SYNGRESS®

1 YEAR UPGRADE
BUYER PROTECTION PLAN

SYNGRESS.COM 1 YEAR UPGRADE BUYER PROTECTION PLAN

Configuring IPv6 for Cisco® IOS

Deploy IPv6 on the Cisco IOS

- Everything You Need to Monitor and Troubleshoot IPv6 Networks
- Your Complete Introduction to IPv6 Architecture
- Special Coverage of IPv4 / IPv6 Internetworking

Sam Brown
Brian Browne
Neal Chen
Paul J. Fong
Robbie Harrell
Eric Knipp
Bart Saylors
Rob Webber
Edgar Parenti, Jr.
Technical Editor

callisma

SYNGRESS®

1 YEAR UPGRADE
BUYER PROTECTION PLAN

Configuring IPv6 for Cisco IOS

Sam Brown
Brian Browne
Neal Chen
Paul J. Fong
Robbie Harrell
Eric Knipp
Bart Saylors
Rob Webber
Edgar Parenti, Jr. Technical Editor

KEY	SERIAL NUMBER
001	44BVHTR46T
002	AKTRT4YHE4
003	KUH4T945T5
004	87U86T6NVH
005	NFGTE4RNAS
006	SGD34B39F4
007	Q2F9R565MR
008	MSVX63N54N
009	GT6YH2BDFC
010	83N5M4B3ES

PUBLISHED BY
Syngress Publishing, Inc.
800 Hingham Street
Rockland, MA 02370

Configuring IPv6 for Cisco IOS

Printed in the United States of America

1 2 3 4 5 6 7 8 9 0

ISBN: 1-928994-84-9

Technical Editor: Edgar Parenti, Jr.
Technical Reviewer: Tony Bautts
Acquisitions Editor: Catherine B. Nolan
Developmental Editor: Kate Glennon
Cover Designer: Michael Kavish
Page Layout and Art by: Shannon Tozier
Copy Editors: Alexandra Kent and Michelle Melani
Indexer: J. Edmund Rush

Distributed by Publishers Group West in the United States and Jaguar Book Group in Canada.

Acknowledgments

We would like to acknowledge the following people for their kindness and support in making this book possible.

Ralph Troupe, Rhonda St. John, Emlyn Rhodes, and the team at Callisma for their invaluable insight into the challenges of designing, deploying and supporting world-class enterprise networks.

Karen Cross, Lance Tilford, Meaghan Cunningham, Kim Wylie, Harry Kirchner, Kevin Votel, Kent Anderson, Frida Yara, Jon Mayes, John Mesjak, Peg O'Donnell, Sandra Patterson, Betty Redmond, Roy Remer, Ron Shapiro, Patricia Kelly, Andrea Tetrick, Jennifer Pascal, Doug Reil, David Dahl, Janis Carpenter, and Susan Fryer of Publishers Group West for sharing their incredible marketing experience and expertise.

Jacquie Shanahan, AnnHelen Lindeholm, David Burton, Febea Marinetti, and Rosie Moss of Elsevier Science for making certain that our vision remains worldwide in scope.

Annabel Dent and Paul Barry of Elsevier Science/Harcourt Australia for all their help.

David Buckland, Wendi Wong, Marie Chieng, Lucy Chong, Leslie Lim, Audrey Gan, and Joseph Chan of Transquest Publishers for the enthusiasm with which they receive our books.

Kwon Sung June at Acorn Publishing for his support.

Ethan Atkin at Cranbury International for his help in expanding the Syngress program.

Jackie Gross, Gayle Voycey, Alexia Penny, Anik Robitaille, Craig Siddall, Darlene Morrow, Iolanda Miller, Jane Mackay, and Marie Skelly at Jackie Gross & Associates for all their help and enthusiasm representing our product in Canada.

Lois Fraser, Connie McMenemy, Shannon Russell, and the rest of the great folks at Jaguar Book Group for their help with distribution of Syngress books in Canada.

Contributors

Sam Brown (CCNP, CCDP, MCSE, MCP+I, CNE, Citrix CCA) is a Consultant with Callisma where he provides technical consulting to a variety of Callisma clients. His specialties include Network Management Systems (NMS) planning and implementation, Cisco routers and LAN switches, Microsoft NT and Novell design and implementation, network planning, network architecture and design, and network troubleshooting and optimization. Sam's background includes positions as a Network Analyst for Opryland USA in Nashville, TN, and and as a Senior Network Engineer at Frost Bank in Fiesta, TX. Sam makes his home in Denver, CO.

Brian Browne (CISSP) is a Senior Consultant with Callisma providing senior-level strategic and technical security consulting to Callisma clients. He has 12 years of experience in the field of information systems security and is skilled in all phases of the security lifecycle. A former independent consultant, Brian has provided security consulting expertise for multiple Fortune 500 clients, and has been published in *Business Communications Review* magazine. His security experience includes network security, firewall architectures, Virtual Private Networks (VPNs), Intrusion Detection Systems (IDSs), UNIX security, Windows NT security, and Public Key Infrastructure (PKI). Brian resides in Willow Grove, PA with his wife Lisa and daughter Marisa.

Neal Chen (CCNP, CCNA) is a Consultant with Callisma providing strategic and technical consulting to all Callisma clients in the Northeast region of the United States. His specialties include Cisco routers and LAN switches, Cisco and Nortel Dense Wavelength Division Multiplexing (DWDM) and SONET equipment, strategic network planning, network architecture and design, and network troubleshooting and optimization. Neal's background also includes a position as a Network Engineer at Raytheon Corporation.

Paul J. Fong (CCDP, CCNP) is a Senior Consultant for Callisma where he provides strategic and technical consulting to service provider clients. Paul's background includes positions as an Advisory Systems Analyst at IBM where he developed a network monitoring system for NASA Space Shuttle telemetry, and as a senior member of the technical staff at MCI Worldcom where he played a key role in the development of the SRDF-over-IP protocol. Paul holds a Bachelor's and a Master's degree from Stanford University. While pursuing his studies, Paul developed speech recognition software at the Xerox Palo Alto Research Center as a research associate and published his work in *IEEE Transactions on Systems, Man and Cybernetics*. Paul is a member of the Colorado Springs Cisco Users Group and lives in Monument, CO, with his wife Sharon and their daughter Shana.

Robbie Harrell (CCIE #3873) is a Principle Architect with Callisma in Atlanta, GA. He has over 10 years of experience and provides strategic, business, and technical consulting services to clients. Robbie specializes in the design and implementation of complex solutions necessary to meet business objectives in the enterprise and service provider market space. His expertise is in routing and switching, and strategic planning. Robbie's background includes positions as a Principle Consultant at International Network Services, Lucent, and Frontway.

Eric Knipp (CCNP, CCDP, CCNA, CCDA, MCSE, MCP+I) is a Consultant with Callisma. He is currently engaged in a broadband optimization project for a major U.S. backbone service provider. He specializes in IP telephony and convergence, Cisco routers, LAN switches, well as Microsoft NT, and network design and implementation. Eric has also passed both the CCIE Routing and Switching written exam as well as the CCIE Communications and Services Optical qualification exam and is currently preparing to take the CCIE lab later this year. Eric's background includes positions as a Project Manager for a major international law firm, and a Project Manager for NORTEL. Eric has contributed to the Syngress publications *Cisco AVVID and IP Telephony Design and Implementation* (ISBN: 1-928994-83-0), *Managing Cisco Network Security, Second Edition* (ISBN: 1-931836-56-6), and the forthcoming *Configuring Cisco Voice Over IP, Second Edition* (ISBN: 1-931836-64-7).

Bart Saylors (CCNP, CCDP) is a Senior Consultant with Callisma. His specialties include Cisco router and LAN switching design, implementation, and troubleshooting as well as providing the business processes and project management needed during the life cycle of these technologies. Bart has 19 years of networking experience and has held positions of Senior Network Support for the JCPenney corporate data center network, Senior Design Engineer at ACS and Data Engineering Support for GTE.

Rob Webber (CCIE #6922) is a Senior Network Consultant with Callisma in Wakefield, MA. He has over 14 years of experience in the data networking industry, and has spent the last four as a consultant. Rob specializes in the design and implementation of complex networks in the financial, medical, manufacturing, and service provider industries. His expertise includes routing, switching, security, and converged voice and data networking solutions from Cisco Systems and Nortel Networks.

Rob is a contributing author to *Cisco AVVID and IP Telephony Design & Implementation* (Syngress Publishing, ISBN: 1-928994-83-0). In addition to networking Rob enjoys Web development and Perl scripting. Rob holds a Bachelor's of Science degree from the University of New Hampshire.

Technical Reviewer

Tony Bautts is a Senior Security Consultant with Astech Consulting. He currently provides security advice and architecture for clients in the San Francisco Bay area. His specialties include Intrusion Detection Systems (IDSs), firewall design and integration, post-intrusion forensics, bastion hosting, and secure infrastructure design. Tony's security experience has led him to work with Fortune 500 companies in the United States as well as to perform two years of security consulting in Japan. Tony was a contributing author to *Hack Proofing Your Wireless Network* (Syngress Publishing, ISBN: 1-928994-59-8). He is also involved with the BerkeleyWireless.net project, which is working to build neighborhood wireless networks for residents of Berkeley, CA.

Technical Editor

Edgar Parenti, Jr. (CCNA, CCDA, CCNP, CCDP, CNE-3/4/5, MCNE, PSE, MCSE 2000, MCT) is currently a Consulting Engineer with UNICOM Technology Group, Inc. where he provides corporate, education, and government customers with a portfolio of cutting-edge networking solutions. Edgar has a strong background in network and directory design, network analysis and optimization, system performance tuning, Web application architecture and support, messaging and infrastructure engineering, operating system support, process engineering, and information security. His background also includes working at numerous corporations of all sizes providing senior-level IT consulting services utilizing a wide array of technologies and over six years of designing and managing Cisco internetworks.

Contents

The show version Command

The *show version* command enables administrators to discern the following system conditions and parameters:

- System Platform
- System IOS version
- System Boot Rom Version
- System Uptime
- Reason for last reboot
- System Image File
- Processor and Memory available
- Physical Interfaces
- Configuration Register

The Benefits of IPv6

The benefits of IPv6 include:

- Increased IP Address Size
- Increased Addressing Hierarchy Support
- Simplified Host Addressing
- Simplified Auto-configuration of Addresses
- Improved Scalability of Multicast Routing
- The Anycast Address
- Streamlined Header
- Improved Security
- Better Mobility
- Better Performance

IPv4 and IPv6 Address Spaces

The allocation and assignment policies are defined in RFC 2050. Specifically, the policy is as follows:

1. End users should request address space from their directly connected upstream provider.
2. If no addresses are available from the upstream provider, request addresses from the provider's provider.
3. If justifiable, request address space directly from ARIN, RIPE, or APNIC.

Configuring IPv6 Addressing

- When configuring your interfaces, remember that *ipv6* must be included in the syntax to distinguish between IPv4 and IPv6 addresses.
- There are three types of addresses that can be assigned on an interface: global, site-local and link-local.
- If you are using the *EUI-64* command when entering an IPv6 address, remember that the router uses its identifier for the last 64 bits of the IPv6 address, therefore if you want to find out the address, you have to go back and get the full IPv6 address.
- With the configuration of duplicate address detection, you can specify the number of solicitation messages that are sent out.

WARNING

BGP uses a ten-step process in calculating its metric. Redistributed routes do not go through this process, unless of course you are redistributing another BGP process; therefore, you should manually configure the redistributed protocols to appropriately fit into the BGP routing process.

IPv6 Deployment Strategies

To verify the platform support and list minimum memory requirements for the intended deployment, follow these steps:

1. Go to www.cisco.com/kobayashi/support/tac/software.shtml.
2. Select either **Software Advisor | Search by Release**, or **Software Upgrade Planner** for Cisco IOS selection.

Extension Headers Overview

When more than one IPv6 extension header is included in a packet, the header should adhere to the following order:

1. IPv6 Header
2. Hop-by-Hop Options Header
3. Destination Options Header (for options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header)
4. Routing Header
5. Fragment Header
6. Authentication Header
7. Encapsulating Security Payload Header
8. Destination Options Header (for options to be processed only by the final destination of the packet)

Foreword

It's time for the next generation of the Internet Protocol and IPv6 has arrived! This new protocol will allow for the continued expansion of the Internet and is one of the most significant developments in the history of the communications industry.

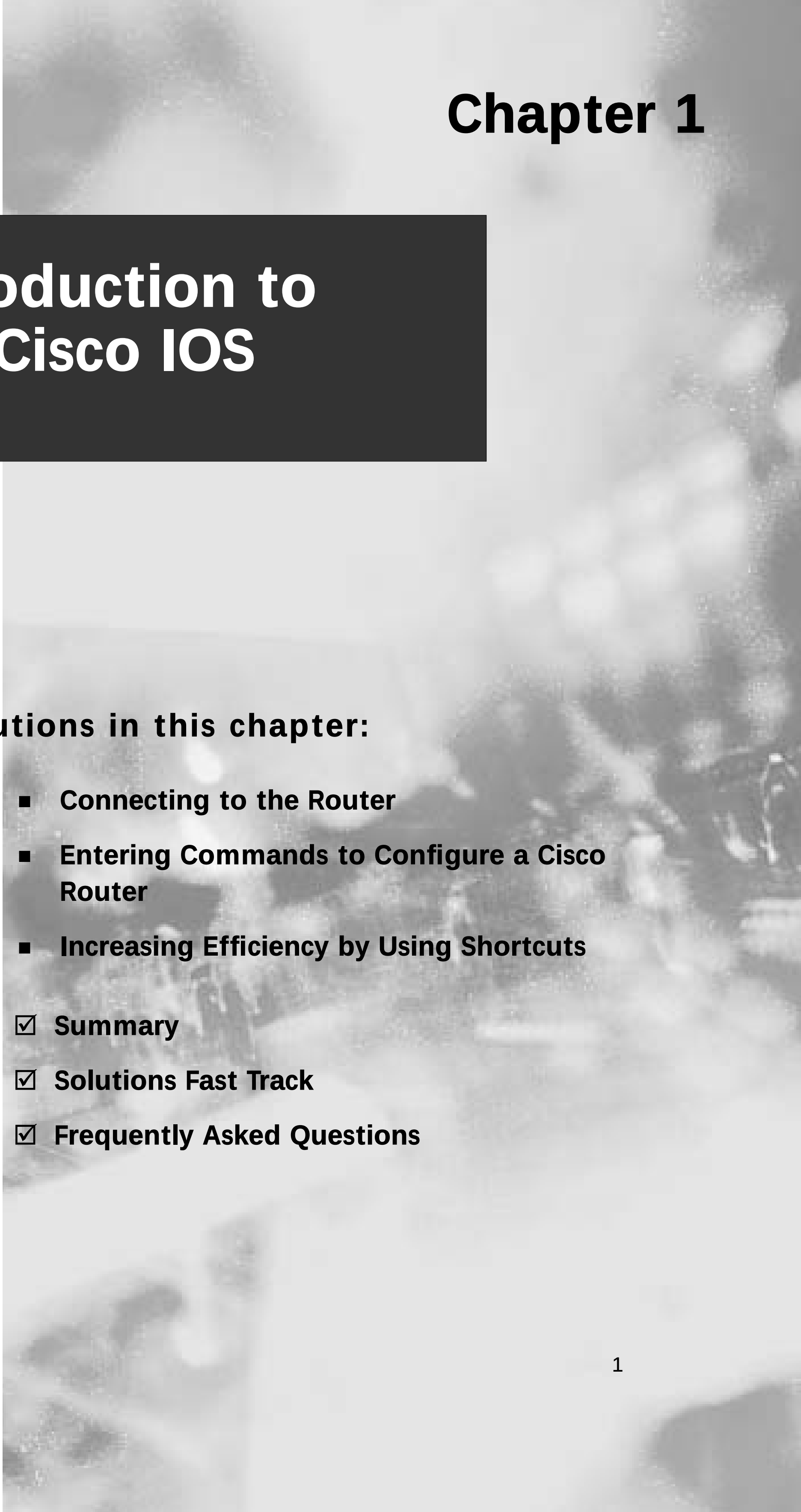
Today's Internet uses mostly IPv4, which is now nearly 20 years old. In spite of its age, IPv4 has been remarkably resilient, although it's limitations are now becoming problematic. Most importantly, there is a growing shortage of IPv4 addresses, which are needed by all new machines added to the Internet.

IPv6 offers multiple advantages over IPv4: larger address space, hierarchical addressing, security, and mobility. One of the most important benefits of IPv6 is hierarchical addressing, because supernetting fosters smaller routing table entries. IPv6 will eventually replace IPv4 in next generation carrier networks, although the probable catalyst for IPv6 might be the expected widespread adoption of 3G data wireless services. This adaptation of IPv6 to next-generation wireless networks is already under way.

Configuring IPv6 for Cisco IOS reviews the IPv6 architecture and features while focusing on the practical aspects of IPv6 implementation, which include addressing and routing. Also, because IPv4 networks will continue to exist for quite some time, requiring efficient and stable IPv4/IPv6 internetworking, the book explores several approaches to IPv6 transitioning. Also, it reviews the monitoring and troubleshooting required for long-term operational support. This book provides understanding of the technology and more important, it offers the real-world experience of writers who have worked with the IPv6 architecture.

As a professional services firm focused on network infrastructure solutions, Callisma provides planning, design and implementation of advanced networking protocols such as IPv6. We regularly assist customers with the considerations this book covers and will educate readers on some of the theory and practical information required to successfully deploy and support their mission critical networks. For more information, visit the Callisma Web site at www.callisma.com.

—*Ralph Troupe*
President and CEO, Callisma



Chapter 1

Introduction to the Cisco IOS

Solutions in this chapter:

- **Connecting to the Router**
- **Entering Commands to Configure a Cisco Router**
- **Increasing Efficiency by Using Shortcuts**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

Introduction

A common management and configuration framework is the goal of any good network equipment provider. Cisco Systems has achieved this goal in commendable fashion. The Cisco Internetwork Operating System (IOS) provides a breadth of features, which facilitates a uniform configuration and management platform across nearly its entire line of routing and switching products. One of Cisco's goals for its routing and switching products has been to unify the core services provided by its IOS to create a standard "look and feel" for its product lines. The Command Line Interface (CLI), which is used to enter configuration and management commands, has been standardized in recent years and has been emulated by other networking vendors due to its effective structure.

The IOS is not unlike any other operating system. Its purpose is to transform the otherwise lifeless collection of electronics within the particular piece of network equipment into a useable, intelligent device. Operating systems such as MS-DOS, UNIX, VMS, NT, and Windows 2000 perform a similar function for mid-range and personal computers. The function of the Cisco IOS is much different from that of a PC operating system but not any more difficult to learn. At first glance, however, configuring a Cisco networking product's IOS through the CLI seems quite intimidating. There are complex sets of commands that require intricate syntax to perform even the simplest tasks. The daunting nature of the large command set and complex syntax is offset by the consistency of command syntax and integrated help features. Gaining access to the router and entering commands become routine activities after you have performed them only a few times, due to the short learning curve necessary for mastering the CLI.

This chapter will focus entirely on the necessary procedures to connect to a Cisco router and enter commands using the CLI. It will cover in detail the steps necessary to connect to the router and will describe how to enter commands, how to work with command shortcuts, and how to access help features. Because this book is about configuring a routed protocol, IOS configuration tasks will be demonstrated on a Cisco router. If you have never configured a Cisco router, this chapter will give you all the information necessary to help you accomplish this often-intimidating task with the greatest of ease.

Connecting to the Router

As logic would dictate, to configure a router, you must first be able to connect to it. Out of the box, the IOS is loaded; however, it has no configuration at all. As such, it has no Layer 3 addressing on the interfaces or Simple Network Management Protocol (SNMP) strings loaded into the router. This makes it impossible to connect remotely. To configure the router, a physical connection must be made to the router through the use of the *router console port*.

Although router configurations differ considerably from platform to platform, all routers generally share two common ports: an *auxiliary port* and a *console port*. Auxiliary ports are typically attached to a modem or other device for remote management in case of network failure or some other catastrophic event. Although the auxiliary port is useful in many cases, we are going to focus our discussion on the console port.

Configuring & Implementing…

One IOS?

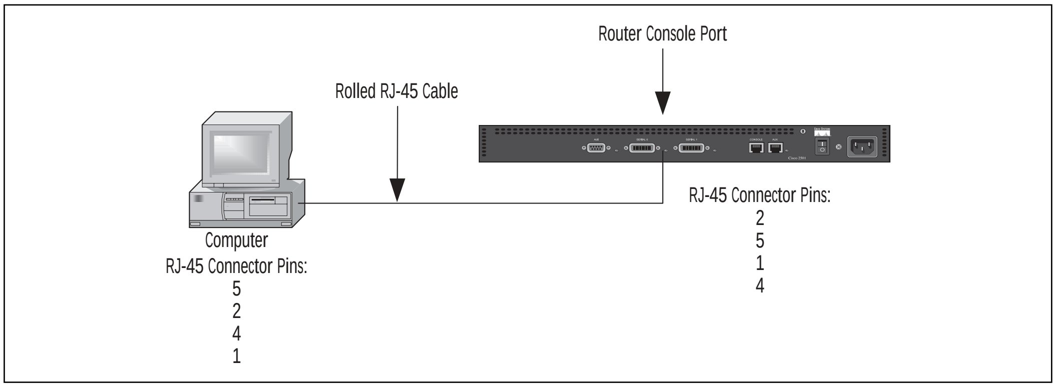
Not just one IOS exists for all Cisco devices; in fact, hundreds of different IOS releases and versions are available. IOS releases are written for each router platform and even more specific releases offer certain features and solutions. IOS releases are custom tailored for almost any environment, such as Service Providers, Enterprises, Systems Network Architecture (SNA) integration, and Internetwork Packet Exchange (IPX) support.

What you must decide as a network administrator is which IOS is right for your network. To assist in this, Cisco offers an IOS software selection tool. This tool enables you to select the individual router/switch platform and displays all the specific IOS releases available and the features that each of them support. The IOS selection tool is available at the Cisco technical assistance (TAC) Web site at www.cisco.com/tac. One point to keep in mind is that in order to access the TAC Web site you must have a Cisco Connection Online (CCO) login. To obtain a CCO login, you must have either a service contract with Cisco Systems for a specific hardware platform or you must be a Cisco partner.

Console Port Connections

The console port, as its name implies, provides a console connection to the router for configuration purposes. To access this port you need to use a rolled-RJ45 cable with a 9– or 25-pin adapter connected to a PC that is running terminal emulation software, connected to the console port of the router. On older routers, the console port is a 25-pin serial port; a 25-pin to RJ-45 connector with a rolled RJ-45 cable is incorporated to connect to the console ports of these routers. A rolled RJ-45 cable is simply a cable in which each pair have been swapped at opposite ends to make the send receive and the receive send. Wires connecting to pins 5, 2, 4, and 1 on one RJ-45 connector interface with pins 2, 5, 1, and 4 on the other connector in order to accomplish this. Figure 1.1 demonstrates the connection between a PC running terminal software and the router's console port. A new router usually comes with a console connection kit with a 9-pin and 25-pin connector as well as an RJ-45 rolled cable.

Figure 1.1 PC Connecting to a Router Console Port

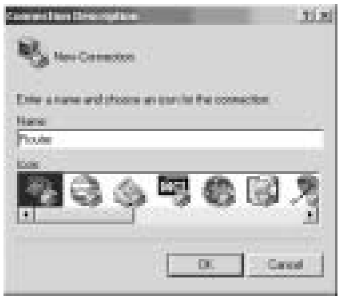


Now that you have an understanding of how the physical connection is made to the router, we can turn our attention to how the logical connection to the router is made. To make a software connection to the router, you must run a terminal-emulation software program on your computer. For purposes of this discussion we will use the HyperTerminal software program, which is included in the accessories software package of most Windows-based computers.

This configuration is as follows:

1. Go to **Start Menu | Programs | Accessories | Communications**, and launch HyperTerminal.
2. In the Connection Description screen, enter a name for your new connection and press **OK** (see Figure 1.2).
3. In the Connect To screen, choose a COM port from the list (2-4 ports are usually available). For the purposes of this example we will use **COM 2**, as shown in Figure 1.3.

Figure 1.2 HyperTerminal Name Settings



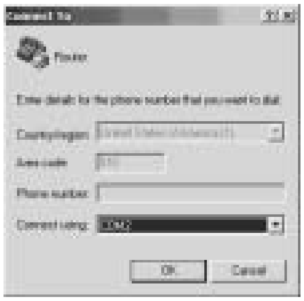
Connection Description

New Connection

Enter a name and choose an icon for the connection:

Name: Router

Figure 1.3 HyperTerminal Connection Settings



Connect To

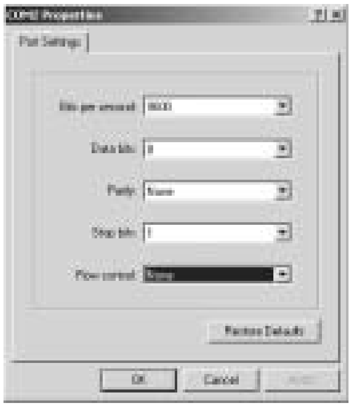
Router

Enter details for the phone number that you want to dial:

Country/region: United States (1)
Area code:
Phone number:
Connect using: COM2

4. Make the following changes to the connection: Choose **9600** bps and **None** for the option for flow control. Figure 1.4 demonstrates what the appropriate connection settings should look like.

Figure 1.4 Com Port Settings



COM2 Properties

Port Settings

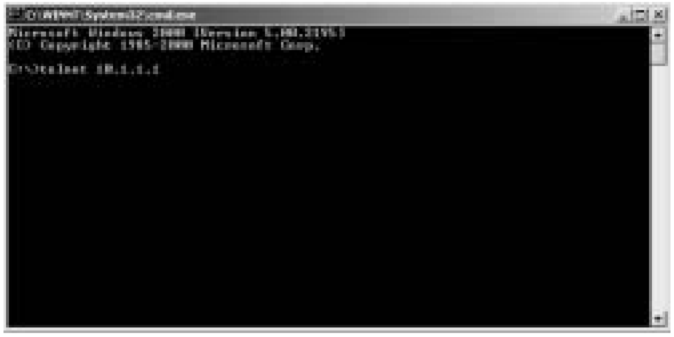
Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

Turn the router on and you should see the system post information come up. This information includes the boot information and pertinent information to the current config. After the router has finished posting, press **Return** twice and the router is ready to accept commands.

Telnet Connection

After you have configured addressing on interfaces and network paths are available to these addresses, you can access the router through the virtual terminal (VTY) interfaces of the router. VTY ports are a set of completely logical interfaces that answer to the TCP port 23 requests to the Internet Protocol (IP) addresses configured for the router. To access a router via Telnet connection, use terminal emulation software from a PC to issue the telnet command to access a router. Several products, such as the Reflection suite of products, are available to provide Telnet connectivity. In the following example, we will be using the Telnet function built into the Windows operating system to access our router. To do this on a PC, go to **Start | Run** and type **Cmd** if you are using Windows 2000 or XP (or **Command**, if you are using Windows 9*x*). This brings up a command prompt. At the command prompt, enter the **telnet <ip address>** command. An example of this command is shown in Figure 1.5.

