.255.255.224
 encapsulation ppp
!
router ospf 64
 network 135.2.0.0 0.0.255.255 area 0
!
no ip classless
!
bridge 1 protocol ieee
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Let's start by connecting to RouterB. The **show dlsw peer** command should indicate that RouterB has a connected peer to RouterA.

```
RouterB# show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
 TCP 135.2.15.73     CONNECT        537       453  conf      0    1   0 01:44:53
```

Now connect to RouterA. RouterA should have a connected peer to RouterB.

```
RouterA# show dlsw peer
Peers:                state     pkts_rx   pkts_tx  type  drops ckts TCP   uptime
TCP 135.2.15.85     CONNECT        453       537  conf      0    1    0 01:44:40
```

The **show access−expression** command will indicate that our access expression has been applied to interface E0/0.

```
RouterA#show access-expression
Interface Ethernet0/0:
   Input: (netbios-host(rest) | netbios-host(test))
```

The reader should check the configuration by going on SYSTEM1 and verifying that only workstations SYSTEM2 and SYSTEM3 are visible from SYSTEM1.

Workstation Configuration to Run NetBEUI

The following steps will enable the reader to configure a workstation to run Windows networking using only the NetBEUI protocol. Workstations running only NetBEUI can be used to verify that the bridging and DLSW labs are functioning properly.

Figure 24−10 shows the Windows Network configuration screen. This screen is displayed when the Network icon is selected in the Windows control panel. We see that the only protocol selected is NetBEUI. In addition, we have loaded the Client for Microsoft Networks, the Microsoft Family Logon, and File and Printer Sharing for Microsoft Networks. We see that our network card on the computer shown in Figure 24−10 is a 3Com 10/100 card.

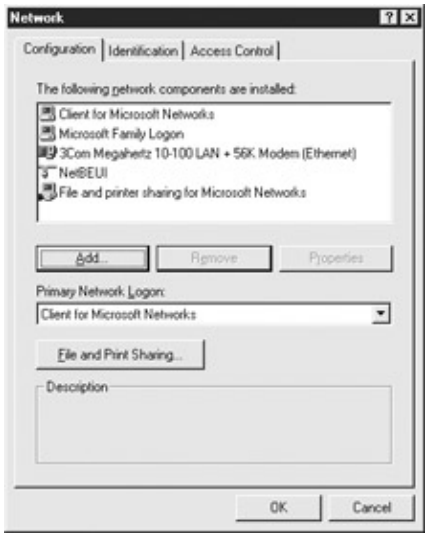


Figure 24−10: Windows networking configuration

Clicking the File and Print Sharing box will bring you to the screen shown in Figure 24−11. You need to select file access in order for others to have access to your files.

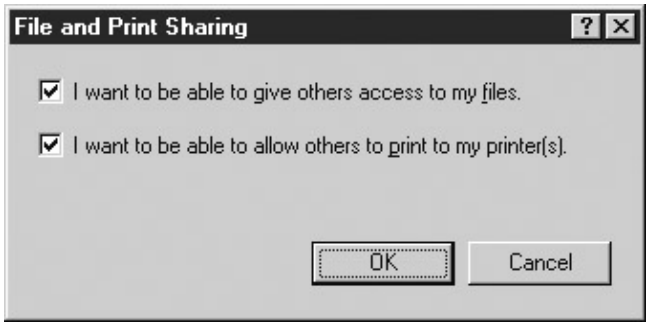


Figure 24−11: Windows file and print sharing configuration

Clicking the Network Adapter card, shown in Figure 24−10, will take you to the Network card configuration screens. We see in Figure 24−12 that the only protocol that is bound to the network adapter card is NetBEUI.

Figure 24−12: Network card bindings

Clicking the Identification tab shown in Figure 24−10 will display the identification screen shown in Figure 24−13. We see that we have set the name of this computer to System3.

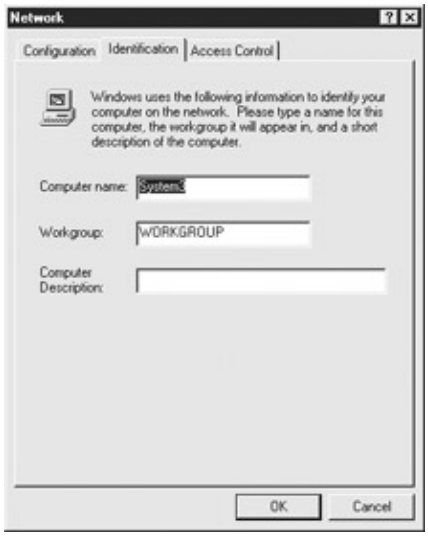


Figure 24−13: Network identification

Once a system has been configured with the steps shown previously, it will be ready to run Windows networking. The system will need to be restarted before the changes take effect. Once the system reloads, you will need to supply a network login password to log into the network. The password is arbitrary and can be set to any value.

The proper configuration of the DLSW and bridging labs can be verified by making sure that workstations attached to the routers can see all of the other workstations via Network Neighborhood. As shown in Figure 24−14, other NetBEUI workstations can be found by right−clicking the Network Neighborhood icon. Select the Find Computer option. When prompted, enter the name of the system that you want to find.

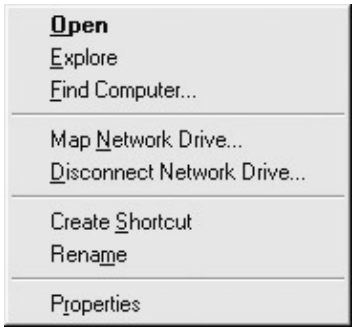


Figure 24−14: Network Neighborhood Find Computer

As an alternative to right−clicking the Network Neighborhood icon to find a neighbor computer, you can double−click the Network Neighborhood icon. This will bring up a screen similar to the one shown in Figure 24−15. Notice that a workstation named System1 has been found on the network.

Figure 24−15: Network Neighborhood

Conclusion

This chapter discussed the topic of bridging and DLSW. We explored several topics, such as DLSW border peers, access expressions, and DLSW backup peers.

Chapter 25: IPSec

Overview

Topics Covered in This Chapter

- Technology Overview
- Basic IPSec Tunnel Mode Using ESP−3DES
- IPSec and NAT
- OSPF over IPSec Using a GRE Tunnel
- Tunnel Endpoint Discovery (TED)

Introduction

IPSec is a framework of open standards for ensuring secure private communications over IP networks. IPSec ensures data integrity and authenticity as well as confidentiality (encryption) across IP networks.

IPSec protocols offer security services at the IP layer through Authentication Header (AH) and the Encapsulating Security Payload (ESP) protocols. These security services include access control, connectionless integrity, data authentication, protection against replay, and encryption.

Technology Overview

Authentication Header (AH)

AH mode provides authentication for the IP header and the payload contained in the IP packet. It does this by using a keyed hash through a shared secret value. The AH service protects the external IP header and payload. It protects all of the fields in the IP header that do not change during transit. Fields such as the TTL, TOS, and Header checksum are zeroed before the integerity check value is calculated. This is necessary since the values may change during transit and the sender has no way of predicting what they might be when they reach the receiver.

The AH header is placed in the packet beween the original IP header and the payload:

```
---------------------------------
|orig IP hdr  |    |     |      |
|(any options)| AH | TCP | Data |
---------------------------------
|<------- authenticated ------->|
```

Encapsulating Security Payload (ESP)

Encapsulating Security Payload (ESP) provides payload encryption at the IP packet layer. It can also provide authentication through the use of an optional Integrity Check Value (ICV). If authentication is used, the value is calculated after the encryption is done.

The following is what an IP packet would look like if ESP with authentication were used in tunnel mode. The two modes of IPSec will be discussed later in the chapter.

```
-----------------------------------------------------------
| new IP hdr  |     | orig IP hdr   |   |    | ESP   | ESP|
|             | ESP |               |TCP|Data|Trailer|Auth|
-----------------------------------------------------------
|<--------- encrypted ---------->|
```

```
|<---------- authenticated ---------->|
```

IPSec Modes of Operation

There are two options for implementing IPSec. The first is for the end stations to provide security directly (transport mode). The advantage of this is it has no impact on the network. The network design, topology, and routing have no impact on the security services. The disadvantage is that all end stations must run IPSec software and users must be "IPSec aware." (Meaning they are responsible for determining what traffic is to be encrypted and/or authenticated.) This requires a certain understanding by the end user since, in most cases, configuration changes are needed.

The second option (tunnel mode) is totally transparent to the end user. With tunnel mode, the network provides the security. The advantage, of course, is the end users are not directly involved and don't need to be "IPSec aware." The disadvantage is the network topology and routing must be taken into consideration and the routers will require additional processing power to perform the IPSec services.

Transport Mode

With transport mode only, the IP payload is encrypted while the original headers are left intact. The ESP header is inserted after the IP header and before the upper−layer protocol header. The upper−layer protocols are encrypted and authenticated along with the ESP header. ESP does not authenticate the IP header itself. Authentication protects the external IP header along with the data payload, protecting all the fields in the header that do not change in transit.

```
-------------------------------------------------
|orig IP hdr  | ESP |     |      |   ESP   | ESP|
|             | Hdr | TCP | Data | Trailer |Auth|
-------------------------------------------------
|<----- encrypted ---->|
|<------ authenticated ----->|
```

Table 25−1 shows the services provided by the different IPSec protocols in transport mode.

Table 25−1: Transport Mode Services Provided

Security Service	AH	ESP	AH & ESP
Access control	Yes	Yes	Yes
Connectionless integrity Data origin authentication	Authenticates parts of original IP header along with upper−layer protocols	Encrypts and optionally authenticates upper−layer protocol and does not authenticate original IP header	Encrypts and authenticates upper−layer protocol and authenticates parts of original IP header
Encryption	No	Yes	Yes
Hiding the original Source & Destination	No	No	No

Tunnel Mode

In tunnel mode, the entire IP packet is encrypted and becomes the payload of a new IP packet. The new IP header has the destination address of its IPSec peer. All of the information from the packet is protected, including the header. Tunnel mode protects against traffic analysis since the true endpoints of the packet cannot be determined — only the endpoints of the IPSec tunnel are visible.

Authentication can be provided through AH or by the authentication option in ESP. If AH is used, both the original and the new header, along with the payload, are authenticated. If the authentication option in ESP is

used, only the original IP datagram and the ESP header are protected; the new IP header is not authenticated.

Table 25−2 shows the services provided by the different IPSec protocols in tunnel mode.

Table 25−2: Tunnel Mode Services Provided

Security Service	AH	ESP	AH & ESP
Access control	Yes	Yes	Yes
Connectionless integrity Data origin authentication	Authenticates all of original IP datagram and parts of new IP header	Encrypts and optionally authenticates original IP datagram and does not authenticate new IP header	Encrypts and authenticates original IP datagram and authenticates parts of new IP header
Encryption	No	Yes	Yes
Hiding the original source & destination	Yes	Yes	Yes

How IPSec Works

Since the scenarios in this lab deal with tunnel mode using preshare authentication and IKE, we will describe how this works.

When the router receives a packet, it checks its security policy to determine whether or not the packet should be protected. With IPSec, you define what data should be secured through the use of access lists. If the traffic matches the access list, it is protected using the defined security protocols.

From the defined policy, the router determines what security services should be applied to the packet and the address to use as the endpoint of the IPSec tunnel. The router then checks to see if a security association already exists.

If none is present, the router will need to negotiate one with the associated peer. To do this, IKE is used to create an authenticated, secure tunnel between two routers over which the security association for IPSec can be negotiated. IKE first authenticates its peer; this can be done using preshared keys, public−key cryptography, or digital signatures. Once the peer is authenticated, the Diffie−Hellman protocol is used to agree on a common session key.

At this point, the IPSec security association is negotiated over the secure tunnel. The peer that initiates the session will send its configured ISAKMP policy to its remote peer. The policy contains the defined encryption algorithm, the hash algorithm, the authentication method, the Diffe−Hellman group, and the lifetime. This policy can be viewed on the router using the command **show crypto isakmp policy.**

Subsequent to receiving the ISAKMP policy, the remote peer will try to find a matching policy. A match is made when the receiving peer finds a configured policy that has the same encryption algorithm, hash algorithm, authentication type, and Diffe−Hellman parameters as the ones being sent by the remote peer. The lifetime of the remote peer must also be less than or equal to the one configured on the router. After the two sides have agreed on which algorithms to use, they derive keys. The key used by IPSec is different than the one used by IKE. The IPSec shared key can be derived by using Diffie−Hellman again or can be derived by taking the IKE negotiated key and hashing it with a pseudo−random number.

Once the policies are agreed upon, IKE will complete the negotiation process and a security association will be created. The security association is used to track all the specifics concerning a given IPSec communication session. It is uniquely identified through the combination of a randomly chosen unique number called the security parameter index (SPI) and the IP address of the destination.

Once established, the security association (SA) is then applied to all traffic that matches that particular access list that caused the initial negotiation. SAs are unidirectional, only requiring two SAs for each session. The corresponding security association is used when processing incoming or outgoing traffic from that peer. Each SA has a lifetime equal to the negotiated value. Once the SA expires, a new SA will need to be established.

Commands Discussed in This Chapter

- **clear crypto sa**
- **crypto dynamic−map**
- **crypto ipsec security−association lifetime**
- **crypto ipsec transform−set**
- **crypto map (global)**
- **crypto map (interface)**
- **debug crypto isakmp**
- **debug crypto ipsec**
- **debug crypto engine**
- **set peer**
- **show crypto ipsec sa**
- **show crypto ipsec transform−set**
- **show crypto dynamic−map**
- **show crypto map**

Definitions

clear crypto sa: This exec command deletes all IPSec security associations. Once the SA is removed, any future IPSec traffic will require new security associations to be negotiated.

crypto dynamic−map: This global configuration command is used to create a dynamic crypto map. A dynamic map is used when processing negotiation requests for new security associations from a remote IPSec peer, when the crypto map parameters required to communicate with that peer are not known.

crypto ipsec security−association lifetime: This global configuration command is used to set the lifetime in seconds of the negotiated security association. Once the lifetime expires, the SA is removed and a new one is established.

crypto ipsec transform−set: This global configuration command defines the security protocols and algorithms that get applied to IPSec protected traffic.

crypto map (global): This global configuration command classifies what traffic should be protected and defines the policy to be applied to that traffic.

crypto map (interface): This interface configuration command assigns a crypto map to a particular interface.

debug crypto isakmp: This debug command displays messages about IKE events.

debug crypto ipsec: This debug command displays messages about IPSec events.

debug crypto engine: This command will display information from the crypto engine.

set peer: This crypto map configuration command specifies an IPSec peer for a crypto map.

show crypto ipsec sa: This command is used to view the settings used by the current security associations.

show crypto ipsec transform−set: This command is used to view all configured transform sets on the router.

show crypto dynamic−map: This command is used to view all dynamic crypto maps configured on the router.

show crypto map: This command is used to view all crypto maps configured on the router.

IOS Requirements

IPSec first appeared in IOS version 11.3. All labs in this chapter were done using IOS 12.0.

Lab #107: Basic IPSec Tunnel Mode Using ESP−3DES

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having two Ethernet and two serial ports
- Cisco IOS 12.0 capable of running IPSec
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- A Cisco rolled cable

Configuration Overview

This lab will demonstrate a basic IPSec configuration. All traffic between network 135.25.3.0 and network 135.25.4.0 will be encrypted using ESP−3DES. The routers will use IKE to create the security association using preshared keys.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. OSPF will be run on all networks on both RouterA and RouterB. The IP addresses are assigned as per Figure 25−1.

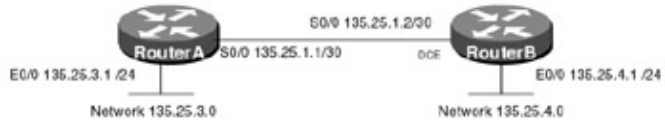


Figure 25−1: Basic IPSec

Router Configurations

The configurations for the two routers in this example are as follows (IPSec configurations have not been added — the step−by−step configuration is provided in the monitoring and testing section).

RouterA

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA

!
interface Ethernet0/0
 ip address 135.25.3.1 255.255.255.0
 no ip directed-broadcast
```

```
 no keepalive
!
interface Serial0/0
ip address 135.25.1.1 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
 line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
interface Ethernet0/0
 ip address 135.25.4.1 255.255.255.0
 no ip directed-broadcast
 no keepalive
!
interface Serial0/0
 ip address 135.25.1.2 255.255.255.252
no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 1000000
!
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

The first step is to create an IKE crypto policy. IKE is used to create the security association (SA) between the routers. Under the IKE policy, the encryption algorithm, hash algorithm, authentication method, Diffie-Hellman group identifier, and SA lifetime are configured. With the exception of the SA lifetime, if these parameters do not match, the SA negotiation will fail. The following command configures the IKE

policy 1 on RouterA and RouterB:

```
RouterA(config)#crypto isakmp policy 1

RouterB(config)#crypto isakmp policy 1
```

Up to 10,000 policies can be defined. The priority value (the number at the end of the command) is used to determine the priority of each policy — the lower the number, the higher the priority. During negotiation, each policy is evaluated in order of priority. If no policies are explicitly configured, the default parameters are used to define the policy. The default parameters are listed in Table 25−3.

The authentication type is then defined under the IKE policy. Three types of authentication can be used: RSA−SIG, RSA−ENCR, or preshare. For this lab, we will be using preshare keys. If RSA−Sig were used, a certificate authority would be needed. The following command enables the IKE policy to use preshare keys:

```
RouterA(config)#crypto isakmp policy 1
RouterA(config-isakmp)#authentication pre-share

RouterB(config)#crypto isakmp policy 1
RouterB(config-isakmp)#authentication pre-share
```

Since preshared keys are being used, a shared key must be defined along with the peer's identity. The identity can be either the peer's IP address or name. For this lab, the IP address of the serial interface of each router will be the peer address and the shared secret will be cisco. The following command defines the key that will be used and the peer address:

```
RouterA(config)#crypto isakmp key cisco address 135.25.1.2

RouterB(config)#crypto isakmp key cisco address 135.25.1.1
```

View the crypto policy on RouterA with the **show crypto isakmp policy** command. The following is the output from the command. Notice that two policies exist: the one we just created and a default policy. The only difference is the authentication method. The default policy is using RSA−SIG as the authentication method, while the policy we just created is using preshared key.

Table 25−3: Default Policy Parameters

Parameter	Default Value
Encryption algorithm	DES
Hash algorithm	SHA−1
Authentication method	RSA signatures
Diffie−Hellman identifier group	Group 1 (768−bit DH)
Security Association's lifetime	86,400 seconds

```
RouterA#show crypto isakmp policy
Protection suite of priority 1
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
Default protection suite
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Rivest-Shamir-Adleman Signature
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
```

The next step is to define the transform set or sets that will be used. A transform is simply the algorithm or algorithms that the router is willing to use for the session. The various transform sets are offered to the receiver during IKE. The receiver selects the one that will be used.

For this lab, we will be using encryption only — just one transform needs to be defined. The following is a list of transforms supported:

```
  h-md5-hmac    AH-HMAC-MD5 transform
  ah-sha-hmac   AH-HMAC-SHA transform
  comp-lzs      IP Compression using the LZS compression algorithm
  esp-3des      ESP transform using 3DES(EDE) cipher (168 bits)
  esp-des       ESP transform using DES cipher (56 bits)
  esp-md5-hmac  ESP transform using HMAC-MD5 auth
  esp-null      ESP transform w/o cipher
  esp-sha-hmac  ESP transform using HMAC-SHA auth
```

The following command defines the transform set on RouterA and RouterB:

```
RouterA(config)#crypto ipsec transform-set encr-only esp-3des


RouterB(config)#crypto ipsec transform-set encr-only esp-3des
```

The transform set that is going to be used can be viewed with the command, **show crypto ipsec transform-set.** The following is the output from the command on RouterA, notice that the transform set encr-only is using esp-3des.

```
RouterA#show crypto ipsec transform-set

Transform set encr-only: { esp-3des  }
   will negotiate = { Tunnel,  },
```

The next step is to define the traffic that will be given security protection. This is done using an extended access list. For this lab, all traffic from network 135.25.3.0 to network 135.25.4.0 should be encrypted. Only the traffic that you wish to protect needs to be defined. In access-list terminology, "permit" means "protect" and "deny" means "don't protect." Any traffic that is denied will pass in the clear.

The following commands define the access lists on RouterA and RouterB:

```
RouterA(config)# access-list 101 permit ip 135.25.3.0 0.0.0.255 135.25.4.0 0.0.0.255

RouterB(config)# access-list 101 permit ip 135.25.4.0 0.0.0.255 135.25.3.0 0.0.0.255
```

The next step is to define a crypto map, which combines the policy and traffic information. The crypto map contains the traffic to which security must be applied (defined by the access list), the actual algorithm to apply (defined by the transform) and the crypto endpoint (the remote peer). An IPSec crypto map is defined with a tag, a sequence number, and the encryption method.