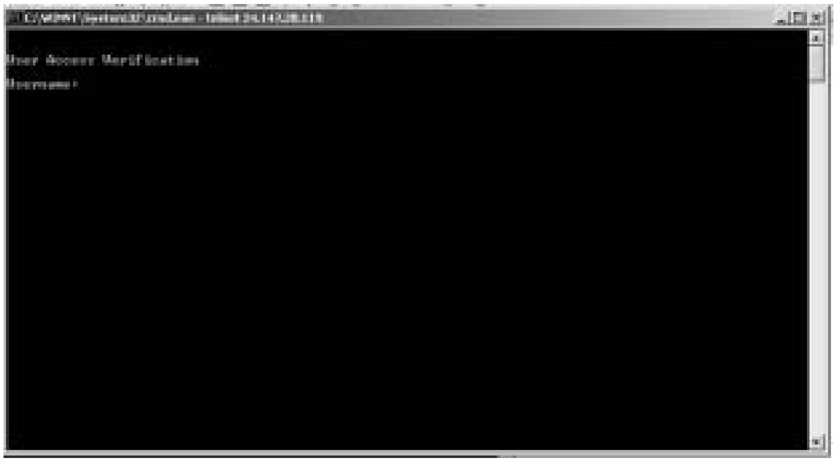
Figure 1.5 Router Telnet Connection



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>telnet 10.1.1.1
```

If you have already set up your router, you can see from our example in Figure 1.6 that we issued the telnet command for the IP address 10.1.1.1 and that this request was successful because we were met with a router prompt. This particular router is prompting us for a username and password because we have configured a username and password on the router. We will discuss this in more detail in the "Entering Commands" section later in this chapter.

Figure 1.6 Successful Connection to a Router via Telnet



```
User Access Verification

Username:
```

SNMP Configurations

The Simple Network Management Protocol is an industry open standard used by administrators and network management systems (NMS) to remotely manage systems. SNMP incorporates a system known as management information bases (MIBs) to accomplish this. MIBs are sets of device characteristics that let the NMS know how the device operates. When a certain criterion is met, an event known as a trap is set off. A trap lets the NMS know that a specific event has occurred and whether or not an action should be taken for the action.

Designing & Planning…

Exploring GUI Options

The goal of this chapter is to familiarize you with the Cisco IOS and its command structure. However, many individuals are very leery of command-line interfaces; indeed, in this age of Windows and Java consoles, command line is starting to become quite passé. As you can see, the command-line interface offered by the Cisco IOS differs from many of the stereotypes faced by these traditional interfaces in that it is a user-friendly and fairly intuitive interface; however, if you wish to have a graphical user interface (GUI), a few options are available to you.

Continued

Network Management Systems:

Through the use of a network management system, of which CiscoWorks 2000 is the flagship, many of the common management functions and statistical data gathering can be accomplished through a GUI or similar interface. This type of system would enable the routers and switches to be managed from a single point of administration. CiscoWorks and other NMSs do offer an alternative to the command-line interface for some but not all functions; initial configuration still has to be done through command lines.

Web Configuration:

In recent years, Cisco has begun writing versions of the IOS to support Web-based GUIs to their products. These are a highly powerful and easy-to-use interfaces that enable almost all configuration and monitoring tasks to be accomplished through use of a Web browser. To accomplish this, you simply type **HTTP://** followed by the router's IP address in a typical Java-enabled browser. Be advised, though, that not all Cisco's products support Web-based configuration; you should check the Cisco Web site to see if your specific router/switch supports Web configuration. Also, you should keep in mind that you still need to do initial configuration through a console connection to the router.

ConfigMaker:

ConfigMaker is a very powerful configuration tool available as a free download from the Cisco Web site. This tool offers a GUI interface and uses a series of user inputs and questions to build a configuration file that can be easily applied to the configuration of a new or working router. Although it is not an all-encompassing solution, and it cannot create a configuration file for every possible scenario, it is a great tool for general configurations and is part of the tool set used by network engineers and administrators from novice to expert levels. ConfigMaker can be downloaded from the Cisco Web site at www.cisco.com.

Cisco routers are fully compliant with the SNMP standard. Although initial configuration cannot take place through the use of SNMP, ongoing management and monitoring is more than possible through it. Cisco routers, through use of SNMP, can interoperate with almost all the major NMS platforms including CiscoWorks, HP OpenView, IBM Tivoli, Micromuse, and SMARTS. SNMP is enabled on a router through the configuration of community strings and privileges.

Entering Commands to Configure a Cisco Router

Now that you know how to connect to the router, let's turn our attention to what we do when we have accessed the router—configure it. Router configuration, as a process, is a relatively simple task, with a basic command structure and a built-in command menu; however, router configuration gets a bad rap for being difficult because of the logic that must be incorporated into many of the configurations and changes. In this section we will discuss the structure and process for entering commands into a Cisco router.

NOTE

For all our examples in this chapter we will be using a 2501 router running 12.2(8)T IOS.

There are two basic levels of router configuration, *user mode* and *privileged mode*. User mode is for basic administration and verification—no configuration can be done on this level. You are limited to basic functions such as *ping*, low-level *show* commands, and trace routes at this level. Privileged mode is where the real fun starts. Privileged mode is the configuration level that enables an administrator to make actual configuration changes and high-level *show* commands. The majority of this section will center on entering commands at a privileged level. If ever in doubt as to what you can do at a particular prompt, you can enter a question mark (**?**) at the command prompt to see the commands available to you. When you enter the question mark after a particular command, it will show you all the different parameters available for that command. The following code example shows all the commands available in user mode, and afterwards all the particular parameters available for the *ping* command:

```
6Router-1>?
Exec commands:
  <1-99>            Session number to resume
  access-enable     Create a temporary Access-List entry
  access-profile    Apply user-profile to interface
  clear             Reset functions
  connect           Open a terminal connection
  disable           Turn off privileged commands
  disconnect       Disconnect an existing network connection
  enable           Turn on privileged commands
  exit             Exit from the EXEC
  help             Description of the interactive help system
  lock             Lock the terminal
  login            Log in as a particular user
  logout           Exit from the EXEC
  mrinfo           Request neighbor and version information from a
                       multicast router
  mstat            Show statistics after multiple multicast traceroutes
  mtrace           Trace reverse multicast path from destination to source
  name-connection  Name an existing network connection
  pad              Open an X.29 PAD connection
  ping             Send echo messages
  ppp              Start IETF Point-to-Point Protocol (PPP)
  resume           Resume an active network connection
  rlogin           Open an rlogin connection
  show             Show running system information
  slip             Start Serial-line IP (SLIP)
  systat           Display information about terminal lines
  telnet           Open a telnet connection
  terminal         Set terminal line parameters
  traceroute       Trace route to destination
  tunnel           Open a tunnel connection
  udptn            Open an udptn connection
  where            List active connections
  x28              Become an X.28 PAD
  x3               Set X.3 parameters on PAD

6Router-1>
6Router-1>ping ?
  WORD  Ping destination address or hostname
  ip    IP echo
  ipv6  IPv6 echo
  srb   srb echo
  tag   Tag encapsulated IP echo

6Router-1>ping
```

As we mentioned earlier, when you first enter into a router, you will enter into basic user mode. You can always tell which level you are in by looking at the router prompt at the far left. When you are in user mode, you will see the router name followed by a greater-than angle bracket (**>**). To enter into privileged mode, enter the **enable** command. This changes the angle bracket to a pound sign (**#**). In most cases, and as generally recommended by Cisco, an enable password governs the capability to enter into privileged mode. In such cases, you are prompted for the enable password after entering the *enable* command. In order to move back into user mode from privileged mode, enter the **disable** command. The following code example demonstrates this process:

```
6Router-1>enable

Password:

6Router-1#disable

6Router-1>
```

Using Configuration Commands

After you have entered into privileged mode on your router, there are two basic types of command that you can enter: *configuration commands* or *show commands*. In this section, we will discuss configuration commands. Configuration commands are commands that you enter into the router that change the way the router operates. As you can probably imagine, thousands of possible configuration commands exist for changing the router's operations. Many changes can be made at this level. Although this is by no means a comprehensive list, some of these changes include interface addressing, routing protocol configuration, password changes, and system-level configuration changes. All these configuration changes must be made through configuration mode. To enter into configuration mode, use the **configure** command at privilege level. When you enter the *configure* command, it will prompt you as to how you wish to configure the router, either by *terminal*, *memory*, or *network*. In this particular instance, select **terminal** to configure it from your terminal connection. Memory and network configuration changes are made if you wish to use the configuration stored either in the router's non-volatile RAM (NVRAM) or if you wish to configure from a network server. To exit the configuration level of the router, enter **Ctrl+Z** to exit completely or **exit** to exit the current layer (when at the top layer, the *exit* command also takes you out of configuration mode). The following code example demonstrates the *configure* command:

```
6Router-1#configure
Configuring from terminal, memory, or network [terminal]? terminal
Enter configuration commands, one per line.  End with CTRL/Z.
6Router-1(config)#^Z
6Router-1#
```

Notice in the preceding example that when entering into configuration mode the *(config)* prompt is present next to the router name. This is always a way to tell that you are in configuration mode. This will change as you move into the different layers of configuration, such as interface configuration. In configuration mode you can see all the possible configuration commands by again entering a question mark. For reasons of brevity, we will shorten this list considerably (more than one hundred possible commands exist for the following output):

```
6Router-1(config)#?
Configure commands:
  aaa                         Authentication, Authorization, and Accounting.
  boot                        Modify system boot parameters
  clock                       Configure time-of-day clock
  config-register             Define the configuration register
  default                     Set a command to its defaults
  do                          To run exec commands in config mode
  downward-compatible-config  Generate a configuration compatible with
                                   older software
  enable                      Modify enable password parameters
  hostname                    Set system's network name
  ip                          Global IP configuration subcommands
  logging                     Modify message-logging facilities
  multilink                   PPP multilink global configuration
  netbios                     NETBIOS access control filtering
  no                          Negate a command or set its defaults
  parser                      Configure parser
  regexp                      regexp commands
  rif                         Source-route RIF cache
  service                     Modify use of network-based services
  username                    Establish User Name Authentication
  virtual-profile
  access-list                 Add an access list entry
  alias                       Create command alias
  alps                        Configure Airline Protocol Support
  arp                         Set a static ARP entry

. . .

6Router-1(config)#
```

Once again you can see the specific parameters for a configuration command by entering it followed by a question mark. In this case, see what possible configuration parameters you can make with the **ipv6** command:

```
6Router-1(config)#ipv6 ?
  access-list       Configure access lists
  hop-limit         Configure hop count limit
  host              Configure static hostnames
  icmp              Configure ICMP parameters
  neighbor          Neighbor
  prefix-list       Build a prefix list
  route             Configure static routes
  router            Enable an IPV6 routing process
  unicast-routing   Enable unicast routing

6Router-1(config)#ipv6
```

It would be nearly impossible to discuss in any degree of detail all of these configuration commands. In fact, many of them do not apply in many situations. The next sections will discuss some of the more general configuration commands.

Using Passwords to Control Router Access

Passwords are used on a Cisco router just like they are used on any computer: to control access. Cisco routers use five different passwords to control access to the router: *console*, *auxillary*, *VTY*, *enable*, and *enable secret*.

Using the console *Password*

Use the *console* password to control physical access to the console port of a router. When utilizing a console password, a user will automatically be prompted for the console password to access the router even before entering user mode. This provides for security of the console port in case an unauthorized user tries to physically connect to the router console port. To assign or change the console password, you must first specify that interface when working in configuration mode. This is accomplished by entering the **line console 0** command in configuration mode. After entering into the line configuration mode, you will see that the prompt changed from just *(config)* to *(config-line)*. To change this password, you can simply enter the **password** command followed by the password that you wish to use:

```
6Router-1(config)#line console 0
6Router-1(config-line)#password cisco
6Router-1(config-line)#exit
6Router-1(config)
```

Using the auxiliary *(*aux*) Password*

As we discussed earlier, the *auxiliary* password is a secondary port for access to the router. Most commonly it is used to attach a modem for remote access to the router in the case of network failure. It can also be used for dial backup in the case of primary network link failure. This port does present a security problem if not secured and, as such, should also be protected with a password. The configuration for this password is nearly identical to that of the configuration for a console password, except here you specify the **aux 0** port:

```
6Router-1(config)#line aux 0
6Router-1(config-line)#password cisco
6Router-1(config-line)#exit
6Router-1(config)
```

Using the VTY Password

As we discussed before, VTY ports are virtual interfaces into the router for Telnet access. In general, routers have five virtual terminal interfaces (0-4). Since there are multiple interfaces, they can be grouped when making configuration changes. Because it is impossible to know which VTY port will answer the Telnet session and because VTY ports allow for remote access to the router, all need to be secured. It is highly recommended that appropriate password protection be implemented on these ports. Configuration for a password on these ports is also very similar to that of the *console* and *aux* passwords:

```
6Router-1(config)#line vty ?
  <0-4>  First Line number
```

```
6Router-1(config)#line vty 0?
<0-4>

6Router-1(config)#line vty 0 ?
  <1-4>  Last Line number
  <cr>

6Router-1(config)#line vty 0 4
6Router-1(config-line)#password cisco
6Router-1(config-line)#exit
```

Using the enable *and* enable secret *Passwords*

Use the *enable* and *enable secret* passwords to gain access to the privileged mode of the router—the configuration mode that you can make all your changes in. Out of the box, privileged mode can be entered by just using the *enable* command. Unsecured, this is a very large security risk. To change this, use the *enable* password command in configuration mode. The *enable secret* password is an extension of the *enable* password. The *enable secret* command overrides the *enable* password, making itself the *enable* password. Also, the *enable* password, unless manually encrypted, is a clear-text password, whereas the *enable secret* password is an encrypted password. The following code examples demonstrate configuration of both the **enable** and **enable secret** passwords:

```
6Router-1(config)# enable password cisco
6Router-1(config)# enable secret cisco
```

Performing Interface Configuration Tasks

Interface configuration is also one of the more fundamental tasks of router configuration. Interface configuration concerns itself with the actual physical, Ethernet, serial, Asynchronous Transfer Mode (ATM), and so on, as well as logical interfaces such as loopback addresses on the routers. As such it is also one of the most important types of configuration because it specifies how the router will connect to the rest of the network. Configuration tasks that take place on this level include logical addressing, line speed, duplex settings, framing, and so on. To enter into interface configuration, enter the **interface** command from configuration mode, followed by the interface type and number of the particular interface.

As usual, you can follow the command with a question mark to see the available options:

```
6Router-1(config)#interface ?
  Async               Async interface
  BVI                 Bridge-Group Virtual Interface
  CTunnel             CTunnel interface
  Dialer              Dialer interface
  Ethernet            IEEE 802.3
  Group-Async         Async Group interface
  Lex                 Lex interface
  Loopback            Loopback interface
  MFR                 Multilink Frame Relay bundle interface
  Multilink           Multilink-group interface
  Null                Null interface
  Serial              Serial
  Tunnel              Tunnel interface
  Vif                 PGM Multicast Host interface
  Virtual-Template    Virtual Template interface
  Virtual-TokenRing   Virtual TokenRing
  range               interface range command
6Router-1(config)#interface
```

For purposes of this discussion, select the Ethernet 0 interface by selecting **Ethernet** from the preceding list and selecting the **0** port. Remember, many routers have more than one type of specific port, so it is important to specify the port that you intend to work with:

```
6Router-1(config)#interface ethernet ?
  <0-0>  Ethernet interface number
6Router-1(config)#interface ethernet 0
6Router-1(config-if)#
```

As you can see from the preceding example, when you enter into interface configuration mode, your prompt changes from *(config)* to *(config-if)* for interface configuration. Again, you can enter a question mark to find out the configuration options for this interface:

```
6Router-1(config-if)#?
Interface configuration commands:
```

```
access-expression        Build a bridge boolean access expression
arp                      Set arp type (arpa, probe, snap) or timeout
backup                   Modify backup parameters
bandwidth                Set bandwidth informational parameter
bridge-group             Transparent bridging interface parameters
carrier-delay            Specify delay for interface transitions
cdp                      CDP interface subcommands
cmns                     OSI CMNS
custom-queue-list        Assign a custom queue list to an interface
default                  Set a command to its defaults
delay                    Specify interface throughput delay
description              Interface specific description
dlsw                     DLSw interface subcommands
dspu                     Down Stream PU
duplex                   Configure duplex operation
exit                     Exit from interface configuration mode
fair-queue               Enable Fair Queuing on an Interface
fras                     DLC Switch Interface Command
help                     Description of the interactive help system
hold-queue               Set hold queue depth
ip                       Interface Internet Protocol config commands
ipv6                     IPv6 interface subcommands
keepalive                Enable keepalive
lan-name                 LAN Name command
llc2                     LLC2 Interface Subcommands
load-interval            Specify interval for load calculation for an
                             interface
locaddr-priority         Assign a priority group
logging                  Configure logging for interface
loopback                 Configure internal loopback on an interface
mac-address              Manually set interface MAC address
max-reserved-bandwidth   Maximum Reservable Bandwidth on an Interface
media-type               Interface media type
mtu                      Set the interface Maximum Transmission Unit (MTU)
multilink-group          Put interface in a multilink bundle
netbios                  Use a defined NETBIOS access list or enable
                             name-caching
```

```
  no                        Negate a command or set its defaults
  ntp                       Configure NTP
  pppoe                     pppoe interface subcommands
  pppoe-client              pppoe client
  priority-group            Assign a priority group to an interface
  random-detect             Enable Weighted Random Early Detection (WRED) on
                                an Interface
  rate-limit                Rate Limit
  rmon                      Configure Remote Monitoring on an interface
  sap-priority              Assign a priority group
  service-policy            Configure QoS Service Policy
  shutdown                  Shutdown the selected interface
  sna                       SNA pu configuration
  snapshot                  Configure snapshot support on the interface
  snmp                      Modify SNMP interface parameters
  standby                   HSRP interface configuration commands
  timeout                   Define timeout values for this interface
  traffic-shape             Enable Traffic Shaping on an Interface or
                                Sub-Interface
  transmit-interface        Assign a transmit interface to a receive-only
                                interface
  tx-ring-limit             Configure PA level transmit ring limit

6Router-1(config-if)#
```

We could go on forever with each of the different interface configuration options, and that is just for this particular interface. Configuration commands are at the very heart of working with routers and at the very heart of this book. We will leave our discussion of configuration options with this piece of advice: *Remember the question mark!* It will be your best friend in knowing the appropriate configuration changes and commands on routers.

Using *show* Commands

Show commands serve as invaluable tools for diagnosing your routers and network conditions. As with configuration commands, literally hundreds of potential *show* commands exist, ranging from the very general, such as showing the overall configuration on a router, to the very specific, such as showing the collisions on an Ethernet interface, and everything in between. Once again, you can use the question mark to learn the available *show* commands at your disposal. The following listing is the command output of a *show ?* command. As before, the output was shortened for purposes of brevity. More than 100 possible *show* commands exist.

```
6Router-1#show ?
  aaa                Show AAA values
  access-expression  List access expression
  access-lists       List access lists
  accounting         Accounting data for active sessions
  adjacency          Adjacent nodes
  aliases            Display alias commands
  alps               Alps information
  arp                ARP table
  async              Information on terminal lines used as router interfaces
  backup             Backup status
  bgp                BGP information
  bridge             Bridge Forwarding/Filtering Database [verbose]
  bsc                BSC interface information
  bstun              BSTUN interface information
  buffers            Buffer pool statistics
  caller             Display information about dialup connections
  cca                CCA information
  cdapi              CDAPI information
  cdp                CDP information
  cef                Cisco Express Forwarding
  class-map          Show QoS Class Map
  clock              Display the system clock
  cls                DLC user information
  cns                CNS subsystem
  compress           Show compression statistics
  configuration      Contents of Non-Volatile memory
  connection         Show Connection
  controllers        Interface controller status
  cops               COPS information
  debugging          State of each debugging option
  derived-config     Derived operating configuration
  dhcp                Dynamic Host Configuration Protocol status
  dialer              Dialer parameters and statistics
  dlsw                Data Link Switching information
  dnsix               Shows Dnsix/DMDP information
  drip                DRiP DB
  dspu                Display DSPU information
  dxi                 atm-dxi information
  entry               Queued terminal entries
  exception           exception informations
  file                Show filesystem information
  flash:              display information about flash: file system
  flh-log             Flash Load Helper log buffer
  frame-relay         Frame-Relay information
  fras                FRAS Information
  fras-host           FRAS Host Information
  funi                FUNI information
  history             Display the session command history
  hosts               IP domain-name, lookup style, nameservers, and host
. . .
6Router-1#show
```

As you can see, quite a few potential *show* commands are available to you. Remember that each one of these commands has specific parameters that you can specify. You can learn what each of these is by issuing the command followed by a question mark. Take a look at the parameters available for *show interface*:

```
6Router-1#show interface ?
  Async               Async interface
  BVI                 Bridge-Group Virtual Interface
  CTunnel             CTunnel interface
  Dialer              Dialer interface
  Ethernet            IEEE 802.3
  Loopback            Loopback interface
  MFR                 Multilink Frame Relay bundle interface
  Multilink           Multilink-group interface
  Null                Null interface
  Serial              Serial
  Tunnel              Tunnel interface
  Vif                 PGM Multicast Host interface
  Virtual-Template    Virtual Template interface
  Virtual-TokenRing   Virtual TokenRing
  accounting          Show interface accounting
  crb                 Show interface routing/bridging info
  irb                 Show interface routing/bridging info
  mac-accounting      Show interface MAC accounting info
  precedence          Show interface precedence accounting info
  rate-limit          Show interface rate-limit info
  summary             Show interface summary
  |                   Output modifiers
  <cr>
  Lex                 Lex interface

6Router-1#show interface
```

Although hundreds of potential *show* commands are available on a router, you will find a few universally useful in understanding and diagnosing your system configuration and operation, including *show version*, *show running-configuration*, and *show interface*, which we will discuss further.

Using the *show version* Command

The *show version* command is a very useful command that enables an administrator to discern the following system conditions and parameters:

- System Platform
- System IOS version
- System Boot Rom Version
- System Uptime
- Reason for the last reboot
- System Image File
- Processor and Memory available
- Physical Interfaces
- Configuration Register

This is a truly useful command in fully understanding the general information about your system. This information is also useful in understanding malfunctioning system states such as reboots and gives a good overall picture as to the operation of your router. The following code example demonstrates this command for a Cisco 2500 router running IOS version 12.2(8)T:

```
6Router-1#show version
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-IS-L), Version 12.2(8)T,  RELEASE SOFTWARE
(fc2)
TAC Support: http://www.cisco.com/tac
Copyright (c) 1986-2002 by cisco Systems, Inc.
Compiled Wed 13-Feb-02 21:11 by ccai
Image text-base: 0x0306DA78, data-base: 0x00001000

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
BOOTLDR: 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE
SOFTWA)

6Router-1 uptime is 1 week, 5 days, 21 hours, 39 minutes
System returned to ROM by reload
System image file is "flash:c2500-is-l.122-8.T.bin"

cisco 2500 (68030) processor (revision N) with 14336K/2048K bytes of
memory.
Processor board ID 05606049, with hardware revision 00000000
Bridging software.
X.25 software, Version 3.0.0.
1 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read ONLY)

Configuration register is 0x2102

6Router-1#
```

Using the *show running-configuration* Command

The *show running-configuration* command is also a very useful tool in getting an overall picture of your router. Whereas the *show version* command discussed the actual system parameters, the *show running-configuration* command shows the actual system configurations that have been made on the router to process traffic. You will obtain the following information from issuing this command:

- IOS version
- Service Information
- Hostname
- Enable password (if not encrypted)
- IP and IPV6 static routes
- Access List information
- Host file entries
- Interface addressing information
- Dynamic Routing Processes
- Console, Aux, and VTY parameters

The following is an example of this command output. Keep in mind that this will vary greatly from system to system depending on the configuration. As with a few of the previous command outputs, we have significantly abbreviated this; it can literally be four to five pages and even more in certain instances.

```
6Router-1#show running-configuration
Building configuration…

Current configuration : 2718 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
service password-encryption
service tcp-small-servers
!
hostname 6Router-1
```

```
!
enable secret 5 $1$kW9A$aKyzOAaklR/ReD6YKiyQa/
!
clock timezone EST -5
clock summer-time EST recurring
ip subnet-zero
ip tcp synwait-time 5
no ip domain-lookup
ip host 6ROUTER-2 2001 192.168.123.50
ip host 2501-2 2001 192.168.123.50
!
ipv6 unicast-routing

interface Ethernet0
 ip address 192.168.123.50 255.255.255.0
 no ip route-cache
 no ip mroute-cache
 ipv6 address 3FFE:4200:1:1::1/64
 ipv6 address 2000:1:2::1/64
 ipv6 rip cisco enable
!
interface Serial0
 ip address 172.16.4.1 255.255.255.0
 no ip route-cache
 no ip mroute-cache
 ipv6 address 2000:1:1::1/64
 ipv6 rip cisco enable

!

!
router rip
 redistribute ospf 1 metric 3
 passive-interface Serial0
 network 172.16.0.0
!
!
```

```
 address-family ipv6
 neighbor cisco activate
 neighbor 2000:1:1::2 peer-group cisco
 network 2000:1:1::/64
 network 2000:1:2::/64
 no synchronization
 exit-address-family
!


!
line con 0
line aux 0
 exec-timeout 0 0
 transport input all
line vty 0 4
 exec-timeout 90 0
 password 7 060A1A2255
 login local
!
end

6Router-1#
```

Using the *show interface* Command

The *show interface* command gives the physical and logical information for each physical and logical interface on the router. You can also use this command to specify a specific interface or type of interface by following it with the interface type, such as *show interface ethernet 0*. Information gathered from this *show* command includes the following:

- Physical up/down status
- Line Protocol up/down status
- IP or logical addressing information
- MTU information

- Encapsulation
- Queuing
- Collisions
- CRC errors
- Dropped packets

This *show* command is very useful in diagnosing problems associated with specific interfaces and is a powerful tool for your system's overall health. The following example is of the *show ethernet 0* command output:

```
6Router-1#show interface ethernet 0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 00e0.b05a.d998 (bia 00e0.b05a.d998)
  Internet address is 192.168.123.50/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue :0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     93866 packets input, 13802455 bytes, 0 no buffer
     Received 47471 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     234089 packets output, 24304032 bytes, 0 underruns
     0 output errors, 2 collisions, 31 interface resets
     0 babbles, 0 late collision, 19 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Increasing Efficiency by Using Shortcuts

Throughout this chapter we have purposefully completely written out the entire string when entering a specific command to ensure that you understood the entire command syntax. Now that you do, we'll discuss how to save you a considerable amount of time and energy by using *shortcuts*. A shortcut is a very useful feature built into the Cisco IOS that enables you to enter only a partial command at the command prompt, just enough for the IOS to recognize it. For example, you can enter **copy run-star**, which the router will recognize to mean "copy running-configuration startup-configuration." Isn't the former much easier? Look at a brief example, in which you will use shortcuts rather than entire commands to configure the router. In this example, you will enter the router via the console and configure an IP address for the Ethernet 0 interface. You will then copy the running configuration to the startup configuration:

```
6Router-1>en
6Router-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#int e0
6Router-1(config-if)#ip add 10.1.1.1
6Router-1(config-if)#ex
6Router-1(config)#^Z
6Router-1(config)# copy run star
```

In the preceding example you connected to the router and entered privileged exec mode by entering **en** instead of typing **enable**, then entered configuration mode by entering **config t**, which the router recognized as **configure terminal**. After that, you entered interface configuration mode by typing **int e0**, which the router recognized as **interface Ethernet 0**; then you configured an IP address by using the **IP add** command. You then exited interface configuration mode by typing **ex** for **exit** and copied the running config to startup config by typing **copy run star** for the command **copy running-config startup-config**.

You probably can understand the syntax from looking at the example, however the rule for shortcuts is that you can reduce a command as far as possible to keep it unique from other commands. For example, you could not reduce disable to DI, because disconnect also starts with DI, so you would need to use DISA, and conversely, disconnect would need to be DISC. If you do not enter enough information to make the command unique, you will be met with an error stating

% Ambiguous command:. This indicates that you need to provide more information to the router.

Another very useful feature inherent in the Cisco IOS is the *auto-complete* feature. This feature automatically completes the entering of a command for you when you simply press the **Tab** key. Once again, the same rules apply as for shortcuts: you must enter enough information for the command to be unique or it will not work. The following code example demonstrates this feature. In this example you are trying to enter the *configure terminal* command. You start by trying to enter **con**, but this does not work because both the *configure* and *connect* commands start with "con," so you have to enter enough information to make it unique by entering **conf**. You follow this by **t** and hit the **Tab** key to get the terminal word to appear:

```
6Router-1#con
6Router-1#conf
6Router-1#configure t
6Router-1#configure terminal
```

A set of shortcuts is also available with specific keystrokes. Here is a list of some of them and their associated actions:

- **CTRL+A** Move to the beginning of a line
- **CTRL+E** Move to the end of a line
- **CTRL+R** Redisplay a line
- **CTRL+U** Erase a line
- **CTRL+W** Erase a Word
- **CTRL+Z** Exit Configuration Mode

Summary

All Cisco routers share two commonalities. They share the same common operating system platform, the Cisco Internetwork Operating System (IOS). This common operating system allows for simplification and generalization of administration, affording administrators ease of use.

The majority of routers also have two physical interfaces: a console port and an auxiliary port. The console port is a crucial part of the router because it is used for initial configuration of the router. To use this port for configuration, a rolled RJ-45 cable as well as a 9– or 25-pin serial adapter is needed. After you have concluded initial configuration, you can administer the router remotely via Telnet connection and via the Simple Network Management Protocol (SNMP). Cisco routers can use SNMP to interoperate with common network management systems such as CiscoWorks, HP OpenView, and SMARTS.

A Cisco router has two basic levels of access: user mode and privileged mode. User mode offers a limited command set to diagnose simple problems and verify operation. Privileged mode offers the capability to make configuration changes and to diagnose complex problems and situations. Although literally hundreds of potential commands are available on a Cisco router, commands generally fall into two types: *configuration* commands and *show* commands. Configuration commands are those that change the operation of the router; for example, routing protocols, interface configurations, and passwords. You can access these commands in the configuration mode by using the *configure terminal* command. Administrators use show commands to display network or router conditions; these function solely as a reporting function, a majority of which must be executed from privileged mode.

Administrators also have the capability to take advantage of *shortcuts* in the Cisco IOS. Shortcuts enable an administrator to enter the beginning of a command and the system will recognize the rest. An important rule governing this procedure is that you must provide enough information for the command entry to be unique. Another feature built into the IOS is the capability to auto-complete a command by pressing the **Tab** key. The same rule about shortcuts applies here: you must provide enough information to make the command unique. Cisco also has several shortcut keystrokes built into the system to make entering commands quicker.

Solutions Fast Track

Connecting to the Router

☑ The initial connection to the router must be made through the console connector on the router. This is accomplished by use of a rolled-RJ45 cable and a 9– or 25-pin serial connector.

☑ After you have applied Internet Protocol (IP) or other logical addresses, you can access a router by Telnet sessions for configuration.

☑ Routers can also use Simple Network Management Protocol (SNMP) settings to integrate with network management systems (NMS) platforms.

Entering Commands to Configure a Cisco Router

☑ The router has two levels of access: *user mode* and *privileged mode*. User mode is a very restricted level of access and allows for only basic functions, whereas privileged mode allows for full configuration and diagnosing capabilities.

☑ Two basic types of commands are available: *configuration* commands and *show* commands. You use configuration commands to change the operation of the router and its processes; you use *show* commands to understand and diagnose the system configuration and operation.

☑ Enter a question mark (**?**) at the command prompt after a particular command to see all the available commands and their configuration parameters.

Increasing Efficiency by Using Shortcuts

☑ Commands can be abbreviated by entering only the first few letters of the command—you must enter enough characters for the IOS to be able to recognize it.

☑ When you press the **Tab** key after entering the beginning of a command, the Cisco IOS's auto-complete feature automatically fills in the rest of the command. The same rule applies as for shortcuts: you must enter enough information to make the command unique.

☑ Several keystroke shortcuts are also available throughout the IOS.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: I am trying to connect to the console port of my router with no luck. I have the 9-pin connector on my computer and an RJ-45 cable; what should I do?

A: To communicate with the console port on your router, you must use a *rolled* RJ-45 cable—not a standard RJ-45 cable. The difference is that pins 5, 2, 4, and 1 are swapped on opposite ends of the cable. Use this cable and you should have no problem.

Q: I have replaced the cable with a rolled RJ-45 cable, but I still cannot access the router. I am using HyperTerminal on my Windows 2000 computer with the standard settings.

A: To make HyperTerminal work with your router, you need to make your line speed 9600, use 8 data bits, 1 stop bit, and turn off flow control. This should enable you to talk to your router.

Q: What is the difference between the *show version* command and the *show running-configuration* command?

A: The *show version* command deals specifically with the information regarding the operation of the router as system such as processor, memory, and software revision. The *show running configuration* command displays information regarding the actual routing configuration on the router. This will display information such as routing protocol information, static routing, passwords, and interface addressing parameters.

Q: I am trying to enter a shortcut for the *disable* command using "dis", but it doesn't seem to work. It also is not working when I press the **Tab** key. What is the problem?

A: When using shortcuts you must remember the fundamental rule of keeping the command unique. There are other commands that also begin with "dis-", so you must extend your shortcut further to allow for this.

Chapter 2

Introduction to IPv6 Architecture

Solutions in this chapter:

Introduction

Version 6 of the Internet Protocol (IPv6) has finally arrived in practical form! Although the protocol has been worked on for close to ten years now, official specifications and standards have only recently been finalized, and there are still some aspects of the protocol that are being worked on by the Internet Engineering Task Force (IETF) working groups. The issues that made version 4 of the Internet Protocol (IPv4) inadequate required complex solutions. This has forced designers of the new protocol to work diligently to ensure that the same issues would not be encountered with the new version of the protocol. Members of the Internet community who were responsible for developing the protocol carefully scrutinized each new Request for Comments (RFC) that was developed. For those not familiar with the RFC process, RFCs are documents that detail the protocol specifications so that hardware and software manufacturers will know how to implement the protocol in a standard and agreed-upon manner. Standardization enables each manufacturer and software vendor to follow the same blueprint rather than developing proprietary versions of the protocol. (This was a common problem with earlier networking protocols.)

The realization of a new, scalable protocol in which considerable thought has been given to future expansion is a very exciting concept. Never before has the Internet community seen such a magnitude of effort or planning put into the development of a new protocol, and there has certainly never been a protocol more specifically tailored to the growth of the Internet. The purpose of this book is to help ensure a smooth implementation of the new protocol by focusing on the configuration of IP version 6 on the Cisco IOS, the most common platform that will utilize the new protocol.

Understanding the Benefits of IPv6

Let's discuss some of the benefits of IPv6 in more detail in order to see how this protocol attempts to deal with the Internet and business network problems of today. Some of the main reasons for the development of the new version of the Internet Protocol were the exhaustion of the Class B address space, the growth of the backbone routing table, security issues, IP options size limitation, and routing performance. We'll look at the two main problems solved by IPv6—namely address depletion and routing scalability—in more detail, and then look at some of the added benefits that IPv6 gives to network designers and administrators. The benefits of IPv6 include:

- Increased IP Address Size
- Increased Addressing Hierarchy Support
- Simplified Host Addressing (unified addressing: global, site, local)
- Simplified Autoconfiguration of Addresses (easier readdressing, DHCPv6, Neighbor Discovery instead of ARP broadcasts)
- Improved Scalability of Multicast Routing
- The *Anycast* Address
- Streamlined Header
- Improved Security (security extension headers, integrated data integrity)
- Better Mobility (home agent, *care-of* address, routing extension header)
- Better Performance (aggregation, Neighbor Discovery instead of ARP broadcasts, no fragmentation, no header checksum, flow, priority, integrated QoS)

Increased IP Address Size

IPv6 has 128 bits available for addressing. Let's examine this briefly in order to appreciate the vastness of this degree of address space. 128 bits of address space means that there are 2^{128} different addresses available. The first three bits of 001 are reserved for Globally Routable Unicast addresses. This means that we have 125 bits left to use for addresses (128–3=125), so we have 2^{125} addresses available before Globally Routable Unicast address space is depleted. This is roughly 4.25 E+037 addresses. To put this into perspective, let's compare this to IPv4. In IPv4, we use all address space between 0.0.0.0 and 223.255.255.255 for unicast routing, which is approximately 3.7 E+09 addresses (we will not take into account the addresses delegated as non-routable reserved addresses as defined by RFC 1918). This means that IPv6 has room for 10^{28} more addresses than IPv4!

Clearly, 128 bits provides enough address space to take current Internet trends well into the future. In fact, it may seem like an inexhaustible amount of address space, but we will see when we get into the details of LAN and WAN configuration that this is not quite the case. Still, it is a much larger address space than IPv4. One thing to think of in order to appreciate IPv6 is the number of *networks* that IPv6 can support. An IPv6 address uses the last 64 bits to describe the host ID for a system on a network. In addition, IPv6 actually uses the last 64 bits of the address to distinguish hosts from one another on the same subnet. Whether using the link-local, site-local, or Globally Routable Unicast address format, the last 64 bits on a machine will remain the same. This is because IPv6 uses the Layer 2 Media Access Control (MAC) address as the host ID for a machine (the Layer 2 MAC address is the address that is burned into all Layer 2 hardware, such as Ethernet cards and other Network Interface Cards). Since MAC addresses are only 48 bits long, each one is padded with a 16-bit prefix, which will be discussed in more detail below. This limits the number of addresses that can be used because there will rarely be 2^{64} addresses in use on a typical Ethernet LAN. Some address space definitely gets wasted. However, if you remove the 64 bits used for host ID and the first three bits used to designate Globally Routable Unicast addresses, you get 2^{61} possible addresses (2.31E+018). So even without using all of the addresses that IPv6 has available, we have the scaling ability to take us well beyond the future of IPv4. Clearly, IPv6 frees up our ability to use addressing efficiently without having to worry about running out of addresses. Please keep in mind that I do not mean to say that address space should be used in a carefree manner; that is how IPv4 came into its current predicament.

Table 2.1 presents a rough comparison of the address sizes of IPv4 and IPv6.

Table 2.1 Address Space Comparison

Specification	IPv4	IPv6
Address Length	32 bits	128 bits
Host Identifier Length	2 - 24 bits	64 bits
Network Identifier Length	7 - 30 bits	61 bits
Maximum Number of Hosts per Subnet	2^{2} – 2^{24}	2^{64}
Maximum Number of Subnets	2^{7} – 2^{30}	2^{61}
Maximum Number of Hosts	3.7 E+09	4.25 E+037

Increased Addressing Hierarchy Support

As we learned earlier in the chapter, IPv6 addressing has restructured the means by which address blocks are delegated. IPv4 first used the classful IP assignment rules, then began to assign based on the principles of Classless Inter-Domain Routing (CIDR). IPv6 corrects the de-aggregation problems associated with each of these by splitting the IPv6 address into a set of definite scopes, or boundaries, by which IPv6 addresses are delegated.

The Format Prefix is used to show that an address is Globally Routable Unicast, or another type of address, and is always set to the same value. This allows a routing system to quickly discern whether or not a packet is Globally Routable Unicast or some other type. By obtaining this information quickly, the routing device can more efficiently pass the packet off to routing subsystems for proper handling.

The Top Level Aggregator (TLA) ID is used for two purposes. First, it is used to designate a large block of addresses from which smaller blocks of addresses are carved in order to give downstream connectivity to those who need access to the Internet. Second, it is used to distinguish where a route has come from. If large blocks of address space are given only to Internet service providers, and then in turn delegated to customers, it becomes easier to see which transit network a route has traversed, or from which transit network the route first originated. With IPv4 many addresses were portable, and the numbers authorities were delegating blocks down to small businesses, so it became impossible to know where a route came from without tracing back towards the source of the packet. Now, with IPv6, determining the source of a route is more feasible. Imagine an Internet consisting of 500 Tier 1 providers. If this were the case (and it is most likely not too far off from today, though what makes a provider a Tier 1 provider is very ambiguous), then at the very least a quick search through a text file could tell you where a route originated, based on the TLA ID of the longest match route. It's even possible to obtain software that has this functionality built into it (if the software is properly written to keep a dynamic list as new delegations are assigned).

Let us look at the size of the TLA in more detail. We discussed in prior sections how the address space would be given only to providers and those who needed their own IPv6 space. (The term *needed* is used cautiously here, as it is ambiguous as well—there are currently no set boundaries or requirements with respect to what *need* means in this context.) This way, we are able to sufficiently aggregate prefixes into big blocks at the Internet core, thereby passing fewer routes between routing domains, as well as internally, which will increase the efficiency of the Internet core.

For fun, let us assume that we have the IPv6 address delegation 3D00::B234::/24. Let us further assume that all of our customers have sufficient need for a /48 delegation for their networks. This leaves us with 24 bits of addressing to delegate out—that's a lot of address space! In fact, the number of networks we can support with this scheme is equivalent to the number of *hosts* we could support with a Class A IPv4 address block! You can see that there will be much more pressure put on Tier 1 service providers to efficiently track the delegation of address space that they make. Currently, a Tier 1 service provider gets addresses in blocks of perhaps /16 or less. If we assume that a service provider today only delegates addresses up to /24, that leaves only 256 delegations (eight bits) that the service provider can make prior to applying for more address space. Most Tier 1 service providers today are required to subnet delegations down to at least /28 in order to qualify for more address space, so this example may not be entirely realistic, but we can still grasp the size of a TLA, which is monumental compared to the size of the assignments that are given today.

NOTE

This amount of address space can present tremendous support infrastructure problems. Service providers that today provide DNS service for their downstream customers will have to think long and hard about the best way to implement these support structures before getting into the IPv6 market.

As we can see from the previous example, Tier 1 service providers will have extremely large address spaces to deal with. This will not only eliminate most of the politics surrounding address delegation and obtaining more address blocks, but will also provide motivation for major support and automation of infrastructure upgrades within an organization. Many service providers today have difficulty upgrading support structure due to the engrained functionality and interdependencies of many support platforms integrated together. IPv6 provides for great challenges and opportunities, not only in network engineering and architecture, but also in IT development and integration. The trick will be making the move from the old world to the new look like a fresh start, rather than a workaround for support.

The Next Level Aggregator (NLA) address block is a block of addresses that are assigned downstream out of a TLA block. We know that these addresses are to be aggregated as much as possible into bigger TLA blocks, when they are exchanged between providers, in the Internet core. Let us look at the benefits of this type of addressing structure from the NLA perspective.

There are two main advantages to getting address space from a provider. The first has to do with individual backbone routing stability. If we are an NLA and wish to provide downstream service to our customers, we will most likely wish to provide the fullest, most robust service we can to our client base in order to retain current clients and to gain market share. Perhaps we wish to allow customers to connect to us at multiple locations, as we are fairly geographically diverse for a given region, and have rich connectivity upstream to Internet Tier 1 (core) providers. Furthermore, we want to allow our customers to receive a full routing table, should they desire one, if they want to use explicit routes to form their routing policy. Perhaps they wish to load-balance between two connections, using some destinations preferred through one connection, and the rest preferred through the other connection to us. To do this, we have to carry full routes in our backbone so that we can pass them down to our customers. Though an Internet core is usually composed of very modern, robust routing equipment, a Tier 2 provider may not be able to afford to constantly upgrade their backbone in order to keep up with new technology and increased routing table size. Luckily, with IPv6, processing power is not as big a worry as it could be. Because the Internet core is fundamentally aggregated efficiently, we now have a much smaller routing table to maintain. We can provide full routes to a customer, and that set of routes may not be too big for us to handle. So by everyone "playing nice" and following aggregation strategies, we are able to reap the benefits of the core's minimized routing table size in our own network backbone.

The second benefit to NLA aggregation has to do with actual route stability of our routes across the Internet core globally. Some background information is needed in order to fully appreciate this point. At the beginning of the Internet's explosion in size, there were times when the Internet was not very stable. BGP speakers would lose routes due to backbone links failing, immature software and the like. Because of this, routes were constantly being advertised and then withdrawn (when the route became unreachable), causing considerably more processing to take place on core routers, which must keep an up-to-date set of full Internet routes at all times. To combat this BGP instability, the concept of *route dampening* emerged. Essentially, route dampening works as follows: every time a route is withdrawn and re-advertised, it is assigned a penalty, which is kept track of at the place of instability (usually an exterior border gateway protocol or eBGP session). The more the route "flaps," the higher the penalty associated with that route. When the penalty associated with the route reaches a certain level, the route is withdrawn and not accepted for advertisement for a given period of time. When this happens, the route is said to be *dampened*. The dampened route must undergo some period of wait time, without flapping more (or the penalty gets even higher) before it can be re-introduced into a router's BGP table. When the route goes long enough without flapping (the penalty decreases with time), the route is again allowed, and it is inserted back into the router's BGP table and treated just like other routes. Route dampening provided a way for the Internet core to deal with instabilities in a manner that minimized the cost of other crucial processing. Border Gateway Protocol (BGP) version 6 and IPv6 routing will be discussed in detail in a later chapter to better explain this concept.

Configuring & Implementing…

Site Renumbering

Site renumbering is performed when there is a need to replace or redistribute existing addresses. An extreme example of site renumbering can occur when a new service provider is chosen and new global addresses need to be issued. Site renumbering in IPv4 can pose an administrative nightmare, but IPv6 simplifies this process immensely. Here are two ways to implement the site renumbering process:

- If stateful autoconfiguration is in use, the Top Level Aggregator and Next Level Aggregator numbers of the new service provider can be placed in the DHCPv6 service and propagated throughout the site.
- If stateless autoconfiguration is in use, the new network prefixes can be placed in the routers and propagated throughout the site.

The advantage of the IPv6 architecture is that the interface identifier remains the same; only the prefixes are changed. In the case of IPv6 stateful autoconfiguration, the prefixes can be propagated from a central point.

Now that we understand route dampening, we can appreciate the second benefit of aggregation. When our upstream provider aggregates this route for us and only announces the aggregate to their peers, this aggregate will, in all likelihood, remain stable, without respect to the stability of our own network. Because of this, we are virtually certain that another provider will never dampen our routes somewhere else on the Internet. None of the more specific routes that we use need be spread across the Internet core, outside of our own upstream provider. This improved routing stability is a major benefit to aggregation as a whole, both in IPv4 and in IPv6. The good part is that it is *required* in IPv6—in IPv4 it is only recommended. So now the only place that we need to worry about being dampened is within our own upstream provider's network. Fortunately, because we are paying for our upstream connectivity in most cases, it is substantially easier to get our own upstream provider to help us remove the penalties on dampened routes than it would be with another provider to whom we had no financial obligation. As you can see, in addition to more addresses and smaller routing table sizes, there are more ramifications of IPv6 aggregation schemes than first meet the eye.

The Site Level Aggregator (SLA) enjoys most of the benefits that an NLA does, except for its size: the SLA is usually a network or network provider with a much smaller network. Because of this, a smaller delegation of address space is needed. It retains the values of aggregations in that its routing tables are kept smaller, even when receiving a full Internet routing table from its upstream provider. It also enjoys the benefits of global route stability, in that its upstream provider, whether an NLA or a TLA, aggregates according to the principles of the IPv6 aggregations model.

Simplified Host Addressing

As we studied earlier, the IPv6 model defines 128 bits of address space. The first 64 bits are used for network numbering, and the last 64 bits are used for host numbering. We also remember that the last 64 bits of the host ID are obtained from the MAC address of the host's Network Interface. You may wonder how the 64-bit address is derived from a MAC address, which is classically only 48 bits. In this section we will look into how addresses are derived, and what developments we may see in the future as a result of IPv6's addressing scheme.

By convention, when assigning a host in IPv4, IPv4 will break up the given subnet and assign host addresses based upon the addresses that are available. Also by convention, the first address is normally given to the designated router, and the rest of the addresses get assigned to hosts on that subnet, with the last address in the subnet reserved for the broadcast addresses of that subnet. In IPv6, the situation is somewhat different. With IPv6, we know that the host ID is a 64-bit address that is obtained from the MAC address. Although the MAC addresses of today are typically 48 bits, we need a way to get the host ID to come out to 64 bits. The solution to this problem is to pad the MAC address with some well-defined set of bits that will be known by routing systems on that subnet. Today, we use the strings **0XFF** and **0xFE** (:FF:FE: in IPv6 terms) to pad the MAC address between the company ID and the vendor-supplied ID of the MAC address. (MAC addresses are delegated in much the same way as IP addresses today, except that companies who make network interface (NIC) cards are given a piece of MAC space, rather than providers being given IPv4 space.) This way, every host will have a 64-bit host ID that is related to their MAC address in the same way. Furthermore, we know that the 64-bit MAC address will be unique on a given network, because every NIC card will have a unique MAC address. This well-defined padding makes it possible to learn the IPv6 addresses (or at least the host IDs) of other machines on the subnet simply by learning the Layer 2 MAC information.

One interesting debate is whether or not MAC addresses will need to become 64 bits in length prior to the widespread deployment of IPv6. If there is a need for MAC addresses to become longer (if all MAC addresses are used), then 64 bits will most likely be the next option for length, as this will supply over 1.8E019 more MAC addresses to use ($2^{64}–2^{48}$). Moreover, if this comes to be the case, we may simply stop the padding of the MAC address, and use the full 64 bits of the MAC address for the Host ID.

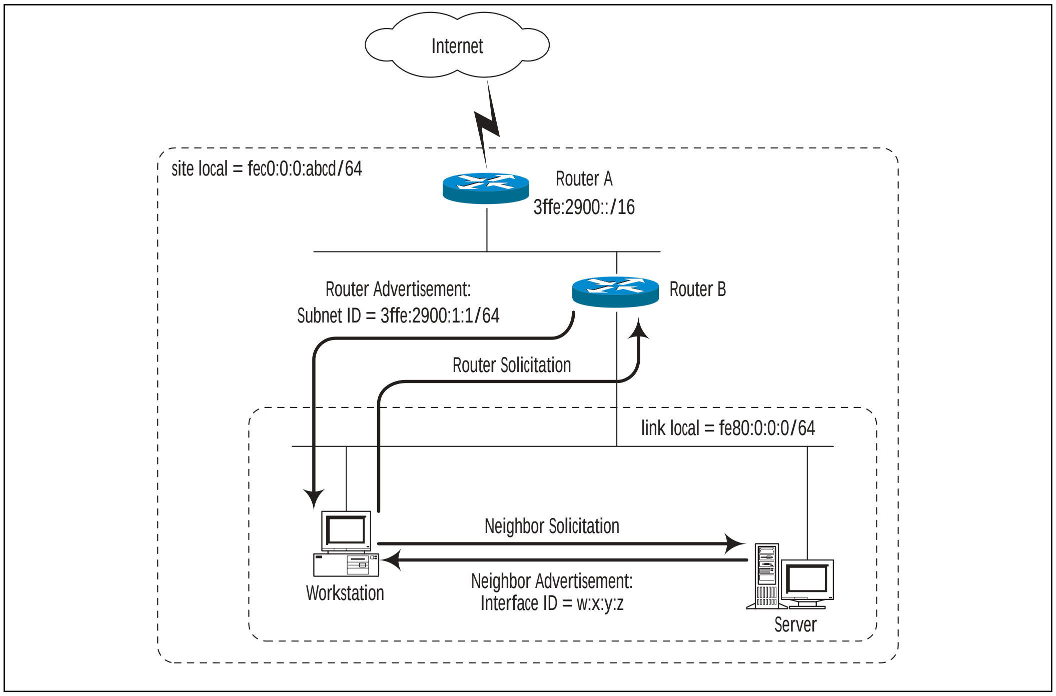
Simpler Autoconfiguration of Addresses

One of IPv6's best perks for administrators is that not only is the Host ID determined prior to configuring an IPv6 speaking machine, but the network on which it resides can be deduced as well, making it possible for autoconfiguration to take place. Before we go into detail about autoconfiguration, a new type of address will have to be brought into our repertoire: the *multicast* address.

A multicast address can be simultaneously assigned to more than one machine. It differs from an anycast address in that anycast packets are routed to the *closest* destination (one of the set of machines with the same address), while multicast packets are routed to *all* machines that are assigned that address. This is fundamentally different than a Globally Routable Unicast address in that more than one host can be numbered with the same address, so the address that a given host is assigned need not necessarily be unique for the scope on which the multicast address is acting. All machines assigned this multicast address are said to be in a *multicast group* whose address is the multicast address they use. Multicast-speaking machines send and receive data from more than one host (every member in that group). This type of addressing and routing has typically been used for 1-to-N or M-to-N type Internet transactions (when one or more people need to get identical information to more than one destination). Multicast provides an efficient means by which to do so.

If we unite the concept of multicast with the concept of the Host ID coming from the hardware on a given machine, we can see how autoconfiguration is possible. When a machine first powers up onto a network and realizes that it is connected and is supposed to speak IPv6, it will send a multicast packet that is well known and has a standard definition out onto the LAN segment to which it is attached. This packet will be destined towards a locally scoped *(see earlier)* multicast address, known as the Solicited Node Multicast address. When the router sees this packet come in, it can reply with the network address from which the machine should be numbered in the payload of the reply packet. The machine receives the packet and, in turn, reads the network number that the router has sent. It then assigns itself an IPv6 address by appending its Host ID (obtained from its MAC address of the interface that it connected to that subnet) to that network number. See Figure 2.1 for a graphical representation of autoconfiguration. Not only does this require no manual intervention by the administrator to configure the machine (though it may or may not involve manual configuration of the router on that subnet), it also ensures that the address is unique. The machine is guaranteed to have a unique address because the network number is assigned uniquely by the router on that network, and the Host ID is unique because the MAC address of the interface by which that machine is attached is provided by the vendor and unique. Furthermore, now that it has a routable address, it can learn the default route that it needs in order to get off of that subnet. Notice the ease of configuration that we now have when we move from one network to another. Not only do we no longer have to manually reconfigure an end-station (and then reboot in most cases), we also no longer have to take time out of our network administrator's busy day for him to delegate an address in order to ensure its uniqueness. Also, the administrator no longer has to keep track of the addresses that he has assigned and which ones are free at any given time! Certainly, this can save a network administrator a great deal of time, not just in the paperwork associated with keeping track of addresses used, but in reconfiguration that must occur in order for a network to be renumbered. Think of the things an administrator could be doing if he isn't constantly being hounded for IP addresses or network numbers! We will go into more detail about the concept of the multicast address and its possibilities in a later section.

Figure 2.1 Autoconfiguration Mechanism



Improved Scalability of Multicast Routing

Now that we have studied unicast addressing, looked at the primary advantages of multicast addressing, and discovered the potential of routing table size scalability in the Internet with IPv6, let's take a moment to discuss the multicast address in a little more detail. Multicast servers are perhaps the most misunderstood technology of the present day. We'll start by examining the concept of multicasting in general.

In the beginning, the Internet was primarily a research network upon which research data flowed between one university and another. This was not big business, so congestion problems were tolerated, and the data that people were sending was not dated because it didn't need to be received in real-time. Today, by contrast, businesses and consumers are using the Internet for a vast array of applications. More and more, we are seeing different types of media going over the Internet, whether it's stock quotes, phone calls, or even our favorite TV channel. We see the need for media to arrive quickly and to be sent to an increased audience. Even things like newsgroups are getting information out to millions of people each day. This 1-to-N transmission trend is bringing about a need for a new type of traffic sending, in which one person can send a piece of data to many people. In the past, if we wanted to send a piece of data to ten friends, we would simply make ten copies of that data and send them to each person one at a time. However, as this type of transmission gains popularity, a scaling problem takes place. For instance, say we have a video or radio show that we wish to send out over the Internet. If we want to send this media to 10,000 people, all of whom want to see or hear the show in a fashion as close to real-time as possible, we have to make sure that our upstream bandwidth is sufficient to handle up to 10,000 times the data rate of one transmission. That requires us to spend much more money purchasing the upstream bandwidth to satisfy our client base (our viewers or listeners). Fortunately, the concept of multicast was conceived some time ago, and has been in a testing phase for quite a while. The idea of taking one piece of data and efficiently sending it to many interested parties at once becomes a complex routing problem, especially if we become caught in the unicast paradigm we are all used to. The concept of multicast addresses this problem.

In a multicast situation, we have a 1-to-N (or M-to-N) relationship between the source and the destination. Instead of using a unicast address to indicate that we are interested in receiving a given multicast feed, we use a multicast address. In IPv4, a multicast address is usually referred to as a *group address*. This group address, when applied to a machine or to an application on that machine, signifies that we are interested in listening to any data that is sent to that address. In IPv4, the address range of 224.0.0.0 to 239.255.255.255 is used to designate multicast group addresses. When someone wants to receive multicast feeds, they (temporarily or permanently, depending on the situation) assign themselves that address, and effectively listen for packets coming along that have that multicast address listed as a destination.

The routing for multicast becomes rather complex and is beyond the scope of this book, but you are encouraged to read more about it. Good information can be found either on the Multicast Forum home page at www.ipmulticast.com, or in the IETF working groups regarding multicast. Some of the IETF working groups you may wish to check out include the Mbone Deployment working group (www.ietf.org/html.charters/mboned-charter.html) and the Inter-Domain Multicast Routing working group (www.ietf.org/html.charters/idmr-charter.html). There are also a number of protocol-specific working groups currently active within the IETF, including the Multicast Source Discovery Protocol working group (MSDP) and the Protocol Independent Multicast working group (PIM). I leave it up to you to learn as much as you wish about the current updates to multicast routing in general. For the purposes of this book, let us assume that multicast works, and that it will help save us bandwidth (since it allows us to send only one stream out of our Internet connection, and the backbone will reproduce it as seen fit to make sure it gets to all the interested destinations).

One particularly good application of multicast is in a corporate network. Perhaps a memo needs to be sent to all employees' workstations at once, or a live videoconference from the CEO needs to be sent over the corporate network to all employees. In a corporate network, we want to save as much money as possible on bandwidth while maintaining an efficient routing structure. Multicast buys us just that. However, in most cases, we only want multicast information (streams) to get to the places that are supposed to see them. We do not want the whole Internet to hear our CEO talk about our newest secret initiative to take over our competition! For this, IPv6 has the concept of multicast scoping built into it. With IPv6, we can designate certain multicast streams to be routed only within a certain area, and never to allow packets to get out of that area for fear of who may see them. This scoping will be well known and understood by all routing entities in order to ensure, through minimal configuration, that multicast data and multicast routes do not get outside the edges of the routing domain for which they are meant to exist. Figure 2.2 presents the multicast addressing format in a little more detail.

Figure 2.2 IPv6 Multicast Address Format

```
|   8    |  4 |  4 |                  112 bits                   |
+------ -+----+----+---------------------------------------------+
|11111111|flgs|scop|                  group ID                   |
+--------+----+----+---------------------------------------------+
```

As we can see, the multicast addressing architecture is a little different than that of the Globally Routable Unicast addressing format. Notice that the first eight bits are all set to one, which will allow a routing device to know immediately that the packet is multicast in nature, and subject to the special handling associated with this packet type. The next four bits are used for flags. Currently, the first three bits in the *flags* field are reserved and undefined, so they should always be set to zero. (You will find that some implementers of protocols will use these bits fallaciously for some sort of proprietary signaling. This is fine until the bits get standardized to something in the future, at which point incompatibilities will arise.) The fourth bit is known as the *T bit* (see RFC 2373), and is used to decide whether the multicast address is a permanently assigned address (also called *well-known*) or a temporary assignment (also known as *transient*). So this field will tell us if the multicast address being used is one that is standard (perhaps a group address used to contact all nodes within a given routing domain, for example) or a temporarily assigned address (perhaps the Monday night football game broadcast over the Internet). The next field is the one we are interested in here: the *scope* field will determine how far the multicast packet can go, in what areas of a routing domain the packet can travel, and the group address that can be advertised. The *scope* field takes the values shown in Table 2.2.

Table 2.2 Scope Definitions

Scope Field Value	Definition
0	Reserved
1	Node-local scope
2	Link-local scope
3	(unassigned)
4	(unassigned)
5	Site-local scope
6	(unassigned)
7	(unassigned)
8	Organization-local scope
9	(unassigned)
A	(unassigned)
B	(unassigned)
C	(unassigned)
D	(unassigned)
E	Global scope
F	Reserved

Depending on how we assign our multicast address, we can control how far the multicast packets will travel, and how widely the routing announcements associated with that multicast group will be advertised. For instance, if you would like to advertise a multicast session of the fish tank in your office, and you would like the whole world to see it, you would assign a scope of E (1110 in binary). However, if you want to set up a multicast group so that you and your coworkers can have a video conference over the corporate network, you would want to make sure to give the address a scope of 5 (0101 in binary), or 2 (0010) if everyone involved is on the same LAN as you. This makes life a little easier in terms of controlling how far information gets propagated. Now, instead of relying on a Network Administrator to apply filters at the borders of each routing domain, we can rely on software (which is generally not susceptible to the same sort of random changes that networks are) to keep our traffic within the scope we want. This allows for privacy at a level that is much easier to implement. This is another benefit of IPv6: Not only are multicast boundaries well defined, they are also easy to maintain.

The Anycast Address

IPv6 defines a new type of address, known as the *anycast* address. Although this form of address is deployed in a limited fashion in IPv4, IPv6 integrates this address type into its operations, which improves routing efficiency. In this section we will look into some of the characteristics of the anycast address in detail and discuss some of the interesting applications of the anycast address in the IPv6 Internet of the future.

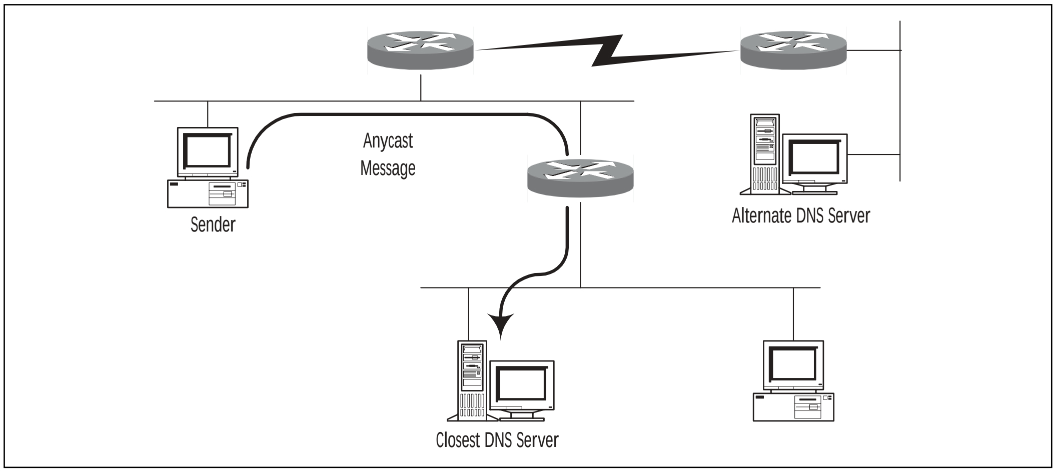
An anycast address is an IPv6 address that is assigned to a group of one or more hosts, all of which serve a common purpose or function. When packets are sent to the IPv6 anycast address, routing will dictate which member of the group receives the packet via the machine closest to the source, as determined by the IGP of the network in question. (IGP is the Interior Gateway Protocol: the routing protocol you use in your routing domain; for example, RIP, EIGRP, or IS-IS,) In this way, it becomes possible to disperse functionality geographically across your network in a way that improves efficiency in two ways. This differs fundamentally from the multicast address. Although both the anycast and the multicast addresses are assigned to more than one host, the anycast address serves for data transmissions that are 1-to-1, whereas multicast addressing is used when a data transmission to multiple destinations is required. Let us look at the two primary benefits of the anycast addressing scheme.

First, if you are going to the closest machine in a group and it does not matter which group member you exchange information with, you will usually save time by communicating with the closest (IGP-wise) group member. Second, communicating with the closest anycast group member saves bandwidth, because the distance a packet has to travel is, in most cases, minimized. So not only can anycast save you time, but it can also save you money (by using less bandwidth).

The anycast address does not have its own set of bits to define it; instead, anycast addressing is derived from either scoped or Globally Routable Unicast addresses. From the point of an IPv6-speaking machine, the anycast address is no different than a unicast address. The only difference is that there may be other machines that are also numbered with the same scope of unicast address within the same region for which that scope is defined (for instance, you may have more than one machine with a site-local anycast address within a given site).

Now that we understand the differences between anycast and multicast addresses, let's look into some possible uses of the anycast address. One application that anycast can help with is DNS (Domain Name Service). If we were to offer DNS to many people or customers, as in the case of most Tier 1 service providers today, we would need to build our DNS in a way that could handle a large number of queries from all parties for which we provide the service. Because of this, it is often more efficient to deploy multiple DNS servers and spread them out geographically. This will allow for fail-over if one DNS server becomes unreachable due to network failures, and it will also allow us to distribute the load of our DNS service between these servers. However, we do not want to make our customers assign too many different IP addresses for DNS servers to point to for resolution, as most people only use one or two. Also, we want some way for one or two IP addresses to be used for all of our geographically diverse service, for the fail-over reason just stated. One way to do this would be to assign each DNS server that has identical configuration and authoritative information the *same* IP address. If we then inject routes to each of these DNS servers into our backbone routing table, when someone queries our DNS the request will be sent to the DNS server that is geographically closest. This will allow us to split up the load among multiple DNS servers and to avoid too much backhauling of DNS queries across our backbone. So by this method of deployment, we are saving both time for our customers (DNS servers are close, so their data transmission takes less time), and money for ourselves (bandwidth = money for service providers). Because DNS is User Datagram Protocol (UDP)-based rather than TCP-based, transactions between DNS servers and end-stations are quick and short, and don't need to be kept track of with sequencing, error checking, etc. When we want to resolve a host name, a packet is sent to the DNS server requesting the address associated with a given Internet Domain Name and a response is sent back with the answer. This makes the anycast addressing model viable for this type of application. An illustration of anycast in use is provided in Figure 2.3. You can read more on this specific type of deployment at www.globecom.net/ietf/draft/draft-catalone-rockell-hadns.00.txt.