The following commands define a crypto map on RouterA and RouterB:

```
RouterA(config)#crypto map peer-RouterB local-address serial 0/0
RouterA(config)#crypto map peer-RouterB 10 ipsec-isakmp
RouterA(config-crypto-map)#set peer 135.25.1.2
RouterA(config-crypto-map)#set transform-set encr-only
RouterA(config-crypto-map)#match address 101

RouterB(config)#crypto map peer-RouterA local-address loopback s0/0
RouterB(config)#crypto map peer-RouterA 10 ipsec-isakmp
RouterB(config-crypto-map)#set peer 135.251.1
RouterB(config-crypto-map)#set transform-set encr-only
RouterB(config-crypto-map)#match address 101
```

The last thing is to apply the crypto map to an interface. You must assign a crypto map set to an interface before that interface can provide IPSec services. The following commands assign the crypto map to the serial interface connecting RouterA and RouterB:

```
RouterA(config)#interface s0/0
RouterA(config-if)#crypto map peer-RouterB

RouterB(config)#interface s0/0
RouterB(config-if)#crypto map peer-RouterA
```

Display the active IPSec connections on RouterA with the command **show crypto engine connections active.** The following is the output from the command. Notice that there are no active connections. This is because no traffic matching the crypto map has been sent.

```
RouterA#sho crypto engine connections active

  ID Interface       IP-Address      State  Algorithm           Encrypt  Decrypt
Crypto adjacency count : Lock: 0, Unlock: 0
```

Turn on the following debug commands on RouterA:

```
RouterA#debug crypto ipsec
RouterA#debug crypto isakmp
```

Now ping from RouterA sourcing the packet from the Ethernet interface (135.25.3.1), to 135.25.4.1. The following is the output from the debug commands:

```
RouterA#ping ← Extended ping sourcing the packet from the Ethernet interface
Protocol [ip]:
Target IP address: 135.25.4.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 135.25.3.1 ← Source address is the Ethernet
                                          interface of RouterA
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 135.25.4.1, timeout is 2 seconds:

00:23:56: IPSEC(sa_request):
,  (key eng. msg.) src= 135.25.1.1, dest= 135.25.1.2, ← Using the serial
                                                        address as the source
                                                        as defined in (crypto
                                                        map peer-RouterB local
                                                        address serial 0/0)
    src_proxy= 135.25.3.0/255 .255.255.0/0/0 (type=4), ← Proxy is the real
                                                         source and destination
    dest_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-3des , ← Using 3DES
    lifedur= 3600s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4004
00:23:56: ISAKMP (0:1): beginning Main Mode exchange
00:23:56: ISAKMP (1): sending packet to 135.25.1.2 (I) MM_NO_STATE
00:23:56: ISAKMP (1): received packet from 135.25.1.2 (I) MM_NO_STATE.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 12/13/16 ms
RouterA#
00:23:56: ISAKMP (0:1): processing SA payload. message ID = 0
00:23:56: ISAKMP (0:1): Checking ISAKMP transform 1 against priority 1
```

```
policy ← Beginning of negotiation

00:23:56: ISAKMP:      encryption DES-CBC
00:23:56: ISAKMP:      hash SHA
00:23:56: ISAKMP:      default group 1
00:23:56: ISAKMP:      auth pre-share
00:23:56: ISAKMP (0:1): atts are acceptable. Next payload is 0
00:23:56: ISAKMP (0:1): SA is doing pre-shared key authentication
00:23:56: ISAKMP (1): SA is doing pre-shared key authentication using id type ID
_IPV4_ADDR
00:23:56: ISAKMP (1): sending packet to 135.25.1.2 (I) MM_SA_SETUP
00:23:56: ISAKMP (1): received packet from 135.25.1.2 (I) MM_SA_SETUP
00:23:56: ISAKMP (0:1): processing KE payload. message ID = 0
00:23:56: ISAKMP (0:1): processing NONCE payload. message ID = 0
00:23:56: ISAKMP (0:1): SKEYID state generated
00:23:56: ISAKMP (0:1): processing vendor id payload
00:23:56: ISAKMP (0:1): speaking to another IOS box!
00:23:56: ISAKMP (1): ID payload
        next-payload : 8
        type         : 1
        protocol     : 17
        port         : 500
        length       : 8
00:23:56: ISAKMP (1): Total payload length: 12
00:23:56: ISAKMP (1): sending packet to 135.25.1.2 (I) MM_KEY_EXCH
00:23:56: ISAKMP (1): received packet from 135.25.1.2 (I) MM_KEY_EXCH
00:23:56: ISAKMP (0:1): processing ID payload. message ID = 0
00:23:56: ISAKMP (0:1): processing HASH payload. message ID = 0
00:23:56: ISAKMP (0:1): SA has been authenticated with 135.25.1.2
00:23:56: ISAKMP (0:1): beginning Quick Mode exchange, M-ID of -346695504
00:23:56: IPSEC(key_engine): got a queue event . . .
00:23:56: IPSEC(spi_response): getting spi 487787706 for SA
        from 135.25.1.2      to 135.25.1.1      for prot 3
00:23:57: ISAKMP (1): sending packet to 135.25.1.2 (I) QM_IDLE
00:23:57: ISAKMP (1): received packet from 135.25.1.2 (I) QM_IDLE
00:23:57: ISAKMP (0:1): processing SA payload. message ID = -346695504
00:23:57: ISAKMP (0:1): Checking IPSec proposal 1
00:23:57: ISAKMP: transform 1, ESP_3DES
00:23:57: ISAKMP:   attributes in transform:
00:23:57: ISAKMP:      encaps is 1
00:23:57: ISAKMP:      SA life type in seconds
00:23:57: ISAKMP:      SA life duration (basic) of 3600
00:23:57: ISAKMP:      SA life type in kilobytes
00:23:57: ISAKMP:      SA life duration (VPI) of  0x0 0x46 0x50 0x0
00:23:57: ISAKMP (0:1): atts are acceptable.
00:23:57: IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) dest= 135.25.1.2, src= 135.25.1.1,
    dest_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),
    src_proxy= 135.25.3.0/255 .255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-3des ,
    lifedur= 0s and 0kb,
     spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
00:23:57: ISAKMP (0:1): processing NONCE payload. message ID = -346695504
00:23:57: ISAKMP (0:1): processing ID payload. message ID = -346695504
00:23:57: ISAKMP (0:1): processing ID payload. message ID = -346695504
00:23:57: ISAKMP (0:1): Creating IPSec Sas ← SAs being created
00:23:57:         inbound SA from 135.25.1.2      to 135.25.1.1      (proxy 135.
25.4.0     to 135.25.3.0    )
00:23:57:         has spi 487787706 and conn_id 2000 and flags 4 ← Conn id 2000
                                                                   is used for
                                                                   inbound
                                                                   traffic
00:23:57:         lifetime of 3600 seconds
00:23:57:         lifetime of 4608000 kilobytes
00:23:57:         outbound SA from 135.25.1.1      to 135.25.1.2      (proxy 135
.25.3.0     to 135.25.4.0    )
00:23:57:         has spi 583664933 and conn_id 2001 and flags 4 ← Conn id 2001
                                                                   is used for
```

```
                                                                    outbound
                                                                    traffic
00:23:57:         lifetime of 3600 seconds
00:23:57:         lifetime of 4608000 kilobytes
00:23:57: ISAKMP (1): sending packet to 135.25.1.2 (I) QM_IDLE
00:23:57: ISAKMP (0:1): deleting node −346695504
00:23:57: IPSEC(key_engine): got a queue event . . .
00:23:57: IPSEC(initialize_sas):,
  (key eng. msg.) dest= 135.25.1.1, src= 135.25.1.2,
    dest_proxy= 135.25.3.0/255 .255.255.0/0/0 (type=4),
    src_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp−3des,
    lifedur= 3600s and 4608000kb,
    spi= 0x1D130CBA(487787706), conn_id= 2000, keysize= 0, flags= 0x4
00:23:57: IPSEC(initialize_sas):,
  (key eng. msg.) src= 135.25.1.1, dest= 135.25.1.2,
    src_proxy= 135.25.3.0/255 .255.255.0/0/0 (type=4),
    dest_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp−3des,
    lifedur= 3600s and 4608000kb,
    spi= 0x22CA0525(583664933), conn_id= 2001, keysize= 0, flags= 0x4
00:23:57: IPSEC(create_sa): sa created,
  (sa) sa_dest= 135.25.1.1, sa_prot= 50,
```

Now display the crypto engine on RouterA with the command **show crypto engine connections active**. The following is the output from the command. Notice that RouterA now has two SAs: one for outgoing traffic and one for incoming traffic.

```
RouterA#show crypto engine connections active

  ID Interface     IP−Address      State   Algorithm           Encrypt  Decrypt
   1 <none>        <none>          set     HMAC_SHA+DES_56_CB        0        0
2000 Serial0/0     135.25.1.1      set     3DES_56_CBC               0        4
2001 Serial0/0     135.25.1.1      set     3DES_56_CBC               4        0
Crypto adjacency count : Lock: 0, Unlock: 0
```

Lab #108: IPSec and NAT

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having two Ethernet and two serial ports
- Cisco IOS 12.0 capable of running IPSec
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- A Cisco rolled cable

Configuration Overview

This lab will demonstrate how to configure IPSec in conjunction with NAT. Static NAT will be configured on RouterA to translate unregistered address 10.1.1.1 to 135.25.1.3 before the packet is sent out the serial interface of RouterA. All traffic between host 10.1.1.1 and network 135.25.4.0 will be encrypted using ESP−3DES. The routers will use IKE to create the security association using preshared keys.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. OSPF will be run on all networks on both RouterA and RouterB, except for network

10.1.1.0. The IP addresses are assigned as per Figure 25−2.

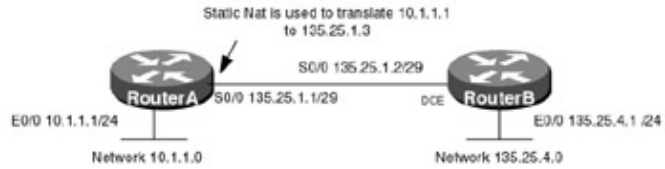


Figure 25−2: Basic IPSec with static NAT

Router Configurations

The configurations for the two routers in this example are as follows (IPSec configurations have not been added — the step−by−step configuration is provided in the monitoring and testing section).

RouterA

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
interface Ethernet0/0
 ip address 10.1.1.1 255.255.255.0
 no ip directed-broadcast
ip nat inside
no keepalive
!
interface Serial0/0
ip address 135.25.1.1 255.255.255.248
 no ip directed-broadcast
ip nat outside
 no ip mroute-cache
 no fair-queue
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip nat inside source static 10.1.1.1 135.25.1.3 ← Translates 10.1.1.1 to
                                                   135.25.1.3

ip classless
no ip http server
!
 line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
```

```
interface Ethernet0/0
 ip address 135.25.4.1 255.255.255.0
 no ip directed-broadcast
 no keepalive
!
interface Serial0/0
 ip address 135.25.1.2 255.255.255.248
no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 1000000
!
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

The first step is to create an IKE crypto policy. IKE is used to create the security association (SA) between the routers. This negotiation process is protected, so the peers must first agree on a shared IKE policy. Under the IKE policy, the encryption algorithm, hash algorithm, authentication method, Diffie−Hellman group identifier, and SA lifetime are configured. With the exception of the SA lifetime, if these parameters do not match, the SA negotiation will fail. The following command configures the IKE policy on RouterA and RouterB:

```
RouterA(config)#crypto isakmp policy 1

RouterB(config)#crypto isakmp policy 1
```

The authentication type is then defined under the IKE policy. Three types of authentication can be used: RSA−SIG, RSA−ENCR, or preshare. For this lab, we will be using preshare keys. If RSA−SIG were used, a certificate authority would be needed. The following command enables the IKE policy to use preshare keys:

```
RouterA(config)#crypto isakmp policy 1
RouterA(config-isakmp)#authentication pre-share

RouterB(config)#crypto isakmp policy 1
RouterB(config-isakmp)#authentication pre-share
```

Since preshared keys are being used, a shared key must be defined along with the peer's identity. The identity can be either the peer's IP address or name. For this lab, the IP address of the serial interface of each router will be the peer address and the shared secret will be cisco. The following command defines the key that will be used and the peer address:

```
RouterA(config)#crypto isakmp key cisco address 135.25.1.2

RouterB(config)#crypto isakmp key cisco address 135.25.1.1
```

The next step is to define the transform set or sets that will be used. A transform is simply the algorithm or algorithms that the router is willing to use for the session. The various transform sets are offered to the receiver during IKE; the receiver selects the one that will be used. For this lab, we will be using encryption

and authentication, so two transforms need to be defined. The following is a list of transforms supported:

```
  h-md5-hmac    AH-HMAC-MD5 transform
  ah-sha-hmac   AH-HMAC-SHA transform
  comp-lzs      IP Compression using the LZS compression algorithm
  esp-3des      ESP transform using 3DES(EDE) cipher (168 bits)
  esp-des       ESP transform using DES cipher (56 bits)
  esp-md5-hmac  ESP transform using HMAC-MD5 auth
  esp-null      ESP transform w/o cipher
  esp-sha-hmac  ESP transform using HMAC-SHA auth
```

The following command defines the transform on RouterA and RouterB:

```
RouterA(config)#crypto ipsec transform-set encr&auth esp-3des esp-md5-hmac


RouterB(config)#crypto ipsec transform-set encr&auth esp-3des esp-md5-hmac
```

The transform set that is going to be used can be viewed with the command **show crypto ipsec transform-set.** The following is the output from the command on RouterA. Notice that the transform set encr&auth is using esp-3des and esp-md5-hmac.

```
RouterA#show crypto ipsec transform-set

Transform set encr-only: { esp-3des  }
   will negotiate = { Tunnel,  },

Transform set encr&auth: { esp-3des esp-md5-hmac  }
   will negotiate = { Tunnel,  },
```

The next step is to define the traffic that will be given security protection. This is done using an extended access list to identify the traffic. For this lab, all traffic from network host 10.1.1.1 to network 135.25.4.0 should be encrypted and authenticated.

Since the IP address of E0/0 on RouterA is being translated, the NATed address is used in the access list. The ordering of processes for the inbound packet is to check the ACLs for the inbound interface, then a crypto map check, and then NAT, followed by policy routing and standard routing. Once a packet is outbound, NAT is checked on the outbound interface, then a crypto map check, and, finally, the outbound ACLs. This means that one needs to set up the access lists for IPSec using the NATed addresses. This is very important to remember when configuring IPSec and NAT on the same router.

The following commands define the access lists on RouterA and RouterB. Notice that the NATed address is being used, not the original source address.

```
RouterA(config)#access-list 101 permit ip host 135.25.1.3 135.25.4.0 0.0.0.255

RouterB(config)#access-list 101 permit ip 135.25.4.0 0.0.0.255 host 135.25.1.3
```

The next step is to define a crypto map, which combines the policy and traffic information. The crypto map contains the traffic to which security must be applied (defined by the access list), the actual algorithm to apply (defined by the transform), and the crypto endpoint (the remote peer). An IPSec crypto map is defined with a tag, sequence number, and the encryption method.

The following commands define a crypto map on RouterA and RouterB:

```
RouterA(config)#crypto map peer-RouterB local-address serial 0/0
RouterA(config)#crypto map peer-RouterB 10 ipsec-isakmp
RouterA(config-crypto-map)#set peer 135.25.1.2
RouterA(config-crypto-map)# set transform-set encr&auth
RouterA(config-crypto-map)#match address 101
```

```
RouterB(config)#crypto map peer-RouterA local-address loopback s0/0
RouterB(config)#crypto map peer-RouterA 10 ipsec-isakmp
RouterB(config-crypto-map)#set peer 135.251.1
RouterB(config-crypto-map)#set transform-set encr&auth
RouterB(config-crypto-map)#match address 101
```

The last thing is to apply the crypto map to an interface. You must assign a crypto map set to an interface before that interface can provide IPSec services.

```
RouterA(config)#interface s0/0
RouterA(config-if)#crypto map peer-RouterB

RouterB(config)#interface s0/0
RouterB(config-if)#crypto map peer-RouterA
```

Display the active IPSec connections on RouterA with the command **show crypto engine connections active.** The following is the output from the command. Notice that there are no active connections. This is because no traffic matching the crypto map has been sent.

```
RouterA#sho crypto engine connections active

  ID Interface       IP-Address      State  Algorithm           Encrypt  Decrypt
Crypto adjacency count : Lock: 0, Unlock: 0
```

Turn on the following debug commands on RouterA:

```
RouterA#debug crypto ipsec
RouterA#debug crypto isakmp
```

Now ping from the Ethernet interface of RouterA to the Ethernet interface of RouterB, using the extended ping command:

```
RouterA#ping
Protocol [ip]:
Target IP address: 135.25.4.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 10.1.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
```

The following is an analysis from the output of the debugs.

```
The ping source and destination matched access list 101,
which is tied to crypo map peer RouterB sequence number 10
                          ↓
02:33:34: IPSEC(sa_request):,
  (key eng. msg.) src= 135.25.1.1, dest= 135.25.1.2,

The scr proxy is the source of the interesting traffic as
defined by the access list. Since NAT was being used, the src-proxy
address is that of the serial interface. The dest proxy address is
the destination address that the packet is going to
                                          ↓
   src_proxy= 135.25.1.3/255 .255.255.255/0/0 (type=1),
    dest_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),

The protocol and the transforms are specified by the crypto map
```

peer RouterB 10. SPI is zero, which means the main mode of negotiation is being started
↓
protocol= ESP, **transform=** esp−3des esp−md5−hmac,
lifedur= 3600s and 4608000kb,
spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4004

These are the attributes that are being offered by RouterA
↓
02:33:34: ISAKMP (0:1): **beginning Main Mode exchange**
02:33:34: ISAKMP (1): sending packet to 135.25.1.2 (I) MM_NO_STATE
02:33:34: ISAKMP (1): received packet from 135.25.1.2 (I) MM_NO_STATE
02:33:34: ISAKMP (0:1): processing SA payload. message ID = 0
02:33:34: ISAKMP (0:1): **Checking ISAKMP transform 1 against priority 1 policy**
02:33:34: ISAKMP: encryption DES−CBC
02:33:34: ISAKMP: hash SHA
02:33:34: ISAKMP: default group 1
02:33:34: ISAKMP: auth pre−share

02:33:34: ISAKMP (0:1): atts are acceptable. Next payload is 0

Policy 1 of this router matches the attributes that were sent by RouterA.
Peer share authentication starts now
↓
02:33:34: ISAKMP (0:1): SA is doing pre−shared key authentication

02:33:34: ISAKMP (1): SA is doing pre−shared key authentication

using id type ID_IPV4_ADDR
02:33:34: ISAKMP (1): sending packet to 135.25.1.2 (I) MM_SA_SETUP
02:33:34: ISAKMP (1): received packet from 135.25.3.1 (I) MM_SA_SETUP
02:33:34: ISAKMP (0:1): processing KE payload. message ID = 0
02:33:34: ISAKMP (0:1): processing NONCE payload. message ID = 0

Nonce from the far end is being processed
↓
02:33:34: ISAKMP (0:1): SKEYID state generated
02:33:34: ISAKMP (0:1): processing vendor id payload
02:33:34: ISAKMP (0:1): speaking to another IOS box!
02:33:34: ISAKMP (1): ID payload
next−payload : 8
type : 1
protocol : 17
port : 500
length : 8
02:33:34: ISAKMP (1): Total payload length: 12
02:33:34: ISAKMP (1): sending packet to 135.25.1.2 (I) MM_KEY_EXCH
02:33:34: ISAKMP (1): received packet from 135.25.1.2 (I) MM_KEY_EXCH
02:33:34: ISAKMP (0:1): processing ID payload. message ID = 0
02:33:34: ISAKMP (0:1): processing HASH payload. message ID = 0
02:33:34: ISAKMP (0:1): SA has been authenticated with 135.25.1.2

At this point, preshare authentication has succeeded and ISAKMP has been negotiated
↓
02:33:34: ISAKMP (0:1): beginning Quick Mode exchange, M−ID of −1594944679

Quick mode authentication starts
↓

02:33:34: IPSEC(key_engine): got a queue event . . .
02:33:34: IPSEC(spi_response): getting spi 513345059 for SA
from 135.25.3.1 to 135.25.4.1 for prot 3
ISAKMP gets the SPI from the IPSEC routine and sends to the other side
↓

02:33:34: ISAKMP (1): sending packet to 135.25.1.2 (I) QM_IDLE
02:33:35: ISAKMP (1): received packet from 135.25.1.2 (I) QM_IDLE
02:33:35: ISAKMP (0:1): processing SA payload. message ID = −1594944679

```
02:33:35: ISAKMP (0:1): Checking IPSec proposal 1
```

RouterA processes the IPSEC attributes offered by the remote end.
Below are the attributes of the transform offered by the remote end

```
                                ↓<ΛINE/>
02:33:35: ISAKMP: transform 1, ESP_3DES
02:33:35: ISAKMP:   attributes in transform:
02:33:35: ISAKMP:      encaps is 1
02:33:35: ISAKMP:      SA life type in seconds
02:33:35: ISAKMP:      SA life duration (basic) of 3600
02:33:35: ISAKMP:      SA life type in kilobytes
02:33:35: ISAKMP:      SA life duration (VPI) of  0x0 0x46 0x50 0x0
02:33:35: ISAKMP:      authenticator is HMAC-MD5
02:33:35: ISAKMP (0:1): atts are acceptable.
```

The IPSEC SA has been successfully negotiated.