Figure 2.3 Anycast Message



Designing & Planning…

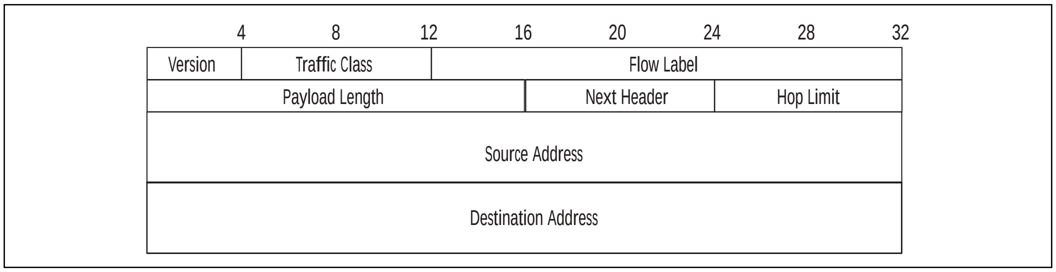
Which Applications Are Good Candidates for Anycast?

Some applications may not be well suited to anycast deployment. For instance, TCP-based applications using anycast addressing for deployment will not provide the fail-over capabilities that the previous example provides. When we are using a UDP-based application, there is no sequencing information to keep track of. With TCP, we run into a problem: when a network problem occurs and users are in the middle of a TCP session with an anycast machine, the TCP sequencing will be all wrong when the traffic gets rerouted to the next-closest anycast server. In the case of Web traffic, which is largely TCP-based, the user would need to reload the Web page (at least), to get to where he or she was when the failure occurred on the network. Other applications could have even more disastrous consequences associated with rerouting, so careful consideration should be given to which types of services are supplied using an anycast model.

Streamlined Header

The new IPv6 header is simpler and more streamlined than the IPv4 header. The new header has only six fields and two addresses, while an IPv4 header contains ten fixed fields, two addresses, and a variable-length *options* field. Figure 2.4 illustrates the format of an IPv6 header.

Figure 2.4 IPv6 Header

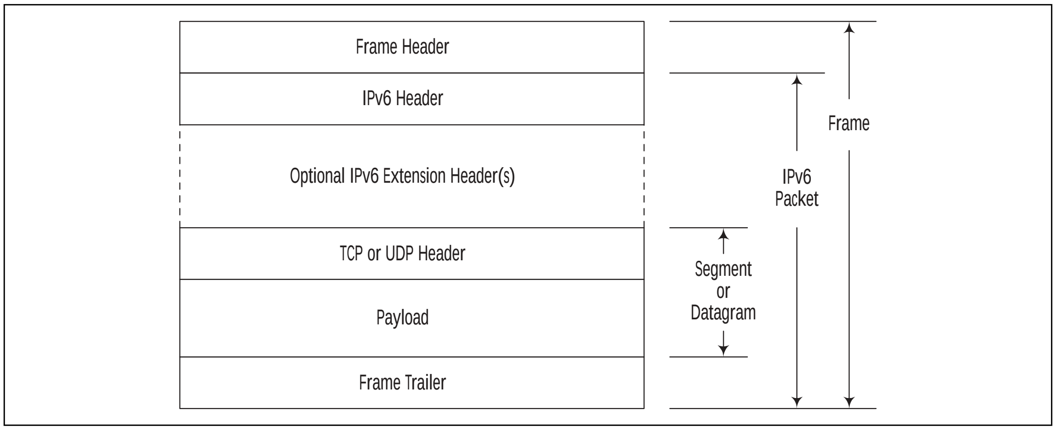


The IPv6 header provides the following simplifications:

- **Simplified Format** The IPv6 header has a fixed format with fewer fields. The variable-length *options* field has been eliminated. Other IPv4 fields have been eliminated or changed to optional extension headers. Not all of the optional extension headers need to be processed by each node—many of the extension headers are intended only for processing by host nodes. This simplified format reduces the protocol overhead of IPv6 and allows for greater flexibility.
- **No Header Checksum** The IPv4 *checksum* field has been eliminated. This field was included in IPv4 because early networks used slow and unreliable links, thus computing the checksum at each hop was necessary for ensuring data integrity. Today's network links are fast and highly reliable, so only hosts need to compute the checksum, not routers.
- **No Hop-by-Hop Fragmentation Procedure** In IPv4, routers fragmented packets that were too large for transmission on the outgoing interface. This added significantly to IPv4's processing overhead. In IPv6, only the host may fragment a packet. To aid the host, IPv6 includes a function that finds the maximum transmission unit (MTU) size from source to destination.

Figure 2.5 illustrates a transmission frame of a TCP segment (or UDP datagram) using IPv6.

Figure 2.5 Transmission Frame with IPv6



Security

One goal of IPv6 designers was to support interoperable encryption-based security. IPv6 integrates security into its architecture by introducing two optional extension headers: the Authentication Header (AH) and the Encrypted Security Payload (ESP) header. These two headers can be used together or separately to support many types of security functions.

- **Authentication Header (AH)** The heart of the Authentication Header is the *integrity check value (ICV)* field. The ICV is computed by the source and computed again by the destination for verification. This procedure provides both connectionless integrity and data origin authentication. Connectionless integrity detects modifications to the payload. Data origin authentication verifies the identity of the source of the data. The AH also contains a *sequence number* field that can be used to detect packet replay attacks, which tie up receiving system resources. By examining the sequence numbers, we can spot the arrival of duplicate IP packets.
- **Encrypted Security Payload (ESP) Header** IPv6 can provide confidentiality by encrypting the payload. The IPV6 ESP header contains a *security parameter index (SPI)* field that refers to a security association telling the destination how the payload is encrypted. ESP headers may be used end-to-end or for tunneling. When tunneling, the original IPv6 header and payload are both encrypted and jacketed by outer IPv6 and ESP headers. Near the destination, a security gateway strips away the

outer headers and decrypts the original header and payload. This encapsulation provides limited traffic flow confidentiality because a traffic analyzer may see the outer headers but not the inner encrypted header and payload.

Mobility

A growing number of Internet users work while traveling. This places primary importance on IPv6's ability to support mobile hosts, such as laptops. IPv6 introduces four concepts that are key to its support of mobile computing:

- Home Address
- Care-of Address
- Binding
- Home Agent

In IPv6, mobile hosts are identified by a home address, regardless of where they are attached at the moment. When a mobile host changes from one subnet to another, it must acquire a *care-of* address through the autoconfiguration process. The association between the home address and the care-of address is called a binding. When the mobile host acquires a care-of address, it notifies its home agent with a Binding Update message. The home agent maintains a mapping between home addresses and care-of addresses, called a *binding cache*.
A mobile host can be reached by sending a packet to its home address. If the mobile host is not connected to its home network, the home agent will forward the packet to the mobile host via its care-of address. The mobile host will then send a Binding Update message to the source node. The source node will update its binding cache and send subsequent packets directly to the mobile node via its care-of address. Only the first packet exchanged between a source node and a mobile host passes through the home agent. All subsequent packets are passed directly between the source node and the mobile host. This redirection function of IPv6 ensures scalability when supporting mobility.

The proposed use of IPv6 for cell phones may provide an example of the intended operation of a mobile host. When an IPv6 cell phone is activated to receive calls, it acquires a care-of address from the cell it is in. The cell phone notifies its home agent, which is maintained by the service provider. Calls received by the home agent are forwarded to the cell phone via the care-of address, and the cell phone sends a Binding Update message to the originating phone. The originating phone or the originating service provider updates its binding cache and sends subsequent packets directly to the cell phone via its care-of address.

If the cell phone moves into another cell, another care-of address is acquired. Binding Update messages are sent to the home agent and to the originating phone or service provider. Subsequent packets are sent directly to the cell phone via the new care-of address. Previous care-of addresses may be maintained in the binding caches to allow the cell phone to receive packets from a previous cell if warranted by changes in signal strength.

Performance

The IPv6 architecture provides advantages in network performance and scalability. These advantages include:

- **Reduced Address Translation Overhead** To overcome limitations in its address space, IPv4 permits the use of private addresses that are restricted for use only within the private network. Network address translation must be used to map private addresses to a limited pool of public addresses. This address translation represents network performance overhead. In IPv6, address translation to overcome address space limitations is unnecessary.
- **Reduced Routing Overhead** Many IPv4 address blocks (such as Class C address blocks) are allocated to users without respect to aggregation. This results in disjointed subnets, each requiring separate routing table entries. They cannot be aggregated and be represented as a single network. This greatly increases routing table sizes and routing performance overhead. In contrast, the IPv6 addresses are allocated via service providers to encourage an addressing hierarchy that reduces routing overhead.
- **Increased Route Stability** In IPv4, route flapping results when an unreliable link is repeatedly withdrawn and re-advertised. The advertising and processing of these routing changes poses a burden to the Internet backbone. In IPv6, a single provider can aggregate the routes of many networks and allow route flapping to be isolated to that provider's network. Routing changes need only to be advertised between peer routers in a provider's network.

- **Reduced Broadcasts** IPv4's Address Resolution Protocol (ARP) uses broadcasts to map link-layer addresses with network-layer addresses. IPv6 uses Neighbor Discovery to perform a similar function during the auto-configuration process without the use of ARP broadcasts.
- **Scoped Multicasts** In IPv6, a multicast address contains a *scope* field that can restrict multicast packets to the node, the link, or the organization. In IPv4, implementing similar restrictions requires the implementation of filters and private address spaces.
- **Streamlined Header** In contrast to the 12 fixed-length fields and one variable-length field in the IPv4 header, the streamlined IPv6 header has only eight fixed-length fields. To implement extended functions, extension headers can be used that need not be checked by intermediate routers. This streamlined header architecture lowers network overhead.
- **No Intermediate Node Fragmentation** In IPv4, when an intermediate node or router receives a packet that is too large to be forwarded, the router may fragment the packet. This costly function is not supported in IPv6. Instead, only the source node will perform packet fragmentation. To assist the source node, IPv6 provides a *Path MTU Discovery* function to determine the MTU size for the path from source to destination.
- **No Header Checksum** The IPv4 header provides a *checksum* field to allow for error detection at each hop in the network. To eliminate the overhead of checksum processing by each hop, IPv6 eliminates the *header checksum* field. While this may cause erroneous packets to be forwarded, the reliability of today's network links reduces that probability. Checksum verification is already performed at the source and destination by upper-layer processes such as TCP and UDP. In IPv6, checksum processing is solely the responsibility of the source and destination. This greatly reduces network overhead.

Comparing IPv6 to IPv4

IPv6 clearly differs from IPv4 is many significant ways. Let's examine the most significant differences between the two versions of the protocol. This will allow those who have worked with IPv4 to easily recognize the major differences with the new protocol version. The most significant differences are:

- Streamlined Header Format
- Flow Label
- 128-bit Network Addresses
- Elimination of Header Checksum
- Fragmentation Only by Source Host
- Extension Headers
- Built-in Security

Addressing Structure

The IPv6 unicast address is 128 bits long and is comprised of a subnet prefix and an interface identifier. For Aggregatable Global Unicast addresses, both the prefix and the interface ID are 64 bits long, as illustrated in Figure 2.6. The subnet prefix is the network number assigned to the link. The interface identifier is derived from the node's Media Access Control (MAC) address. During IPv6 address autoconfiguration, the host node supplies its own interface ID from a ROM and queries the local router or DHCPv6 server for a subnet prefix.

Figure 2.6 IPv6 Addressing Format

```
|          64 bits              |          64 bits              |
+-------------------------------+-------------------------------+
|       Subnet Identifier       |     Interface Identifier      |
+-------------------------------+-------------------------------+
```

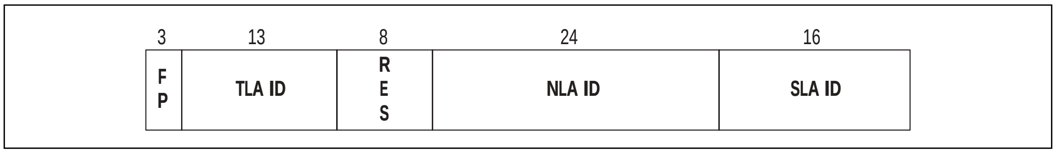
Today's MAC addresses are only 48-bits long so 16 bits in the interface ID is reserved. The IEEE has proposed a MAC address that is 64 bits long, known as EUI-64.

In contrast, the IPv4 addresses are only 32 bits long. They are also comprised of a subnet number and a host number. The IPv4 addresses are not derived from MAC addresses. Network administrators assign both the subnet numbers and the host numbers.

Address Administration

The 64-bit subnet prefix of an IPv6 Aggregatable Global Unicast address is further subdivided into five fields. These five fields are illustrated in Figure 2.7. The first field is the *format prefix (FP)* field, which identifies an Aggregatable Global Unicast address with a binary value of 001. The third field is reserved for future use.

Figure 2.7 Subnet Prefix of Aggregatable Global Unicast Address



Two fields, the *TLA ID* and the *NLA ID*, are key to understanding IPv6's support for an aggregatable addressing hierarchy. The TLA ID is the Top-Level Aggregation Identifier. IPv6 global addresses will be assigned to service providers or TLA organizations. The TLA organizations will in turn allocate addressing space to the Next-Level Aggregation (NLA) organizations. This hierarchical method of allocating address space encourages address aggregation to reduce the size of core routing tables.

In contrast, IPv4 addresses are commonly allocated by CIDR blocks. Each CIDR block is comprised of one or more Class C addresses. Each Class C block can address approximately 254 devices. Unfortunately, CIDR blocks allocated to different organizations cannot be easily aggregated, and each CIDR block may require a separate routing table entry in a core router. The issuance of CIDR blocks has caused explosive growth in the size of core routing tables.

To local network administrators, the most important field may be the *Site-Level Aggregation (SLA) Identifier*. Unlike the TLA ID and the NLA ID, the SLA ID is not usually delegated to a downstream organization with a pre-assigned value. Per RFC 2374, the SLA ID allows an organization to define its own local subnets and addressing hierarchy. The 16 bits provided by the SLA ID for subnet identifiers can support 65,535 subnets, enough for all but the largest organizations. To support even larger networks, a downstream organization may request that a lower-order portion of the NLA ID be delegated.

Herein lies a key advantage of the IPv6 addressing structure. A network administrator need not worry about allocating addresses for both subnet and host identifiers as he must in IPv4. In IPv4, both subnet and host identifiers often come from the same limited pool of available addresses. An IPv4 network administrator often "borrows" host identifier fields to number his subnets. In IPv6, the top half of the IPv6 address structure provides enough addressing space for subnet identifiers. In a separate and virtually inexhaustible field, the IPv6 host identifiers are uniquely generated for each host by a separate autoconfiguration process.

IPv6 eases the network administrator's burden in another way: Aggregatable Global Unicast addresses do not require address translation when used to access external networks such as the Internet. In IPv4, private address spaces are used when global addresses are unavailable. These private addresses must be translated to a limited set of global addresses when accessing external networks. IPv4 address translation schemes include Network Address Translation (NAT) and Port Address Translation (PAT). IPv6 virtually eliminates the need for address translation as a means of accessing external networks.

Table 2.3 illustrates the reduced address administration burden placed upon IPv6 network administrators.

Table 2.3 Address Administration Comparison

Address Administration Issues	IPv4 Private Class A Block	IPv6 Aggregatable Global Unicast
Address Length	32 bits	128 bits
Length of Pre-assigned Upstream Fields	8 bits	48 bits
Length of Delegated Addressing Fields	24 bits	80 bits
Host Identifier Length	24 – subnet bits	64 bits
Subnet Identifier Length	24 – host bits	16 bits (SLA ID)
Allocate host addresses for subnet identifiers	Yes	No
Determine subnet identifiers	Yes	Yes
Determine host identifiers	Yes	No
Address Translation Required (NAT/PAT)	Yes	No

Header Comparison

IPv6 provides a more streamlined header than that of IPv4. Five fields are eliminated, including the variable-length IPv4 *options* field. Removal of the variable-length field and other fields permits the IPv6 header to have a fixed format of 40 bytes in length. A comparison of the two types of headers is summarized in Table 2.4.

Table 2.4 Header Comparison

Header	IPv4	IPv6
Header Format	Variable	Fixed
Header Fields	13	8
Header Length	20-60 bytes	40 bytes

Continued

Table 2.4 Continued

Header	IPv4	IPv6
Address Length	32 bits	128 bits
Header Checksum	Yes	No
Fragmentation Fields	Yes	No
Extension Headers	No	Yes

To provide for additional options, IPv6 defines the following extension headers, which will be discussed in detail in the next chapter:

- Hop-by-Hop Options header
- Destination Options header
- Routing header
- Fragment header
- Authentication header
- Encapsulating Security Payload header

Feature Comparison

The IPv6 architecture contains integrated features that are not contained in IPv4. Table 2.5 contrasts the features of IPv4 and IPv6.

Table 2.5 Feature Chart

Feature	IPv4	IPv6
Anycast Address	No	Yes
Multicast Scoping	No	Yes
Security Support	No	Yes
Mobility Support	No	Yes
Autoconfiguration	No	Yes
Router Discovery	No	Neighbor Discovery
Multicast Membership	IGMP	Multicast Listener Discovery
Router Fragmentation	Yes	Source only

Anycast addresses and multicast scoping are not found in IPv4. While security is a function of upper-layer protocols in IPv4, IPv6 provides integrated security support through the use of authentication and encryption headers. For mobility, IPv6 provides the functions of the home agent, care-of addresses, and binding caches.

IPv6 provides Plug and Play functionality. A host address and prefix length can be autoconfigured rather than manually entered, and routers and neighbors can be discovered. IPv4 contains no built-in discovery function—it requires that basic IP parameters be manually configured or obtained from external sources, such as DNS servers.

For multicast membership support, IPv4 relies on augmentations such as Internet Group Management Protocol (IGMP). In contrast, IPv6 has a built-in Multicast Listener Discovery (MLD) protocol. MLD allows a router to determine which of its ports contain multicast listeners and to add or prune multicast transmissions appropriately.

IPv4 allows routers to fragment packets, which causes greater network overhead. IPv6 only permits the source node to fragment a packet. It provides a *Path MTU Discovery* function that allows the source to fragment packets efficiently. This eliminates the need for network devices to further fragment a packet along the path to its destination.

Examining IPv6 Network Architecture

The Internet has become a victim of its own success. The worldwide success of the Internet has resulted in an explosion of new users and applications, and this growth has placed new demands upon the network infrastructure and upon network administrators. In the early 1990s, the IETF began to design IPv6 with the objective of improving IPv4 to meet these new demands. The areas targeted for improvement included:

- **Address Space Depletion** The depletion of IP addresses has been foreseen for many years and many patches and extensions have been added to IPv4 to alleviate and postpone the looming crisis. Among those extensions are variable-length subnet masking (VLSM), Classless Inter-Domain Routing (CIDR), Network Address Translation (NAT), Port Address Translation (PAT), and private address spaces. IPv6 introduces a large unified addressing structure that will make many of these complex extensions obsolete.

- **Network Performance** The Internet has outgrown many features in IPv4 that now encumber network performance. Among these features are header checksums, MTU size, and packet fragmentation. IPv6 is streamlined to reduce protocol overhead.
- **Security** IPv4 was not designed with security in mind—security was considered the responsibility of higher layers in the Open Systems Interconnect (OSI) model. IPv6 provides integrated security support for encryption and authentication.
- **Plug and Play** Configuring nodes in an IPv4 network has always been complicated. Many configuration tasks are manually intensive and not practical in large networks. A case in point is the renumbering of a network when a new Internet service provider is selected. The growth of mobile computing has also added to the workload of network administrators. IPv6's autoconfiguration facilities take us a step closer to true "Plug and Play" computing.

IPv6 Communication Fundamentals

In the following sections, we will examine in further detail how communication occurs between the devices on a network and how IPv6 facilitates that communication. We will examine communication between hosts of the same subnet as well as host and router communication between subnets.

Intra-Subnet Communications

How many of us have cringed at the prospect of connecting our computers or laptops to a network for the first time? A computer must be configured to be able to communicate on a network. Likewise, a network administrator must configure his network devices to facilitate communication between hosts. Nodes at both ends of each network link must have compatible configurations. Many patches have been added to IPv4 to make the configuration process less manually intensive.

IPv6 is designed for Plug and Play. Support for autoconfiguration is built into IPv6. We will first examine stateless autoconfiguration and its role in intra-subnet communications. Later, we will discuss stateful autoconfiguration and its role in inter-subnet communications.

The keys to understanding intra-subnet communications are the following concepts:

- Stateless Autoconfiguration
- Link-Local Address
- Link-Local Prefix
- Interface Identifier
- Neighbor Solicitation message
- Neighbor Advertisement message
- Neighbor Cache

Let's imagine a subnet in a dentist's office. The subnet is comprised of a few workstations and printers. It has no routers, no connections to the Internet, and no servers to assist in the configuration process. A host on such a subnet must configure its own IPv6 address with a process known as *stateless autoconfiguration*.

When a workstation is connected to a port on the subnet, the workstation automatically configures a tentative address, known as the link-local address. This address is formed using the hardware address of the workstation's network interface. The link-local address configured by the workstation is 128 bits long and is comprised of a local-link prefix and the workstation's interface identifier. The local-link prefix is an all-zero network identifier prefaced with the hex digits, FE8. The interface ID, also known as the Media Access Control (MAC) address, resides in a ROM on the interface hardware. Today's MAC addresses are 48 bits long, but new specifications will support 64-bit MAC addresses. A typical local-link address takes the following form, where the x's indicate the 64-bit interface ID:

```
FE80:0:0:0:xxxx:xxxx:xxxx:xxxx.
```

To ensure a unique address, the workstation will send a special Neighbor Solicitation message to the newly configured address and wait one second for a reply. If no Neighbor Advertisement message is returned, the new link-local address is assumed to be unique. (Later, we will see that the Neighbor Solicitation and Neighbor Advertisement messages are also used for other functions that are part of the IPv6 Neighbor Discovery protocol.)

After verifying the uniqueness of the link-local address, the next phase is to query for neighboring routers on the network. In our example of a subnet in a dentist's office, no routers will be found. The workstation is now ready to begin communications with its neighbors.

To communicate with a destination host on the same subnet, the workstation must discover the destination's interface identifier. To do so, the workstation uses the functions provided by the IPv6 Neighbor Discovery protocol. The workstation sends a Neighbor Solicitation message to the destination and the interface identifier is returned in a Neighbor Advertisement message. This interface ID is placed in a header before the IPv6 header and transmitted on the subnet. The workstation then adds an entry to its Neighbor Cache. The entry contains the destination's IPv6 address, its interface identifier, a pointer to packets pending transmission, and a flag indicating whether or not the destination is a router. This cache will be used for future transmissions instead of sending another solicitation message.

The link-local addresses cannot be used for communications outside the local subnet. For inter-subnet communications, site-local addresses or global addresses must be used in conjunction with routers. We will
discuss inter-subnet communications in the next section.

Inter-Subnet Communications

Suppose that in the previous example, our workstation discovered that a router *did* exist on the subnet. How would the autoconfiguration process differ, and how would the workstation communicate with hosts on different subnets? To discuss inter-subnet communications, we will expound on the stateless autoconfiguration process and introduce the following concepts:

- Neighbor Discovery
- Site-Local Address
- Subnet Identifier
- Router Solicitation Message
- Router Advertisement Message
- Default Router List Cache
- Destination Cache
- Prefix List Cache
- Redirect Message
- Path MTU Discovery

During and after autoconfiguration, the workstation relies heavily on the IPv6 Neighbor Discovery protocol. The Neighbor Discovery protocol allows nodes on the same subnet to discover each other and to find routers for use as the next hop towards a destination on another subnet. The Neighbor Discovery protocol replaces the IPv4 Address Resolution Protocol (ARP), the IPv4 default gateway process, and the IPv4 redirect process.

During the autoconfiguration process, after the workstation generates a unique link-local address, it queries for a router. The workstation sends a Router Solicitation message and a router responds with a Router Advertisement message.

The presence of a router indicates that there may be other subnets connected to the router. Each subnet must have its own subnet identifier, because routing is dependent on unique subnet numbers. Host identifiers are not used for making routing decisions. The workstation address must now have a unique subnet identifier. The link-local address with its zero subnet ID is not sufficient for inter-subnet communications.

To support stateless autoconfiguration, the Router Advertisement contains a subnet identifier. Router Advertisements for each router interface contain a different subnet identifier. This identifier will be concatenated with the interface identifier to form the workstation's IPv6 address.

The workstation will discard its tentative link-local address and configure a new address that is known as the *site-local* address. The site-local address contains a 16-bit subnet ID and has the following format, where the x's indicate the 64-bit interface ID:

```
FEC0:0:0:<subnet ID>:xxxx:xxxx:xxxx:xxxx.
```

The workstation will use information from the Router Advertisement to update its caches. The subnet ID is added to the workstation's Prefix List cache. This cache will be used to determine if an address is on the workstation's subnet (on-link) or not (off-link). The router's information will be added to the Neighbor cache and Destination cache. If the router can be used as a default router, an entry will be added to the Default Router List cache.

When the workstation is ready to send a packet to a destination host, it queries the Prefix List to determine whether the destination's IPv6 address is on-link or off-link. If the destination host is off-link, the packet will be transmitted to the next hop, which is the router in the Default Router List. The workstation will then update its Destination cache with an entry for the destination host and its next hop address. If the default router selected is not the optimal next hop to the destination, the router will send a Redirect message to the source workstation with the new recommended next hop router for the destination. The workstation will then update its Destination cache with the new next hop for the destination.

The caches are maintained by each IPv6 host and are queried before solicitation messages are transmitted. The caches reduce the number of solicitation and advertisement messages that need to be sent. The caches are periodically purged of expired information, and they are constantly updated.

To facilitate inter-subnet communications, IPv6 provides another useful service, Path MTU Discovery. IPv6 does not allow routers to fragment packets that are too large to be forwarded through the next hop link or interface; only the source node may fragment a packet. Using IPv6's Path MTU Discovery service, a source node can determine the largest packet that may be sent to the destination. With this information, the source node can appropriately resize its packets prior to transmission.

Site-local addresses can only be used for communications within the site. For communications beyond the site, global addresses must be assigned with a more scalable autoconfiguration procedure.

Internetwork Communications

In stateless autoconfiguration, each node is responsible for configuring its own address and caches using its interface identifier and information provided by the distributed Neighbor Discovery protocol. In small networks, stateless autoconfiguration is advantageous for its simplicity and ease of use. Its disadvantages include relying on multicast discovery mechanisms, using address space inefficiently, and lacking in security and control over policy and access.

To facilitate communications in larger and more complex internetworks, it may be desirable to manage the autoconfiguration process using a procedure known as *stateful autoconfiguration*. During our discussion of this process, we will introduce the following concepts:

- Stateful Autoconfiguration
- Dynamic Host Configuration Protocol Version 6 (DHCPv6)
- DHCPv6 Client, Relay, Agent, Server

Stateful autoconfiguration relies on servers to provide the bulk of the configuration information, including the network information required for obtaining an Aggregatable Global Unicast address. These servers are known as Dynamic Host Configuration Protocol version 6 (DHCPv6) servers. From a network administrator's point of view, stateful autoconfiguration is more complex than stateless autoconfiguration because it requires that configuration information be entered into a DHCPv6 database. On the other hand, stateful autoconfiguration provides greater scalability when administering to large networks.

Stateful autoconfiguration can be used simultaneously with stateless autoconfiguration. For example, a node may follow the stateless procedure upon startup to obtain a link-local address. After obtaining the link-local address, it may use stateful autoconfiguration to interact with and obtain additional information from a DHCPv6 server.

To obtain configuration information, a workstation first locates a DHCPv6 server by issuing a DHCP Solicit message or by listening for a DHCPv6 Advertisement. The workstation then issues a unicast DHCPv6 Request. If a DHCPv6 server is not on the local subnet, then a DHCPv6 Relay or Agent will forward the request to a server on behalf of the workstation. The server will respond with a DHCPv6 Reply that contains configuration information for the workstation.

The use of a DHCPv6 service has several advantages:

- **Control** The DHCPv6 service controls the distribution and assignment of addresses from a central control point.
- **Aggregation** Through the thoughtful distribution of addresses, an addressing hierarchy can be built to ensure address aggregation.
- **Renumbering** When a new Internet service provider is chosen to replace the old provider, new addresses can more easily be distributed with the DHCPv6 service.
- **Security** A host registration system can be enforced with the DHCPv6 service. This registration system can selectively provide network services to registered hosts and deny access to unregistered hosts.

Designing & Planning…

Autoconfiguration

Stateless and stateful autoconfiguration can coexist in IPv6. When designing and planning for the IPv6 numbering of a site, it is often desirable to plan for both autoconfiguration methods. We'll begin by classifying various types of nodes based on their need to access the site and the Internet:

- Workstations that do not need to communicate beyond their local subnet may use stateless autoconfiguration to obtain link-local addresses.
- Workstations that need to communicate throughout the site but don't need to access the Internet can use stateless

Continued

autoconfiguration to obtain site-local addresses from their local routers.

- Workstations that need access to the Internet can obtain aggregatable global unicast addresses from a DHCPv6 server. DHCPv6 allows address distribution to be controlled from a central point.

Upper-Layer Protocol Issues

In general, the layered architecture shields the upper-layer protocols from changes in the network layers. However, there are several issues need to be addressed. Upper layer protocols that compute checksums over packets must account for changes in IPv6, including the use of 128-bit addresses, having a final destination instead of intermediate ones when the Routing header is used, and so on.

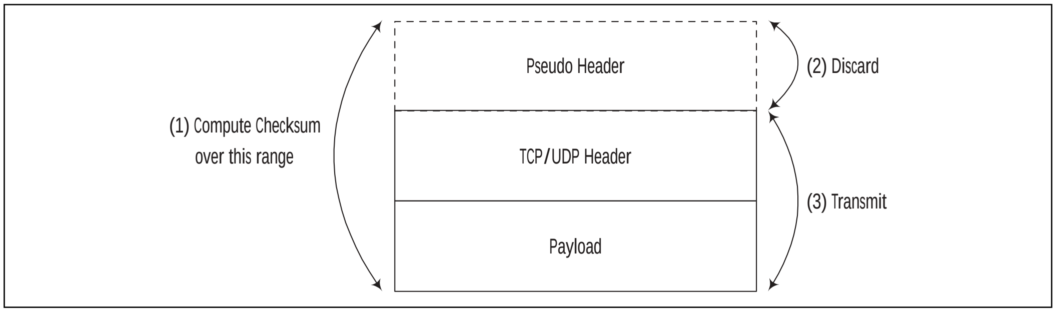
As discussed earlier, the *Time To Live* field (which behaves differently than its original definition) has been renamed the *hop limit*. Any upper-layer protocol that relies on the original meaning of the Time To Live may have to make necessary adjustments. The maximum upper-layer payload size also needs to be adjusted to reflect the length of the IPv6 header (40 bytes).