Here, RouterA validates the proposal that it negotiated with the remote side

```
                           ↓<ΛINE/>
02:33:35: IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) dest= 135.25.1.2, src= 135.25.1.1,
    dest_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),
    src_proxy= 135.25.1.3/255 .255.255.255/0/0 (type=1),
    protocol= ESP, transform= esp-3des esp-md5-hmac,
    lifedur= 0s and 0kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
02:33:35: ISAKMP (0:1): processing NONCE payload. message ID = -1594944679
02:33:35: ISAKMP (0:1): processing ID payload. message ID = -1594944679
02:33:35: ISAKMP (0:1): processing ID payload. message ID = -1594944679
02:33:35: ISAKMP (0:1): Creating IPSec SAs
```

At this point, crypto engine entries have been created, inbound traffic has a security parameter index (spi) of 513345059 and a conn_id 2000, and outbound traffic has a spi of 71175277 and conn_id of 2001.

When RouterA sends a packet that requires IPSec protection (any packet from 10.1.1.1 to 135.25.4.0), it looks up the security association in its database, applies the specified processing, and then inserts the SPI from the security association into the IPSec header. When RouterB receives the packet, it looks up the security association in its database by destination address and SPI and then processes the packet as required.

```
02:33:35:         inbound SA from 135.25.1.2      to 135.25.1.1
(proxy 135.25.4.0        to 135.25.1.3  )
02:33:35:         has spi 513345059 and conn_id 2000 and flags 4
02:33:35:         lifetime of 3600 seconds
02:33:35:         lifetime of 4608000 kilobytes
02:33:35:         outbound SA from 135.25.1.1      to 135.25.1.2
(proxy 135.25.1.3     to 135.25.4.0  )
02:33:35:         has spi 71175277 and conn_id 2001 and flags 4
02:33:35:         lifetime of 3600 seconds
02:33:35:         lifetime of 4608000 kilobytes
02:33:35: ISAKMP (1): sending packet to 135.25.1.2 (I) QM_IDLE
02:33:35: ISAKMP (0:1): deleting node -1594944679
02:33:35: IPSEC(key_engine): got a queue event . . .
02:33:35: IPSEC(initialize_sas):,
  (key eng. msg.) dest= 135.25.1.1, src= 135.25.1.2,
    dest_proxy= 135.25.1.3/255 .255.255.255/0/0 (type=1),
    src_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-3des esp-md5-hmac,
    lifedur= 3600s and 4608000kb,
    spi= 0x1E990623(513345059), conn_id= 2000, keysize= 0, flags= 0x4
02:33:35: IPSEC(initialize_sas):,
  (key eng. msg.) src= 135.25.1.1, dest= 135.25.1.2,
    src_proxy= 135.25.1.3/255 .255.255.255/0/0 (type=1),
    dest_proxy= 135.25.4.0/255 .255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-3des esp-md5-hmac,
```

```
    lifedur= 3600s and 4608000kb,
    spi= 0x43E0C6D(71175277), conn_id= 2001, keysize= 0, flags= 0x4
02:33:35: IPSEC(create_sa): sa created,
  (sa) sa_dest= 135.25.1.1, sa_prot= 50,
    sa_spi= 0x1E990623(513345059),
    sa_trans= esp-3des esp-md5-hmac, sa_conn_id= 2000
02:33:35: IPSEC(create_sa): sa created,
  (sa) sa_dest= 135.25.1.2, sa_prot= 50,
    sa_spi= 0x43E0C6D(71175277),
    sa_trans= esp-3des esp-md5-hmac, sa_conn_id= 2001
```

Now display the active crypto connections with the command **show crypto engine connection active**. The following is the output from the command. Notice there are two IPSec SAs: one for incoming traffic **(2000)** and one for outgoing traffic **(2001)**.

```
RouterA#show crypto engine connections active
  ID Interface      IP-Address     State   Algorithm           Encrypt  Decrypt
   1 <none>         <none>         set     HMAC_SHA+DES_56_CB        0        0
2000 Serial0/0      135.25.1.1     set     HMAC_MD5+3DES_56_C        0        9
2001 Serial0/0      135.25.1.1     set     HMAC_MD5+3DES_56_C        9        0
Crypto adjacency count : Lock: 0, Unlock: 0
```

Lab #109: OSPF over IPSec Usinga GRE Tunnel

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having two Ethernet and two serial ports
- Cisco IOS 12.0 capable of running IPSec
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- A Cisco rolled cable

Configuration Overview

This lab will demonstrate how to run OSPF across an IPSec tunnel using generic routing encapsulation (GRE). All traffic between network 135.25.3.0 and network 135.25.4.0 will be encrypted using ESP−3DES. The routers will use IKE to create the security association using preshared keys.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. OSPF will be run to advertise the Ethernet networks that are attached. Since OSPF is not supported natively over IPSec, a GRE tunnel will be created between the two routers. OSPF will run over the GRE tunnel, which in turn runs over the IPSec tunnel. The IP addresses are assigned as per Figure 25−3.

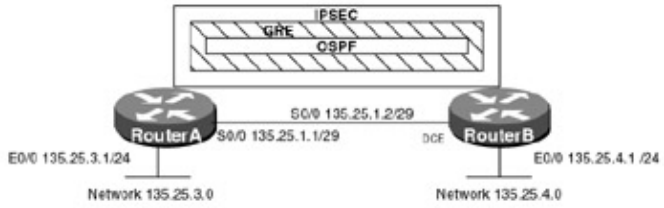


Figure 25−3: Running OSPF over IPSec

Router Configurations

The configurations for the two routers in this example are as follows (IPSec configurations have not been added — the step−by−step configuration is provided in the monitoring and testing section).

RouterA

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterA
!
interface Ethernet0/0
 ip address 135.25.3.1 255.255.255.0
 no ip directed−broadcast
 no keepalive
!
interface Serial0/0
ip address 135.25.1.1 255.255.255.252
 no ip directed−broadcast
 no ip mroute−cache
 no fair−queue
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
 line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterB
!

!
interface Ethernet0/0
 ip address 135.25.4.1 255.255.255.0
 no ip directed−broadcast
 no keepalive
!
interface Serial0/0
 ip address 135.25.1.2 255.255.255.252
no ip directed−broadcast
 no ip mroute−cache
 no fair−queue
 clockrate 1000000
!
```

```
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

The first step is to create the GRE tunnel between the two routers. This procedure involves creating the tunnel interface on both routers and defining the tunnel endpoints. For this lab, the serial interfaces of both routers are the endpoints of the tunnel. The following commands configure a GRE tunnel between RouterA and RouterB:

```
RouterA(config)#interface Tunnel0
RouterA(config-if) ip address 135.25.2.1 255.255.255.0
RouterA(config-if)tunnel source 135.25.1.1
RouterA(config-if)tunnel destination 135.25.1.2

RouterB(config)#interface Tunnel0
RouterB(config-if) ip address 135.25.2.2 255.255.255.0
RouterB(config-if)tunnel source 135.25.1.2
RouterB(config-if)tunnel destination 135.25.1.1
```

Verify that the tunnel is up and operational with the command **show interface tunnel 0**. The following is the output from the command on RouterA. Notice that the line protocol is up.

```
RouterA#show int tunnel 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 135.25.2.1/24
  MTU 1476 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive set (10 sec)
  Tunnel source 135.25.1.1, destination 135.25.1.2
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Checksumming of packets disabled,  fast tunneling enabled
  Last input never, output 00:00:09, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/0, 6 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     46 packets output, 6624 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Next, we enable OSPF on the Ethernet interface and the tunnel interface of each router:

```
RouterA(config)#router ospf 64
RouterA(config-router)#network 135.25.3.0 0.0.0.255 area 0
RouterA(config-router)#network 135.25.2.0 0.0.0.255 area 0
```

```
RouterB(config)#router ospf 64
RouterB(config-router)#network 135.25.4.0 0.0.0.255 area 0
RouterB(config-router)#network 135.25.2.0 0.0.0.255 area 0
```

Display the OSPF neighbors on RouterA with the command **show ip ospf neighbors**. Notice that RouterA has formed an adjacency with RouterB over the tunnel interface.

```
RouterA#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
135.25.5.1        1   FULL/ -         00:00:30    135.25.2.2      Tunnel0
```

Now display the routing table on RouterA. The following is the output. Notice that RouterA has learned of network 135.25.4.0 via the tunnel interface.

```
RouterA#sho ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     135.25.0.0/16 is variably subnetted, 4 subnets, 2 masks
O       135.25.4.0/24 [110/11121] via 135.25.2.2, 00:04:49, Tunnel0
C       135.25.2.0/24 is directly connected, Tunnel0
C       135.25.3.0/24 is directly connected, Ethernet0/0
C       135.25.1.0/30 is directly connected, Serial0/0
```

The next step is to enable IPSec on the router so that the tunnel traffic is encrypted. To do this, first create an IKE crypto policy. The following command configures the IKE policy on RouterA and RouterB:

```
RouterA(config)#crypto isakmp policy 1

RouterB(config)#crypto isakmp policy 1
```

The authentication type is then defined under the IKE policy. For this lab, we will be using preshare keys. If RSA-SIG were used, a certificate authority would be needed. The following command enables the IKE policy to use preshare keys:

```
RouterA(config)#crypto isakmp policy 1
RouterA(config-isakmp)#authentication pre-share

RouterB(config)#crypto isakmp policy 1
RouterB(config-isakmp)#authentication pre-share
```

Since preshared keys are being used, a shared key must be defined along with the peer's identity. The identity can be either the peer's IP address or name. For this lab, the IP address of the serial interface of each router will be the peer address and the shared secret will be cisco. The following command defines the key that will be used and the peer address:

```
RouterA(config)#crypto isakmp key cisco address 135.25.1.2

RouterB(config)#crypto isakmp key cisco address 135.25.1.1
```

The next step is to define the transform set or sets that will be used. For this lab, we will be using encryption only and just one transform needs to be defined. The following command defines the transform on RouterA and RouterB:

```
RouterA(config)#crypto ipsec transform-set encr-only esp-3des

RouterB(config)#crypto ipsec transform-set encr-only esp-3des
```

The transform set that is going to be used can be viewed with the command, **show crypto ipsec transform-set.** The following is the output from the command on RouterA. Notice that the transform set encr-only is using esp-3des.

```
RouterA#show crypto ipsec transform-set

Transform set encr-only: { esp-3des  }
   will negotiate = { Tunnel,  },
```

The next step is to define the traffic that will be given security protection. This is done using an extended access list to identify the traffic. For this lab, only the GRE tunnel needs to be encrypted since all routing protocol exchanges and traffic between LANs will follow over it.

In access-list terminology, "permit" means "protect" and "deny" means "don't protect." Any traffic that is denied will pass in the clear.

The following commands define the access lists on RouterA and RouterB. Notice that gre is specified as the protocol and the tunnel source address is the source of the traffic, while the tunnel destination address is the destination. All traffic that goes over the tunnel will be encrypted.

```
RouterA(config)# access-list 101 permit gre host 135.25.1.1 host 135.25.1.2

RouterB(config)# access-list 101 permit gre host 135.25.1.2 host 135.25.1.1
```

The next step is to define a crypto map, which combines the policy and traffic information. The crypto map contains the traffic to which security must be applied (defined by the access list), the actual algorithm to apply (defined by the transform), and the crypto endpoint (the remote peer). An IPSec crypto map is defined with a tag, a sequence number, and the encryption method.

The following commands define a crypto map on RouterA and RouterB:

```
RouterA(config)#crypto map gre-tunnel local-address serial 0/0
RouterA(config)#crypto map gre-tunnel 10 ipsec-isakmp
RouterA(config-crypto-map)#set peer 135.25.1.2
RouterA(config-crypto-map)# set transform-set encr-only
RouterA(config-crypto-map)#match address 101

RouterB(config)#crypto map gre-tunnel local-address loopback s0/0
RouterB(config)#crypto map gre-tunnel 10 ipsec-isakmp
RouterB(config-crypto-map)#set peer 135.25.1.1
RouterB(config-crypto-map)#set transform-set encr-only
RouterB(config-crypto-map)#match address 101
```

The last step is to apply the map to an interface. You must assign a crypto map set to an interface before that interface can provide IPSec services. When using GRE, the crypto map needs to be applied to both the physical interface and to the tunnel interface.

```
RouterA(config)#interface s0/0
RouterA(config-if)#crypto map gre-tunnel
RouterA(config)#interface tunnel 0
RouterA(config-if)#crypto map gre-tunnel

RouterB(config)#interface s0/0
RouterB(config-if)#crypto map gre-tunnel
RouterB(config)#interface tunnel 0
RouterB(config-if)#crypto map gre-tunnel
```

Now display the active crypto connections with the command **show crypto engine connection active.** The following is the output from the command. Notice there are two IPSec SAs: one for incoming traffic (2000) and one for outgoing traffic (2001). At this point, all traffic that transverses the GRE tunnel will be encrypted.

```
RouterA#show crypto engine connections active

  ID Interface       IP-Address      State  Algorithm           Encrypt  Decrypt
   1 <none>          <none>          set    HMAC_SHA+DES_56_CB        0        0
2000 Serial0/0       135.25.1.1      set    3DES_56_CBC               0       21
2001 Serial0/0       135.25.1.1      set    3DES_56_CBC              22        0
Crypto adjacency count : Lock: 0, Unlock: 0
```

Lab #110: Tunnel Endpoint Discovery (TED)

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having two Ethernet and two serial ports
- Cisco IOS 12.0 capable of running IPSec
- A PC running a terminal emulation program
- One Cisco DTE/DCE crossover cable
- A Cisco rolled cable

Configuration Overview

Tunnel endpoint discovery (TED) automatically determines the endpoint of the IPSec tunnel, greatly reducing the amount of configuration needed. Without TED, you must define which traffic gets protected and the remote endpoint of the IPSec tunnel. In large implementations, this is not feasible. Take, for example, a large fully meshed 100−router network. If TED is not used, every router will need to have (*n* — 1) = 99 endpoints defined along with traffic protection profiles for each. If additional routers are added, every router in the mesh will require further configuration.

When TED is used, the only thing that needs to be defined on the router is the traffic that is going to be protected. The tunnel endpoint is not configured — it is automatically discovered through a probe. When the router receives traffic that matches its protect policy, it sends a probe towards the destination address of the packet. The packet flows through the network until it reaches another TED−enabled router. The remote TED−enabled router responds to the request and sends its IP address to the originating router. Once received, the originating router can establish the IPSec tunnel since it now knows the endpoint.

RouterA and RouterB are connected serially via a crossover cable. RouterB will act as the DCE supplying clock to RouterA. OSPF will be run to advertise the Ethernet networks that are attached.

All traffic between network 135.25.3.0 and network 135.25.4.0 will be encrypted using ESP−3DES. The routers will use IKE to create the security association using preshared keys. Tunnel endpoint (TED) discovery will be used to automatically discover the IPSec endpoints. The IP addresses are assigned as per Figure 25−4.

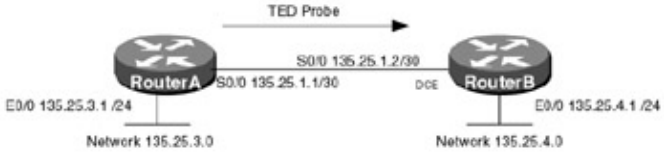


Figure 25−4: Tunnel endpoint discovery

When RouterA receives a packet from network 135.25.3.0 destined for network 135.25.4.0, it checks to see if an SA exits. If no SA is present, RouterA will send a TED probe packet in order to determine the remote peer

— once determined, the two routes can then establish a SA.

Router Configurations

The configurations for the two routers in this example are as follows (IPSec configurations have not been added — the step−by−step configuration is provided in the monitoring and testing section).

RouterA

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterA
!
interface Ethernet0/0
 ip address 135.25.3.1 255.255.255.0
 no ip directed−broadcast
 no keepalive
!
interface Serial0/0
ip address 135.25.1.1 255.255.255.252
 no ip directed−broadcast
 no ip mroute−cache
 no fair−queue
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
 line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

RouterB

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterB
!
interface Ethernet0/0
 ip address 135.25.4.1 255.255.255.0
 no ip directed−broadcast
 no keepalive
!
interface Serial0/0
 ip address 135.25.1.2 255.255.255.252
no ip directed−broadcast
 no ip mroute−cache
 no fair−queue
 clockrate 1000000
!
```

```
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
!
end
```

Monitoring and Testing the Configuration

The first step is to create an IKE crypto policy on the routers. IKE is used to create the security association (SA) between the two routers. The following command configures an IKE policy on RouterA and RouterB:

```
RouterA(config)#crypto isakmp policy 1

RouterB(config)#crypto isakmp policy 1
```

The authentication type is then defined under the policy. For this lab, we will be using preshare keys. The following commands enable the IKE policy to use preshare keys:

```
RouterA(config)#crypto isakmp policy 1
RouterA(config-isakmp)#authentication pre-share

RouterB(config)#crypto isakmp policy 1
RouterB(config-isakmp)#authentication pre-share
```

Since preshared keys are being used, a shared key must be defined along with the peer's identity. The identity can be either the peers IP address or name. Since TED is being used to discover the endpoints, an all-zeros address is used as the peer. The following command defines the key that will be used and the peer address:

```
RouterA(config)#crypto isakmp key cisco address 0.0.0.0

RouterB(config)#crypto isakmp key cisco address 0.0.0.0
```

The next step is to define the transform set or sets that will be used. For this lab, we will be using encryption only and just one transform needs to be defined. The following command defines the transform on RouterA and RouterB:

```
RouterA(config)#crypto ipsec transform-set encr-only esp-3des

RouterB(config)#crypto ipsec transform-set encr-only esp-3des
```

The transform set that is going to be used can be viewed with the command **show crypto ipsec transform-set.** The following is the output from the command on RouterA. Notice that the transform set encr-only is using esp-3des.

```
RouterA#show crypto ipsec transform-set

Transform set encr-only: { esp-3des  }
   will negotiate = { Tunnel,  },
```

The next step is to define the traffic that will be given security protection. This is done using an extended access list to identify the traffic. For this lab, traffic between network 135.25.3.0 and network 135.25.4.0 needs to be encrypted.

The following commands define the access lists on RouterA and RouterB:

```
RouterA(config)# access-list 101 permit ip 135.25.3.0 0.0.0.255 135.25.4.0 0.0.0.255

RouterB(config)# access-list 101 permit ip 135.25.4.0 0.0.0.255 135.25.3.0 0.0.0.255
```

The next step is to define a dynamic crypto map, which combines the policy and traffic information. The dynamic crypto map creates a policy template that is used to process negotiation requests for new security associations from a remote IPSec peer, even if you don't know the remote peer's address.

The crypto map TED contains the traffic to which security must be applied (defined by the access list) and the actual algorithm to apply (defined by the transform), but does not include the remot peer's IP address. For this entry and any other entry that is not defined, the crypto map "wildcard" parameters will be accepted.

The only configuration required in a dynamic crypto map is to set the transform set. All other information is optional. The following commands define a crypto map on RouterA and RouterB:

```
RouterA(config)#crypto dynamic-map TED 10
RouterA(config-crypto-map)# set transform-set encr-only
RouterA(config-crypto-map)#match address 101

RouterB(config)#crypto dynamic-map TED 10
RouterB(config-crypto-map)#set transform-set encr-only
RouterB(config-crypto-map)#match address 101
```

Dynamic crypto map entries, like regular static crypto map entries, are grouped into sets. After you define a dynamic crypto map set, you must define a global or parent crypto map. Under the global crypto map, you then add the defined dynamic crypto map.

When a router receives a negotiation request via IKE from another IPSec peer, the request is examined to see if it matches a crypto map entry. The crypto map entries are searched in order, starting with the lowest number.

In our case, crypto map global 10 contains a reference for dynamic crypto map TED, it will accept the wildcard parameter for the peer's address, allowing the router to negotiate IKE without knowing the remote peer's IP address beforehand.

In this lab, we are only defining one instance of the crypto map (sequence number 10), which is a dynamic map. However, normally, multiple static crypto maps would also be defined for peers that remain static. In those cases, it is important that the dynamic map have the highest sequence number so that it is evaluated last. Only after the negotiation request does not match any of the statically defined maps should the dynamic crypto map be used.

The following commands add the global or parent crypto map global and reference the dynamic map TED:

```
RouterA(config)# crypto map Global 10 ipsec-isakmp dynamic TED discover

RouterB(config)# crypto map Global  10 ipsec-isakmp dynamic TED discover
```

The last thing is to apply the global crypto map to an interface. You must assign a crypto map set to an interface before that interface can provide IPSec services. The following commands add the global map to the serial interfaces of RouterA and RouterB:

```
RouterA(config)#interface s0/0
RouterA(config-if)#crypto map Global

RouterB(config)#interface s0/0
RouterB(config-if)#crypto map Global
```

Now verify that the dynamic crypto map is configured correctly with the command **show crypto dynamic−map.** The following is the output from the command on RouterA. Notice that the dynamic map covers traffic from network 135.25.3.0 to network 135.25.4.0.

```
RouterA#show crypto dynamic−map
Crypto Map Template"TED" 10
        Extended IP access list 101
            access−list 101 permit ip 135.25.3.0 0.0.0.255 135.25.4.0 0.0.0.255
        Current peer: 0.0.0.0
        Security association lifetime: 4608000 kilobytes/3600 seconds
        PFS (Y/N): N
        Transform sets={ encr−only, }
```

Enable Crypto ISAKMP, Crypto Engine, and Crypto IPSec debugging on RouterA. From RouterA, ping network 135.25.4.1 using the extended ping command to source the packet from 135.25.3.1.

```
RouterA#ping
Protocol [ip]:
Target IP address: 135.25.4.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Source address or interface: 135.25.3.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100−byte ICMP Echos to 135.25.4.1, timeout is 2 seconds:
```

The following is the output from the debug commands on RouterA.