- **Upper-Layer Checksums** Currently, an upper-layer transport protocol such as TCP and UDP will concatenate a pseudo header before its payload when computing the transport layer checksum. This pseudo header contains the source and destination IPv4 addresses. For IPv6, the pseudo header must be expanded to include the larger addresses, the upper-layer packet length and the *next header* field. The checksum is computed over the IPv6 pseudo header, the TCP or UDP header, and the TCP or UDP payload. Figure 2.8 illustrates the checksum being calculated over an IPv6 pseudo header.
- **Maximum Packet Lifetimes** The IPv4 header has a *Time To Live* field that is used to determine when a packet can be discarded if it has not reached its destination. In contains either a hop count or a time in seconds. IPv6 has renamed this field as the *Hop Limit*, and the *time in seconds* measurement is no longer supported. Applications using this field for time data must be revised.
- **Maximum Upper-Layer Payload Size** The nominal IPv6 header is 40 bytes long. The nominal IPv4 header is 20 bytes long. Replacing the

IPv4 header with a longer IPv6 header will result in larger packet sizes, which may have upper-layer consequences.

- **Routing Headers & Security** The IPv6 routing header extension contains the intermediate nodes that the packet must traverse on the way to its destination. When a packet with a routing header is received by the destination, it should not assume that the reverse path to the source is appropriate. In fact, responding along the reverse path may facilitate certain types of security breaches.
- **Domain Name System (DNS)** The DNS is a distributed database system that defines a hierarchical naming convention for host nodes and maps these host names to IP addresses. For example, www.syngress.com maps to IPv4 address 216.238.176.55. IPv6 enhancements to DNS include new record types with the 128-bit IPv6 address and a new service that can return a hostname when given its IPv6 address.
- **Application Programming Interface (API)** Application programs written for IPv4 must be converted to use APIs written for IPv6. The application programs must contain new data structures for the longer IPv6 addresses. Functions that manipulate IPv4 addresses must be substituted with functions that can manipulate IPv6 addresses.

Figure 2.8 TCP/UDP Checksum Calculation



Understanding ICMPv6

The Internet Control Message Protocol version 6 (ICMPv6) is a key part of the IPv6 architecture. ICMPv6 performs the feedback functions necessary to ensure the smooth operation of IPv6 processes. These functions include:

- Packet Processing Error Reporting
- Diagnostics
- Neighbor Discovery
- Multicast Membership Reporting

ICMPv6 is streamlined to delete ICMPv4 functions that are no longer used, and it combines the functions of three separate IPv4 protocols: ICMPv4, Internet Group Membership Protocol (IGMP), and Address Resolution Protocol (ARP). ICMPv6 messages can be divided into *error messages* and *information messages*.

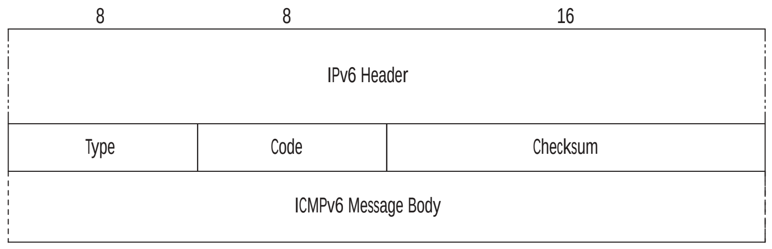
Error Messages

ICMPv6 issues error messages pertain to packet processing. These error messages include:

- **Destination Unreachable** The source host or a router generates this message when a packet cannot be delivered for any reason other than congestion.
- **Packet Too Big** The router generates this message when a packet cannot be forwarded because it is larger than the MTU of the next hop link. The MTU of the next link is returned in the message. This message is used by the Path MTU Discovery function.
- **Time Exceeded** The IPv6 header contains a Hop Limit that is decremented by each router that forwards the packet. When this hop limit reaches zero, the router discards the packet and returns a Time Exceeded message to the source node. This function is the basis of the traceroute function; it traces the route to a destination by sending packets towards the destination with incremental hop limits. The Time Exceeded messages returned are used to identify the routers along the path.
- **Parameter Problem** When a problem with some part of an IPv6 header keeps a router from successfully processing the packet, the packet is discarded and the router returns a Parameter Problem message to the source node.

Each ICMPv6 error message includes three fixed-length fields plus a variable-length message body. Figure 2.9 illustrates the format of the error message.

Figure 2.9 ICMPv6 Error Message



Type	ICMPv6 Error Message
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem

Informational Messages

ICMPv6 can also report informational messages. These include:

- **Diagnostic Messages** Diagnostic messages include the echo request and reply. When a destination receives the echo request, it returns a reply. This echo request and reply are used to implement the ping diagnostic function. The ping function is important for determining whether or not a particular destination is connected to the same network as the source.
- **Multicast Listener Discovery (MLD) Messages** These messages enable a router to discover neighboring nodes that wish to receive multicast packets, and what multicast addresses are of interest to them. A router transmits an MLD Query to learn whether a multicast address (or addresses) has listeners on a link. Nodes respond affirmatively with an MLD Report message. When a node ceases to listen to a multicast address, it sends an MLD Done message.
- **Neighbor Discovery Messages** ICMPv6 provides messages necessary for the Neighbor Discovery protocol. These messages include router solicitation and advertisement, neighbor solicitation and advertisement, and redirect. These messages include information options such as the source link-layer address, the target link-layer address, prefix information, the redirected header, and the MTU size.

ICMPv6 informational messages have the same format as the ICMPv6 error message displayed in Figure 2.9. The *type* field values for informational messages range from 128 to 255. Table 2.6 shows some of the common *type* fields for ICMPv6 informational messages.

Table 2.6 ICMPv6 Informational Messages

Type Field Value	ICMPv6 Informational Message
128	Echo Request
129	Echo Reply
130	Multicast Listener Query
131	Multicast Listener Report
132	Multicast Listener Done
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect

Understanding Neighbor Discovery

IPv6's Neighbor Discovery protocol is used to obtain information that facilitates the packet forwarding process. The information gathered by the Neighbor Discovery protocol can be used for:

- Next Hop Determination
- Address Resolution
- Prefix Discovery
- Parameter Discovery
- Redirection

Five ICMPv6 messages are used in the Neighbor Discovery protocol. We will discuss Neighbor Discovery with respect to these five messages.

Router Solicitation and Advertisement

During the autoconfiguration process, after the workstation generates a unique link-local address, it queries for a router. The workstation sends a Router Solicitation message and listens for a Router Advertisement message.

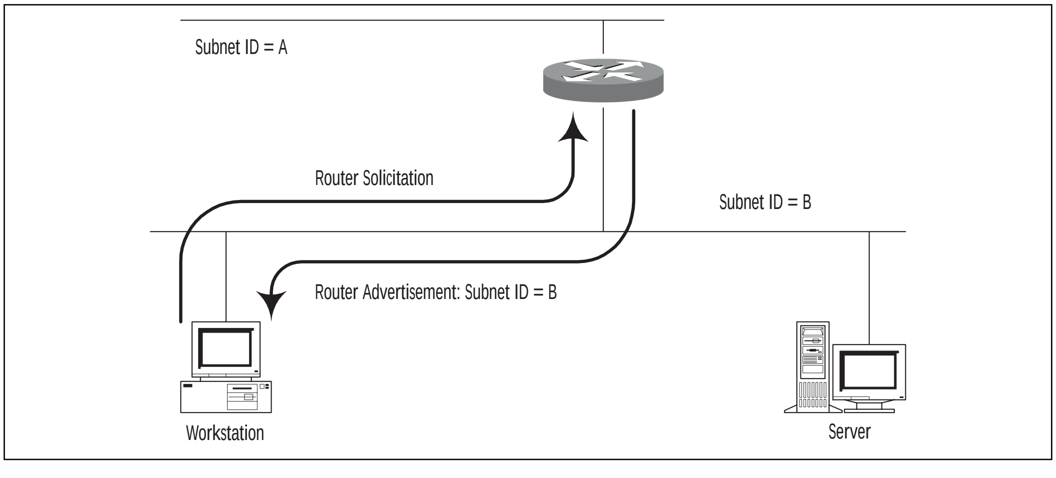
The presence of a router indicates that there may be other subnets connected to the router. Each subnet must have its own subnet identifier because routing is dependent on unique subnet numbers. Host identifiers are not used to make routing decisions. The workstation address must now have a unique subnet identifier. The link-local address, with its zero subnet ID, is not sufficient for inter-subnet communications.

The Router Advertisement contains a network number or prefix. The prefix may contain an aggregatable global unicast prefix or simply a subnet identifier. Router Advertisements for each router interface contain different prefixes. This prefix will be concatenated with the interface identifier to form the workstation's IPv6 address.

The workstation uses information from the Router Advertisement to update its caches. The subnet ID is added to the workstation's Prefix List cache. This cache will be used to determine if an address is on the workstation's subnet (on-link) or not (off-net). The router's information will be added to the Neighbor cache and the Destination cache. If the router can be used as a default router, an entry will be added to the Default Router List cache.

Figure 2.10 illustrates a workstation during the autoconfiguration process. The workstation solicits the local router and receives the subnet identifier it needs to complete its host IPv6 address.

Figure 2.10 Router Discovery

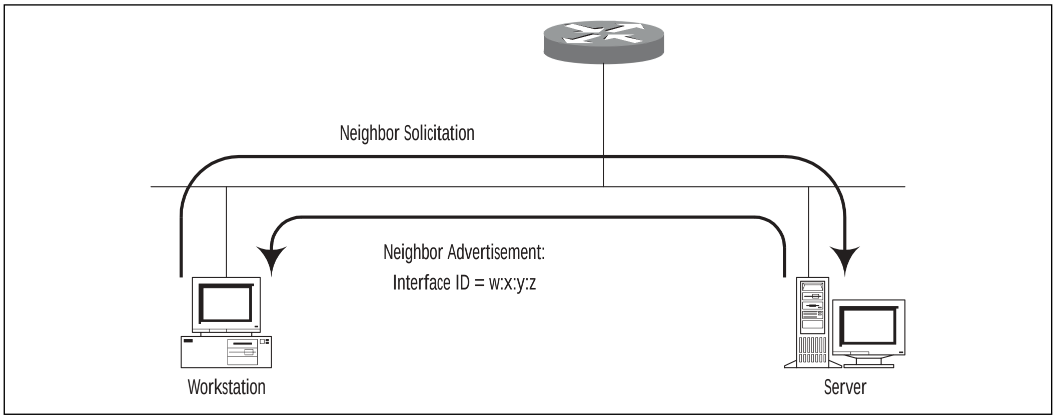


Neighbor Solicitation & Advertisement

To communicate with a destination host on the same subnet, the workstation must discover the destination's interface identifier. To do so, the workstation uses the functions provided by the IPv6 Neighbor Discovery protocol. The workstation sends a Neighbor Solicitation message to the destination, and the interface identifier is returned in a Neighbor Advertisement message. This interface ID is placed in a header before the IPv6 header and transmitted on the subnet. The workstation then adds an entry to its Neighbor Cache containing the destination's IPv6 address and interface identifier, a pointer to packets pending transmission, and a flag indicating whether the destination is a router. This cache will be used for future transmissions (instead of sending duplicate solicitation messages).

Figure 2.11 illustrates how Neighbor Solicitation and Advertisement messages play a key role in the Neighbor Discovery process.

Figure 2.11 Neighbor Discovery



Redirect Message

Routers issue the Redirect message to inform other nodes of a better first hop to the destination. A node can be redirected to another router on the same link. Here's how redirection works in the packet processing process:

When the workstation is ready to send a packet to a destination host, it queries the Prefix List to determine whether the destination's IPv6 address is on-link or off-link. If the destination host is off-link, the packet will be transmitted the next hop, which is the router in the Default Router List. The workstation will then update its Destination cache with an entry for the destination host and its next hop address. If the default router selected is not the optimal next hop to the destination, the router will send a Redirect message to the source workstation with the new recommended next hop router for the destination. The workstation will then update its Destination Cache with the new next hop for the destination.

Message Options

Neighbor Discovery messages may contain additional information options. These options include:

- **Source Link-Layer Address Option** This option contains the link-layer address of the *source* of the message. It is used in Router Solicitation, Router Advertisement, and Neighbor Solicitation messages.
- **Target Link-Layer Address Option** This option contains the link-layer address of the *target* of the message. It is used in Neighbor Advertisement and Redirect messages.
- **Prefix Information Option** This option contains prefixes for address autoconfiguration. It is used in Router Advertisements.
- **Redirected Header Option** This option contains all or part of the packet that is being redirected. It is used in Redirect messages.
- **MTU Option** This option contains the MTU size of the link. It is used in Router Advertisements.

Summary

IP and its extensions have withstood the test of time over the last three decades, during which we saw an explosion of new users and new applications. Today, its success has prompted its re-examination. The Internet Engineering Task Force (IETF) has designed the next version of IP to meet the growing needs of the Internet. IPv6 is much more than simply an extension of IPv4 with a larger address space—its architecture was designed to provide easier administration and greater performance, security and mobility.

IPv6 takes us closer to Plug and Play computing and greatly eases the burden put upon network administrators. IPv6's stateless autoconfiguration allows subnet communications literally "out of the box." The Neighbor Discovery protocol automatically solicits and caches information needed for packet processing. DHCPv6 uses IPv6's hierarchical addressing structure to allow even the daunting task of site renumbering to proceed with relative ease. ICMPv6 provides an effective set of diagnostic and information-gathering functions.

IPv6's Plug and Play support extends to mobile computing: Mobile hosts acquire care-of addresses as they travel, and home agents bind the care-of addresses to home addresses to ensure that mobile users retain connectivity away from home. Redirection allows a source host to communicate directly with a mobile host.

IPv6 provides integrated security support with two extension headers. The Authentication Header's *Integrity Check Value (ICV)* field supports both connectionless integrity and data origin authentication. The *sequence number* field can be used to detect packet replay attacks. Encrypted payloads can be transmitted with the Encrypted Security Payload (ESP) Header. The header's *SPI* field contains a security association that tells the destination how the payload is encrypted. ESP headers may be used end-to-end or for tunneling.

The address space of IPv6 is not only larger than IPv4's address space; it provides an aggregation hierarchy to simplify address administration and to reduce core routing table sizes. The use of MAC addresses to form the host portion of the address increases the probability of uniqueness during autoconfiguration.

IPv6's design enhances network performance. Its larger address space eliminates the need for network and port address translation, reducing overhead. Core routing overhead is significantly reduced by the ability to aggregate addresses and reduces the size of routing tables. Aggregation enhances core router stability by allowing route flapping to be isolated to a provider's network. IPv6's streamlined header architecture eliminates header checksums and router fragmentation.

Scoped multicast traffic and the introduction of anycast addresses further reduce multicast traffic.

The layered network model generally protects upper-layer protocols from an expensive transition to IPv6. Obvious transitional issues related to the upper-layer protocols, which were discussed earlier, are solvable.

This author's hope is that this book will help you to understand the solutions and benefits provided by IPv6 and encourage you to explore IPv6 further.

Solutions Fast Track

Understanding the Benefits of IPv6

☑ The IP address size is greatly increased.

☑ A "top-down" allocation plan supports addressing hierarchies, enabling greater address aggregation.

☑ Host addressing is simplified by using the MAC address to form the host portion of the IP address.

☑ Plug and Play functionality is supported by simpler autoconfiguration of IP addresses.

☑ Multicast routing is scoped to provide greater scalability.

☑ Anycast addresses increase routing efficiency.

☑ Streamlined headers lower routing overhead.

☑ Security is built in to support IP security at a lower level in the OSI stack.

☑ Mobility is enhanced with greater scalability and ease.

☑ Performance is enhanced with greater efficiency and streamlining.

Comparing IPv6 to IPv4

☑ IPv6 enlarges the addressing structure without address translation and private address spaces.

☑ IPv6 address administration is "top-down" to allow for greater address aggregation.

- ☑ IPv6 headers are simpler and more efficient thanks to a fixed size and fewer fields.
- ☑ IPv6 features not included in IPv4 include enhanced support for multicasting, security, mobility, and discovery.

Examining IPv6 Network Architecture

- ☑ The impending address space depletion provided the initial impetus for developing IPv6.
- ☑ A streamlined architecture improves network performance.
- ☑ Security on the IP level is built into IPv6.
- ☑ Plug and Play has become a reality with IPv6.
- ☑ Intra-subnet communications can be autoconfigured in a Plug and Play fashion.
- ☑ Inter-subnet communications can be set up by the stateless autoconfiguration function.
- ☑ Internetwork communications can be set up by stateful autoconfiguration.

Upper-Layer Protocol Issues

- ☑ Upper-layer checksums need to be computed over a larger IPv6 pseudoheader.
- ☑ The maximum packet lifetime is expressed as an IPv6 Hop Limit, not a Time To Live.
- ☑ Maximum upper-layer payload sizes may need revision to allow for the larger IPv6 header.
- ☑ The IPv6 Routing Header must be used with care to avoid the security risks of source routing.
- ☑ The Domain Name Service (DNS) is being enhanced to support IPv6.
- ☑ Application Programming Interfaces (APIs) are being developed to support the larger IPv6 addresses.

Understanding ICMPv6

☑ There are four types of IPv6 error messages:

- Destination Unreachable messages arise when a packet cannot be delivered for any reason other than congestion.
- Packet Too Big messages arise when a packet exceeds the MTU size of the outgoing interface.
- Time Exceeded messages arise when a packet exceeds its Hop Limit.
- Parameter Problem messages arise when a router cannot process an IPv6 header.

☑ There are three types of Informational Messages.

- Diagnostic messages include the echo request and reply messages.
- Multicast Listener Discovery (MLD) messages are used to discover multicast listeners and the multicast addresses that are of interest to them.
- Neighbor Discovery messages support the Neighbor Discovery protocol.

Understanding Neighbor Discovery

☑ Router Solicitation and Advertisement messages are used to discover local routers and to advertise router and subnet information.

☑ Neighbor Solicitation and Advertisement messages are used to discover neighbors and to advertise their addresses.

☑ A Redirect message is used to inform a node of a better first hop to a destination.

☑ Message Options are used to include additional information in Neighbor Discovery messages.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: Under IPv6, if my organization were to change network service providers, would we have to renumber our host IP addresses?

A: No, the 64-bit host portion of your IPv6 address will remain the same. Only the 64-bit network number of your IPv6 address will change. The network portion of your IPv6 address is provided by a host configuration server. The server's network numbering program will change, but the host numbering will remain the same.

Q: What is the core set of RFCs specifying IPv6 header and extension headers?

A: Newer RFCs may render these obsolete, but most of the information in this chapter is based on the following RFCs:

- **RFC 2374** An IPv6 Aggregatable Global Unicast Address Format
- **RFC 2460** Internet Protocol, Version 6 (IPv6) Specification
- **RFC 2402** IP Authentication Header
- **RFC 2406** IP Encapsulating Security Payload Header

Q: What is the implementation status of IPv6?

A: IPv6 is currently being developed for many host systems and routers, including 3Com, Cisco Systems, Digital, IBM and Microsoft. A preview version of Windows 2000 that supports IPv6 can be downloaded from this URL: http://msdn.microsoft.com/downloads/sdks/platform/tpipv6.asp.

Q: How should my organization transition from IPv4 to IPv6?

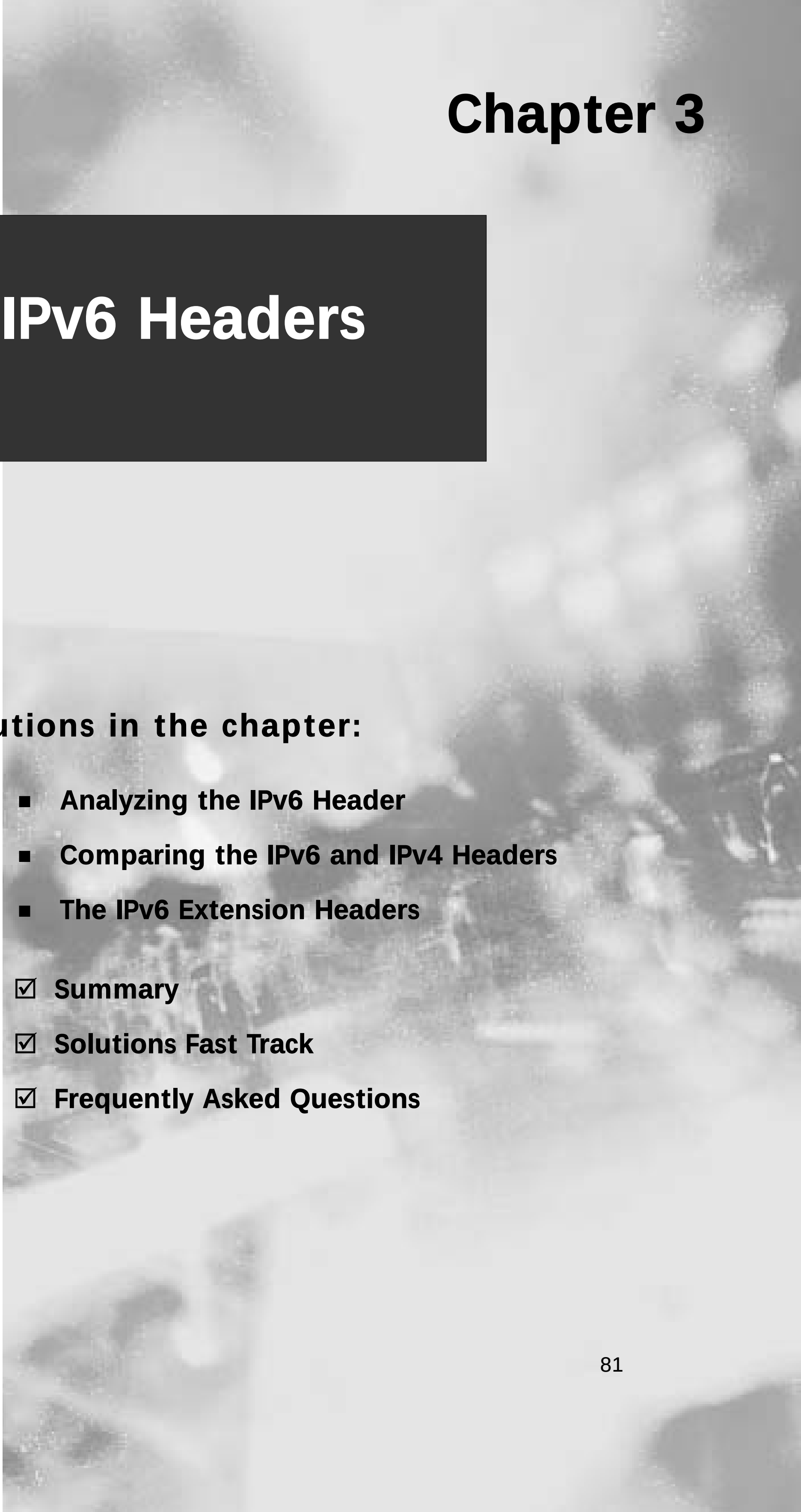
A: Strategies have been devised to ease the difficulty of transitioning from IPv4 to IPv6. These strategies involve dual stack approaches and tunneling approaches to provide simultaneous support for both IPv4 and IPv6 during the transition phase. There is ongoing research aimed at making this transition as easy as possible.

Q: What happened to IPv5?

A: Version 5 was assigned to an experimental protocol, the *Streams Protocol*, known as ST2. ST2 was designed to carry real-time traffic in parallel with IP.

Q: Will IPv6 be more complicated to implement than IPv4?

A: IPv6 was designed with autoconfiguration and Neighbor Discovery functions that will make it much easier to install and implement.



Chapter 3

The IPv6 Headers

Solutions in the chapter:

- **Analyzing the IPv6 Header**
- **Comparing the IPv6 and IPv4 Headers**
- **The IPv6 Extension Headers**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

Introduction

This chapter examines the purpose and use of the IPv6 header within the IPv6 protocol. The IPv6 header can be divided into a basic header that appears in every IPv6 packet and several extension headers. Each extension header has its own function or functions and is not required in an IPv6 packet. Examples of these headers are the *fragmentation* header and the *authentication* header.

The basic IPv6 header is only 40 bytes in length. Considering that 32 bytes are used for the source and destination IPv6 addresses, this only leaves eight additional bytes in the header. This was clearly by design—the creators of IPv6 wanted to keep the standard header small and fixed in size. Keeping it small reduces the overhead on each packet, and keeping it fixed in size makes it easier for all IPv6 nodes to process the header.

The IPv6 header can have optional extension headers. None of these are required, but an IPv6 packet can have one or several of them. Each header serves a general purpose, and many have option types that can serve more specific purposes. This is similar to the way that Internet Control Message Protocol (ICMP) in IPv4 is used for general control, but has many codes for specific control functions.

When examining the IPv6 header, we can treat the header as if it follows the Open System Interconnection (OSI) reference model. In the OSI model each layer (Layer 2, Layer 3, Layer 4, etc.) has its own header. The headers are sequenced in order, starting with Layer 2, with each layer indicating the exact type of header that follows. For example, the Layer 2 header indicates that the Layer 3 header is IP. The IP header indicates that the Layer 4 header is TCP, and so on.

The basic IPv6 header has a standard format that will always appear in an IPv6 packet, much like the Ethernet (Layer 2) header in a packet. The header then has a field to indicate what comes next—an IPv6 extension header, the next protocol (TCP), or something else. This is very similar to the way an Ethernet header indicates that an IPv4 header follows. An IPv6 extension header likewise has a *next header* field. This will indicate what comes next—another IPv6 extension header or the TCP header (or other upper-layer protocol). This is similar to the way the IPv4 header indicates that a TCP header follows.

Furthermore, an IPv6 router or node starts at the beginning of an IPv6 packet and works in sequence though each header (for however many headers need to be processed). It first examines the IPv6 header, then each extension header (if any), then the upper-layer protocol header. This is the exact same way that nodes process packets in the OSI model. First Layer 1/Layer 2 is processed, then Layer 3, then Layer 4, and so on.

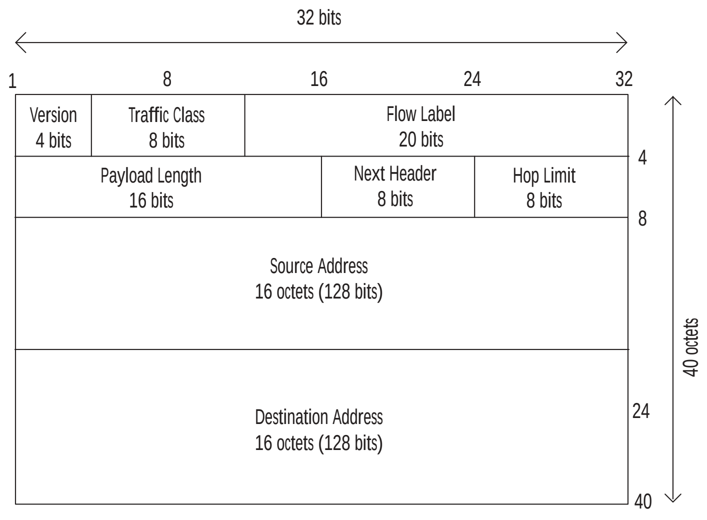
When the format of the IPv6 header was created, the IPv4 header was used as a template. Some fields, such as *version* and *payload length*, were maintained from IPv4 to IPv6. Other fields, such as *Time To Live (TTL)*, were maintained from IPv4 to IPv6, but renamed to reflect their actual use ("*hop limit*"). Some, such as *identification* and *fragment offset*, were changed to IPv6 extension headers, and still others, including *header checksum* and *options*, were dropped completely from the IPv6 header.

Analyzing the IPv6 Header

The IPv6 header is fixed in length and aligned at 8-octet boundary, unlike the IPv4 header, which is of variable length and aligned at 4-octet boundary. Most modern computer architectures are optimized to read eight octets at a time. Thus, the length of the IPv6 header (or extension headers) is designed to be a multiple of eight octets for 8-octet alignment. With a fixed IPv6 header, a router can efficiently process a packet. For instance, a router must decide if there are any options in an IPv4 packet by reading the *header length* field. Processing a variable-length header can lead to inefficient router implementation.

The changes from the IPv4 header to the IPv6 header will be covered in the next section. In this section, we will describe each field in the IPv6 header and its intended role. Figure 3.1 shows the format of an IPv6 header.

Figure 3.1 The IPv6 Header



The IPv6 header stores the information necessary to route and deliver packets to their destination. The headers are processed by each node along the path. The first 4-bit field, *version*, indicates the version of the Internet Protocol being used, and its value is 6 for IPv6. This field is necessary because it allows both protocols to coexist on the same segment without conflicts.

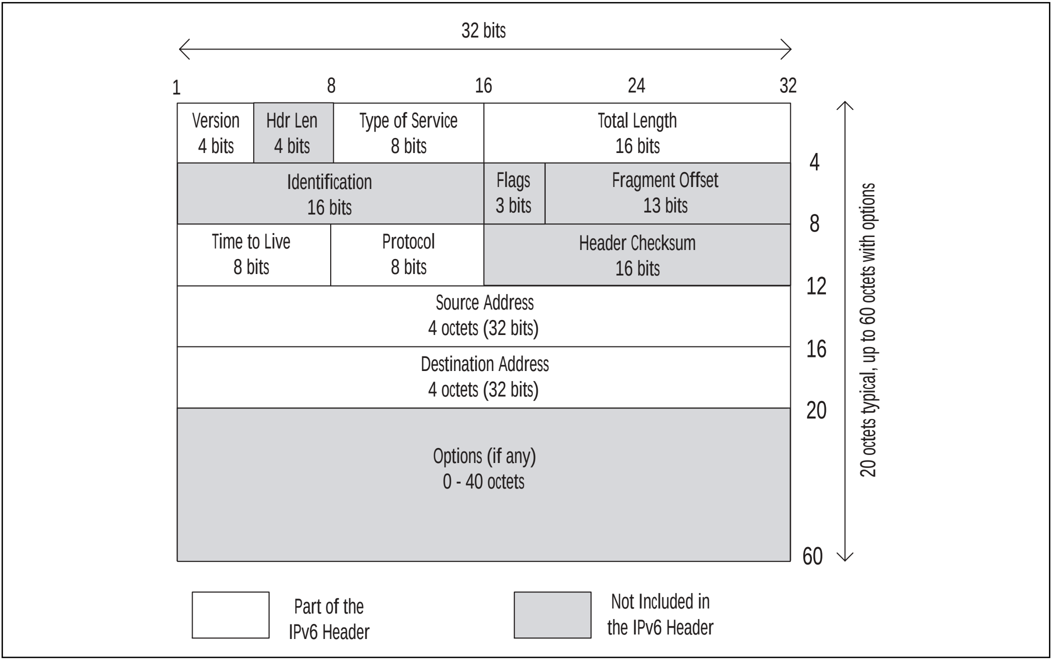
The next two fields, *traffic class* and *flow label*, are used to provide differentiated services and support applications requiring special per-flow handling. The 8-bit *traffic class* field can be used to provide differentiated services based on the nature of the data being transmitted. This field is similar to the intended use of the *type of service* field in the IPv4 header. For instance, an organization may set up its network to prioritize network traffic based on applications or source and destination information, and hosts and/or routers can use the *traffic class* field to differentiate the priority. The values and the exact use of this field are yet to be determined. The flow label, in combination with source and destination addresses, can uniquely identify a flow that requires special handling by intermediate routers. When a router identifies a flow for the first time, it remembers the flow and any special handling this flow requires. Once per-flow handling has been set up, the processing of subsequent packets belonging to this flow can be shorter than processing individual packets.

The 16-bit *payload length* field, similar to the *total length* field in the IPv4 header, indicates the length of the packet, not including the length of the IPv6 header. The 8-bit *Next Header* field is used to indicate the next header following the IPv6 header. The intended use of this field is identical to that of the *protocol* field in the IPv4 header. The *hop limit* can be used to limit the number of intermediate hops a packet is allowed to visit, which can prevent packets from being circularly routed in a network. (In IPv4, the *TTL* field has been used to prevent packets from being routed circularly—the new name for this field was chosen to more accurately reflect accurately its purpose.) As in IPv4 headers, IPv6 headers contain *source* and *destination* IP addresses. Unlike IPv4 nodes, IPv6 nodes use 128-bit addresses.

Comparing the IPv6 and IPv4 Headers

The IPv6 header shares some similarities with its predecessor, the IPv4 header. We will review the IPv4 header and discuss its similarities and differences to the IPv6 header. Figure 3.2 illustrates the format of an IPv4 header.

Figure 3.2 The IPv4 Header



Let's take a look at each of the fields in an IPv4 header and how they relate to the IPv6 header:

Version The first 4-bit *version* field in the IPv4 header is used to indicate the current version of the IP being used. The same field is used in the IPv6 header and is necessary in order to make IPv6 backward compatible.

Header Length The 4-bit *header length* field is necessary for the IPv4 header to indicate the length of the header since the total length of the IPv4 header is variable between 20 and 64 bytes, depending on the presence and the length of options in the *option* field. However, this field is not necessary in an IPv6 header, because an IPv6 header is a fixed length of 40 bytes.

Type of Service (ToS) The intent of the *type of service* field in IPv4 is similar to that of the *traffic class* field in the IPv6 header. Nevertheless, this field has not been widely accepted or used in IPv4 implementations.

Total Length This field indicates the entire length of the IP portion of the packet. It only includes the IP portion, so it does not include parts of a packet that are not related to IP, such as the Ethernet header or the Frame Check Sequence (also used on Ethernet packets), for example. IPv6 uses a similar field called *payload length*.

Identification, Flags, Fragment Offset The next three fields in the IPv4 header, the *identification*, *flags* and *fragmentation offset* fields, are all related to the handling of fragmentation and the reassembly of packets. In IPv4, an intermediate hop may further fragment a packet when the maximum transmission unit (MTU) on the outgoing link is smaller than the size of the packet that is to be transmitted on that link. Unlike IPv4, fragmentation processing in IPv6 takes place only at the source node, using a path MTU. Further, information related to fragmentation is encoded in the *fragmentation* header as an extension header in an IPv6 packet. Therefore, these three fields are not necessary in the IPv6 header.

Time To Live (TTL) In the original design of IPv4, the *TTL* field was used to indicate the number of seconds to live in a network, thus preventing packets from being circularly routed if a circular route existed in a network. However, in implementations, this field has been used to limit the number of hops the packet is allowed to visit. At each hop, a router decrements this field, and when it reaches zero, the packet is removed from the network. In IPv6, this field has been renamed *hop limit*, a more accurate description of its implementation.

Protocol The *protocol* field, which is used to indicate the next protocol (header) following the IPv4 header, is similar to the *next header* field in the IPv6 header.

Header Checksum The *header checksum* field is used to maintain the integrity of the IPv4 header. However, the higher layer calculates the checksum again for the entire packet, making this field redundant. Therefore, this field is not used in IPv6 header. If applications require a higher degree of integrity, they can achieve it through appropriate use of the Authentication and Encapsulating Security Payload extension headers.

Source and Destination Address The *source* and *destination* fields in IPv4 header remain the same in IPv6, except that the IPv4 node addresses are 32 bits and the IPv6 node addresses are 128 bits.

Options The use of options in IPv4 implies that each intermediate node in the path needs to examine the *options* field in the IPv4 header, although the options may be pertinent only to the destination node. This leads to inefficient router performance when options are used. In IPv6, optional information is encoded in extension headers.

The IPv6 Extension Headers

Extension headers, placed between the IPv6 header and the upper-layer protocol header, are used to carry optional Internet-layer information in a packet. An IPv6 packet may carry zero, one, or more extension headers. The *Next Header* field in the IPv6 header and extension headers is used to indicate which extension header or upper-layer protocol header follows the current header.

NOTE

Table 3.1 provides the Next Header value and the corresponding headers. Except for the *Hop-by-Hop Options* header, the Next Header value appears in the immediately preceding header. When the *Hop-by-Hop Options* header is used, it must follow immediately after the IPv6 header. Therefore, the Next Header value of zero can appear only in IPv6 header.

Table 3.1 Next Value Headers

Next Header Value	Next Header
0	Hop-by-Hop Options header
4	Internet Protocol
6	Transmission Control Protocol
17	User Datagram Protocol
43	Routing header
44	Fragment header
45	Inter Domain Routing Protocol
46	Resource Reservation Protocol
50	Encapsulating Security Payload
51	Authentication header
58	Internet Control Message Protocol
59	No next header
60	Destination Options header

When a TCP header immediately follows an IPv6 header without an extension header, the value of the *Next Header* field in the IPv6 header indicates that the following header is a TCP header. When a packet using TCP as its upper-layer protocol carries one extension header, the *Routing* header, this extension header is placed between the IPv6 header and the TCP header. The *Next Header* field in the IPv6 header indicates that the Routing header follows the IPv6 header and the *Next Header* field in the Routing header indicates that the TCP header immediately follows the Routing header. The Next Header value of 59 indicates that there is no extension or upper-layer protocol header following the current header.

A full implementation of IPv6 includes the following extension headers: Hop-by-Hop Options, Routing (Type 0), Fragment, Destination Options, Authentication, and Encapsulating Security Payload. The recommended ordering of extension headers when multiple extension headers are present in a packet is as follows:

- IPv6 header
- Hop-by-Hop Options header
- Destination Options header (to be processed by all destination nodes appearing in the routing header)
- Routing header
- Fragment header
- Authentication header
- Encapsulating Security Payload header
- Destination Options header (to be processed only by the final destination of the packet)
- Upper-layer header

Except for the Destination Options header, each extension header should appear no more than once in a packet. The Destination Options header contains information to be processed by the final destination node. When the Routing header is present, an additional Destination Options header may be used for options to be processed by all nodes listed in the Routing header; in this case, there will be at most two occurrences of Destination Options headers in an IPv6 packet.

When an IPv4 packet carries an option that is applicable only to its destination node, all intermediate nodes must examine and process the packet before forwarding, thus impacting the performance of the forwarding nodes.

WARNING

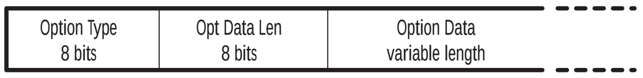
Most often, routers are implemented such that packets containing options are handled after packets without options. For this reason, the use of options is discouraged in IPv4 networks.

Except for the Hop-by-Hop Options header, extension headers are examined or processed only by the destination node (or nodes, in the case of multicast) of the packet. Thus, an IPv6 packet may carry optional information applicable only to its destination node without impacting the performance of the intermediate nodes. The Hop-by-Hop Options header can be used to carry optional information that needs to be examined or processed at all intermediate nodes.

The value of the *Next Header* field in the current header determines the next action to be taken, and the semantics of the current extension header determine whether to continue processing the next header. Thus extension headers must be examined in the order they appear in a packet. When a node receives an unrecognized Next Header value in a packet, it discards the packet and sends an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1—*unrecognized Next Header type encountered*. Because the Hop-by-Hop Options header must immediately follow the IPv6 header, a Next Header value of zero in any header other than IPv6 header will be treated as a packet with an unrecognized Next Header value.

Currently, the Hop-by-Hop Options header and the Destination Options header carry a variable number of options encoded in Type-Length-Value (TLV) format, as seen in Figure 3.3.

Figure 3.3 TLV-Encoded Option Format



Option Type	8-bit identifier of the type of option
Opt Data Len	8-bit unsigned integer. Length of the Option Data field of this option, in octets.
Option Data	Variable-length field. Option-Type-Specific data.

The Option Type identifiers are encoded in such a way that the highest-order two bits specify the action to be taken when the processing node does not recognize the Option Type, and the third highest bit specifies whether or not the

Option Data of that option can change en route to the packet's final destination. For instance, when a node encounters an unknown Option Type value of 130 (1000 0010), the highest-order two bits indicate that the node must discard the packet and send an ICMP Parameter Problem, Code 2, message to the source of the packet. Table 3.2 describes the encoding of Option Type.

Table 3.2 Option Type Encoding

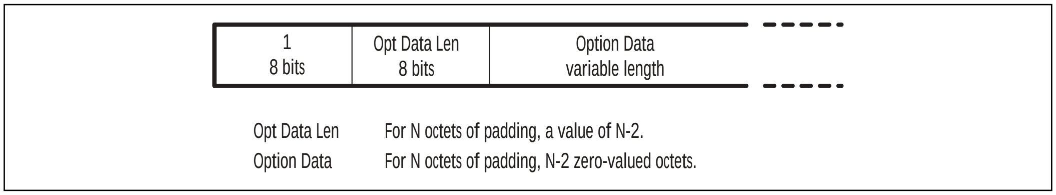
Highest-order two bits	Action to be taken
00	Skip over this option and continue processing the header.
01	Discard the packet.
10	Discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.
11	Discard the packet and, only if the packet's Destination Address was *not* a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

Some Option Type values may change as the packet progresses through the route to its destination. The third highest-order bit of the Option Type is used to indicate whether its data value can be changed en-route or not. The third highest-order bit is zero when Option Data does not change en-route, and one when it may change. This can create problems for an authentication service, since its purpose it to assure the contents of the packet do not change. When the Authentication header is used, the source of the packet computes the authenticating value over the packet and places the result in the Authentication header. For an Option Type whose Option Data may change en route, the Option Data is treated as a set of zero-valued octets when computing the packet's authenticating value.

As stated before, extension headers are designed to be multiples of eight octets in length. To ensure that the end of the *option data* field is aligned with the 8-octet boundary, specific Option Types may be associated with alignment requirements in the form $xn+y$, indicating that the Option Type must appear at an integer multiple (n) of x octets from the start of the header, plus y octets. For instance, a $4n+2$ alignment requirement indicates that the Option Type must start at any 4-octet offset from the start of the header, plus two octets, such as 2, 6, 10, 14, and so on.

Two padding options, the Pad1 option and the PadN option, may be used to force headers containing options to be multiples of eight octets in length. The Pad1 option is used to insert one zero-valued octet of padding, and the PadN option is used to insert more than one octet of padding. The format of the PadN option is shown in Figure 3.4. To insert two octets of padding (Pad2), one octet with the value of one and one octet (*option data length* field) with the value of zero can be used. The Pad2 option is a special case in that there is no Option Data, or Option Data of zero length is used.

Figure 3.4 PadN Option Format

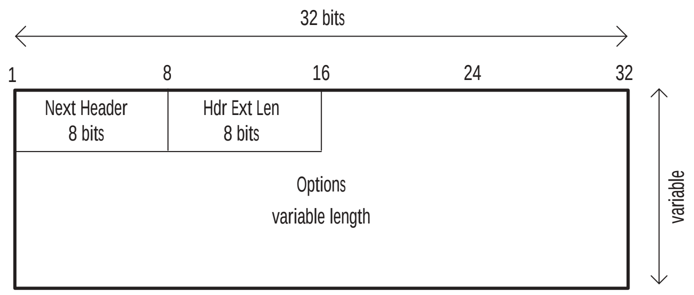

1
8 bits
Opt Data Len
8 bits
Option Data
variable length
Opt Data Len
For N octets of padding, a value of N-2.
Option Data
For N octets of padding, N-2 zero-valued octets.

Hop-by-Hop Options Header

The Hop-by-Hop Options header, identified by a Next Header value of zero in the IPv6 header, carries optional information that must be processed by every node along a packet's delivery path. For instance, it may be necessary for a router to examine and process a packet containing control messages for new protocols, such as *RSVP*. The use of the Hop-by-Hop Options header allows routers to selectively examine packets for special handling, if necessary. The format of the Hop-by-Hop Options header is shown in Figure 3.5. Note that the *header extension length* field is the length of the Hop-by-Hop Options header in 8-octet units, not including the first eight octets. In other words, when the length of the TLV-encoded option(s) is less than or equal to six octets, the *header extension length* field is zero. Examples of Hop-by-Hop Options include the Router Alert Option and Jumbo Payload Option.

A call set-up control message using RSVP protocol needs special provisioning at each router along the path of the connection. Using the Router Alert Hop-by-Hop option, routers can provide special handling. Processing a Hop-by-Hop option may result in processing an upper-layer protocol such as RSVP. RSVP is a protocol used for end-to-end flow control and is detailed in numerous RFCs. Using RSVP for flow control and quality of services (QoS) is a complex subject that is just briefly mentioned here to illustrate header processing.

Figure 3.5 Hop-by-Hop Options Header

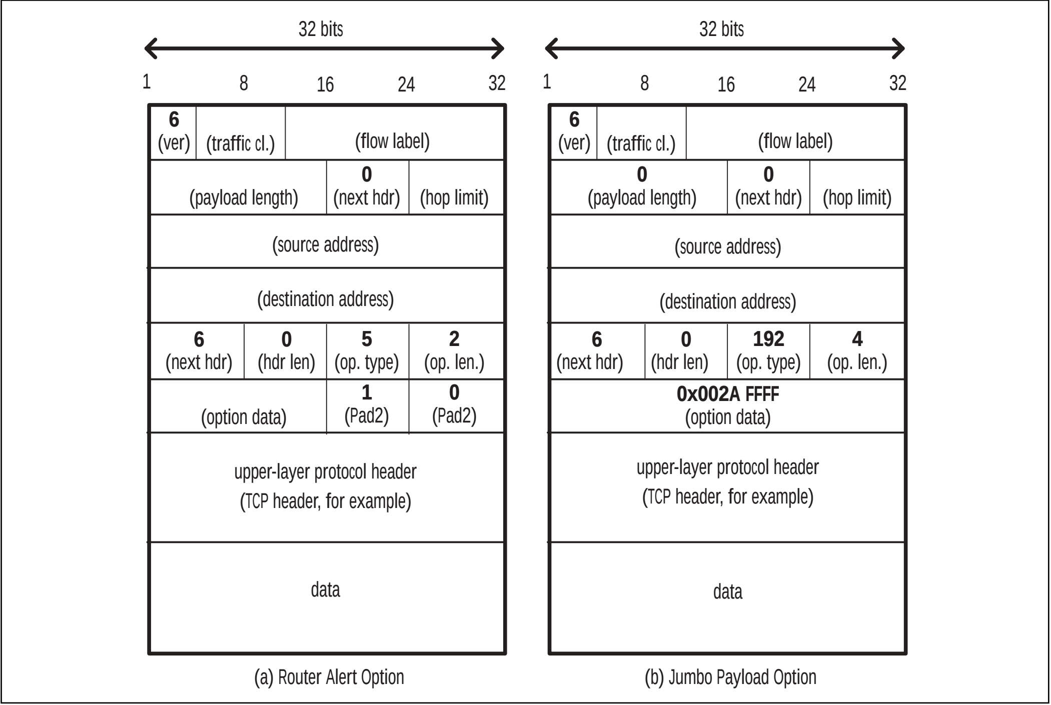


Next Header	8-bit selector. Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the *IPv4 Protocol* field [RFC-1700].
Hdr Ext Len	8-bit unsigned integer. Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.
Options	Variable-length field, of length such that the complete Hop-by-Hop Options header is an integer multiple of 8 octets. Contains one or more TLV-encoded options.

The Option Type of the Router Alert option is 5 (00000101), indicating that nodes not recognizing this option should skip it and continue processing the header, and the Option Data must not change its value en route. The Option Length of the Router Alert option is two; thus the valid range of the Option Data is 0 to 65535. Currently, only 0, 1, and 2 have been defined to indicate a packet containing an ICMPv6 Group Membership message, an RSVP message, or an Active Network message, respectively. No alignment requirement has been associated with this option.

Figure 3.6 (a) illustrates a packet containing a Router Alert Hop-by-Hop Option. The value of the *next header* field in the IPv6 header is zero, indicating that a Hop-by-Hop Options header follows. All nodes in the path of this packet are to examine and process this packet. The *next header* field of the Hop-by-Hop Options header indicates the next header following this Hop-by-Hop header—a TCP header in this sample packet. The *extension header length* field is zero since there is only one option (the Router Alter option), and the total length of the TLV encoding of this option is four octets. Since there is no alignment requirement associated with this option, its TLV-encoded option is placed first, and the Pad2 Option is used to make the length of this Hop-by-Hop Options header exactly eight octets.

Figure 3.6 Packets with the Hop-by-Hop Options Header



The IPv6 header uses the 16-bit *payload length* field, which limits the maximum length of a packet to 65536. However, advances in hardware have enabled the transmission of a jumbogram, a packet with payload larger than 65536 octets. This option supports jumbograms up to 4,294,967,296 octets. When a path MTU can support payloads larger than 65535, this option may be used to transmit jumbograms.

The Option Type of the Jumbo Payload Option is 192 (1100 0010), indicating that nodes not recognizing this option type must discard this packet and send an ICMP, Parameter Problem, Code 2, message to its sender (only if the destination is not a multicast), and the Option Data must not change en route. The *option length* field of this option is four octets, and the Option Data is the length of the IPv6 jumbogram, not including the IPv6 header. When this option is used, the *payload length* field in IPv6 is set to zero. This option has an alignment requirement of 4n+2.

Figure 3.6 (b) illustrates a packet that includes the Jumbo Payload Hop-by-Hop option. The *next header* field in the IPv6 header indicates that the Hop-by-Hop Options header follows. Note that the *payload length* field in the IPv6 header is set to zero. The *next header* field in this Hop-by-Hop options header indicates that the next header is a TCP header. The *extension header* field has a value of zero because the total length of the TLV-encoded Jumbo Payload option is six octets. The value of the Option Data of this packet indicates that the payload of is 2,818,048 octets (0x002A FFFF). Since the end of the Option Data is aligned with an 8-octet boundary, no padding option is necessary.

The processing of a Jumbo Payload option must detect several possible format errors and send an appropriate ICMP Parameter Problem message if one is present. These format errors include the absence of the Jumbo Payload option when the IPv6 Payload and the IPv6 Next Header are both zero, the use of the Jumbo Payload option when the IPv6 Payload is not zero, the use of the Jumbo Payload option when the actual payload is less than 65,535, and the use of the Jumbo Payload option when the Fragment Header is present.

NOTE

The Fragment header uses the 13-bit *fragment offset* field, in 8-octet units, to indicate the offset of the fragmented data relative to the original packet. In other words, the maximum offset can be the 65536th octet. Therefore, it makes little or no sense to use the Jumbo Payload option and Fragment header at the same time.

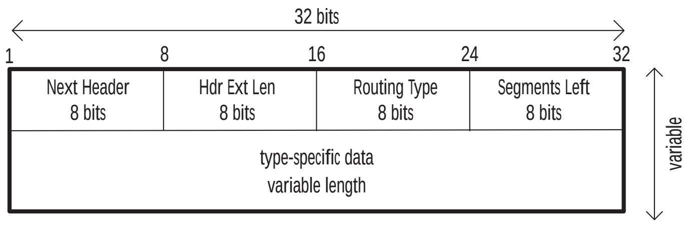
Routing Header

The Routing Header, identified by a Next Header value of 43 in the header immediately preceding it, allows an IPv6 source to determine routes to reach its destination by listing one or more intermediate nodes to be visited. (This is very similar to IPv4's Loose Source and Record Route options.) The format of the Routing Header is shown in Figure 3.7.

When a node encounters an unrecognized Routing Type, and the value of *segment left* is zero, it ignores the Routing header and continues to process the next header. However, if Segment Left is not zero, a node discards the packet and sends an ICMP Parameter Problem, Code 0, message to the packet's Source Address. Currently, only Type 0 has been defined; Figure 3.8 shows the format of a Type 0 Routing header.

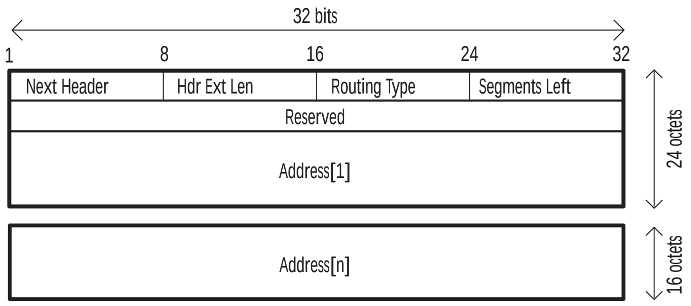
One application of the Type 0 Routing header is in supporting new protocols, such as RSVP.

Figure 3.7 Routing Header



Next Header	8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the *IPv4 Protocol* field [RFC-1700].
Hdr Ext Len	8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets.
Routing Type	8-bit identifier of a particular Routing header variant.
Segments Left	8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
type-specific data	Variable-length field, of format determined by the Routing Type, and of length such that the complete Routing header is complete Routing header is an integer multiple of 8 octets.

Figure 3.8 Type 0 Routing Header



Next Header	8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol field [RFC-1700].
Hdr Ext Len	8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets. For the Type 0 Routing header, Hdr Ext Length is equal to two times the number of addresses in the header.
Routing Type	0
Segments Left	8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
Reserved	32-bit reserved field. Initialized to zero for transmission; ignored on reception.
Address[1...n]	Vector of 128-bit addresses, numbered 1 to n.

In RSVP, a connection path may be established whereby all packets belonging to that connection follow the same path to reach the destination. The source of this connection may then use a Type 0 Routing header to specify the path to its destination.

Another use of a Routing header is to communicate with a mobile node away from its home network without triangle routing. Without Route Optimization, which may or may not be supported, packets may have to be sent to the mobile node's home network and be forwarded by the home agent (creating triangle routing) when a mobile node is away from its home network. The source of such a connection can specify the path using a Type 0 Routing header in order to allow the source of a connection to specify its path and avoid triangle routing.

For the connection between source node *s* and destination node *d* via routers *r1* and *r2*, source node *s* creates an IPv6 packet with a routing header, as shown in Figure 3.9(a). Notice that the *destination* field is *r1*, the first router in the path, rather than the final destination node *d*. Recall that except for the Hop-by-Hop Options header, all other extension headers are examined only by the packet's destination node. Since router *r1* is the destination of this packet, after examining the IPv6 header, it will continue to process the next header as indicated by the *Next Header* field. In this case, the Routing header will be processed by router *r1*.

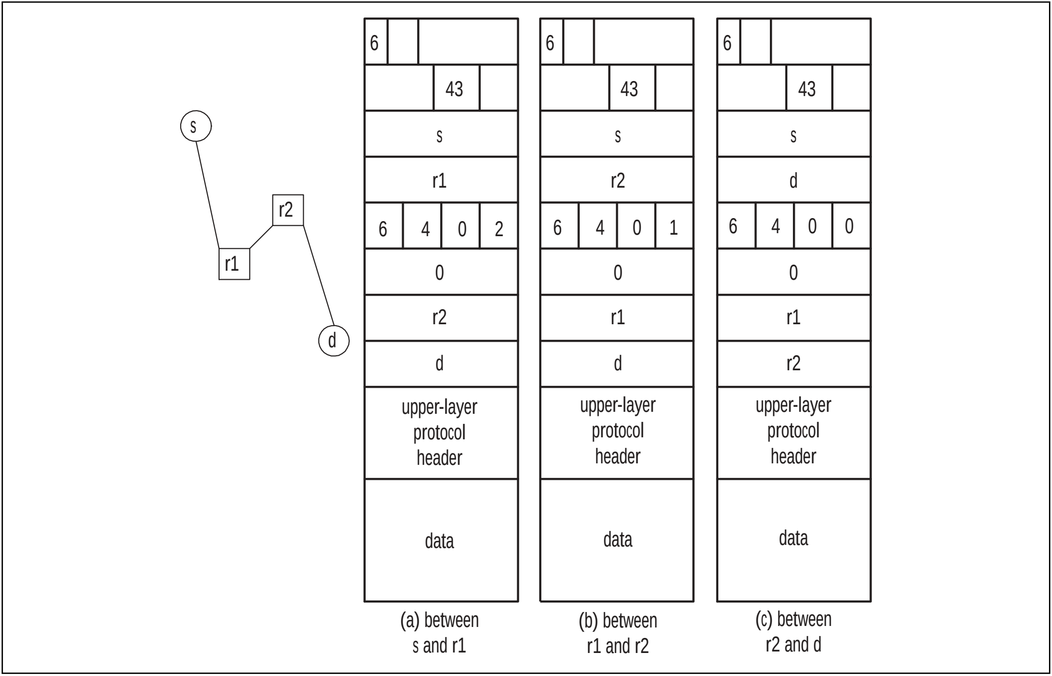
The *extension header length* field has a value of four, indicating that the length of the Routing header is four 8-octet sets, not counting the first eight octets. The value of four is also twice the number of addresses in this Routing header (two, as indicated in the *segments left* field). The first address in the Routing header is the next router in the path, *r2*, followed by the final destination node *d*.

Router *r1* decrements the *segments left* field and swaps the values of the *destination* field in the IPv6 header and the first address in the Routing header. Figure 3.9 (b) shows the packet sent from router *r1* to router *r2*. Similarly, after examining the IPv6 header, router *r2* continues to process the Routing header, since the *destination* field of the IPv6 header is *r2*. Again, router *r2* decrements the *segment left* field and swaps the values in the *destination* field of the IPv6 header and the second address in the Routing header, as shown in Figure 3.9(c). When processing the Routing header, the index of the address to visit can be computed using the *header extension length* and the *segment left* fields:

```
(Header Extension Length ÷ 2) – (Segment Left + 1)
```

When the Segment Left value reaches zero, the node handling this Routing header proceeds to process the next header in the packet, whose type is identified in the *next header* field in the Routing header.

Figure 3.9 Packets with a Routing Header



NOTE

The Type 0 Routing header is processed by all nodes appearing in the *address* field, and each node decrements the *segment left* field and swaps the values in the *destination* field in the IPv6 node and the address. Thus, multicast addresses must not appear in the address of the Type 0 Routing header or in the IPv6 *destination* field of a packet containing the Routing header.

When processing the Type 0 Routing header, format checking is performed. Recall that the Header Extension length is two times the number of addresses in the Routing header. Thus, the Header Extension length must not be odd. A node processing this packet discards the packet and sends an ICMP Parameter Problem, Code 0, message to the source node. Since the Header Extension length is two times the number of addresses in the Routing header, the largest value in the *segment left* field cannot be more than half of the Header Extension length. If the Segment Left value is larger, the node handling the packet also discards this packet and sends an ICMP Parameter Problem, Code 0, message to the source.

Fragment Header

The 16-bit *total length* field in the IPv4 header limits the maximum size of a packet to be 64k bytes. However, depending on the link technology used, the actual size of a packet may be further limited. In IPv4 packet transmission, each IP layer is responsible for fragmenting packets, if necessary, to ensure that the packet size does not exceed the maximum transmission unit. Thus, the user data sent in a single packet from a source node may arrive at the destination node in multiple packets if there is a link whose MTU is smaller than the link MTU at the source node. This approach, however, may not be the most optimal solution for the path.

Designing & Planning…

Packet Fragmentation Overhead

Consider an application that transfers a segment of 3000 bytes at a regular interval, where the link MTU at the source is 3000 bytes. The next link MTU is 1500 bytes; thus, a packet is fragmented into two packets of 1500 bytes each. However, the following link MTU is 1000 bytes. Then each of 1500-byte packets will be further fragmented into two packets, 1000 bytes and 500 bytes, for a total of four packets. If the path MTU had been known at the source, the source node would have fragmented the 3000 bytes in three packets. In IPv6's implementation of fragmentation, the source node discovers the lowest MTU in the end-to-end path and performs any fragmentation necessary.

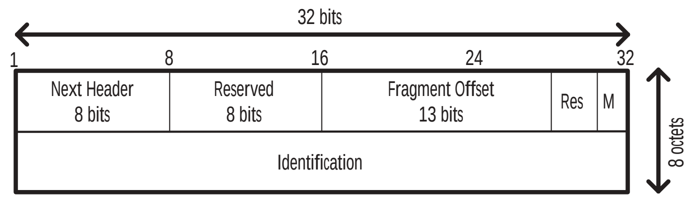
In general, fragmentation involves high overhead, so the use of fragmentation should be discouraged. (In IPv4, the overhead involved in the transmission of 3000 bytes is 20 bytes for an IP header alone, without options. The overhead is higher when upper-layer protocol header overhead is considered.) Links with configurable MTUs (e.g. PPP links) should configure their MTU to at least 1280 octets, which is the required MTU in IPv6.

In IPv6, only source nodes perform fragmentation. A source node first finds the path MTU and then segments the fragmentable part of the original packet so that the length of each fragmented packet does not exceed the path MTU. Before fragmentation, the original packet consists of two parts: the unfragmentable part and fragmentable part. The IPv6 header and any extension headers that need to be processed at each hop on the way to the destination are unfragmentable, and extension headers processed only by the final destination node (or nodes in the case of multicast) are considered to be fragmentable.

The Hop-by-Hop Options header is always unfragmentable, since it must be processed by each hop in the path. Thus when the Hop-by-Hop Options header is present, the unfragmentable part of the original packet includes the IPv6 header and the Hop-by-Hop Options header. Whether the Destination Options header is fragmentable depends on whether there is a Routing Options header. If there is no Routing Options header, the Destination Options header is fragmentable, since it only needs to be processed by the final destination. If there is a Routing Options header, the Destination Options header is unfragmentable, since it will need to be processed by every node appearing in the Routing header.