RouterA receives a packet sourced from 135.25.3.1 destined for 135.25.4.1. It first checks to see if it already has a SA established. In this case, none exist, so it needs to create one in order to encrypt the traffic. Since the remote peers address is not configured, the router must send a probe packet to determine its peer for negotiation.

```
 02:34:55: IPSEC(tunnel discover request):
  (key eng. msg.) src= 135.25.3.1, dest= 135.25.4.1,
    src_proxy= 135.25.3.0/255.255.255.0/0/0 (type=4),
    dest_proxy= 135.25.1.1/255.255.255.255/0/0 (type=1),
    protocol= ESP, transform= esp−3des,
    lifedur= 3600s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4044
02:34:55: GOT A PEER DISCOVERY MESSAGE FROM THE SA MANAGER!!!src =
135.25.3.1 ← No peer defined router must use ted to discover
   to 135.25.4.1    , protocol 3, transform 3, hmac 0
02:34:55: proxy source is 135.25.3.0    /255.255.255.0  and my address (not us
ed now) is 135.25.1.1
02:34:55: ISAKMP (3): ID payload
        next−payload : 5
        type         : 1
        protocol     : 17
        port         : 500
        length       : 8
02:34:55: ISAKMP (3): Total payload length: 12
02:34:55: 1st ID is 135.25.1.1
02:34:55: 2nd ID is 135.25.3.0 /255.255.255.0
```

RouterA sends a TED packet with a source address of 135.25.3.1 and a destination address of 135.25.4.1; the IP address of RouterA is contained in the payload.

```
02:34:55: ISAKMP (0:3): beginning peer discovery exchange
02:34:55: ISAKMP (3): sending packet to 135.25.4.1 (I) PEER_DISCOVERY
```

RouterA receives a TED response containing RouterB's IP address and begins IKE negotiation with that address.

```
02:34:56: ISAKMP (3): received packet from 135.25.4.1 (I) PEER_DISCOVERY
02:34:56: ISAKMP (0:3): processing vendor id payload
02:34:56: ISAKMP (0:3): speaking to another IOS box!
02:34:56: ISAKMP (0:3): processing ID payload. message ID = 0
02:34:56: ISAKMP (0:3): processing ID payload. message ID = 998864500
02:34:56: ISAKMP (3): ID_IPV4_ADDR_SUBNET dst 135.25.4.0/255 .255.255.0 prot 0 po
rt 0
02:34:56: ISAKMP (3): received response to my peer discovery probe!ISAKMP:
initiating IKE to 135.25.1.2 in response to probe.
02:34:56: ISAKMP (4): sending packet to 135.25.1.2 (I) MM_NO_STATE
02:34:56: ISAKMP (0:3): deleting SA
02:34:56: ISAKMP (4): received packet from 135.25.1.2 (I) MM_NO_STATE
02:34:56: ISAKMP (0:4): processing SA payload. message ID = 0
02:34:56: ISAKMP (0:4): Checking ISAKMP transform 1 against priority 1 policy
02:34:56: ISAKMP:      encryption DES-CBC
02:34:56: ISAKMP:      hash SHA
02:34:56: ISAKMP:      default group 1
02:34:56: ISAKMP:      auth pre-share
02:34:56: ISAKMP (0:4): atts are acceptable. Next payload is 0
02:34:56: CryptoEngine0: generate alg parameter
02:34:56: CRYPTO_ENGINE: Dh phase 1 status: 0
02:34:56: CRYPTO_ENGINE: Dh phase 1 status: 0
02:34:56: ISAKMP (0:4): SA is doing pre-shared key authentication
```

Now display the active crypto connections with the command **show crypto engine connection active.** The following is the output from the command. Notice there are two IPSec SAs: one for incoming traffic **(2000)** and one for outgoing traffic **(2001)**.

```
RouterA#show crypto engine connections  active

  ID Interface       IP-Address      State  Algorithm           Encrypt  Decrypt
   4 <none>          <none>          set    HMAC_SHA+DES_56_CB        0        0
2000 Serial0/0       135.25.1.1      set    3DES_56_CBC               0        4
2001 Serial0/0       135.25.1.1      set    3DES_56_CBC               4        0
Crypto adjacency count : Lock: 0, Unlock: 0
```

Troubleshooting IPSec

The Cisco IOS provides many tools for troubleshooting IPSec. The following is a list of key commands along with sample output from each.

{show crypto ipsec transform-set} This privileged exec command displays all configured transform sets on the router.

```
RouterA#show crypto ipsec transform-set

Transform set encr-only: { esp-3des  }
   will negotiate = { Tunnel,  },
```

{show crypto dynamic-map} This privileged exec command displays all dynamic crypto maps configured on the router.

```
RouterA#show crypto dynamic-map
```

```
Crypto Map Template"TED" 10
        Extended IP access list 101
            access-list 101 permit ip 135.25.3.0 0.0.0.255 135.25.4.0 0.0.0.255
        Current peer: 0.0.0.0
        Security association lifetime: 4608000 kilobytes/3600 seconds
        PFS (Y/N): N
        Transform sets={ encr-only, }
permit 150.1.1.0, wildcard bits 0.0.0.255
```

{show crypto engine connections active} This privileged exec command, displays all active crypto connections. The following is the output from the command. Notice that there are two IPSec SAs: one for incoming traffic **(2000)** and one for outgoing traffic **(2001)**.

```
RouterA#show crypto engine connections active

  ID Interface      IP-Address      State   Algorithm           Encrypt  Decrypt
   1 <none>         <none>          set     HMAC_SHA+DES_56_CB        0        0
2000 Serial0/0      135.25.1.1      set     3DES_56_CBC               0        4
2001 Serial0/0      135.25.1.1      set     3DES_56_CBC               4        0
Crypto adjacency count : Lock: 0, Unlock: 0
```

{show crypto isakmp policy} This exec command displays all of the crypto policies configured on the router. The following shows two policies: a user-defined policy and a default policy.

```
RouterA#show crypto isakmp policy
Protection suite of priority 1
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
Default protection suite
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Rivest-Shamir-Adleman Signature
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
```

Conclusion

IPsec provides a secure transport mechanism for the transportation of sensitive information over the public Internet, providing data privacy, data integrity, authenticity as well as confidentiality through a flexible suite of encryption and authentication protocols. Unlike other encryption technologies, IPSec operates at the IP layer, providing greater choices to service providers when it comes to the underlying network topology.

Chapter 26: Voice

Overview

Topics Covered in This Chapter

- Basic Voice Configuration
- Private Line Automatic Ringdown (PLAR)
- Number Expansion
- IP Precedence
- Custom Queuing for Voice Traffic
- Priority Queuing for Voice Traffic

Introduction

Combining voice and data over the same network is a major trend in networking today. Cisco provides extensive support for integrating voice traffic onto existing data networks. This chapter will explore this technology as well as demonstrate Cisco's voice over data capabilities with six hands−on labs.

Voice Technology Overview

Voice over IP (VoIP) is a technology that enables analog traffic (telephone calls and faxes) to be carried over an IP network. As shown in Figure 26−1, with VoIP technology a digital signal processor (DSP) that resides in a Cisco Voice Network Module takes analog traffic (voice and fax), converts the analog traffic to digital signals, compresses the digital signals, and then sends the analog traffic as IP packets. These voice packets are then sent as encapsulated IP traffic using the ITU H.323 specification, which defines the transmission of voice,

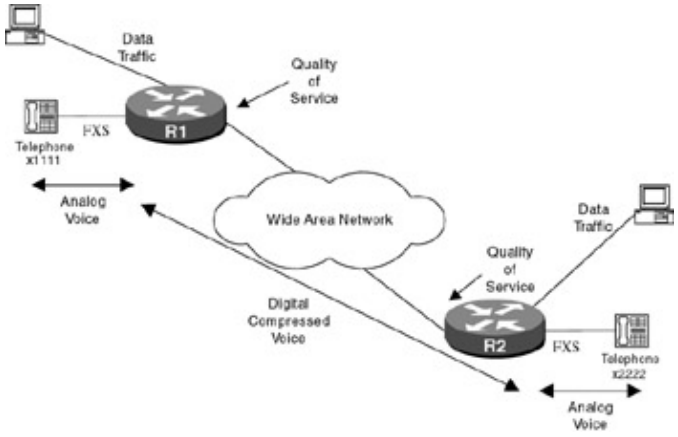


Figure 26−1: Voice over IP overview

data, and video across a network. Voice traffic is very sensitive to delay. This is because voice traffic has the characteristics of constant bit rate traffic; i.e., there is a constant stream of information being sent. Unlike data packets, if voice packets are lost or received in error, it does no good to retransmit the original traffic, since there is a constant stream of traffic that is being sent in real time. Because of the delay sensitivity of voice traffic, it is important to properly design your network. The Cisco IOS supports several protocols and enhanced features, which can improve the quality of service of voice traffic.

In order to bring voice and fax traffic into the router, you have to install special voice interface cards (VICs) in the router.

VoIP Technology

The current public phone network (PSTN) takes analog traffic from a customer's home or office and brings it into a voice switch such as a Lucent 5ESS or a Nortel DMS−250. Once at the voice switch, the analog signal is converted into a digital signal using a technology known as pulse code modulation (PCM). The analog signal is converted into a digital signal by sampling the analog signal 8,000 times per second. Each of the 8,000 samples is given an 8−bit value; this gives a sample rate of 64,000 bits/second.

Voice Compression

Voice compression techniques have been standardized by the ITU and are supported by the major vendors of VoIP equipment. The most popular voice compression technique is the G.729 algorithm. The G.729 specification describes a compression scheme where voice is coded into 8 Kbps streams. Encapsulation overhead that is added to the 8 Kbps voice stream actually creates a VoIP call that is in the range of 22 Kbps, but this number can be brought down to the 11−Kbps range using header compression.

Compressed Voice Quality

A common test that is used to determine the quality of sound produced by different types of voice compression is the mean opinion score (MOS). MOS works by having several people listen to a voice recording and judge the quality of the voice. Samples are rated on a scale of 1 (poor) to 5 (excellent). Although very subjective, the MOS test is the best measurement of voice quality. Uncompressed voice, sometimes referred to as toll−quality voice, is given a MOS score or 5. The G.729 algorithm, which is the most widely used voice compression method, is usually given a MOS rating around 4.0. It should be noted that voice compression is a type of compression that is known as lossy compression. Lossy compression is any type of compression where the original signal cannot be recreated. Voice compression and graphic compression methods such as JPEG are examples of lossy compression. Care must be taken in designing networks that use lossy compression. Multiple compression hops can greatly reduce the quality of the voice, and therefore the MOS score.

Voice Packet Delay and Jitter

An important consideration to take into account when implementing a voice network is to minimize the end−to−end delay. Since voice traffic is real−time traffic, it is very susceptible to delay. It is desirable to keep the end−to−end delay in a voice network to less than 200 milliseconds. The following are types of delay:

- **Propagation delay:** Caused by the electrical voice signals traveling via a fiber− or copper−based medium.
- **Serialization delay:** Caused by the devices that handle voice packets. There are several types of serialization delays:

    - Compression algorithm−induced delays are considered a type of serialization delay. The serialization delay introduced by the G.729 algorithm is in the range of 15 ms. In contrast, the delay introduced by the standard uncompressed PCM algorithm is in the range of 5 ms.
    - A second type of serialization delay is the time it takes to generate a voice packet. A VoIP data frame is generated every 10 milliseconds. Two data frames are encapsulated within a single voice packet. This creates a packet delay in the range of 20 milliseconds.
    - A third type of serialization delay is the time it takes for voice packets to be moved to the output queue. The Cisco IOS has several QoS features that are designed to minimize this type of serialization delay.

Jitter

Jitter is defined as a variation between when a voice packet is supposed to be received and when it actually is received. Jitter can be thought of as variable latency for different packets or cells. Jitter causes a disruption to the real−time voice stream. The Cisco IOS employs buffering techniques, which help adjust for packet jitter.

Echo

Echo occurs when a speaker hears his or her own voice in the telephone receiver while speaking. When properly timed, echo is reassuring to the speaker. On the other hand, if the echo is greater then 25 milliseconds it can be an annoyance. Echo is caused by a mismatch in impedance when the voice signal is converted at the central office from a four−wire signal at the switch to a two−wire signal at local loop. The Cisco IOS provides echo cancellation capability.

Voice Interface Cards

Cisco supports a variety of VIC cards and Network modules, which enable voice traffic to be brought into a router. These interfaces include FXS, FXO, E&M, and T1 or E1 trunk interfaces.

Commands Discussed in This Chapter

- connection plar *string*
- custom−queue−list *list*
- destination−pattern [+]*string*
- dial−peer voice *number* {voip | pots}
- ip precedence *number*
- num−exp *extension−number expanded−number*
- priority−group *list−number*
- priority−list *list−number* default {high | medium | normal | low}
- priority−list *list−number* protocol *protocol−name* {high | medium | normal | low}
- queue−list *list−number* default *queue−number*
- queue−list *list−number* protocol *protocol−name queue−number queue−keyword keyword−value*
- queue−list *list−number* queue *queue−number* byte−count *byte−count−number*
- session target {ipv4:*destination−address* | dns:[s. | d. | u.] *host−name* | loopback:rtp | loopback:compressed | loopback:uncompressed}
- show call active voice
- show dial−peer voice [*number*]
- show dialplan number *dial string*
- show num−exp [*dialed− number*]
- show queuing
- show voice port *slot−number*/*subunit−number*/*port*

Definitions

connection plar: This voice−port configuration command is used to configure a private line automatic ringdown connection for a specified voice port.

custom−queue−list: This interface configuration command is used to assign a custom queue list to an interface.

destination−pattern: This dial−peer configuration command specifies either the prefix or the full E.164 telephone number that will be used for a dial peer.

dial−peer voice: This global configuration command is used to enter dial−peer configuration mode and specify the method of voice−related encapsulation.

ip precedence: This dial−peer configuration command is used to set the IP precedence bits for packets sent by the specified dial peer.

num−exp: This global configuration command defines how the router will expand an extension number into a specific destination pattern.

priority−group: This interface configuration command assigns the specified priority list to an interface.

priority−list default: This global configuration command assigns a priority queue for those packets that do not match any other rule in a priority list.

priority−list protocol: This global configuration command is used to establish queuing priorities based upon the type of protocol being used.

queue−list default: This global configuration command is used to assign a priority queue for those packets that do not match any other rule in the queue list.

queue−list protocol: This global configuration command is used to establish queuing priority based upon the protocol type.

queue−list queue byte−count: This global configuration command is used to specify how many bytes the system allows to be delivered from a given queue during a particular cycle.

session target: This dial−peer configuration command is used to configure a network−specific address for a specified dial peer.

show call active voice: This privileged EXEC command displays the active call table.

show dial−peer voice: This privileged EXEC command displays configuration information for dial peers.

show dialplan number: This privileged EXEC command shows which dial peer is reached when a particular telephone number is dialed.

show num−exp: This privileged EXEC command displays the number expansions that have been configured on a router.

show queuing: This privileged EXEC command causes the router to list all configured queuing strategies.

show voice port: This privileged EXEC command displays configuration information about a specific voice port.

IOS Requirements

Voice capabilities were first introduced in IOS 11.3. All labs in this chapter were done using IOS 12.1.

Lab #111: Basic Voice Configuration

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface and two FXS interfaces.
- One Cisco router with one serial interface and one FXS interface.
- Three analog phone handsets.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.

- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports voice.

Configuration Overview

This lab will demonstrate a basic Voice over IP configuration. Three handsets will be connected to two routers. A dial plan will be implemented that will allow four−digit dialing from any handset to any other handset.

The routers are connected as shown in Figure 26−2. RouterA acts as a DCE and supplies clocking to RouterB.

Figure 26−2: Basic voice configuration

Router Configuration

The configurations for the two routers in this example are as follows. Voice commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
voice-port 0/0/0
!
voice-port 0/0/1
!
!
dial-peer voice 1 pots
 destination-pattern 2222 ← Extension 2222 is assigned to physical voice port
                            0/0/0
 port 0/0/0
!
dial-peer voice 2 voip
 destination-pattern 3333 ← Define a dial peer for the remote extension number 3333
 session target ipv4:135.25.3.1 ← Extension 3333 is reachable at 135.25.3.1
!
dial-peer voice 3 pots
 destination-pattern 2223 ← Extension 3333 is assigned to physical voice port
                            0/0/1
 port 0/0/1
!
!
interface Loopback0
 ip address 135.25.2.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.1 255.255.255.252
```

```
 clockrate 800000
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
voice-port 1/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 3333
 port 1/0/0
!
dial-peer voice 2 voip
 destination-pattern 2223
 session target ipv4:135.25.2.1
!
dial-peer voice 3 voip
 destination-pattern 2222
 session target ipv4:135.25.2.1
!
interface Loopback0
 ip address 135.25.3.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.2 255.255.255.252
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
no ip classless
no ip http server
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
password cisco
login
!
end
```

Monitoring and Testing the Configuration

Once the configurations have been entered into RouterA and RouterB, the reader should verify that dial tone is present on all three handsets. The lab configuration should be functional and the reader should try to place phone calls between the three handsets. The configuration can be verified by connecting to RouterA. The **show voice port** command, shown next, can be used to verify the settings of a voice port on the router:

```
RouterA#show voice port

Foreign Exchange Station 0/0/0
 Type of VoicePort is FXS ← Port type
 Operation State is DORMANT
 Administrative State is UP
 No Interface Down Failure
 Description is not set
 Noise Regeneration is enabled
 Non Linear Processing is enabled
 Music On Hold Threshold is Set to −38 dBm
 In Gain is Set to 0 dB
 Out Attenuation is Set to 0 dB
 Echo Cancellation is enabled
 Echo Cancel Coverage is set to 8 ms
 Initial Time Out is set to 10 s
 Interdigit Time Out is set to 10 s
 Call−Disconnect Time Out is set to 60 s
 Ringing Time Out is set to 180 s
 Region Tone is set for US

 Analog Info Follows:
 Currently processing unknown
 Maintenance Mode Set to None (not in mtc mode)
 Number of signaling protocol errors are 0
 Impedance is set to 600r Ohm

 Voice card specific Info Follows:
 Signal Type is loopStart
 Ring Frequency is 25 Hz
 Hook Status is On Hook
 Ring Active Status is inactive
 Ring Ground Status is inactive
 Tip Ground Status is inactive
 Digit Duration Timing is set to 100 ms
 InterDigit Duration Timing is set to 100 ms
```

As shown next, the **show dial−peer voice** command can be used to verify proper configuration for voice ports.

```
RouterA#show dial−peer voice
VoiceEncapPeer1
     information type = voice,
tag = 1, destination−pattern = '2222', ← Number associated with this port
     answer−address = '', preference=0,
     group = 1, Admin state is up, Operation state is up,
     incoming called−number = '', connections/maximum = 0/unlimited,
     application associated:
     type = pots, prefix = '',
     session−target = '', voice−port = '0/0/0',
     direct−inward−dial = disabled,
     register E.164 number with GK = TRUE
     Connect Time = 1409, Charged Units = 0,
     Successful Calls = 19, Failed Calls = 1,
     Accepted Calls = 20, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2566162.
```

```
VoiceEncapPeer3
     information type = voice,
tag = 3, destination-pattern = '2223', ← Number associated with this port
     answer-address = '', preference=0,
     group = 3, Admin state is up, Operation state is up,
     incoming called-number = '', connections/maximum = 0/unlimited,
     application associated:
     type = pots, prefix = '',
     session-target = '', voice-port = '0/0/1',
     direct-inward-dial = disabled,
     register E.164 number with GK = TRUE
     Connect Time = 1731, Charged Units = 0,
     Successful Calls = 12, Failed Calls = 0,
     Accepted Calls = 12, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2564412.
```

You can use the **show dial-peer voice** command to verify that the dial peer has been properly configured.

```
RouterA#show dial-peer voice 2
VoiceOverIpPeer2
     information type = voice,
     tag = 2, destination-pattern = '3333', ← Destination number
     answer-address = '', preference=0,
     group = 2, Admin state is up, Operation state is up,
     incoming called-number = '', connections/maximum = 0/unlimited,
     application associated:
type = voip, session-target = 'ipv4:135.25.3.1', ← Establish calls
                                                    to extension 3333 via this IP address
     technology prefix:
     ip precedence = 0, UDP checksum = disabled,
     session-protocol = cisco, req-qos = best-effort,
     acc-qos = best-effort,
     fax-rate = voice,   payload size =  20 bytes

          Compression Algorithm
               ↓
     codec = g729r8,   payload size =  20 bytes,
     Expect factor = 10, Icpif = 30,signaling-type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 3126, Charged Units = 0,
     Successful Calls = 25, Failed Calls = 4,
     Accepted Calls = 29, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2566163.
```

The **show dialplan number** command can be used to verify that a given called number will be dialed properly. We see next that we have verified that dialing 3333 will establish a VoIP call to the target address 135.25.3.1.

```
RouterA#show dialplan number 3333
Macro Exp.: 3333

VoiceOverIpPeer2
     information type = voice,
     tag = 2, destination-pattern = '3333',
     answer-address = '', preference=0,
     group = 2, Admin state is up, Operation state is up,
     incoming called-number = '', connections/maximum = 0/unlimited,
     application associated:
     type = voip, session-target = 'ipv4:135.25.3.1',
     technology prefix:
     ip precedence = 0, UDP checksum = disabled,
     session-protocol = cisco, req-qos = best-effort,
```

```
     acc-qos = best-effort,
     fax-rate = voice,   payload size =  20 bytes
     codec = g729r8,   payload size =  20 bytes,
     Expect factor = 10, Icpif = 30,signaling-type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 3126, Charged Units = 0,
     Successful Calls = 25, Failed Calls = 4,
     Accepted Calls = 29, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2566163.
Matched: 3333   Digits: 4
Target: ipv4:135.25.3.1
```

Now connect to RouterB and verify that the dial peers have been properly configured with the **show dial-peer voice** command.

```
RouterB#show dial-peer voice
VoiceOverIpPeer2
     information type = voice,
     tag = 2, destination-pattern = '2223',
     answer-address = '', preference=0,
     group = 2, Admin state is up, Operation state is up,
     incoming called-number = '', connections/maximum = 0/unlimited,
     application associated:
     type = voip, session-target = 'ipv4:135.25.2.1',
     technology prefix:
     ip precedence = 0, UDP checksum = disabled,
     session-protocol = cisco, req-qos = best-effort,
     acc-qos = best-effort,
     fax-rate = voice,   payload size =  20 bytes
     codec = g729r8,   payload size =  20 bytes,
     Expect factor = 10, Icpif = 30,signaling-type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 1727, Charged Units = 0,
     Successful Calls = 6, Failed Calls = 0,
     Accepted Calls = 6, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2274609.
VoiceOverIpPeer3
     information type = voice,
     tag = 3, destination-pattern = '2222',
     answer-address = '', preference=0,
     group = 3, Admin state is up, Operation state is up,
     incoming called-number = '', connections/maximum = 0/unlimited,
     application associated:
     type = voip, session-target = 'ipv4:135.25.2.1',
     technology prefix:
     ip precedence = 0, UDP checksum = disabled,
     session-protocol = cisco, req-qos = best-effort,
     acc-qos = best-effort,
     fax-rate = voice,   payload size =  20 bytes
     codec = g729r8,   payload size =  20 bytes,
     Expect factor = 10, Icpif = 30,signaling-type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 1406, Charged Units = 0,
     Successful Calls = 15, Failed Calls = 0,
     Accepted Calls = 15, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2277544.
```

The **show call active voice** command, an excerpt of which is shown next, can be used while a voice call is active to get information pertaining to the call.

```
RouterB#show call active voice
 GENERIC:
SetupTime=2473944 ms
Index=1
PeerAddress=2222
PeerSubAddress=
PeerId=3
PeerIfIndex=11
LogicalIfIndex=0
ConnectTime=2474064
CallDuration=00:00:47
CallState=4
CallOrigin=2
ChargedUnits=0
InfoType=2
TransmitPackets=848 ← Transmission packet counts
TransmitBytes=16960
ReceivePackets=689
ReceiveBytes=13780
VOIP:
ConnectionId[0x29C834F4 0x2287004F 0x0 0x1A5888C]
RemoteIPAddress=135.25.9.1
RemoteUDPPort=17532
RoundTripDelay=9 ms
SelectedQoS=best-effort
tx_DtmfRelay=inband-voice
```

Lab #112: Private Line Automatic Ringdown (PLAR)

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface and two FXS interfaces.
- One Cisco router with one serial interface and one FXS interface.
- Three analog phone handsets.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports voice.

Configuration Overview

This lab will demonstrate an advanced Voice over IP feature known as *private line automatic ringdown* (PLAR). PLAR enables a number to be automatically called as soon as a specified handset is taken off hook. In the case of this lab, when the handset on extension 2222 is taken off hook, it will immediately place a call to extension 3333.

The routers are connected as shown in Figure 26−3. RouterA acts as a DCE and supplies clocking to RouterB.

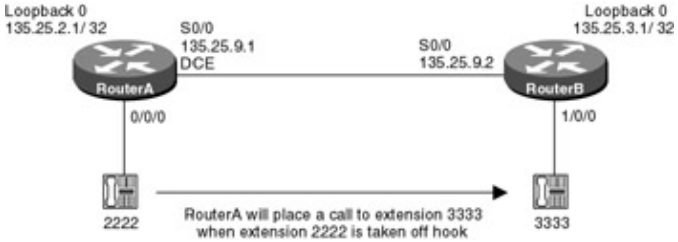


Figure 26−3: Private line automatic ringdown

Router Configuration

The configurations for the two routers in this example are as follows. PLAR commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
voice-port 0/0/0
 connection plar 3333 ← RouterA will dial extension 3333 as soon as the handset
                        connected to voice port is taken off hook
!
!
dial-peer voice 1 pots
 destination-pattern 2222
 port 0/0/0
!
dial-peer voice 2 voip
 destination-pattern 3333
 session target ipv4:135.25.3.1
!
!
interface Loopback0
 ip address 135.25.2.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.1 255.255.255.252
 clockrate 800000
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
voice-port 1/0/0
!
!
```

```
dial-peer voice 1 pots
 destination-pattern 3333
 port 1/0/0
!
dial-peer voice 3 voip
 destination-pattern 2222
 session target ipv4:135.25.2.1
!
interface Loopback0
 ip address 135.25.3.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.2 255.255.255.252
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
no ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
password cisco
login
!
end
```

Monitoring and Testing the Configuration

Once the configurations have been entered for RouterA and RouterB, the user should verify that dial tone is present on all three handsets. The lab should be tested by taking extension 2222 off hook. Extension 3333 should ring at the far end.

Lab #113: Number Expansion

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface and two FXS interfaces.
- One Cisco router with one serial interface and one FXS interface.
- Three analog phone handsets.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports voice.

Configuration Overview

This lab will demonstrate a voice feature known as *number expansion.* Number expansion allows a corporate telephone network to be configured so that you can reach a destination by dialing only a portion of the full telephone number. Number expansion configures the router to recognize extension numbers and expand them into their full phone numbers. For this lab, we will have three handsets, which will be connected as shown in Figure 26−4. RouterB will be configured with number expansion so that when extension 3333 dials the number 611, a call will be placed to extension 2222.

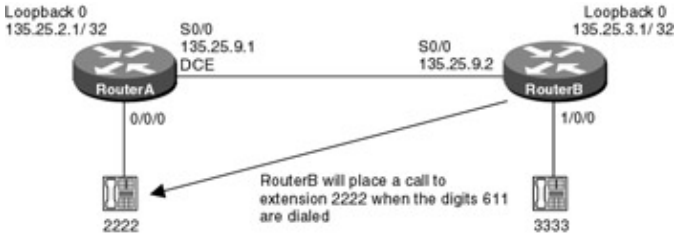


Figure 26−4: Number expansion

The routers are connected as shown in Figure 26−4. RouterA acts as a DCE and supplies clocking to RouterB.

Router Configuration

The configurations for the two routers in this example are as follows. Number expansion commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
voice-port 0/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 2222
 port 0/0/0
!
dial-peer voice 2 voip
 destination-pattern 3333
 session target ipv4:135.25.3.1
!
!
interface Loopback0
 ip address 135.25.2.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.1 255.255.255.252
 clockrate 800000
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
```

```
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
voice-port 1/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 3333
 port 1/0/0
!
dial-peer voice 2 voip
 destination-pattern 2222
 session target ipv4:135.25.2.1
!
num-exp 611 2222 ← Far end extension 2222 can be reached by dialing 611
!
interface Loopback0
 ip address 135.25.3.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.2 255.255.255.252
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
no ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Once the configurations have been entered for RouterA and RouterB, the user should verify that dial tone is present on all three handsets. The lab should be tested by taking extension 3333 off hook on RouterB and dialing the digits 666. Extension 2222 on RouterA should ring at the far end.

Number expansion can be verified by connecting to RouterB. The **show num-exp** command can be used to display all defined number aliases.

```
RouterB#show num-exp
Dest Digit Pattern = '611'     Translation =
                   '2222'
```

Lab #114: IP Precedence

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface and two FXS interfaces.
- One Cisco router with one serial interface and one FXS interface.
- Three analog phone handsets.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports voice.

Configuration Overview

This lab will demonstrate the capability of the Cisco IOS to set the IP precedence value for individual voice conversations. This feature allows you to give real−time voice traffic a higher priority than other network traffic. In this lab, RouterB will be configured to set the IP precedence value to 1 for calls placed to extension 2223. Calls placed from RouterB to extension 2222 will have an IP precedence value of 5.

The routers are connected as shown in Figure 26−5. RouterA acts as a DCE and supplies clocking to RouterB.

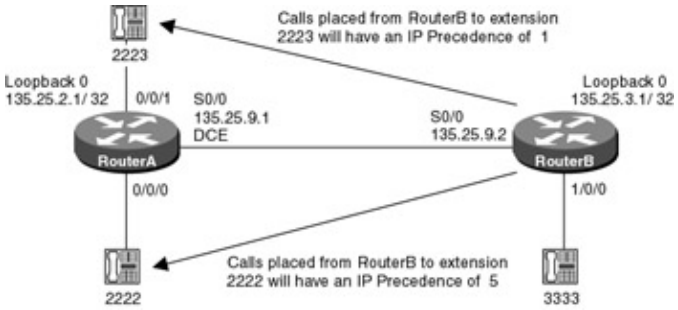


Figure 26−5: IP precedence

Router Configuration

The configurations for the two routers in this example are as follows. IP precedence commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterA
!
voice−port 0/0/0
!
voice−port 0/0/1
!
!
dial−peer voice 1 pots
 destination−pattern 2222
 port 0/0/0
!
dial−peer voice 2 voip
 destination−pattern 3333
 session target ipv4:135.25.3.1
!
dial−peer voice 3 pots
 destination−pattern 2223
 port 0/0/1
!
!
```

```
interface Loopback0
 ip address 135.25.2.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.1 255.255.255.252
 clockrate 800000
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
voice-port 1/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 3333
 port 1/0/0
!
dial-peer voice 2 voip
 destination-pattern 2223
 ip precedence 1 ← Set the IP precedence of this call to 1
 session target ipv4:135.25.2.1
!
dial-peer voice 3 voip
 destination-pattern 2222
 ip precedence 5 ← Set the IP precedence of this call to 5
 session target ipv4:135.25.2.1
!
interface Loopback0
 ip address 135.25.3.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.2 255.255.255.252
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
no ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password cisco
 login
```

```
!
end
```

Monitoring and Testing the Configuration

Once the configurations have been entered for RouterA and RouterB, the user should verify that dial tone is present on all three handsets. The user should verify that a call can be made from any extension to any other extension.

The IP precedence settings can be confirmed by connecting to RouterB. The **show dial−peer voice** command will verify that both of the voice peers on RouterB are using the proper IP precedence bits.

```
RouterB#show dial−peer voice
VoiceOverIpPeer2
     information type = voice, type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 1727, Charged Units = 0,
     Successful Calls = 6, Failed Calls = 0,
     Accepted Calls = 6, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2274609.
VoiceOverIpPeer3
     information type = voice,
     tag = 3, destination−pattern = '2222',
     answer−address = '', preference=0,
     group = 3, Admin state is up, Operation state is up,
     incoming called−number = '', connections/maximum = 0/unlimited,
     application associated:
     type = voip, session−target = 'ipv4:135.25.2.1',
     technology prefix:
     ip precedence = 5, UDP checksum = disabled,
     session−protocol = cisco, req−qos = best−effort,
     acc−qos = best−effort,
     fax−rate = voice,   payload size =  20 bytes
     codec = g729r8,   payload size =  20 bytes,
     tag = 2, destination−pattern = '2223',
     answer−address = '', preference=0,
     group = 2, Admin state is up, Operation state is up,
     incoming called−number = '', connections/maximum = 0/unlimited,
     application associated:
     type = voip, session−target = 'ipv4:135.25.2.1',
     technology prefix:
     ip precedence = 1, UDP checksum = disabled,
     session−protocol = cisco, req−qos = best−effort,
     acc−qos = best−effort,
     fax−rate = voice,   payload size =  20 bytes
     codec = g729r8,   payload size =  20 bytes,
     Expect factor = 10, Icpif = 30,signaling−
     Expect factor = 10, Icpif = 30,signaling−type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 1406, Charged Units = 0,
     Successful Calls = 15, Failed Calls = 0,
     Accepted Calls = 15, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2277544.
```

Lab #115: Custom Queuing for Voice Traffic

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface and two FXS interfaces.
- One Cisco router with one serial interface and one FXS interface.
- Three analog phone handsets.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports voice.

Configuration Overview

This lab will demonstrate how custom queuing can be used to dedicate more network bandwidth to voice traffic. Custom queuing allows the user to define up to 16 traffic queues. RouterA will be configured so that traffic going out its S0/0 interface will have 75 percent of the link bandwidth dedicated to voice traffic and 25 percent of the link bandwidth dedicated to all other traffic.

The routers are connected as shown in Figure 26−6. RouterA acts as a DCE and supplies clocking to RouterB.

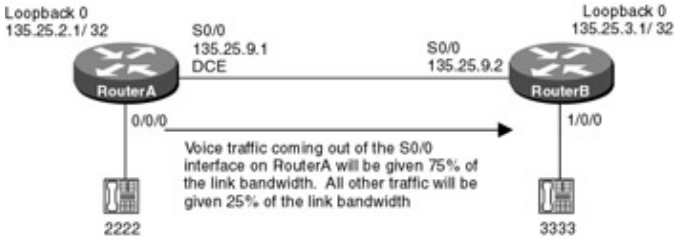


Figure 26−6: Custom queuing

Router Configuration

The configurations for the two routers in this example are as follows. Custom queuing commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
voice-port 0/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 2222
 port 0/0/0
!
dial-peer voice 2 voip
 destination-pattern 3333
 session target ipv4:135.25.3.1
!
!
interface Loopback0
 ip address 135.25.2.1 255.255.255.255
!
```

```
interface Serial0/0
 ip address 135.25.9.1 255.255.255.252
 custom-queue-list 1 ← Apply the queue list applied to an interface
 clockrate 800000
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
access-list 169 permit udp any any range 16384 16484 ← Voice traffic uses UDP
                                                        ports 16384 through
                                                        16484 and TCP port 1720
access-list 169 permit tcp any any eq 1720
queue-list 1 protocol ip 1 list 169 ← Put voice traffic into queue 1. Access
                                      list 169 will determine what goes into
                                      queue 1
queue-list 1 default 4 ← Queue 4 will handle all other traffic besides voice
queue-list 1 queue 1 byte-count 7500 ← Give 75% of the bandwidth to the voice
                                       traffic
queue-list 1 queue 4 byte-count 2500 ← Give 25% of the bandwidth to all other
                                       traffic
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
voice-port 1/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 3333
 port 1/0/0
!
dial-peer voice 2 voip
 destination-pattern 2222
 session target ipv4:135.25.2.1
!
interface Loopback0
 ip address 135.25.3.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.2 255.255.255.252
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
no ip classless
no ip http server
!
line con 0
```

```
 transport input none
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Once the configurations have been entered for RouterA and RouterB, the user should verify that dial tone is present on all three handsets. The user should verify that a call can be made from any extension to any other extension.

The custom queuing configuration can be verified by connecting to RouterA. The **show queuing** command will display the queuing configuration on the router.

```
RouterA#show queuing
Current fair queue configuration:

  Interface           Discard     Dynamic      Reserved
                      threshold   queue count  queue count
  Serial1/0           64          256          0

Current priority queue configuration:
Current custom queue configuration:

List   Queue  Args
1      4      default
1      1      protocol ip          list 169
1      1      byte-count 7500
1      4      byte-count 2500
Current random-detect configuration:
```

Lab #116: Priority Queuing for Voice Traffic

Equipment Needed

The following equipment is needed to perform this lab exercise:

- One Cisco router with one serial interface and two FXS interfaces.
- One Cisco router with one serial interface and one FXS interface.
- Three analog phone handsets.
- One Cisco crossover cable. If a Cisco crossover cable is not available, then you can use a Cisco DTE cable connected to a Cisco DCE cable.
- A Cisco rolled cable for console port connection to the routers.
- A Cisco IOS image that supports voice.

Configuration Overview

This lab will demonstrate how priority queuing can be used to dedicate more network bandwidth to voice traffic. Priority queuing uses four queues in a strict algorithm, which will always service the higher−priority queue if that queue has traffic to send. RouterB will be configured to give voice traffic going out its S0/0 interface a high priority. All other network traffic going out over RouterB's S0/0 interface will be given medium priority.

The routers are connected as shown in Figure 26−7. RouterA acts as a DCE and supplies clocking to RouterB.

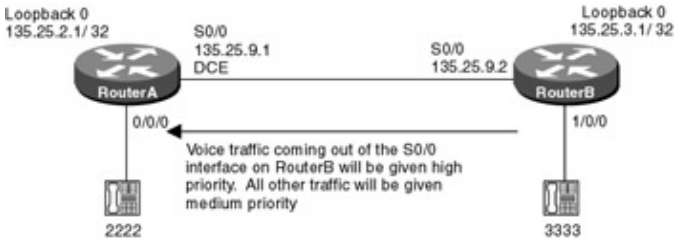


Figure 26−7: Priority queuing

Router Configuration

The configurations for the two routers in this example are as follows. Priority queuing commands are highlighted in bold.

RouterA

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterA
!
voice-port 0/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 2222
 port 0/0/0
!
dial-peer voice 2 voip
 destination-pattern 3333
 session target ipv4:135.25.3.1
!
!
interface Loopback0
 ip address 135.25.2.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.1 255.255.255.252
 clockrate 800000
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
```

```
no service password-encryption
!
hostname RouterB
!
voice-port 1/0/0
!
!
dial-peer voice 1 pots
 destination-pattern 3333
 port 1/0/0
!
dial-peer voice 2 voip
 destination-pattern 2222
 session target ipv4:135.25.2.1
!
interface Loopback0
 ip address 135.25.3.1 255.255.255.255
!
interface Serial0/0
 ip address 135.25.9.2 255.255.255.252
 priority-group 1 ← Assign the priority queue to the interface
!
router ospf 64
 network 135.25.0.0 0.0.255.255 area 0
!
no ip classless
no ip http server
!
access-list 105 permit udp any any range 16384 16484 ← Access list 105 will
                                                        match voice traffic
                                                        since voice traffic uses
                                                        UDP ports 16384 through
                                                        16484 and TCP port 1720
access-list 105 permit tcp any any eq 1720
priority-list 1 protocol ip high list 105 ← traffic which meets the criteria of
                                            access list 105 is high priority
                                            traffic
priority-list 1 default medium ← all other traffic will be assigned to the
                                 medium priority queue
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Monitoring and Testing the Configuration

Once the configurations have been entered for RouterA and RouterB, the user should verify that dial tone is present on all three handsets. The user should verify that a call can be made from any extension to any other extension.

To verify the priority queuing configuration, connect to RouterB. The **show interface s0/0** command will display the following information.

```
RouterB#show interface s0/0
Serial0/0 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 135.25.9.2/30
  MTU 1500 bytes, BW 800 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, loopback not set
```

```
  Keepalive set (10 sec)
  LCP Open
  Open: IPCP, CDPCP
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters 06:49:59
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queuing strategy: priority-list 1
  Output queue (queue priority: size/max/drops):
     high: 0/20/0, medium: 0/40/0, normal: 0/60/0, low: 0/80/0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     8797 packets input, 436014 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     8144 packets output, 396307 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

The **show queuing** command displays the queuing configuration of the router. We see that there are two queues defined. The high−priority queue is used for traffic that meets the conditions of access list 105, and the medium queue is used for all other traffic.

```
RouterB#show queuing
Current fair queue configuration:
Current priority queue configuration:

List   Queue  Args
1      medium default
1      high   protocol ip          list 105
Current custom queue configuration:
Current random-detect configuration:
```

Voice Monitoring and Troubleshooting Commands

This section will discuss key voice monitoring and troubleshooting commands.