The Fragmentation header is identified by a Next Header value of 44 in the immediately preceding header. Figure 3.10 shows the format of the Fragmentation header. The source node generates a unique 32-bit identifier for every fragmented packet sent to the same destination. Except for the last fragmented packet, the fragmentable part of the original packet is divided so that each fragmented part is of a length that is an integer multiple of eight octets. The *fragment offset* field is used to

Figure 3.10 The Fragmentation Header



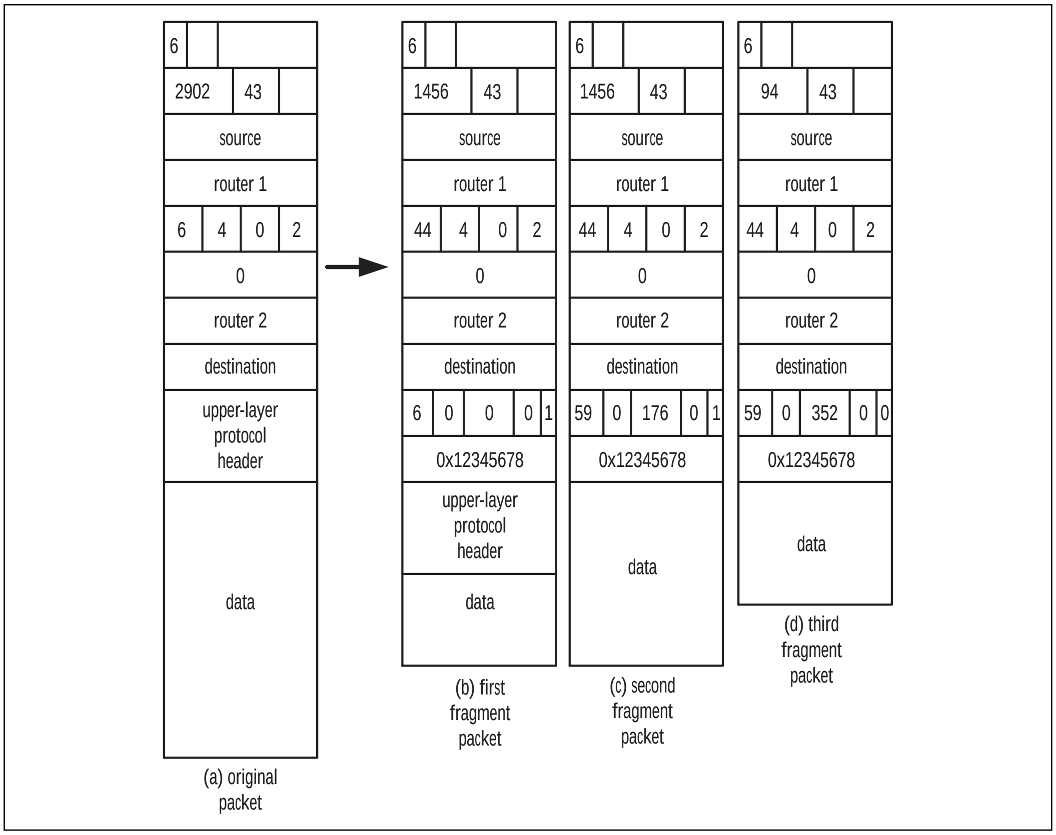
Next Header	8-bit selector. Identifies the initial header type of the fragmentable part of the original packet. Uses the same values as the *IPv4 Protocol* field [RFC-1700].
Reserved	8-bit reserved field. Initialized to zero for transmission; ignored on reception.
Fragment Offset	13-bit unsigned integer. The offset, in 8-octet units, of the data following this header, relative to the start of the fragmentable part of the original packet.
Res	2-bit reserved field. Initialized to zero for transmission; ignored on reception.
M flag	1 = more fragments; 0 = last fragment.
Identification	32-bit identifier

indicate the offset of the data following this Fragmentation header relative to the start of the Fragmentable part of the original packet.

Consider the packet shown in Figure 3.11 (a). This packet needs to be further fragmented by the source node because its path MTU is 1514 bytes. The unfragmentable part of the original packet in this example includes the IPv6 header and the Routing header (the Next Header of the IPv6 is 43). The original packet is broken into three parts. Since the Ethernet header is 14 bytes, the IPv6 packet including the IPv6 header cannot be longer than 1500 bytes. Since the IPv6 header is 40 bytes, the headers and data after the IPv6 header cannot be longer than 1460 bytes. The Routing header contains two destinations (router 2 and the destination), so its length will be 40 bytes. Since the Routing header is part of the unfragmented part, each fragment includes the Routing header, so the headers and data after the Routing header cannot be longer than 1420 bytes. Further, the Fragmentation header (eight octets) is added, leaving 1412 bytes for user data. However, the *fragmentation offset* field in the Fragmentation header identifies the number of 8-octet units of user data. Thus the user data must be a multiple of eight octets. Since 1412 is not a multiple of eight octets, the maximum size of the fragmentable part of the original packet is limited to 1408 (176 8-octet units). This explains the value of 1456 in the *payload length* field in the IPv6 Header of the first fragmentation as shown in Figure 3.11 (b) (1408 of user data + 8 for the Fragmentation header + 40 for the Routing header).

The *next header* field in the Routing header of each fragment (Figure 3.11 (b), (c), and (d)) has a value of 44, indicating that the next header following this Routing header is the Fragmentation header. The *next header* field in the first fragment is 6, indicating that the upper-layer protocol header (TCP) follows this Fragmentation header. However, the *next header* field in each of the other two fragments is 59, indicating that there are no more headers following this Fragmentation header. In this example, the hexadecimal 0x12345678 is used to indicate that the same identifier is used for all fragments. The *fragmentation offset* field is used to indicate the offset, in 8-octet units, of the data following the Fragmentation header, relative to the start of the fragmentable part of the original packet. Thus, the Fragmentation offset in Figure 3.11 (c) indicates that the data following this Fragmentation header should be positioned in the 176^8 byte in the fragmentable part when reassembled at the destination node.

Figure 3.11 Fragmentation Example

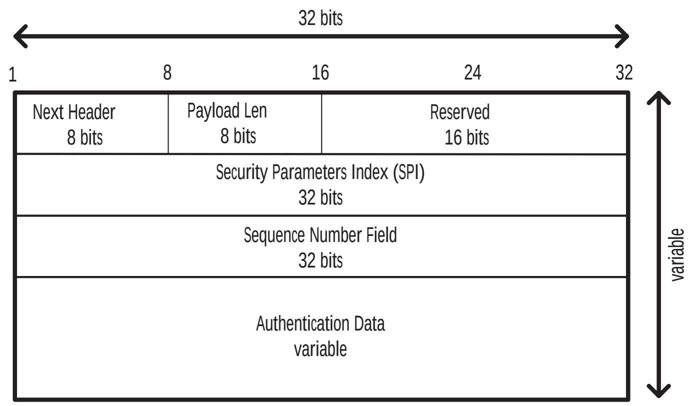


Authentication Header

In an IP network (both IPv4 and IPv6), the Authentication header is used to provide integrity and data origin authentication for IP packets and to protect against replays. In this section, we will provide terms based on an IPv6 network. The Authentication header provides authentication for the IPv6 header, upper-layer protocol headers and user data, and IPv6 extension header fields that may not change en route. For instance, the *destination address* field in the IPv6 header changes at every hop when the Type 0 Routing header is used, so in this case the Authentication Header cannot provide the authentication of the *destination address* field. Figure 3.12 shows the format of the Authentication Header.

Note that the *payload length* field is in a 4-octet unit (32-bit word), not including the first eight octets (or two units of four octets each).

Figure 3.12 The Authentication Header


32 bits
1
8
16
24
32
Next Header 8 bits
Payload Len 8 bits
Reserved 16 bits
Security Parameters Index (SPI) 32 bits
Sequence Number Field 32 bits
Authentication Data variable
variable

Next Header	8-bit selector. Identifies the type of header immediately following the Authentication header. Uses the same values as the *IPv4 Protocol* field [RFC-1700].
Payload Len	8-bit unsigned integer. Length of the Authentication header in 4-octet units, not including the first eight octets.
Reserved	16-bit reserved field. Initialized to zero for transmission; ignored on reception.
Security Parameter Index	32-bit unsigned integer. Combination of this field, destination address, and security protocol identifies the Security Association for this packet.
Sequence Number	32-bit unsigned integer. Monotonically increasing counter value.
Authentication Data	Variable-length field containing the Integrity Check Value (ICV) for this packet. This field must be an integral multiple of eight octets in length.

NOTE

All other IPv6 header extension lengths are encoded by measuring the header length in 8-octet units, then subtracting one (since the first eight octets are not counted).

Thus with a 96-bit Authentication Data value, the payload length will be four. For debugging purposes, the Null Authentication algorithm may be used. In this case, the value of the *payload length* field will be two.

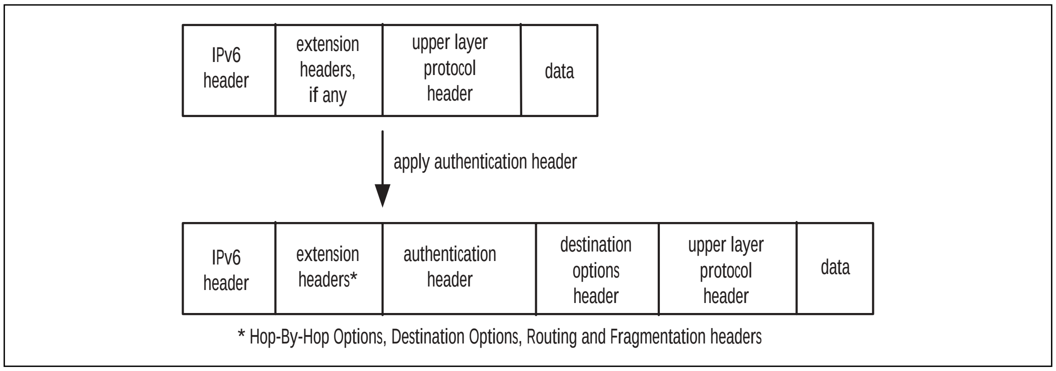
The *sequence number* field is used to provide protection against anti-replay. When a Security Association is established between source and destination nodes, counters at the sending and receiving ends are initialized to zero. It is mandatory for the sender to increment this field for every transmission; however, the receiver may elect not to process the transmission. This service is effective only if the receiver processes this field.

The *authentication data* field contains the Integrity Check Value (ICV) for the packet. The authentication algorithm, selected when the Security Association is established between the sender and the receiver, specifies the length of the ICV, the comparison rules, and the necessary processing steps. This value is computed over the packet by the source node and verified by the destination node, which compares the value to its own recomputed value.

The Authentication header may be applied in transport or tunnel mode. The transport mode Authentication header, implemented in hosts, provides protection for the upper-layer protocol header and any fields in the IPv6 header, as well as extension headers that do not change in transit. The tunnel mode Authentication header is applied to the original IPv6 packet, encapsulating the original packet by constructing a new IPv6 packet that uses a distinct IPv6 addresses, such as a security gateway.

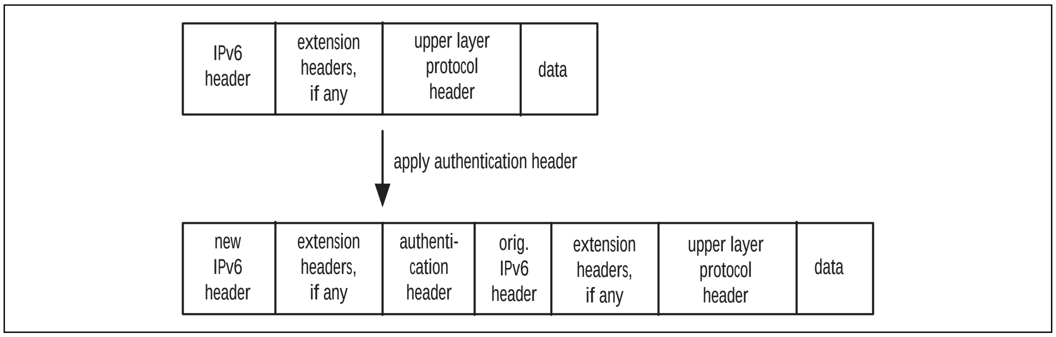
In transport mode, the Authentication header is viewed as an end-to-end payload and is placed after the IPv6 header and Hop-by-Hop, Routing, and Fragmentation extension headers. Recall that the Destination Options header may appear once before the Routing header, as the options in the Destination Options header are applicable to intermediate nodes specified in the Routing header. In this case, the Authentication header comes after the Destination Options header, as shown in Figure 3.13.

Figure 3.13 Header Order with Authentication Header in Transport Mode


IPv6 header
extension headers, if any
upper layer protocol header
data
apply authentication header
IPv6 header
extension headers*
authentication header
destination options header
upper layer protocol header
data
* Hop-By-Hop Options, Destination Options, Routing and Fragmentation headers

In tunnel mode, the Authentication Header is applied to the original IPv6 packet using distinct IPv6 addresses as communication end points, and a new IPv6 header is constructed using addresses of security gateways for source and destination addresses. Fragmentation processing may be necessary after applying the Authentication header; thus a newly constructed IPv6 packet may undergo further processing. Figure 3.14 shows the order of headers after applying the Authentication header in tunnel mode.

Figure 3.14 Header Order with Authentication Header in Tunnel Mode


IPv6 header
extension headers, if any
upper layer protocol header
data
apply authentication header
new IPv6 header
extension headers, if any
authenti-cation header
orig. IPv6 header
extension headers, if any
upper layer protocol header
data

Encapsulating Security Payload

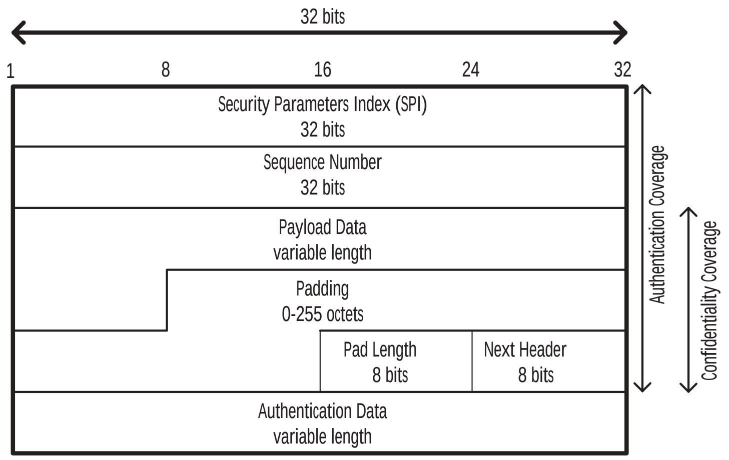
The Encapsulating Security Payload header, used in transport mode or in tunnel mode, also provides security services in both IPv4 and IPv6 networks. The security services provided through the Encapsulating Security Payload include confidentiality, authentication (data origin authentication and connectionless integrity), an anti-replay service, and limited traffic flow confidentiality. The implementation and options chosen at the time the Security Association is established determine the security services provided.

As in the case of the anti-replay service provided by the Authentication header, the source increments the Sequence Number; however, the destination node must check this field to enable the anti-replay service. To provide traffic flow confidentiality service, true source and destination information should be hidden. Thus, this service requires that the Encapsulating Security Payload header be used in a tunnel mode.

Figure 3.15 shows the format of the Encapsulating Security Payload header. The Next Header value of 50 in the immediately preceding header indicates that the Encapsulating Security Payload header processing is necessary.

The mandatory *payload data* field contains encrypted data described by the *next header* field. The encryption algorithm used specifies the length and the location of the structure of the data within the *payload data* field. Padding may be necessary to fulfill the encryption algorithm requirement of the length of the plain text or the 4-octet boundary alignment of the *payload data* field.

Figure 3.15 The Encapsulation Security Payload Header



Security Parameter Index	32-bit unsigned integer. Combination of this field, destination address, and Security Protocol (ESP) identifies the Security Association for this packet.
Sequence Number	32-bit unsigned integer. Monotonically increasing counter value.
Payload Data	Variable-length field containing data described by the *Next Header* field.
Padding	Variable-length field containing 0 to 255 octets of 8-bit padding.
Pad Length	8-bit unsigned integer. Indicates the number of pad octets immediately preceding it.
Next Header	8-bit selector. Identifies the type of data contained in the Payload Data.
Authentication Data	Variable-length field containing the Integrity Check Value (ICV) computed over the ESP packet minus the Authentication Data.

Figures 3.16 and 3.17 illustrate the sequence of an IPv6 packet with its encrypted portion when Encapsulating Security Payload headers are used in transport mode and tunnel mode, respectively.

Figure 3.16 Header Order with Encapsulating Security Payload in Transport Mode

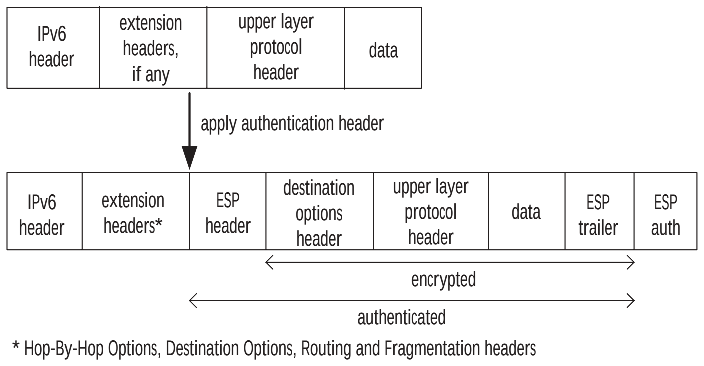
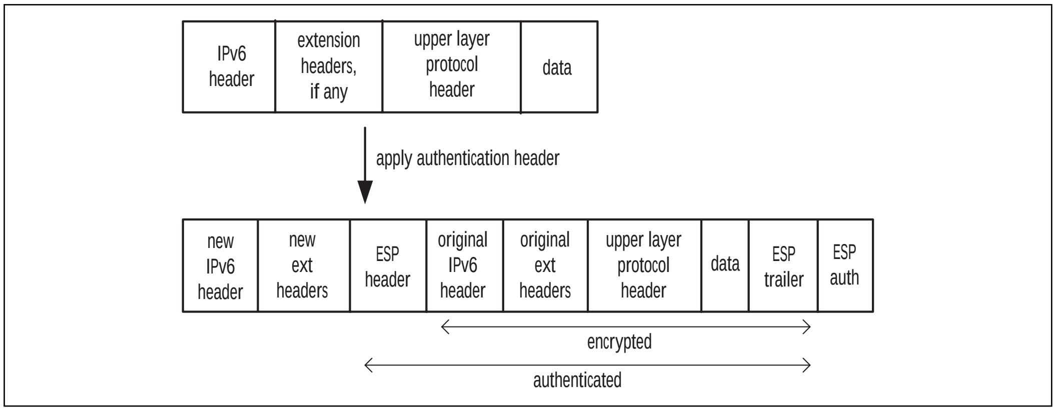

IPv6 header
extension headers, if any
upper layer protocol header
data
apply authentication header
IPv6 header
extension headers*
ESP header
destination options header
upper layer protocol header
data
ESP trailer
ESP auth
encrypted
authenticated
* Hop-By-Hop Options, Destination Options, Routing and Fragmentation headers

Figure 3.17 Header Order with Encapsulating Security Payload in Tunnel Mode



Destination Options Header

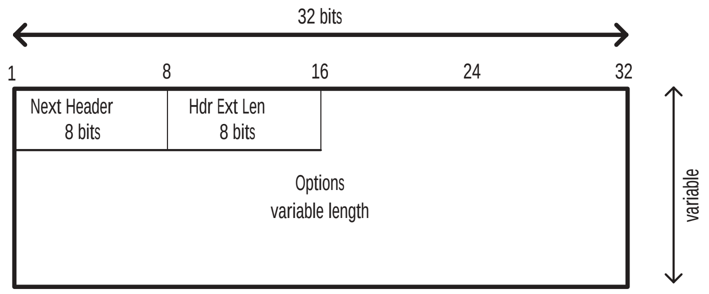
A source node may need to convey optional information that needs to be processed by a destination node. For instance, when a mobile node is away from its home network, a home agent (such as a router at the home network) may be a proxy forwarding packets to the mobile node. A mobile node away from its home network needs to send control messages to its home agent so that the home agent can set up the proxy service and forward packets destined for the mobile node to its current address. In an IPv4 network, a packet that contains options in the IPv4 header will be subject to an examination at every hop on the path.

In an IPv6 network, such optional messages can be handled efficiently using either an extension header dedicated for handling specific optional information or using the Destination Options header. Packet fragmentation or authentication information is handled as an extension header, as shown previously. The IPv6 Mobility Support Internet-Draft proposes four Destination Options to support Mobile IPv6.

The optional information may be encoded either in a separate extension header or in the Destination Options header, based on the desired action to be taken at the destination node when the node does not recognize the option. Optional information that requires a few octets to send an ICMP Unrecognized Type message to the sender only if the destination node is not a multicast address may be encoded in a separate extension header.

The Destination Options header, identified by a Next Header value of 60 in the immediately preceding header, carries optional information that needs to be examined and processed only by a packet's destination node (or nodes, in multicast). The format is shown in Figure 3.18.

Figure 3.18 The Destinations Options Header



Next Header	8-bit selector. Identifies the type of header immediately following the Destination Options header. Uses the same values as the *IPv4 Protocol* field [RFC-1700].
Hdr Ext Len	8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.
Options	Variable length field, of length such that the complete Destination Options header is an integer multiple of 8 octets. Contains one or more TLV-encoded options.

Summary

The IPv6 header strikes a delicate balance between being small and efficient (the standard IPv6 header), yet also supporting the many legacy and new variations needed within IP (the extension headers). The IPv6 header and extension headers maintain lengths in multiples of eight octets wherever possible. This provides maximum efficiency for today's 64-bit architectures.

The design of IPv6 places considerable importance on the future by dedicating 28 bits of the standard header to QoS-related fields. This includes the 8-bit *traffic class* and 20-bit *flow label*. The *flow label* field may not always be explicitly used for QoS—it may simply be used to identify a particular flow of traffic within the network. However, it is likely that the network will often identify flows to perform some type of preferential service. Even though the IPv6 definition does not completely detail the exact uses of these fields, it is understood these will be important in the future.

The *payload length* field in IPv6, as in IPv4, is 16 bits long. This allows a packet to be as large as 64K bytes. Although 64K bytes may be too large to be practical, it provides substantial flexibility to increase the maximum packet size over today's *de facto* standard of 1514 bytes. Even packets as large as 5K or 10K bytes would reduce packet overhead and transfer data more quickly and efficiently. As for the concern of large packets occupying a link for too long and delaying time-sensitive traffic, most ISP links operate at such a high speed that 5K or 10K packets take very little time to transport. For slower links, the MTU can be set lower. This will force nodes to transmit in smaller sizes (such as 1514 bytes), allowing time-sensitive packets to be sent in a timely manner. Of course, the Ethernet MTU will need to be increased before widespread use of larger packet sizes becomes a reality.

The IPv6 header maintains the most useful features of the IPv4 header while getting rid of outdated or unused features. The *payload length*, *version*, *Next Header*, and *hop limit* fields are taken almost directly from the IPv4 header, making the IPv6 header familiar and fairly easy to understand for anyone who is familiar with the IPv4 header.

Certain needed but rarely used features like fragmentation have been moved to their own optional headers in IPv6. In this way, fragmentation remains available without burdening the vast majority of IPv6 packets that don't require it. Similarly, other IPv4 options (and some new options) have been redesigned as extension headers, to be used only if required.

IPv4 fields that would not be useful in IPv6 were dropped, including header length and header checksum.

IPv6 implements a variety of extension headers. Two of these headers, Hop-by-Hop Options and Routing, are specifically designed to be easily processed by all nodes in a path. This will become useful if path-specific protocols such as RSVP become widespread. They can also be useful for protocols such as multicast, where you may want to indicate to all the routers in a path that multicast traffic is or is not desired.

IPv6 created a Destinations header specifically designed to deliver information to the destination node. Much like the traffic class and flow label, the details of this header were not all originally defined in IPv6. Instead, IPv6 created a flexible framework (the Type-Length-Value option) for options and features to be easily added in the future.

Two other extension headers, Authentication and Encapsulation Security Payload, provide various security services. These are very similar to the Authentication and Encapsulation Security Payload protocols used in IP Security (IPSec) today, but they are implemented as extension headers in IPv6. This familiarity should allow these to be easily implemented and quickly adopted in IPv6.

With IPv6's 128-bit available address space, IPv6 addresses should be plentiful. This has obvious benefits, since obtaining IPv6 address blocks should be easy. The large number of IPv6 addresses will also allow companies to assign legitimate IPv6 addresses to every internal node, should they desire. One advantage of this is that it will allow security headers, such as the Authentication header, to protect integrity of the IPv6 packet. Today most packets sent across the Internet (when you want the most security) go through at least one NAT process. This changes the source IP address and makes full authentication difficult, if not impossible. With IPv6's plentiful address space NAT should not be required, allowing the Authentication header to better protect packets.

Solutions Fast Track

Analyzing the IPv6 Header

☑ The IPv6 header reserves two fields (totaling 28 bits) for prioritizing and/or identifying packets and packet flows that require special handling, such as prioritization. As real-time applications such as video and voice become more prevalent, these fields should be heavily used.

- ☑ The *Next Header* field in IPv6 uses the same definitions used in IPv4 (and defined in RFC 1700). Thus TCP uses "6," UDP uses "17," and so on.
- ☑ Immediately following the destination IPv6 address is the next header—an IPv6 extension header, an upper-layer protocol header, etc.

Comparing the IPv6 and IPv4 Headers

- ☑ Source and destination IPv6 addresses are four times as long as IPv4 addresses (128 bits versus 32 bits)
- ☑ The IPv4 header adds optional functions within the header in the *options* field. IPv6 has a completely fixed header, but adds extension headers for optional functions such as routing, fragmentation, or authentication.
- ☑ In IPv6, many IPv4 fields were maintained, a few were added, and several were dropped (or changed to extension headers)

The IPv6 Extension Headers

- ☑ Different extension headers were created based on the nodes that would need to examine and process them. For example, the Destination Header and Fragmentation Headers were created strictly for viewing by the endpoint, whereas the Routing and Hop-by-Hop Headers were created to be viewed by all routers along the path. This improves efficiency, since some headers do not need to be processed by intermediate routers.
- ☑ Authentication and Encryption are incorporated into the IPv6 standard as extension headers, supporting the need for increased security in the digital world.
- ☑ Although the Fragmentation header is part of the IPv6 standard, its use is discouraged. In reality, it will likely get very little use.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: Since the routing header can control the path of a packet, can it be used to optimize the path used between a client and server?

A: While in theory this could be done, in reality it will probably be used only in conjunction with protocols such as RSVP. This is primarily because without RSVP (or an equivalent protocol), the client and the server will not be aware of the underlying network infrastructure. Instead, this feature will most likely be used by mobile users or for diagnostic purposes by network management applications that will allow you to control the flow of the packet for testing purposes.

Q: Doesn't the IPv6 header create more overhead—and thus less efficiency—than IPv4?

A: The IPv6 header is larger and does add more overhead to each packet than IPv4. Assuming there are no options or extensions, the standard IPv6 header is 40 bytes, while the standard IPv4 header is only 20 bytes. So IPv6 adds 20 bytes to the header, but with IPv6 there are 24 more bytes just in the source and destination addresses (32 bytes versus 8 bytes). Without the larger source and destination addresses, the IPv6 header is smaller than IPv4. However, given the shortage of IPv4, I think there is a good argument for increasing the size of IPv6 addresses.

Q: I see that IPv6 supports packets up to 64 KB, or—with the Hop-by-Hop Options header—even much larger packets. Is this feature likely to be used?

A: Larger packets have advantages, such as reduced processing at the end nodes. However, packet sizes larger than 1514 bytes have been proposed in the industry several times and have received surprisingly little support. One previous opposing point was that network designers did not want a very large packet to "clog" a slower link, preventing time-sensitive traffic (such as video or voice) to pass in a timely manner. As WAN speeds are increasing over time, this is becoming less of a problem. Over time this may become a common option, but I do not anticipate this feature being used in the near future.

Q: Many of the extension headers have fields where options can be added—why are only a few of those discussed here?

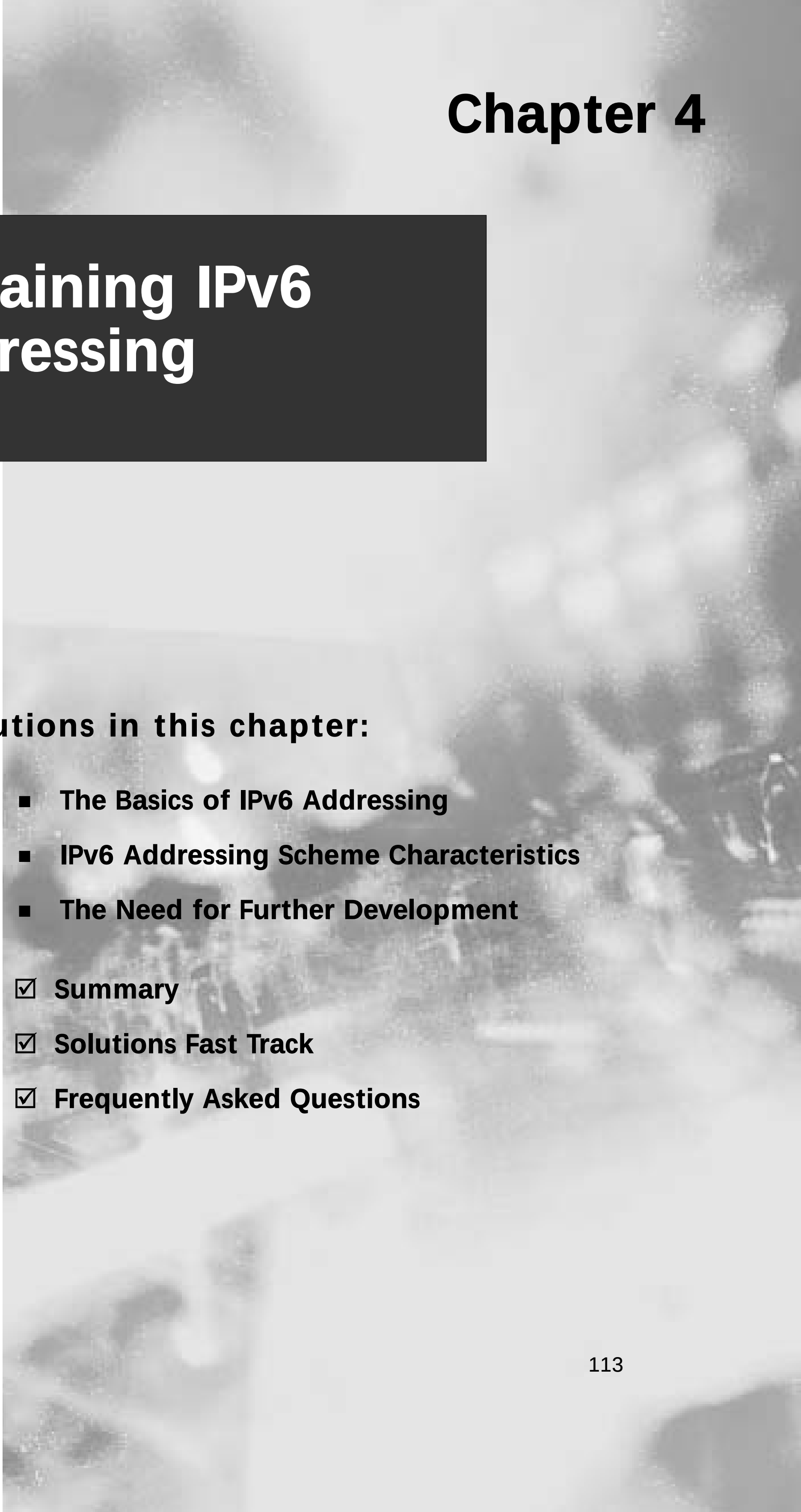
A: The current IPv6 specification, RFC 2460, wanted to create a framework that would be flexible enough to support changes over time. As such, they defined several extension headers, most with fields for options, yet they did not define many of these options. Rather, they wanted these to be further defined in later RFCs as the needs and requirements of IPv6 became more clear. Most of these options are still evolving and being refined as more and more users adopt IPv6.