{show call active voice} The **show call active voice** command can be used while a voice call is active to get information pertaining to the call. Information items such as peer address, call duration, connect time, and setup time are displayed.

```
RouterB#show call active voice
 GENERIC:
SetupTime=2473944 ms
Index=1
PeerAddress=2222
PeerSubAddress=
PeerId=3
PeerIfIndex=11
LogicalIfIndex=0
ConnectTime=2474064
CallDuration=00:00:47
CallState=4
CallOrigin=2
ChargedUnits=0
InfoType=2
TransmitPackets=848 ← Transmission packet counts
TransmitBytes=16960
ReceivePackets=689
ReceiveBytes=13780
VOIP:
ConnectionId[0x29C834F4 0x2287004F 0x0 0x1A5888C]
RemoteIPAddress=135.25.9.1
RemoteUDPPort=17532
```

```
RoundTripDelay=9 ms
SelectedQoS=best-effort
tx_DtmfRelay=inband-voice
```

{show dial-peer voice} You can use the **show dial-peer voice** command to verify that the dial peer has been properly configured. Notice how the target IP address for the specified extension is displayed.

```
RouterA#show dial-peer voice 2
VoiceOverIpPeer2
     information type = voice,
     tag = 2, destination-pattern = '3333', ← Destination number
     answer-address = '', preference=0,
     group = 2, Admin state is up, Operation state is up,
     incoming called-number = '', connections/maximum = 0/unlimited,
     application associated:
type = voip, session-target = 'ipv4:135.25.3.1', ← Establish calls to extension
                                                    3333 via this IP address
     technology prefix:
     ip precedence = 0, UDP checksum = disabled,
     session-protocol = cisco, req-qos = best-effort,
     acc-qos = best-effort,
     fax-rate = voice,   payload size =  20 bytes

          Compression Algorithm
               ↓
     codec = g729r8,   payload size =  20 bytes,
     Expect factor = 10, Icpif = 30,signaling-type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 3126, Charged Units = 0,
     Successful Calls = 25, Failed Calls = 4,
     Accepted Calls = 29, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2566163.
```

{show dialplan number} The **show dialplan number** command can be used to verify that a given called number will be dialed properly. We see next that dialing 3333 will establish a VoIP call to the target address 135.25.3.1.

```
RouterA#show dialplan number 3333
Macro Exp.: 3333

VoiceOverIpPeer2
     information type = voice,
     tag = 2, destination-pattern = '3333',
     answer-address = '', preference=0,
     group = 2, Admin state is up, Operation state is up,
     incoming called-number = '', connections/maximum = 0/unlimited,
     application associated:
     type = voip, session-target = 'ipv4:135.25.3.1',
     technology prefix:
     ip precedence = 0, UDP checksum = disabled,
     session-protocol = cisco, req-qos = best-effort,
     acc-qos = best-effort,
     fax-rate = voice,   payload size =  20 bytes
     codec = g729r8,   payload size =  20 bytes,
     Expect factor = 10, Icpif = 30,signaling-type = cas,
     VAD = enabled, Poor QOV Trap = disabled,
     Connect Time = 3126, Charged Units = 0,
     Successful Calls = 25, Failed Calls = 4,
     Accepted Calls = 29, Refused Calls = 0,
     Last Disconnect Cause is "10  ",
     Last Disconnect Text is "normal call clearing.",
     Last Setup Time = 2566163.
Matched: 3333   Digits: 4
Target: ipv4:135.25.3.1
```

{show num−exp} The **show num−exp** command can be used to display all defined number aliases. In the example that follows, dialing the digits 611 will cause an expansion to the digits 2222.

```
RouterB#show num−exp
Dest Digit Pattern = '611'     Translation =
                   '2222'
```

{show queuing} The **show queuing** command will display the queuing configuration on the router.

```
RouterA#show queuing
Current fair queue configuration:

  Interface           Discard     Dynamic       Reserved
                      threshold   queue count   queue count
  Serial1/0           64          256           0

Current priority queue configuration:
Current custom queue configuration:

List   Queue  Args
1      4      default
1      1      protocol ip          list 169
1      1      byte−count 7500
1      4      byte−count 2500
Current random−detect configuration:
```

{show voice port} The **show voice port** command can be used to verify the settings of a voice port on the router:

```
RouterA#show voice port

Foreign Exchange Station 0/0/0
 Type of VoicePort is FXS ← Port type
 Operation State is DORMANT
 Administrative State is UP
 No Interface Down Failure
 Description is not set
 Noise Regeneration is enabled
 Non Linear Processing is enabled
 Music On Hold Threshold is Set to −38 dBm
 In Gain is Set to 0 dB
 Out Attenuation is Set to 0 dB
 Echo Cancellation is enabled
 Echo Cancel Coverage is set to 8 ms
 Initial Time Out is set to 10 s
 Interdigit Time Out is set to 10 s
 Call−Disconnect Time Out is set to 60 s
 Ringing Time Out is set to 180 s
 Region Tone is set for US

 Analog Info Follows:
 Currently processing unknown
 Maintenance Mode Set to None (not in mtc mode)
 Number of signaling protocol errors are 0
 Impedance is set to 600r Ohm

 Voice card specific Info Follows:
 Signal Type is loopStart
 Ring Frequency is 25 Hz
 Hook Status is On Hook
 Ring Active Status is inactive
 Ring Ground Status is inactive
 Tip Ground Status is inactive
 Digit Duration Timing is set to 100 ms
 InterDigit Duration Timing is set to 100 ms
```

Conclusion

This chapter discussed the topic of Voice over IP. Six labs demonstrated the Cisco IOS voice capabilities.

Chapter 27: MPLS

Overview

Topics Covered in This Chapter

- Detailed technology overview
- Basic MPLS
- Building MPLS VPNs Using Static Routing
- Building MPLS VPNs Using OSPF

Introduction

Multiprotocol Label Switching (MPLS) is an emerging technology that uses labels to switch IP packets across a network. In traditional destination−based routing, each intermediate node makes a routing decision based on the destination IP address of the packet. With MPLS, a label is assigned to each packet at the edge of the network. When an intermediate node receives a labeled packet, it uses the label to make its switching decision, similar to layer 2 switching technologies like frame relay or ATM.

The main differences between MPLS and traditional layer 2 switching technologies are how the labels are assigned and the capability of MPLS to carry a stack of labels attached to a packet. The capability to stack labels allows new applications such as VPNs and traffic engineering across a network. It is these new applications that are driving many service providers to implement MPLS in their networks.

Terminology

Label switching: The technology where labels are assigned to packets for transport across a network. The packet carries a short fixed−length label that is used by intermediate nodes throughout the network to determine how to process and forward the data.

In a traditional IP network, packets are analyzed on a hop−by−hop basis at each intermediate node. The layer 3 header is checked and routing lookup is performed to determine how to forward the packet. With label switching, the packet is switched at intermediate nodes based on the label — similar to layer 2 switching techniques like frame relay and ATM.

Label switching provides the benfits of packet forwarding based on layer 2 switching along with benefits of layer 3 routing. The difference between label switching and traditional layer 2 switching protocols like frame relay is the way the labels are assigned and the ability to stack labels. This ability to stack labels is what makes MPLS such a promising technology. Label stacking allows MPLS to be used for new applications such as traffic engineering and VPNs.

CE Router: This is the customer edge router that is used to connect to the provider's network.

MPLS: Multiprotocol Label Switching is an emerging technology which uses labels instead of layer 3 address to switch IP packets through a network.

MP−BGP: Multiprotocol Border Gateway Protocol is an extension of the existing BGP protocol, which among other things allows it to carry VPN routes between PE routers.

P router: A provider router is a router in the provider's backbone that attaches directly with a PE router and is a routing peer with other P routers.

PE (provider edge) router: A router at the edge of a provider's network. PE routers translate IPv4 addresses into VPN−IPv4 12−byte addresses.

VRF: This stands for VPN routing/forwarding instance. VRF defines the VPN membership of a customer site attached to a PE router. Each VPN has its own VRF, so a customer belonging to a VPN only has access to a set of routes contained within that VRF.

Each VRF contains an IP routing table, a set of interfaces that use the forwarding table, and a set of rules and routing protocols for a particular customer. A customer's VRF contains all the routes available to the site from the VPNs of which it is a member.

VPN−IPv4: A 12−byte address that is a combination of the IPv4 address and the route distinguisher (RD). The first 8 bytes are the RD; the next 4 bytes are an IPv4 address.

RD: The route distinguisher (RD) is a 64−bit prefix that consists of a 2−byte Type field, and a 6−byte Value field. The RD is added to the beginning of the customer's IPv4 prefixes to change them into globally unique VPN−IPv4 prefixes. An RD is either ASN relative, in which case it is composed of an autonomous system number and an arbitrary number, or it is IP address relative, in which case it is composed of an IP address and an arbitrary number.

Technology Overview

MPLS Header

At the edge of the network, PE routers label IP packets with a 32−bit MPLS header. The MPLS header, also referred to as a shim header, is placed between the layer 2 header and the layer 3 packet. Let's take, for example, frame relay — in a traditional IP over frame relay environment, the layer 3 IP packet will follow the frame relay header in the layer 2 frame. With MPLS, a new "shim header" is placed between the frame relay header and the layer 3 packet.

Frame relay header	**Layer 3 packet**

Position of layer 3 packet in a layer 2 frame

Frame relay header	**MPLS label layer 3**	**Layer 3 packet**

Position of MPLS label in a layer 2 frame shim header

This MPLS header is composed of an MPLS label, which is 20 bits, a 3−bit experimental field that is used to carry class of service information (similar to the TOS field in an IP packet), a 1−bit field (called the S bit) that is used to indicate if this is the last label in the stack, and an 8−bit Time to Live field (TTL) that is used in loop prevention — similar to the way the TTL field is used in IP.

20 bits	3bits	1bit	8bits
LABEL	***EXP***	***S***	***TTL***

Since the MPLS label is inserted between the layer 2 frame header and the layer 3 packet, the receiving router needs some way to determine that an incoming frame is an MPLS−labeled packet versus a traditional IP datagram. To accomplish this, the IETF has defined new protocol types to identify MPLS−labeled packets within various layer 2 protocols. In the example of frame relay, a snap header is used with an ethertype value of 0x8847.

Label Distribution

IP packets are labeled at the edge of the network and that label is used throughout the domain to switch the packet. How does the edge router know what label to assign to a packet, and how do the intermediate routers know how to forward the packet based on that label? The answer is through label bindings created by label−binding protocols.

Cisco implements two label−binding protocols that are used to bind IP prefixes with MPLS labels: Tag Distribution Protocol (TDP), which is Cisco's proprietary protocol, and Label Distribution Protocol (LDP), which is the IETF standard.

LDP or TDP are run between MPLS−enabled routers to distribute label−binding information. When a router is first configured for MPLS, a label information base (LIB) structure is created in the router. At that point, every IGP prefix in the routing table is assigned an MPLS label and the mapping between the two is stored in the LIB. LDP or TDP is used to distribute the IP prefix/label−binding information to all neighboring MPLS routers.

The neighboring routers store the binding information in their label−forwarding information base (LFIB) if the binding for the prefix comes from a downstream neighbor, meaning that the binding comes from the neighbor that would normally be used as the next hop by the IGP to reach a destination. This label binding from the next−hop router and the local label binding are entered into the LFIB.

If no label−binding information is received from the next−hop router, the LFIB entry marks the prefix as unlabeled, which tells the router to forward the packet with no label. If the network is directly connected to the router, the LFIB assigns a null label to the prefix. This is used to tell the router that IP layer forwarding is needed for that packet.

Let's look at the following simple scenario to see how label binding works:

1. RouterA is advertising network 192.1.1.0 via OSPF. RouterD knows that to reach network 192.1.1.0 it should forward the packet out its serial interface 0 to next hop 195.1.1.1. RouterC knows that to reach 192.1.1.0 it should forward the packet out serial interface 0 to next hop 194.1.1.1. RouterB knows that to reach 192.1.1.0 it should forward the packet out serial interface 1 to next hop 193.1.1.1 (see Figure 27−1). RouterA knows that network 192.1.1.0 is directly connected to interface Ethernet 0.

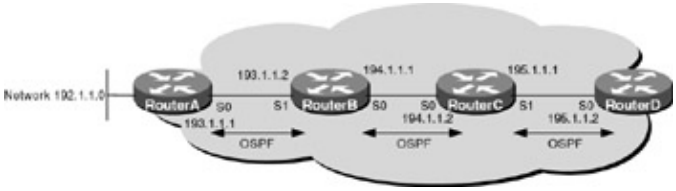


Figure 27−1: Advertising reachability via OSPF

2. MPLS is enabled and LDP or TDP is configured on each of the routers.
3. A soon as MPLS is enabled, the router builds an LIB assigning every prefix in the IGP table an MPLS label. For simplicity, we will only look at the label bindings for prefix 192.1.1.0 (see Figure 27−2).

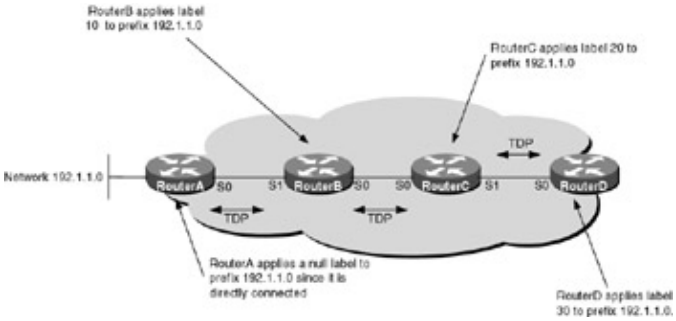


Figure 27−2: TDP exchange

4. RouterA applies a null label to the prefix since it is directly connected. RouterB assigns label 10 and RouterC assigns label 20.
5. Using TDP or LDP, RouterA sends its binding information to RouterB indicating that it should use a null label when forwarding a packet to RouterA destined for network 192.1.1.0. The null label tells RouterB that it should pop the label and forward the packet as normal IP. RouterB sends its binding information to RouterC indicating that it should use label 10 when sending a packet destined for network 192.1.1.0. There are no downstream neighbors to RouterC.

6. The information gets loaded into the LFIB of each router.
7. A packet arrives at RouterD destined for network 192.1.1.0. RouterD looks at its LFIB, which indicates that the packet should be labeled as 20 and sent out interface S0. RouterC sees an incoming MPLS−labeled packet with a label of 20 coming in serial interface S1. It looks in its LFIB, which indicates that the label should be swapped with label 10 and forwarded out interface S0. RouterB sees an incoming MPLS−labeled packet with a label of 10 and pops the label before forwarding to RouterA (see Figure 27−3).

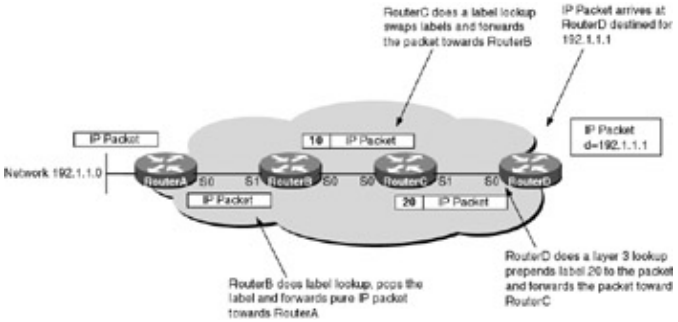


Figure 27−3: Label switching to destination 192.1.1.0

MPLS/VPNs

A virtual private network (VPN) is a network where customer connectivity among multiple sites is deployed on a shared infrastructure. The network appears to the user as a private network, providing the same policies and benefits. An example of a VPN would be one built using traditional layer 2 technologies like frame relay or ATM.

Other IP VPN technologies exist, such as IPSEC, L2TP, L2F, and GRE — all of which work well in a hub−and−spoke topology. However, today's networks require any−to−any communication. To support this using the frame relay or tunneling protocols mentioned previously would require a full mesh of tunnels or PVCs to be created between all member sites. Provisioning and managing a full−mesh topology using these traditional technologies is not supportable in a network that requires thousands or tens of thousands of VPNs.

MPLS/VPN allows full−mesh VPNs to be built in a scalable and manageable fashion over an IP backbone. MPLS/VPNs provide traffic separation amongst subscribers by assigning a unique VRF to each customer's VPN. Since users in one VPN have no knowledge of users in another VPN, the same level of user separation is achieved as with traditional layer 2 VPN technologies like frame relay and ATM.

Technology Overview

So, how are secure VPNs built using MPLS? There are four major technologies that provide the ability to build MPLS−VPNs: multiprotocol BGP, route filtering based on the "route target" extended BGP community attribute, MPLS forwarding to carry the packets across the backbone, and support for routing and forwarding instances in the provider edge routers.

Multiprotocol BGP is run between provider edge routers to exchange VPN prefix information. Multiprotocol BGP is an extension to the existing BGP protocol, allowing it to carry customer VPN−IPv4 address prefixes. A customer VPN−IPv4 address is a 12−byte address that is a combination of the IPv4 address and the route distinguisher (RD). The first 8 bytes are the RD; the next 4 bytes are an IPv4 address.

The RD is a 64−bit prefix that consists of a 2−byte Type field and a 6−byte Value field. The RD is added to the beginning of the customer's IPv4 prefixes to change them into globally unique VPN−IPv4 prefixes. An RD is either ASN relative, in which case it is composed of an autonomous system number and an arbitrary number, or it is IP address relative, in which case it is composed of an IP address and an arbitrary number. This is needed since VPNs may not be using unique addresses, which may overlap with other VPNs. The combination of the RD with the IP address ensures that the new VPN−IPv4 address is unique.

A VPN routing/forwarding instance (VRF) is defined on each PE router for each VPN. The VRF defines the VPN membership of a customer site attached to a PE router. Each VPN has its own VRF, so a customer belonging to a VPN only has access to a set of routes contained within that VRF.

Each VRF contains an IP routing table, a set of interfaces that use the forwarding table, and a set of rules and routing protocols for a particular customer. A customer's VRF contains all the routes available to the site from the VPNs of which it is a member. Standard IP forwarding is used between the PE and CE routers. The PE associates each CE with a per−site forwarding table that contains only the set of routes available to that CE router. Between the CE and the PE, static or dynamic routing can be used to populate the VRF forwarding table. Between the PE routers, multiprotocol BGP is used to advertise VPN prefixes. When a PE router advertises VPN−IPv4 addresses to another PE, it uses a 32−bit IP address (usually its loopback address) as the BGP next−hop address. Also, the PE that originates the VPN route assigns a label to that route. The label is passed in the multiprotocol BGP update along with the VPN−IPv4 address. This label is used by the egress PE to direct data packets to the correct CE.

MPLS forwarding is used across the provider backbone. Each PE router has a label binding for the 32−bit next−hop multiprotocol BGP address for every other PE. When a data packet is forwarded across the backbone, two labels are used. The top label directs the packet to the appropriate egress PE router. This is the label that was assigned to the 32−bit address of the neighboring PE. The second label, which was assigned by the originating PE, indicates how that egress PE should forward the packet.

Let's look at the following simple scenario to see how MPLS/VPNs are built:

1. MPLS is run in the core. Each PE router advertises its 32−bit loopback address: PE 1 advertises 1.1.1.1/32 and PE 2 advertises 2.2.2.2/32 . As shown earlier in the chapter, TDP or LDP is used to distribute binding information between MPLS−enabled routers. On each PE router the LFIB contains a label binding for the 32−bit loopback address of the other PE router. When PE1 forwards a packet to 2.2.2.2 on PE2, it will append label 20 to the packet and when PE2 forwards a packet to 1.1.1.1, it will prepend label 10 to the packet (see Figure 27−4).

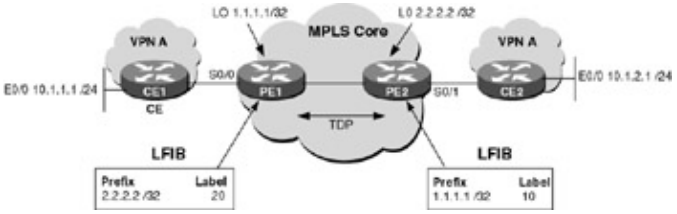


Figure 27−4: MPLS VPNs

2. A VPN routing and forwarding instance is created on PE1 and PE2, called VPNA.
3. PE1 places interfaces S0/0 in this VPN and PE2 places interface S0/1 in this VPN.
4. OSPF is run between PE1 and CE1 and PE2 and CE2.
5. When PE1 receives the route to network 10.1.1.0 from CE1, it places it in VPNA's routing table. At this time, it assigns label (5) to the prefix. When PE2 receives the route to network 10.1.2.0 from CE2, it places it in VPNA's routing table. At this time, it assigns label (6) to the prefix (see Figure 27−5).

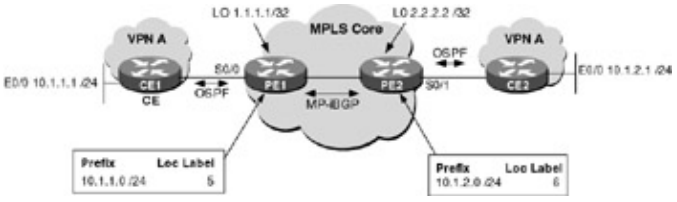


Figure 27−5: MPLS VPNs

6. PE1 then sends a multiprotocol MP−iBGP update to PE2 advertising network 10.1.1.0. The update also contains the label (5) that PE1 assigned to prefix 10.1.1.0, and PE2 should append to any packet destined to network 10.1.1.0 before forwarding it. When PE1 advertised the route, it set the BGP next−hop address to 1.1.1.1/32, which is its loopback address.

7. PE2 then sends a multiprotocol iBGP update to PE1 advertising network 10.1.2.0. The update also contains the label (6), which PE2 assigned to prefix 10.1.2.0 and PE1 should append to the packet destined to network 10.1.2.0 before forwarding it. When PE2 advertised the route, it set the BGP next−hop address to 2.2.2.2/32, which is its loopback address.
8. PE1 loads prefix 10.1.2.0 into VPNA's routing table and PE2 loads prefix 10.1.1.0 into VPNA's routing table.
9. At this point, if you were to display VPNA's routing table on PE1, you would see that 10.1.2.0 is reachable via 2.2.2.2. Similarly, if you were to display VPNA's routing table on PE2, you would see that 10.1.1.0 is reachable via 1.1.1.1.
10. The routes are propagated down to the CE routers using OPSF, and at this point the network has converged.
11. CE1 now sends a packet destined for host 10.1.2.1. The packet is forwarded to PE1. PE1 prepends an inner label of 6 to the packet. It then looks up the destination in VPNA's routing table. It determines that that the next−hop IP address is 2.2.2.2. It looks in its LFIB to determine what the outgoing label should be. At this point, PE1 prepends an outer label of 20 on the packet and forwards out its serial interface towards PE2. The outer label is 20 and the inner label is 5 (see Figure 27−6).

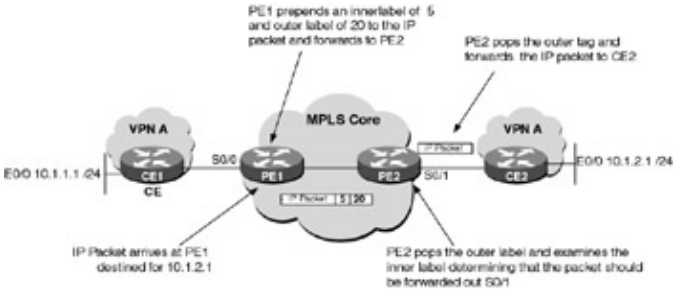


Figure 27−6: MPLS VPNs

12. When PE2 receives the labeled packet, it strips the outer label 20 off and examines the inner label. The inner label (5) tells the router which interface it should forward the packet out of. The packet is then forwarded to CE2.

Commands Discussed in This Chapter

- **address−family vpnv4**
- **clear ip route vrf**
- **exit−address−family**
- **ip cef**
- **ip route vrf**
- **ip vrf forwarding**
- ip vrf
- **neighbor activate**
- **rd**
- route−target
- **show ip bgp vpnv4**
- **show ip route vrf**
- show ip vrf
- **tag−switching ip**

Definitions

address−family vpnv4: The **address−family** command puts you in address family configuration submode. Within this mode, you can configure address−family−specific parameters for routing protocols, such as BGP, that can accommodate multiple layer 3 address families.

clear ip route vrf: This global command clears routes from the VPN routing table.

exit−address−family: This address family submode command is used to exit from the address family submode configuration.

ip cef: This global configuration command enables Cisco express forwarding (CEF) on the router.

ip route vrf: This global command is used to establish static routes for a VRF.

ip vrf forwarding: This interface command is used to associate a VRF with an interface or subinterface.

ip vrf: This global configuration command is used to configure a VRF routing table.

neighbor activate: This router configuration command is used to enable the exchange of information with a BGP neighboring router.

RD: This VRF submode command creates routing and forwarding tables and specifies the default route distinguisher (RD) for a VPN. The RD is added to the beginning of the customer's IPv4 prefixes to change them into globally unique VPN−IPv4 prefixes.

route−target: This VRF subcommand creates a list of import and/or export route target communities for the VRF.

show ip bgp vpnv4: This exec command is used to display VPN address information from the BGP table.

show ip route vrf: This exec command is used to display the IP routing table associated with a VRF.

show ip vrf: This exec command displays the set of defined VRFs and interfaces on the router.

tag−switching ip: This interface command enables tag switching on the interface and allows MPLS to function between the adjacent LSRs.

IOS Requirements

MPLS first appeared in IOS version 12.0. All labs in this chapter were done using IOS 12.1.6.

Lab #117: Basic MPLS

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having one Ethernet and one serial port
- Two Cisco routers, each having two serial interfaces
- Cisco IOS capable of running MPLS
- PC running a terminal emulation program
- Three Cisco DTE/DCE crossover cables
- A Cisco rolled cable

Configuration Overview

This lab will demonstrate a basic MPLS configuration. All of the routers will be configured for MPLS and use Tag Distribution Protocol (TDP) to distribute label bindings between routers.

All of the routers are connected serially via crossover cables. RouterB will act as the DCE supplying clock to RouterA and RouterC, and RouterC will supply clock to RouterD. OSPF will be run on all networks on all of the routers. TDP will be run on each router to distribute the label−binding information. The IP addresses are assigned as per Figure 27−7.

Figure 27−7: Basic MPLS configuration

Router Configurations

The configurations for the four routers in this example are as follows. Key MPLS configurations are in bold. Cisco express forwarding (CEF) must be enabled in all routers running MPLS. Routers that are receiving unlabeled IP packets that will be propagated as labeled packets must have the ingress interface configured for CEF.

RouterA

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterA

ip subnet−zero
no ip finger
!
ip cef ← CEF must be enabled in all routers running MPLS
!
interface Ethernet0/0
 ip address 192.1.1.1 255.255.255.0
 no keepalive
!
interface Serial0/0
 ip address 193.1.1.1 255.255.255.252
 no ip mroute−cache
 tag−switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets
no fair−queue

!
router ospf 64
 log−adjacency−changes
 network 192.1.1.0 0.0.0.255 area 0
 network 193.1.1.0 0.0.0.255 area 0
!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
ip subnet-zero
no ip finger
!
ip cef ← CEF must be enabled in all routers running MPLS
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets

 clockrate 1000000
!
interface Serial0/1
 ip address 194.1.1.1 255.255.255.252
tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                   unlabeled packets

 clockrate 1000000
!
router ospf 64
 log-adjacency-changes
 network 193.1.1.0 0.0.0.255 area 0
 network 194.1.1.0 0.0.0.255 area 0
!
ip classless
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterC
!
!
ip subnet-zero
no ip finger
!
ip cef ← CEF must be enabled in all routers running MPLS
!
!
interface Serial0/0
 ip address 194.1.1.2 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets
```

```
!
interface Serial0/1
 ip address 195.1.1.1 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets

 clockrate 1000000
!
router ospf 64
 log-adjacency-changes
 network 194.1.1.0 0.0.0.255 area 0
 network 195.1.1.0 0.0.0.255 area 0
!
ip classless
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterD
!
ip subnet-zero
no ip finger
!
ip cef ← CEF must be enabled in all routers running MPLS
!
interface Ethernet0/0
 ip address 191.1.1.1 255.255.255.0
 no keepalive

!
interface Serial0/0
 ip address 195.1.1.2 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets
!
router ospf 64
 log-adjacency-changes
 network 195.1.1.0 0.0.0.255 area 0
network 191.1.1.0 0.0.0.255 area 0
!
ip classless
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

Verify that the serial interface on RouterA is configured for tag switching with the command **show tag−switching interfaces.** The following is the output from the command. Notice that serial interface S0/0 is configured for tag switching and is operational, and is not configured for tunneling. "Operational" means that packets are being tagged, while "tunneling" refers to TSP tunnel tagging.

```
RouterA#show tag−switching interfaces
Interface              IP    Tunnel   Operational
Serial0/0              Yes   No       Yes
```

Display the TDP neighbors on RouterA with the command **show tag−switching tdp neighbor.** The following is the output from the command. Notice that RouterA has one TDP peer (194.1.1.1), which is the TDP identifier of RouterB.

The TDP identifier is determined in the same way as an OSPF router ID — the highest loopback IP address is used. If no loopbacks are configured on the router, the highest IP address of an operational interface is used. Like OSPF, the identifier can also be manually set. The global command **tag−switching tdp router−id** can be used to manually set the TDP identifier.

```
RouterA#show tag−switching tdp neighbor
Peer TDP Ident: 194.1.1.1:0; Local TDP Ident 193.1.1.1:0
        TCP connection: 194.1.1.1.11000 − 193.1.1.1.711
        State: Oper; PIEs sent/rcvd: 230/232; ; Downstream
        Up time: 03:18:39
        TDP discovery sources:
          Serial0/0
        Addresses bound to peer TDP Ident:
          193.1.1.2       194.1.1.1
```

Display the TDP binding information on RouterA with the command **show tag−switching tdp bindings**. The following is the output from the command. Notice that RouterA has two bindings for each routing entry: a local binding and a remote binding. The remote binding is the outgoing tag for this destination learned from a neighboring tag switching router (TSR). The local tag is the tag applied for the particular prefix by the local router. For each remote binding, the TSR from which the outgoing tag was learned is listed as well as the tag itself. The TSR is identified by its TDP identifier.

Some of the entries have an imp−null as the tag. An imp−null is a tag value that instructs the router to pop the tag entry off the tag stack before forwarding the packet.

```
RouterA#show tag−switching tdp bindings
  tib entry: 192.1.1.0/24, rev 3
        local binding:  tag: imp−null
        remote binding: tsr: 194.1.1.1:0, tag: 26
  tib entry: 193.1.1.0/30, rev 4
        local binding:  tag: imp−null
        remote binding: tsr: 194.1.1.1:0, tag: imp−null
  tib entry: 194.1.1.0/30, rev 6
        local binding:  tag: 26
        remote binding: tsr: 194.1.1.1:0, tag: imp−null
  tib entry: 195.1.1.0/30, rev 8
        local binding:  tag: 27
        remote binding: tsr: 194.1.1.1:0, tag: 27
```

Now let's examine the tag−forwarding tables of each router in the network, looking particularly at network 192.1.1.0. Display the tag−forwarding table on RouterD with the command **show tag−switching forwarding−table.** The following is the output from the command. Notice that to reach network 192.1.1.0, RouterD will tag the packet as 26 and forward out interface S0/0 to RouterC.

```
RouterD#show tag−switching forwarding−table
Local  Outgoing    Prefix           Bytes tag  Outgoing   Next Hop
```

```
tag    tag or VC   or Tunnel Id      switched   interface
26     26          192.1.1.0/24      0          Se0/0      point2point
27     27          193.1.1.0/30      0          Se0/0      point2point
28     Pop tag     194.1.1.0/30      0          Se0/0      point2point
```

Display the tag−forwarding table on RouterC. The following is the output. When RouterC receives an incoming packet with a tag of 26, it will forward the packet out interface S0/0 to RouterB with a tag of 26.

```
RouterC#show tag−switching forwarding−table
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
26     26          192.1.1.0/24      653940     Se0/0      point2point
27     Pop tag     193.1.1.0/30      0          Se0/0      point2point
```

Now display the tag−forwarding table on RouterB. The following is the output. When RouterB receives an incoming packet with a tag of 26, it will forward the packet out interface S0/0 to RouterA untagged.

```
RouterB#show tag−switching forwarding−table
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
26     Pop tag     192.1.1.0/24      629720     Se0/0      point2point
27     Pop tag     195.1.1.0/30      629720     Se0/1      point2point
```

Enable tag−switching debugging on RouterC and RouterB with the command **debug tag−switching** packets. Now ping from RouterD to 192.1.1.1.

The following is the output from the debug command on RouterC. Note that RouterC received a packet on interface S0/1 from RouterD with a tag of 26 and transmitted the packet out interface S0/0 to RouterB with a tag of 26.

```
04:32:04: TAG: Se0/1: recvd: CoS=0, TTL=255, Tag(s)=26
04:32:04: TAG: Se0/0: xmit: CoS=0, TTL=254, Tag(s)=26
04:32:04: TAG: Se0/1: recvd: CoS=0, TTL=255, Tag(s)=26
04:32:04: TAG: Se0/0: xmit: CoS=0, TTL=254, Tag(s)=26
04:32:04: TAG: Se0/1: recvd: CoS=0, TTL=255, Tag(s)=26
04:32:04: TAG: Se0/0: xmit: CoS=0, TTL=254, Tag(s)=26
04:32:04: TAG: Se0/1: recvd: CoS=0, TTL=255, Tag(s)=26
```

The following is the output from the debug command on RouterB. Note that RouterB received a packet on interface S0/1 from RouterC with a tag of 26 and transmitted the packet out interface S0/0 to RouterA untagged.

```
04:32:44: TAG: Se0/1: recvd: CoS=0, TTL=254, Tag(s)=26
04:32:44: TAG: Se0/0: xmit: (no tag)

04:32:44: TAG: Se0/1: recvd: CoS=0, TTL=254, Tag(s)=26
04:32:44: TAG: Se0/0: xmit: (no tag)
```

Lab #118: Building MPLS VPNs UsingStatic Routing

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having one Ethernet and one serial port
- Two Cisco routers, each having two serial interfaces
- Cisco IOS capable of running MPLS
- PC running a terminal emulation program

- Three Cisco DTE/DCE crossover cables
- A Cisco rolled cable

Configuration Overview

This lab will demonstrate a basic MPLS VPN configuration. All of the routers will be configured for MPLS and use Tag Distribution Protocol (TDP) to distribute label bindings between routers. RouterB and RouterC will be configured as provider edge routers, and RouterA and RouterD will be configured as customer edge routers in VPNA.

MPLS/VPN will be used to create a virtual private network between RouterA and RouterD across an MPLS core.

All of the routers are connected serially via crossover cables. RouterB will act as the DCE supplying clock to RouterA and RouterC, and RouterC will supply clock to RouterD. OSPF will be run between RouterB and RouterC. MP−BGP will be run between RouterB and RouterC to advertise customer VPN routes between the two PE routers. RouterB and RouterC learn the routes from VPNA through static routing. Tag Distribution Protocol will be run on each router to distribute the label binding information.

RouterA and RouterD will have a default route pointing to RouterB and RouterC, respectively. The IP addresses are assigned as per Figure 27−8.

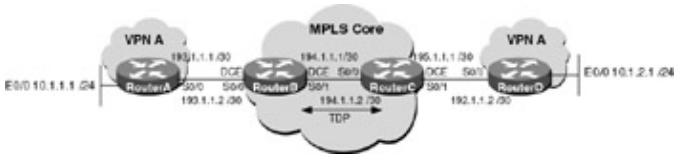


Figure 27−8: Basic MPLS configuration

Router Configurations

The configurations for the four routers in this example are as follows. Key MPLS configurations are in bold. Cisco express forwarding (CEF) must be enabled in all routers running MPLS. Routers that are receiving unlabeled IP packets that will be propagated as labeled packets must have the ingress interface configured for CEF. MPLS/VPN commands will be added and explained in the monitoring and troubleshooting section.

RouterA

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterA

ip subnet−zero
no ip finger
!
!
interface Ethernet0/0
 ip address 10.1.1.1 255.255.255.0
 no keepalive
!
interface Serial0/0
 ip address 193.1.1.1 255.255.255.252
 no ip mroute−cache
 no fair−queue

!
ip classless
```

```
ip route 0.0.0.0 0.0.0.0 193.1.1.2
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
ip subnet-zero
no ip finger
!
ip cef ← CEF must be enabled in all routers running MPLS
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets
 clockrate 1000000
!
interface Serial0/1
 ip address 194.1.1.1 255.255.255.252
tag-switching ip  ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets
 clockrate 1000000
!
router ospf 64
 log-adjacency-changes
 network 194.1.1.0 0.0.0.255 area 0
!
ip classless
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterC
!
!
ip subnet-zero
no ip finger
!
```

```
ip cef ← CEF must be enabled in all routers running MPLS
!
!
interface Serial0/0
 ip address 194.1.1.2 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets

!
interface Serial0/1
 ip address 195.1.1.1 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets

 clockrate 1000000
!
router ospf 64
 log-adjacency-changes
 network 194.1.1.0 0.0.0.255 area 0
!
ip classless
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterD
!
ip subnet-zero
no ip finger
!
!
interface Ethernet0/0
 ip address 10.1.2.1 255.255.255.0
 no keepalive
!
interface Serial0/0
 ip address 195.1.1.2 255.255.255.252
!
ip classless
ip route 0.0.0.0 0.0.0.0 195.1.1.1
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

PE routers (RouterB and RouterC) use MP−iBGP to distribute VPN routes to one another. When a PE router advertises VPN routes to another PE, it does so using itself as the BGP next−hop address. This address should be the 32−bit loopback address on the router. This 32−bit address also needs to be advertised into the IGP routing tables of the backbone. This enables MPLS to assign a label corresponding to the route to each PE router. The following commands configure a 32−bit loopback address on RouterB and RouterC, and advertise this address via OSPF.

```
RouterB(config)#interface loopback 0
RouterB(config−if)#ip address 1.1.1.1 255.255.255.255
RouterB(config−if)#exit
RouterB(config)#router ospf 64
RouterB(config−router)#network 1.1.1.1 0.0.0.0 area 0

RouterC(config)#interface loopback 0
RouterC(config−if)#ip address 2.2.2.2 255.255.255.255
RouterC(config−if)#exit
RouterC(config)#router ospf 64
RouterC(config−router)#network 2.2.2.2 0.0.0.0 area 0
```

As discussed earlier in the chapter, MPLS/VPN uses MP−iBGP to distribute VPN routes from PE to PE. The next step is to configure MP−iBGP between the two PE routers — in our case, RouterC and RouterB. By default, when a BGP session on a Cisco router is configured, it is activated to carry IPV4 addresses. The command **no bgp default ipv4−unicast** disables this behavior. The following commands enable the BGP process and disable the default ipv4 Unicast behavior on RouterC and RouterB:

```
RouterB(config)#router bgp 1
RouterB(config−router)#no bgp default ipv4−unicast

RouterC(config)#router bgp 1
RouterC(config−router)#no bgp default ipv4−unicast
```

The BGP neighbors are now defined under the global BGP process. The neighbor address is the 32−bit loopback address of the remote router. The update source (the address that BGP advertises as the next hop) should be the loopback address of the local router. The last step is to activate the neighbor. The following commands configure the IBGP session between RouterB and RouterC.

```
RouterB(config)#router bgp 1
RouterB(config−router)#neighbor 2.2.2.2 remote−as 1
RouterB(config−router)#neighbor 2.2.2.2 update−source loopback 0
RouterB(config−router)#neighbor 2.2.2.2 activate

RouterC(config)#router bgp 1
RouterC(config−router)#neighbor 1.1.1.1 remote−as 1
RouterC(config−router)#neighbor 1.1.1.1 update−source loopback 0
RouterC(config−router)#neighbor 1.1.1.1 activate
```

The BGP session now needs to be activated to carry VPN−IPv4 prefixes. This is done through the use of an address family. The VPN−IPv4 address family is configured under the BGP process and then the neighbor is activated. The following commands configure the BGP neighbor session on RouterB and RouterC to carry VPN−IPv4 prefixes.

```
RouterB(config)#router bgp 1
RouterB(config−router)#address−family vpnv4
RouterB(config−router−af)#neighbor 2.2.2.2 activate

RouterC(config)#router bgp 1
RouterC(config−router)#address−family vpnv4
RouterC(config−router−af)#neighbor 1.1.1.1 activate
```

Verify that the BGP session is established and configured to carry VPN−IPv4 prefixes on RouterC. To do this, display the BGP neighbors with the command **show ip bgp neighbors**. The following is the truncated output from the command on RouterC. Notice the BGP state is established and the neighbor is configured to support VPN−IPv4.

```
RouterC#show ip bgp neighbors
BGP neighbor is 1.1.1.1,  remote AS 1, internal link
  BGP version 4, remote router ID 1.1.1.1
  BGP state = Established, up for 00:03:03
  Last read 00:00:03, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
    Address family VPNv4 Unicast: advertised and received
  Received 22 messages, 0 notifications, 0 in queue
  Sent 22 messages, 0 notifications, 0 in queue
  Route refresh request: received 0, sent 0
  Default minimum time between advertisement runs is 5 seconds
```

Next, a VRF instance must be configured on RouterB and RouterC for VPNA. Under the VRF, a route distinguisher (RD) must be configured.

Routes from that VRF will be tagged with the RD. The purpose of the RD is to create distinct routes to a common IPv4 address prefix, allowing the same IP address to be used across multiple VPNs. The PE can be configured to associate all routes leading to the same CE with the same RD, or it may be configured to associate different routes with different RDs — even if they lead to the same CE. In this lab, all routes in VPNA will have the RD of 1:100.

The following commands configure vrf VPNA using RD 1:100 on the two routers.

```
RouterB(config)#ip vrf VPNA
RouterB(config−vrf)#rd 1:100

RouterC(config)#ip vrf VPNA
RouterC(config−vrf)#rd 1:100
```

Configure the import and export policies for each VRF. These policies are used to advertise routes out of the VRF (export) and populate routes into the VRF (import). The following commands configure VPNA on RouterA and RouterB to export all routes with the extended community route target of 1:100 and to import all routes into the VRF that have a route target of 1:100.

```
RouterB(config)#ip vrf VPNA
RouterB(config−vrf)#route−target both 1:100

RouterC(config)#ip vrf VPNA
RouterC(config−vrf)#route−target both 1:100
```

After the VRF is configured on the PE, you must tell the router which interfaces belong in that particular VRF. This is done under the interface with the command **ip vrf forwarding.** The following commands associates interface S0/1 on RouterB connecting to RouterA and interface S0/1 on RouterC connecting to RouterD with vrf VPNA.

```
RouterB(config)#interface s0/1
RouterB(config−if)#ip vrf forwarding VPNA

RouterC(config)#interface s0/1
RouterC(config−if)#ip vrf forwarding VPNA
```

After you configure the interface, you will receive the following message indicating the IP address has been removed. The IP address will need to be re−added.

```
% Interface Serial0/1 IP address 195.1.1.1 removed due to enabling VRF VPNA
```

Verify the VPN configuration on RouterC with the command **show ip vrf detail VPNA**. The following is the output from the command. Notice that the RD is set to 1:100, interface S0/1 is associated with the VRF, and the import and export policies are configured.

```
RouterC#show ip vrf detail VPNA
VRF VPNA; default

   RD 1:100
  Interfaces:
    Serial0/1
  Connected addresses are not in global routing table
  Export VPN route-target communities
    RT:1:100
  Import VPN route-target communities
    RT:1:100
  No import route-map
  No export route-map
```

The final step is to get the customer prefixes (routes from RouterA and RouterB) into the VRF. This can be accomplished by running a dynamic routing protocol such as RIP or OSPF between the PE and CE (RouterA and RouterB) or through static routes. This lab will use static routing. The static route for vrf VPNA will need to be configured and redistributed into MP−iBGP under the address family.

The following commands configure the static route and redistribute it into MP−iBGP on RouterB and RouterC.

```
RouterB(config)#ip route vrf VPNA 10.1.1.0 255.255.255.0 serial 0/0

RouterB(config)#router bgp 1
RouterB(config-router)#address-family ipv4 vrf VPNA
RouterB(config-router-af)#redistribute static

RouterC(config)# ip route vrf VPNA 10.1.2.0 255.255.255.0 serial 0/1

RouterC(config)#router bgp 1
RouterC(config-router)#address-family ipv4 vrf VPNA
RouterC(config-router-af)#redistribute static
```

From RouterB, view the routing table for VPNA with the command **show ip route vrf VPNA.** The following is the output from the command. Notice we have a route to network 10.1.2.0.

```
RouterB#show ip route vrf VPNA
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 2 subnets
B       10.1.2.0 [200/0] via 2.2.2.2, 00:11:26
S       10.1.1.0 is directly connected, Serial0/0
C    193.1.1.0/24 is directly connected, Serial0/0
```

Lab #119: Building MPLS VPNs Using OSPF

Equipment Needed

The following equipment is needed to perform this lab exercise:

- Two Cisco routers, each having one Ethernet and one serial port
- Two Cisco routers, each having two serial interfaces
- Cisco IOS capable of running MPLS
- A PC running a terminal emulation program
- Three Cisco DTE/DCE crossover cables
- A Cisco rolled cable

Configuration Overview

This lab will demonstrate a basic MPLS VPN configuration using OSPF from PE to CE. All of the routers will be configured for MPLS and use Tag Distribution Protocol (TDP) to distribute label bindings between routers. RouterB and RouterC will be configured as provider edge routers, and RouterA and RouterD will be configured as customer edge routers in VPNA. MPLS/VPN will be used to create a VPN between RouterA and RouterD across an MPLS core.

All of the routers are connected serially via crossover cables. RouterB will act as the DCE supplying clock to RouterA and RouterC, and RouterC will supply clock to RouterD. OSPF will be run between RouterB and RouterC. MP−iBGP will be run between RouterB and RouterD to advertise customer VPN routes between the two PE routers. RouterB and RouterC learn the routes from VPNA through OSPF. TDP will be run on each router to distribute the label−binding information.

RouterA and RouterD will have a default routing point to RouterB and RouterC, respectively. The IP addresses are assigned as per Figure 27−9.

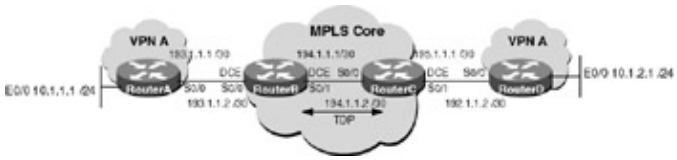


Figure 27−9: Building MPLS VPNs using OSPF from PE to CE

Router Configurations

The configurations for the four routers in this example are as follows. Key MPLS configurations are bolded. Cisco express forwarding (CEF) must be enabled in all routers running MPLS. Routers that are receiving unlabeled IP packets that will be propagated as labeled packets must have the ingress interface configured for CEF. MPLS/VPN commands will be added and explained in the monitoring and trouble shooting section.

RouterA

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password−encryption
!
hostname RouterA

ip subnet−zero
no ip finger
!
!
interface Ethernet0/0
```

```
 ip address 10.1.1.1 255.255.255.0
 no keepalive
!
interface Serial0/0
 ip address 193.1.1.1 255.255.255.252
 no ip mroute-cache
 no fair-queue
!

!
ip classless
no ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterB

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterB
!
ip subnet-zero
no ip finger
!
ip cef ← CEF must be enabled in all routers running MPLS
!
interface Serial0/0
 ip address 193.1.1.2 255.255.255.252
 tag-switching ip  ← CEF must be enabled in all ingress interfaces receiving
                     unlabeled packets
 clockrate 1000000
!
interface Serial0/1
 ip address 194.1.1.1 255.255.255.252
tag-switching ip  ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets
 clockrate 1000000
!
router ospf 64
 log-adjacency-changes
 network 194.1.1.0 0.0.0.255 area 0
!
ip classless
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterC

```
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterC
!
!
ip subnet-zero
no ip finger
!
ip cef ← CEF must be enabled in all routers running MPLS
!
!
interface Serial0/0
 ip address 194.1.1.2 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets

!
interface Serial0/1
 ip address 195.1.1.1 255.255.255.252
 tag-switching ip ← CEF must be enabled in all ingress interfaces receiving
                    unlabeled packets

 clockrate 1000000
!
router ospf 64
 log-adjacency-changes
 network 194.1.1.0 0.0.0.255 area 0
!
ip classless
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

RouterD

```
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname RouterD
!
ip subnet-zero
no ip finger
!
!
interface Ethernet0/0
 ip address 10.1.2.1 255.255.255.0
 no keepalive
!
interface Serial0/0
 ip address 195.1.1.2 255.255.255.252
!
ip classless
```

```
ip http server
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Monitoring and Testing the Configuration

PE routers (RouterB and RouterC) use IBGP to distribute VPN routes to one another. When a PE router advertises VPN routes to another PE, it does so using itself as the BGP next−hop address. This address should be the 32−bit loopback address on the router. This 32−bit address also needs to be advertised into the IGP routing tables of the backbone. This enables MPLS to assign a label corresponding to the 32−bit loopback address of each PE router. The following commands configure a 32−bit loopback address on RouterB and RouterC and advertise this address via OSPF.

```
RouterB(config)#interface loopback 0
RouterB(config−if)#ip address 1.1.1.1 255.255.255.255
RouterB(config−if)#exit
RouterB(config)#router ospf 64
RouterB(config−router)#network 1.1.1.1 0.0.0.0 area 0

RouterC(config)#interface loopback 0
RouterC(config−if)#ip address 2.2.2.2 255.255.255.255
RouterC(config−if)#exit
RouterC(config)#router ospf 64
RouterC(config−router)#network 2.2.2.2 0.0.0.0 area 0
```

As discussed earlier in the chapter, MPLS/VPN uses MP−iBGP to distribute VPN routes from PE to PE. The next step is to configure MP−iBGP between the two PE routers — in our case, RouterC and RouterB. By default, when a BGP session on a Cisco router is configured, it is activated to carry IPV4 addresses. The command **no bgp default ipv4−unicast** disables this behavior. The following commands enable the BGP process and disable the default ipv4 Unicast behavior on RouterC and RouterB.

```
RouterB(config)#router bgp 1
RouterB(config−router)#no bgp default ipv4−unicast

RouterC(config)#router bgp 1
RouterC(config−router)#no bgp default ipv4−unicast
```

The BGP neighbors are now defined under the global BGP process. The neighbor address is the 32−bit loopback address of the remote router. The update source (the address that BGP advertises as the next hop) should be the loopback address of the local router. The last step is to activate the neighbor. The following commands configure the MP−iBGP session between RouterB and RouterC.

```
RouterB(config)#router bgp 1
RouterB(config−router)#neighbor 2.2.2.2 remote−as 1
RouterB(config−router)#neighbor 2.2.2.2 update−source loopback 0
RouterB(config−router)#neighbor 2.2.2.2 activate

RouterC(config)#router bgp 1
RouterC(config−router)#neighbor 1.1.1.1 remote−as 1
RouterC(config−router)#neighbor 1.1.1.1 update−source loopback 0
RouterC(config−router)#neighbor 1.1.1.1 activate
```

The BGP session now needs to be activated to carry VPN−IPv4 prefixes. This is done through the use of an address family. The VPN−IPv4 address family is configured under the BGP process and then the neighbor is activated. The following commands configure the BGP neighbor session on RouterB and RouterC to carry VPN−IPv4 prefixes.

```
RouterB(config)#router bgp 1
RouterB(config-router)#address-family vpnv4
RouterB(config-router-af)#neighbor 2.2.2.2 activate

RouterC(config)#router bgp 1
RouterC(config-router)#address-family vpnv4
RouterC(config-router-af)#neighbor 1.1.1.1 activate
```

Verify that the BGP session is established and configured to carry VPN−IPv4 prefixes on RouterC. To do this, display the BGP neighbors with the command **show ip bgp neighbors**. The following is the truncated output from the command on RouterC. Notice the BGP state is established and the neighbor supports VPN−IPv4.

```
RouterC#show ip bgp neighbors
BGP neighbor is 1.1.1.1,  remote AS 1, internal link
  BGP version 4, remote router ID 1.1.1.1
  BGP state = Established, up for 00:03:03
  Last read 00:00:03, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
    Address family VPNv4 Unicast: advertised and received
  Received 22 messages, 0 notifications, 0 in queue
  Sent 22 messages, 0 notifications, 0 in queue
  Route refresh request: received 0, sent 0
  Default minimum time between advertisement runs is 5 seconds
```

Next, a VRF must be configured on RouterB and RouterC for VPNA. Under the VRF, a route distinguisher (RD) must be configured.

Routes from that VRF will be tagged with the RD. The purpose of the RD is to create distinct routes to a common IPv4 address prefix, allowing the same IP address to be used across multiple VPNs. The PE can be configured to associate all routes leading to the same CE with the same RD, or it may be configured to associate different routes with different RDs, even if they lead to the same CE. In this lab, all routes in VPNA will have the RD of 1:100.

The following commands configure the vrf VPNA using RD 1:100 on the two routers.

```
RouterB(config)#ip vrf VPNA
RouterB(config-vrf)#rd 1:100

RouterC(config)#ip vrf VPNA
RouterC(config-vrf)#rd 1:100
```

Configure the import and export policies for each VRF. These policies are used to advertise routes out of the VRF (export) and populate routes into the VRF (import). The following commands configure VPNA on RouterA and RouterB to export all routes with the extended community route target of 1:100 and to import all routes into the VRF that have a route target of 1:100.

```
RouterB(config)#ip vrf VPNA
RouterB(config-vrf)#route-target both 1:100

RouterC(config)#ip vrf VPNA
RouterC(config-vrf)#route-target both 1:100
```

After the VRF is configured on the PE, you must tell the router which interfaces belong in that particular VRF. This is done under the interface with the command **ip vrf forwarding.** The following commands associate interface S0/1 on RouterB connecting to RouterA and interface S0/1 on RouterC connecting to RouterD with vrf VPNA.

```
RouterB(config)#interface s0/1
RouterB(config-if)#ip vrf forwarding VPNA
```

```
RouterC(config)#interface s0/1
RouterC(config-if)#ip vrf forwarding VPNA
```

After you configure the interface, you will receive the following message indicating the IP address has been removed. The IP address will need to be re-added.

```
% Interface Serial0/1 IP address 195.1.1.1 removed due to enabling VRF VPNA
```

Verify the VPN configuration on RouterC with the command **show ip vrf detail VPNA**. The following is the output from the command. Notice that the RD is set to 1:100, interface S0/1 is associated with the VRF, and the import and export policies are configured.

```
RouterC#show ip vrf detail VPNA
VRF VPNA; default RD 1:100
  Interfaces:
    Serial0/1
  Connected addresses are not in global routing table
  Export VPN route-target communities
    RT:1:100
  Import VPN route-target communities
    RT:1:100
  No import route-map
  No export route-map
```

The final step is to get the customer prefixes (routes from RouterA and RouterB) into the VRF. This can be accomplished by running a dynamic routing protocol such as RIP or OSPF between the PE and CE (RouterA and RouterB) or through static routes. This lab will use OSPF. The following commands configure OSPF on RouterA and RouterD.

```
RouterA(config)#router ospf 100
RouterA(config-router)#network 10.1.1.0 0.0.0.255 area 0
RouterA(config-router)#network 193.1.1.0 0.0.0.255 area 0


RouterD(config)#router ospf 100
RouterD(config-router)#network 10.1.2.0 0.0.0.255 area 0
RouterD(config-router)#network 195.1.1.0 0.0.0.255 area 0
```

An OSPF process is needed for vrf VPNA on RouterC and RouterB. Once configured, this process needs to be redistributed into MP-iBGP under the address family. The following commands configure OSPF on RouterB and RouterC. Notice that the router OSPF command has been extended to support VPNs.

```
RouterB(config)#router ospf 100 vrf VPNA
RouterB(config-router)#network 193.1.1.2 0.0.0.0 area 0

RouterC(config)#router ospf 100 vrf VPNA
RouterC(config-router)#network 195.1.1.1 0.0.0.0 area 0
```

Verify that the OSPF process is configured on RouterB with the command **show ip ospf**. The following is the output from the command. Notice that OSPF process 100, which is configured for VPNA, is connected to the MPLS VPN backbone.

```
RouterB#show ip ospf
 Routing Process "ospf 100" with ID 193.1.1.2 and Domain ID 0.0.0.100
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Connected to MPLS VPN Superbackbone
 It is an area border router
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 Number of external LSA 0. Checksum Sum 0x0
 Number of opaque AS LSA 0. Checksum Sum 0x0
 Number of DCbitless external and opaque AS LSA 0
```

```
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 External flood list length 0
    Area BACKBONE(0)
        Number of interfaces in this area is 1
        Area has no authentication
        SPF algorithm executed 6 times
        Area ranges are
        Number of LSA 2. Checksum Sum 0x10BA3
        Number of opaque link LSA 0. Checksum Sum 0x0
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0

 Routing Process "ospf 64" with ID 1.1.1.1 and Domain ID 0.0.0.64
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 Number of external LSA 0. Checksum Sum 0x0
 Number of opaque AS LSA 0. Checksum Sum 0x0
Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 External flood list length 0
    Area BACKBONE(0)
        Number of interfaces in this area is 2
        Area has no authentication
        SPF algorithm executed 2 times
        Area ranges are
        Number of LSA 2. Checksum Sum 0xBB67
        Number of opaque link LSA 0. Checksum Sum 0x0
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0
```

Verify that vrf VPNA on RouterB has learned the networks from RouterA, with the command **show ip route vrf VPNA**. The following is the output from the command. Note that networks 10.1.1.0 and 193.1.1.0 are in VPNA's routing table.

```
RouterB#show ip route vrf VPNA
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, ia − IS−IS inter area
       * − candidate default, U − per−user static route, o − ODR
       P − periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
O       10.1.1.0 [110/58] via 193.1.1.1, 00:13:35, Serial0/0
     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
O       193.1.1.0/30 [110/96] via 193.1.1.1, 00:13:35, Serial0/0
C       193.1.1.0/24 is directly connected, Serial0/0
```

Once configured, OSPF process 100 needs to be redistributed into MP−iBGP under the address family. The following commands redistribute OSPF process 100 into MP−IBGP on RouterB and RouterC.

```
RouterB(config)#router bgp 1
RouterB(config−router)#address−family ipv4 vrf VPNA
RouterB(config−router−af)#redistribute ospf  100
```

```
RouterC(config)#router bgp 1
RouterC(config-router)#address-family ipv4 vrf VPNA
RouterC(config-router-af)#redistribute ospf 100
```

From RouterB, view the routing table for VPNA with the command **show ip route vrf VPNA**. The following is the output from the command. Notice we now have a route to networks 10.1.2.0 and 195.1.1.0.

```
RouterB#show ip route vrf VPNA
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 2 subnets
B       10.1.2.0 [200/58] via 2.2.2.2, 00:00:42
O       10.1.1.0 [110/58] via 193.1.1.1, 00:23:05, Serial0/0
     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
O       193.1.1.0/30 [110/96] via 193.1.1.1, 00:23:05, Serial0/0
C       193.1.1.0/24 is directly connected, Serial0/0
     195.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
B       195.1.1.0/24 [200/0] via 2.2.2.2, 00:00:42
B       195.1.1.0/30 [200/96] via 2.2.2.2, 00:00:42
```

Display the routing table on RouterA, with the command **show ip route**. The following is the output. Notice that RouterA has not learned about networks 10.1.2.0 or 195.1.1.0.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Ethernet0/0
     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
O       193.1.1.0/24 [110/96] via 193.1.1.2, 00:02:45, Serial0/0
C       193.1.1.0/30 is directly connected, Serial0/0
```

The reason is that the routes that are in vrf VPNA that have been learned by MP-iBGP on RouterB need to be redistributed into OSPF process 100. The following commands redistribute the routes in vrf VPNA on RouterB and RouterC learned from MP-iBGP into OSPF process 100.

```
RouterB(config)#router ospf 100 vrf VPNA
RouterB(config-router)#redistribute bgp 1 subnets metric 20

Routerc(config)#router ospf 100 vrf VPNA
Routerc(config-router)#redistribute bgp 1 subnets metric 20
```

Display the routing table on RouterA with the command **show ip route**. The following is the output. Notice that RouterA now has learned about network 10.1.2.0 or 195.1.1.0.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 2 subnets
O IA    10.1.2.0 [110/68] via 193.1.1.2, 00:04:29, Serial0/0
C       10.1.1.0 is directly connected, Ethernet0/0
     193.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
O       193.1.1.0/24 [110/96] via 193.1.1.2, 00:04:29, Serial0/0
C       193.1.1.0/30 is directly connected, Serial0/0
     195.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
O IA    195.1.1.0/30 [110/68] via 193.1.1.2, 00:04:29, Serial0/0
O IA    195.1.1.0/24 [110/68] via 193.1.1.2, 00:04:30, Serial0/0
```

Verify that RouterA can reach network 10.1.2.1 on RouterD using the ping command on RouterA.

```
RouterA#ping 10.1.2.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.1, timeout is 2 seconds:
!!!!!
```

Troubleshooting MPLS

The Cisco IOS provides many tools for troubleshooting MPLS. The following is a list of key commands along with sample output from each.

{show tag-switching interfaces} This exec command displays information about the requested interface or about all interfaces on which tag switching is enabled. The following example shows a sample output from the command.

```
RouterA#show tag-switching interfaces
Interface              IP    Tunnel   Operational
Serial0/0              Yes   No       Yes
```

{show tag-switching tdp neighbor} This exec command displays the status of Tag Distribution Protocol (TDP) sessions. The following example shows a sample output from the command.

```
RouterA#show tag-switching tdp neighbor
Peer TDP Ident: 194.1.1.1:0; Local TDP Ident 193.1.1.1:0
        TCP connection: 194.1.1.1.11000 - 193.1.1.1.711
        State: Oper; PIEs sent/rcvd: 230/232; ; Downstream
        Up time: 03:18:39
        TDP discovery sources:
          Serial0/0
        Addresses bound to peer TDP Ident:
          193.1.1.2       194.1.1.1
```

{show tag-switching tdp bindings} This exec command displays the contents to the tag information base (TIB). The following example shows a sample output from the command.

```
RouterA#show tag-switching tdp bindings
  tib entry: 192.1.1.0/24, rev 3
        local binding:  tag: imp-null
        remote binding: tsr: 194.1.1.1:0, tag: 26
  tib entry: 193.1.1.0/30, rev 4
        local binding:  tag: imp-null
        remote binding: tsr: 194.1.1.1:0, tag: imp-null
```

{show tag−switching forwarding−table} This exec command displays the contents of the tag−forwarding information base (TFIB). The following example shows a sample output from the command.

```
RouterD#show tag−switching forwarding−table
Local  Outgoing    Prefix           Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id     switched   interface
26     26          192.1.1.0/24     0          Se0/0      point2point
27     27          193.1.1.0/30     0          Se0/0      point2point
28     Pop tag     194.1.1.0/30     0          Se0/0      point2point
```

{show ip bgp neighbors} This exec command displays information about the TCP and BGP connections to neighbors. This command can be used with the argument received routes or advertised−routes, which displays all updates that are sent to or received from a particular neighbor. The following example shows a sample output from the command.

```
RouterC#show ip bgp neighbors
BGP neighbor is 1.1.1.1,  remote AS 1, internal link
  BGP version 4, remote router ID 1.1.1.1
  BGP state = Established, up for 00:03:03
  Last read 00:00:03, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
    Address family VPNv4 Unicast: advertised and received
  Received 22 messages, 0 notifications, 0 in queue
  Sent 22 messages, 0 notifications, 0 in queue
  Route refresh request: received 0, sent 0
  Default minimum time between advertisement runs is 5 seconds
```

{show ip vrf detail} This exec command displays the set of defined VRFs and interfaces on the router. The following example shows a sample output from the command.

```
  RD 1:100
  Interfaces:
    Serial0/1
  Connected addresses are not in global routing table
  Export VPN route−target communities
    RT:1:100
  Import VPN route−target communities
    RT:1:100
```

{show ip route vrf} This exec command is used to display the IP routing table associated with a VRF. The following example shows a sample output from the command.

```
RouterB#show ip route vrf VPNA
Codes: C − connected, S − static, I − IGRP, R − RIP, M − mobile, B − BGP
       D − EIGRP, EX − EIGRP external, O − OSPF, IA − OSPF inter area
       N1 − OSPF NSSA external type 1, N2 − OSPF NSSA external type 2
       E1 − OSPF external type 1, E2 − OSPF external type 2, E − EGP
       i − IS−IS, L1 − IS−IS level−1, L2 − IS−IS level−2, ia − IS−IS inter area
       * − candidate default, U − per−user static route, o − ODR
       P − periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 2 subnets
B       10.1.2.0 [200/0] via 2.2.2.2, 00:11:26
S       10.1.1.0 is directly connected, Serial0/0
C    193.1.1.0/24 is directly connected, Serial0/0
```

Conclusion

MPLS is an emerging technology that aims to address many of the issues associated with packet forwarding in today's IP networks. The ability to stack multiple labels on a packet has given rise to new applications such as traffic engineering, fast reroute, and VPNs.