Q: Since IPv6 will often be sent over Ethernet (which has its own packet checksum), and it frequently delivers TCP data (which uses a checksum), is the authentication header necessary?

A: In many cases the Authentication header will not be used. However, in environments requiring higher levels of security, the Authentication header is still useful. The Ethernet and TCP checksums use open, well-known methods for calculating the checksum. Thus, in theory, someone could alter the packet and calculate the new, correct checksum. More importantly, though, the authentication header provides end-to-end protection from changes for the IPv6 packet (as opposed to relying on Ethernet or TCP). It also provides additional benefits, such as anti-replay protection—anti-replay is a real threat, especially if someone uses a sniffer on the original conversation).

Q: In today's Internet it is not unusual for packets to traverse different paths, even between the same two endpoints. Wouldn't this make it difficult for an IPv6 source node to effectively use MTU discovery and be sure they had correctly detected the smallest MTU?

A: While today's packets do take different paths, it is unlikely this condition will create a problem in the future. Packet sizes today are standardized at 1514 bytes, even in IPv6 environments. Should the MTU of Internet links increase (creating a possible MTU problem), it is likely all link MTUs would increase at approximately the same time (or at the very least, all links within an ISP). This would help to minimize the problem. If this is not the case, potential problems could occur, since intermediate routers do not have the ability to fragment a packet the way IPv4 routers do today.



Chapter 4

Explaining IPv6 Addressing

Solutions in this chapter:

- **The Basics of IPv6 Addressing**
- **IPv6 Addressing Scheme Characteristics**
- **The Need for Further Development**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

Introduction

The explosion of Internet growth has created a severe depletion of the current Internet Protocol v4 (IPv4) address space. The current IPv4 address space mainly serves the address requirements for the PC computer market in today's service provider and enterprise business markets. However, the development of new markets has required networks to support an increasing variety of network attachments, including mobile IP elements such as wireless and infrared devices and IP phones. This expansion of the market drives the need for a substantially larger IP address space. Routing and addressing using the IPv4 address space has become increasingly constrained, and as the Internet has grown, inefficiencies have arisen in the capability to deploy a hierarchal infrastructure that promotes address aggregation. The utilization of Classless Inter-Domain Routing (CIDR) has extended the lifetime of the IPv4 address space considerably, but it is increasingly evident that the current IPv4 address space will be depleted in the near future. IPv6 represents the next generation of Internet protocol that can meet both the current addressing requirements and those generated by emerging markets.

IPv6 addresses are much different from IPv4 addresses from both an addressing structure and aggregation and routing functionality. These differences require designers of supporting protocols to re-design the operations of the protocols. The mapping of addresses to hostnames via the Domain Name System (DNS) has been updated and is becoming increasingly more important as IPv6 starts to become widely used. Management protocols such as Simple Network Management Protocol (SNMP) will change to accommodate IPv6. Internet Control Message Protocol (ICMP) has been updated to ICMPv6 to support the monitoring and troubleshooting of IPv6 networks.

The task of modifying and updating network equipment to support the new protocol is daunting. The scope of modification needed and the time it will take to fully implement the necessary changes will require not only a huge amount of planning, but also a thorough understanding of exactly how IPv6 works and how addressing is handled.

This chapter explains the key concepts related to IPv6 addressing, including IPv6 address syntax, the address space, the types of addresses (unicast, multicast, anycast), prefixes and subnetting, address assignment (where to get addresses: TLA/NLA/SLA, and so on), link-local/site-local addressing, the unspecified address, the loopback address (of course), and any other appropriate addressing topics. This chapter also gives a short introduction to the history and development of the IPv6 protocol, through its current accepted form.

We will look at some of the key aspects of IPv6 that separate the protocol from IPv4, and look into the benefits that we can gain by utilizing IPv6 and its addressing schemas to build a more scalable network. From there, we can begin to build real-world examples of how this addressing can be deployed in Internet-connected networks to come.

Finally, we will look into some of IPv6's yet unsolved issues, and some of the proposed solutions to cope with them. Also in this section, we will give a brief introduction to the IPv6 test network, the 6Bone.

The Basics of IPv6 Addressing

By the early 1990s, it was clear that the Internet was going to take off. The average person was becoming aware of its existence, and the killer-apps of today (Web browsers) were coming into their own. Address space delegations increased at an alarming rate, and it was clear that the Internet Protocol version 4 had a foreseeable upper limit in terms of the number of entities it could connect to the ever-increasing worldwide Internet. The Internet Engineering Task Force (IETF), the standards group from which a large portion of Internet technologies emerge, was beginning to see this as an issue that needed to be tackled earlier rather than later. In a three-year span between 1996 and 1999, authority agents responsible for address allotment allocated 150 million IP addresses. Although the Internet *is* growing at an alarming rate, it is clear that 150 million hosts were not added. Address allocation has a major problem, even after the efforts of CIDR were implemented. Address space is being wasted and will run out eventually.

In addition, demand dictated enhanced features on the network layer (Layer 3 on the Open Systems Interconnection [OSI] reference model stack), such as end-to-end encryption, authentication of packets, source-routing, and Quality of Service (QoS). As people began to see these factors as a reality, many proposals for a new Internet protocol emerged.

The first draft that gained widespread notice was loosely based on the Connection-Less Network Protocol (CLNP), which was based upon another protocol suite, the OSI stack. This stack originally ran on the early Internet, but was quickly replaced by IPv4 when the Internet began to take on size and popularity. The proposal was coined TUBA (TCP/UDP over Bigger Addresses). CLNP did provide for a much larger address range than the current IPv4. Its Network Service Access Point (NSAP) address consisted of 20 octets and would provide adequate addressing ranges for the Internet's foreseeable future. However, this proposal was rejected because CLNP lacked some of the value-added features that were already installed into the current IP (QoS, multicast, and so on.), and these were determined to be important to the Internet's growth.

There was a proposal that attempted to create a packet format compatible with current IP, CLNP, and Internetwork Packet Exchange (IPX). Yet another proposal, known as Simple IP Plus (SIPP), simply advocated increasing the current IP addressing format to 64 bits and fine-tuning some of the feature sets of IPv4, as well as establishing better routing strategies. SIPP turned out, after some modifications, to be the closest match for what the Internet needed. The addressing range was changed from 64 to 128 bits, and the name was changed to IP version 6, or IPv6 (IPv5 was already delegated to another protocol). This would be the protocol to solve the Internet scalability problems and put us into the next millennium.

IPv6 Addressing Scheme Characteristics

This section provides an overview of the IPv6 header and the characteristics of the IPv6 addressing scheme, and discusses how they can improve routing stability and efficiency. Topics of discussion include:

- IPv6 header
- IPv4 and IPv6 address space
- IPv6 address structure

IPv6 Header Overview

The IPv6 header is 40 bytes long and contains fields pertaining to version, traffic class, flow label, payload length, next header, hop limit, source address, and destination address. Figure 4.1 depicts the IPv6 header as expressed in Request for Comments (RFC) 2460 (www.ietf.org/rfc/rfc2460.txt).

Version Field

The *version* field in the IPv6 header is so that Internet mechanisms that know how to route, or even speak routing protocols, will know what type of routing protocol they are about to deal with. Notice the similarities to IPv4. In the case of IPv6, the Version field is a 4-bit integer, with the value of 6 (0110 in binary), to designate this packet as an IP version 6 packet.

Figure 4.1 IPv6 Header

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Traffic Class Field

The *traffic class* field is an 8-bit field in which some sort of traffic differentiation identifier can be placed. Currently, in the IETF, many working groups are dedicated to coming up with the best way to utilize this type of differentiation mechanism (though they mostly concentrate on IPv4 today). One example of such a group is the DiffServ (Differentiated Services). The members of DiffServ are trying to come up with a way to give more important traffic a higher priority for routing on the Internet today. This field was designed for things such as IP Precedence bits (giving certain values of this field higher priority, and then using differentiated queuing strategies in the router to tell "who goes first"). You can learn more about DiffServ on the Web at www.ietf.org/html.charters/diffserv-charter.html. A number of drafts and RFCs have been written with ideas as to how to implement such a policy. The list of current open drafts (they are good for only six months after writing, at which time they need to be resubmitted, just to keep things current) and RFCs is at the bottom of the aforementioned URL.

Flow Label Field

The *flow label* field is a 20-bit field used when special handling of a packet is needed. Common interpretation of this field at the time of this writing is that the field will assign flow labels in order to engineer different traffic patterns in an IPv6 network. The major player in this (though for mostly IPv4 at this time) is the Multi-Protocol Label Switching (MPLS) working group. To see the group's charter, please see www.ietf.org/html.charters/mpls-charter.html. This group's main intention is to come up with an efficient way to assign labels to flows, and to come up with an efficient and scalable way to route based on these flows. A flow can be defined as any class of traffic going from one point to anther, whether point-to-point traffic, or a TCP flow from one end-station at a given port to a destination end-station on a given port. The possibility of assigning flows opens up many interesting options for deployment. Perhaps QoS (quite a buzzword in the field today) can be deployed with scalability this way. Many Internet providers are keeping their eyes wide open as this working group develops, because advanced services that the MPLS working group sees as feasible could lead to ground-breaking new developments in the Internet industry as a whole.

Payload Length Field

The *payload length* field is a 16-bit integer used to designate the length of the payload (the data) in the IPv6 packet, in octets. Notice this field is 16 bits long (2 raised to the power 16), which gives us more than 64,000 different possibilities, enabling IPv6 to have fairly big packets (more than 64,000 bytes). The capability to make big packets can increase the efficiency of the Internet as a whole. When your packets are bigger, the number of packets needed to send a given amount of data becomes smaller for a given flow. When a router has fewer packets to route, it has more time to route other packets, or to perform other tasks (routing table maintenance, cache aging, and so on.). You can see how this can help to increase Internet efficiency altogether. Note that any extension headers outside this header are included in the total packet length in this case. Compare this with the IPv4 case (RFC 791) where the total length field *includes* the IPv4 main header.

Next Header Field

The *next header* field is designated to tell routers if any other headers need to be looked at for the packet to route according to instruction. This feature differs drastically from the IPv4 case, where only one header has a fixed length. The IPv6 main header has a fixed length as well (enabling routers to know beforehand how much of the packet they need to read), but has built-in functionality to stack other headers that provide other value-added services on top of the main header. This field is 8 bits in length, allowing for up to 255 types of Next-Headers. Currently, only a finite amount of Next-Headers are developed. Here is a list of the ones currently on the plate:

- Hop by Hop Options Header
- Destination Options Header I
- Routing Header
- Fragment Header
- Authentication Header
- Encapsulating Security Payload Header
- Destination Options Header II

The preceding list shows the selection of Next-Header fields that can occur in an IPv6 packet. These headers are listed in order of the appearance they would make in an IPv6 packet utilizing this extra functionality. All these headers will be discussed in detail in the previous chapter.

IPv4 and IPv6 Address Space

IPv4 and IPv6 addresses are similar in that they both utilize network and host portion subnetting and CIDR notation to express the address. However, IPv6 utilizes a much larger address space and provides a much different policy for allocation of addresses to support aggregation. In this section, we'll discuss these differences and present the IPv6 address syntax.

As discussed in the introduction, the growth and stability of the IPv4 address space is severely constrained. IPv4 addressing utilizes a 32-bit value that is represented in a four-section dotted decimal format. Each section contains 8 bits that represent byte values between 0 and 255. This allows for a theoretical number of addresses in the range of four billion. However, many of these addresses are reserved, resulting in an actual number of addresses much lower than four billion.

While this may seem to be an inordinate amount of address space, the majority of this space has been allocated to Internet service providers. It is almost impossible for independent businesses (non service providers) to obtain their own IPv4 address space. Most service providers have obtained their own IPv4 address space from address allocation agents such as the American Registry for Internet Numbers (ARIN), Réseaux IP Européens (RIPE), or the Asia Pacific Network Information Centre (APNIC). The aforementioned agents must follow stringent policies for allocation of address space. Most providers adopt these allocation policies when assigning address space to their customers. ARIN, RIPE, and APNIC allocate subsets of the IPv4 address space to providers who in turn assign IPv4 addresses to end users. The allocation and assignment policies are defined in RFC 2050. Specifically, the policy is as follows:

1. End users should request address space from their directly connected upstream provider.
2. If no addresses are available from the upstream provider, request addresses from the provider's provider.
3. If justifiable, request address space directly from ARIN, RIPE, or APNIC.

ARIN, RIPE, and APNIC are responsible for determining the number of addresses assigned to providers. Allocations are based solely on need and not on the providers' projected customer allotments and are defined on CIDR bit boundaries. Generally, a /20 or lower (/19, /18, and so on) is the smallest prefix assigned to the providers. If a prefix of /20 or longer is needed, the preceding rules are utilized. ARIN, RIPE, and APNIC use a slow-start model for allocating the IPv4 address space. This means that addresses are allocated out of a subset of a CIDR block of addresses. The CIDR block is reserved and providers can request additional addresses out of the reserved space, when exhaustion of the allocated addresses is justified.

Address space allocated by ARIN, RIPE, or APNIC is globally routable, but the prefixes assigned to end customers may not be if the customer has dual providers or if they have smaller prefixes that are not advertised by their upstream providers. CIDR aggregation blocks are used to minimize the size of the Internet routing table. Most providers have policies regarding the advertisement of small prefixes (such as a /24 prefix). The exhaustion of the IPv6 address space is the primary reason for the introduction of IPv6 addressing.

The IPv6 address space utilizes a 128-bit format that consists of an eight-part hex address separated by colons (**:**). Each part of the IPv6 address space represents sixteen bits, which provides a theoretical address space of 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses. Due to the fact that IPv6 has reserved addresses, the total assignable address space is smaller than this. The size and scope of the IPv6 address space enables allocations of addresses in a much more hierarchal fashion than IPv4, which in turn enables independent customers of service providers to obtain and deploy globally routable addresses within their environments. This functionality is especially useful in environments that have been forced to utilize Network Address Translation (NAT) due to the inability of the customer to obtain public IPv4 address space. The basic premise of NAT is that it takes a non-globally routable (private) address space from a service provider customer and translates it to a globally routable (public) address before forwarding it on to the Internet. NAT solutions usually require additional hardware and software (that is, a firewall) or resource intensive processing on a customer's router. In addition, NAT introduces unnecessary complexities that can be avoided completely with the deployment of IPv6 addressing. The IPv6 address space provides more than enough address space to reassign public IPv6 addresses to all entities that require global routing over the Internet.

IPv6 Address Structure

As discussed previously, IPv6 utilizes hex notation. This is a fundamental change from the dotted decimal notation used in the IPv4 addressing. Utilization of dotted decimal notation to express and address space equivalent to the size of the IPv6 scope would be complex and cumbersome. To express the same address space as IPv6 in decimal notation, the current IPv4 string would have to be expanded four times. If dotted decimal notation were used, addresses would appear as 15.25.35.45.55.65.75.85.95.105.115.125.135.145.155.165, which would make it very difficult to remember IP addresses. Using hexadecimal notation allows for the expression of these numbers using two hexadecimal numbers. For example, an IPv4 address can be expressed in hexadecimal notation with two numbers per octet. The IPv4 address 100.64.172.255 converts to 64.40.AC.FF in hexadecimal. It is important to understand how to convert from decimal to hexadecimal values in order to express IPv4 addresses as IPv6 addresses. Decimal to hexadecimal conversion is easily done with a conversion calculator; however, you may not always have access to a calculator. Conversion from decimal to hexadecimal and vice versa is a necessary evil for anyone considering IPv6, especially if

IPv4 addresses are incorporated into the IPv6 address. Table 4.1 depicts decimal to hexadecimal equivalents. This table is useful when converting smaller numbers.

Table 4.1 Decimal to Hexadecimal Equivalents

Decimal Notation	Hex Notation
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	a
11	b
12	c
13	d
14	e
15	f

I like to use a simple procedure when converting from decimal to hex for numbers larger than 255. The procedure is as follows.

1. Divide the decimal number by 16.
2. Write down the remainder as a hex number.
3. Take the result from step 1 and divide by 16 again.
4. Continue until you cannot divide by 16 again (or until the result is 0).
5. The conversion to hex is the hex numbers found in step 2 written from last to first.

The entire formula and procedure can be summarized by the following chart and procedural steps.

X	Number you want to convert
Y	Number of times 16 goes into X
Y1	Number of times 16 goes into Y
Y2	Number of times 16 goes into Y2
Yn	Number of times 16 goes into Yn-1
R1	Remainder1
R2	Remainder2
Rn	Remaindern
H1	R1 in hex
H2	R2 in hex
Hn	Rn in hex

1. X/16 = Y remainder R1
2. R1 converts to H1
3. Y/16 = Y1 remainder R2
4. R2 converts to H2
5. Y1/16 = Y2 remainder R3
6. R3 converts to H3
7. Yn/16=Yn-1 remainder Rn
8. Rn converts to Hn
9. Hexadecimal equivalent is HnH3H2H1

For example, to convert 35,000 decimal to hexadecimal, do the following:

1. 35,000/16 = Y = 2187 with a remainder of 8 = R1
2. R1 = 8, which converts to 8 in hex, which equals H1
3. From step 1, Y = 2187, so 2,187/16 = Y1 = 136 with a remainder of 11 = R2
4. R2 = 11, which converts to b in hex, which equals H2
5. From step 3, Y1 = 136, so 136/16 = Y2 = 8 with a remainder of 8 = R3
6. R3 = 8, which converts to 8 in hex, which equals H3
7. 8/16 is not divisible in this scenario so the R4 = 8

8. R4 = 8, which converts to 8 in hex which, equals H4
9. HEX equivalent is expressed as H4H3H2H1 (from above H4=8, H3=8, H2=b, H1=8)
10. 35,000 = 88B8

This formula is good to know, but conversion from hex to decimal with numbers less than 255 is more likely to be useful as the initial deployments of IPv6 may use many IPv4 addresses. The procedure is as follows.

1. Divide the decimal number by 16.
2. Record the value as a hex number.
3. Record the remainder as a hex value.

For example: 232 decimal converted to hexadecimal is done as follows:

1. 232/16 = 14 with a remainder of 8.
2. 14 = E in hex.
3. 8 = 8 in hex.
4. Hex equivalent equals the two numbers together starting with step 2; therefore, 232 = E8.

You might ask how hexadecimal numbers are used within the IPv6 addressing architecture? RFC 2373 provides an addressing structure for IPv6 addressing architectures. Cisco requires that all addressing comply with RFC 2373. Although the hexadecimal notation shortens the number of digits required to express a decimal value, the address structure is much longer than IPv4 due to the amount of address space.

As previously discussed, IPv6 addresses utilize a 128-bit format that consists of an eight-part hex address separated by colons (**:**). Therefore an IPv6 address is expressed as follows: ADBF:0:FEEA:0:0:00EA:00AC:DEED

IPv6 provides two methods for compressing the syntax of the address space. The first is the omission of leading zeroes and the second is the replacement of multiple groups of zeroes by double colons (**::**). Using these methods, the preceding address can be shortened considerably.

For example, using the first method, omitting the leading zeroes, provides an address of ADBF:0:FEEA:0:0:EA:AC:DEED. If the second method is applied, the address is represented as ADBF:0:FEEA::EA:AC:DEED. However, the double colon can appear only once in the address. In addition to replacing multiple groups of zeroes within the address, the double colon can be used to represent the leading or trailing zeroes in an address.

RFC 2373 outlines a structure for representing IPv4 addresses. The six high-order bits of the IPv6 address space are represented as leading zeroes and the two remaining 16-bit spaces are broken into four 8-bit spaces. These represent standard IPv4 addresses. This provides an address in the form of 0:0:0:0:0:0.A.B.C.D, where ABCD is expressed in standard IPv4 syntax. For example, the IPv4 address of 192.168.100.10 can be represented as 0:0:0:0:0:0.192.168.100.10. Notice that within the IPv6 address the IPv4 address space is delimited by dots (**.**), not colons. The IPv6 address can be further truncated by omitting the leading zeroes. The result is an IPv6 address in the form of ::192.168.100.10.

The Internet community adopted CIDR notation to extend the life of the IPv4 address space. IPv6 also utilizes CIDR notation. CIDR does not solve the problem of multiple entries in the Internet routing tables. CIDR presents a method for aggregating address space only outside the normal Class A, B, and C network boundaries. Let's say for example that a customer has a connection to Provider A, which has been assigned the address block of 100.100.0.0 – 100.100.255.255 (or 100.100.0.0/16). Provider A can allocate addresses out of this block for a customer to use as public address space. Let's say that the provider allocates 100.100.100.0-100.100.100.255 (or 100.100.100.0/24) to the customer. Provider A announces an aggregate address of 100.100.0.0/16 to everyone on the Internet. This simply means that anyone trying to reach an address in the 100.100.0.0/16 address space can forward traffic to Provider A. Therefore, all traffic destined for 100.100.100.0/24 (the customer) goes through Provider A's backbone. This works fine until the customer decides that he wants to implement a dual-homed connection to the Internet. Provider B enters the picture as a secondary provider of Internet connectivity. Because the customer already has the address space of 100.100.100.0/24, Provider B must advertise this route in its routing table. This means that now two routes to get to 100.100.100.0/24 exist, one through Provider A's aggregated route of 100.100.0.0/16 and one through Provider B's 100.100.100.0/24 route. It is not as important to understand how the routing works as to understand that CIDR in combination with smaller prefixes carved out of the CIDR aggregation create multiple routes in the Internet routing table. For every customer who wishes to dual home to different providers, multiple entries to the customer's networks are injected into the Internet routing tables. If the multiple routing entries seen in IPv4 routing tables also were present in the IPv6 address space, the impact would be enormous. RFC 2373 provides mechanisms for subnetting the IPv6 address space so that it does not negatively impact the Internet routing tables. A discussion of the mechanisms used for subnetting is found in the "Aggregatable Global Unicast Address" section later in this chapter.

The next sections discuss the types of addresses defined within IPv6 and their uses. As with IPv4 addresses, specific types of reserved addresses divisions are within the IPv6 addressing architecture. The leading bits within the address identify reserve addresses. Table 4.2 shows the reserved addresses.

Table 4.2 IPv6 Address First-Bits Standards

Allocation	Prefix (binary)	Fraction of Address Space
Reserved	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128
Reserved for IPX Allocation	0000 010	1/128
Unassigned	0000 011	1/128
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Aggregatable Global Unicast Addresses	001	1/8
Unassigned	010	1/8
Unassigned	011	1/8
Unassigned	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512
Link-Local Unicast Addresses	1111 1110 10	1/1024
Site-Local Unicast Addresses	1111 1110 11	1/1024
Multicast Addresses	1111 1111	1/256

Each of these reserved address spaces defines a specific type of address. The addresses defined have scope. A node or router may have multiple addresses configured that define the scope as local to the link, local to the site, or globally significant. In addition, there are loopback and local use, multicast, and anycast addresses. The types of addresses are as follows:

- Unicast Address
- Multicast Address
- Anycast Address
- Nodes-Required Address

Unicast Addresses

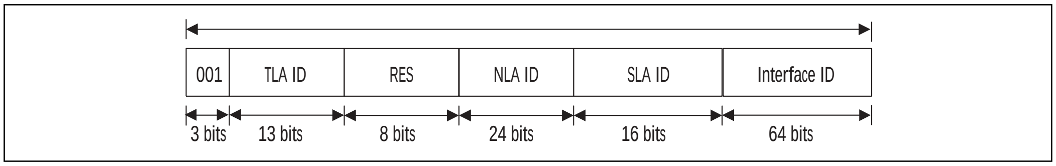
A *unicast* address represents an end node or host. Unicast addresses are aggregatable and require the use of contiguous bits to represent the subnet mask. By defining clear and concise definitions within the address structure, IPv6 allows for a robust addressing architecture that provides a much more structured hierarchy when allocating address space. IPv6 unicast addresses are segmented into the following formats, which we'll look into in the next sections:

- Aggregatable Global Unicast Addresses
- The Loopback Address
- The Unspecified Address
- Interface Identifiers
- Local-use Unicast Addresses
- NSAP Addresses
- IPX Addresses

Aggregatable Global Unicast Address

The *Aggregatable Global Unicast* address format is used to define a tiered structure for the assignment and allocation of the IPv6 address space. The new structure calls for the address space to be broken into six separate components. These components are the Format Prefix (FP), the Top-Level Aggregation Identifier (TLA ID), Reserved (RES), Next-Level Aggregation Identifier (NLA ID), Site-Level Aggregation Identifier (SLA ID), and the Interface Identifier (Interface ID). Figure 4.2 depicts the global unicast address format.

Figure 4.2 Global Unicast Address Format



The public routing topology prefix consists of the first 48 bits of unicast address space. This includes the FP bits plus the TLA, Reserved, and NLA bits. The next 16 bits define the site topology and the final 64 bits are the interface identifier. The public topology is address space assigned to exchanges and ISPs that provide transit Internet services. The site topology can be defined as the customers of the providers and exchanges and the interface identifier is an end host or node interface identifier.

FP The *Format Prefix* for Globally Routable Unicast prefixes will always have the same three bits (in the initial deployment of IPv6). These first three bits will always be set to 001 and are there to designate (to any routing entity on the Internet) that this address is a Globally Routable Unicast address. For each type of IPv6 address that we discuss, the FP will be unique to that type of address, thus making it easier for routing entities to discern packet types and to process them according to the rules that apply to the respective packet type. For example, multicast packets and unicast packets are routed in very different ways. Unicast packet routing is 1-to-1 (a packet with an IPv6 Globally Routable Unicast destination originates from one host and is delivered to one host), and multicast packets are 1-to-N (one multicast packet may be delivered to N interested destination hosts), or N-to-N (N sources delivering packets to N destinations), so these packets are handled in vastly different ways on an Internet backbone. The FP serves as a delimiter, so a routing device can make a quick decision as to how to handle the incoming packet and ensure that it is handled correctly. Note that using the first few bits of an address to designate type of address is more efficient than putting it into the packet, because now we can utilize more of the packet for other valuable features, discussed earlier.

TLA ID The *TLA ID* utilizes 13 bits that provide for 8,192 TLAs. This means there can be 8192 providers or exchanges at this level. This is analogous to today's Tier-1 providers. The TLAs reside at the highest point of the routing hierarchy. TLAs will be assigned one of the 8,192 TLA IDs and will own responsibility for allocating addresses to downstream customers.

NOTE

The Internet community discusses intently what defines a Tier-1 provider. Possibly the simplest definition is that these are providers that do not pay peering costs to exchange routing information with each other. However, no standard definition exists, nor are there any rules governing who can peer with whom. This lack of standards and rules leads to the question of who will be TLA providers and who will make this decision. This is important in terms of which providers will be classified as TLAs and own one of the 8,192 address blocks at the TLA level and how they plan their address allocations.

RES These bits are *reserved* for now. It has not been determined by the IETF what course of action should be used for these bits. At this stage, it is appropriate for TLAs to subnet their assignment using these 8 bits to increase the amount of Globally Routable Unicast address space that a TLA can use to delegate to their customers or use on their Backbone.

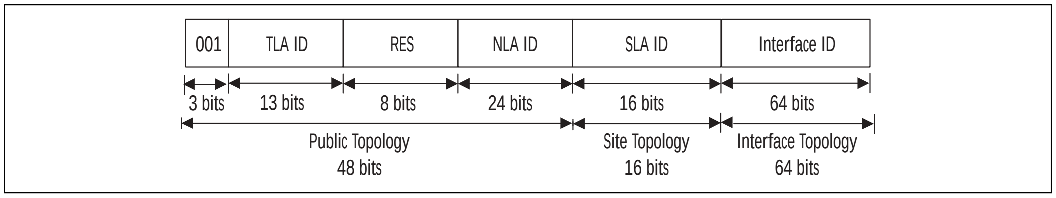
NLA ID These 24 bits depict the *Next-Level Aggregator Identifier*. A Next-Level Aggregator can be thought of today as a Tier-2 network service provider or ISP. An NLA can range from a small organization with one TLA connection, to a large, regional provider with many upstream TLA connections and complex backbones. An NLA will receive an NLA ID from its upstream TLA, and, in turn, will break its NLA ID into chunks, which will be delegated to its customers.

SLA ID A *Site-Level Aggregator Identifier* describes an entity that has no downstream customers who are network service providers. An SLA could be a small to large business, or a small service provider who does not delegate address space to its providers (for instance, today's cable-modem providers could fit into an SLA arrangement).

Interface ID The final 64 bits of the Globally Routable IPv6 Unicast address is reserved for the *Interface Identifier*. In IPv4 terms, this is known as the host ID. These 64 bits will be designated to distinguish one host from another on a given network segment. Each Interface ID on a given network segment must be unique. We will see that IPv6 builds in a clever way to ensure this is so.

To effectively understand IPv6 addressing, it is necessary to understand how the address space is allocated. The policies governing the assignment and allocation of the IPv6 address space have been set forth in a published standard entitled ripe-196. This abstract defines the Internet registry system for the distribution of IPv6 unicast addresses. The goal is to ensure that the IPv6 address space is managed properly. This management is necessary to eliminate the inconsistencies and unfairness of address allocation seen in the IPv4 address space. This minimizes waste of address space and maximizes aggregation. The authority for all IPv6 address space is the Internet Assigned Numbers Authority (IANA). IANA allocates IPv6 address space to regional Internet Registries (IRs). There are three IRs: ARIN, RIPE Network Cordination Centre (NCC), and APNIC. ARIN serves North and South America, the Caribbean, and sub-Saharan Africa; RIPE NCC handles Europe, the Middle East, and some parts of Africa; APNIC handles areas in the Asia Pacific region. The IRs can handle areas outside their administrative control if necessary. Additional IRs can be created as the deployed base of IPv6 addresses grows. The policies governing the assignment of IPv6 unicast address space must be done in a manner that ensures that each unicast address is efficiently allocated, globally routable, unique, and supports aggregation. RFC 2374 (the update to RFC 2373) organizes the IPv6 address space into a topological hierarchy. The topologies are public topology, site topology, and interface. Figure 4.3 depicts the mapping of these topologies to the IPv6 address space.

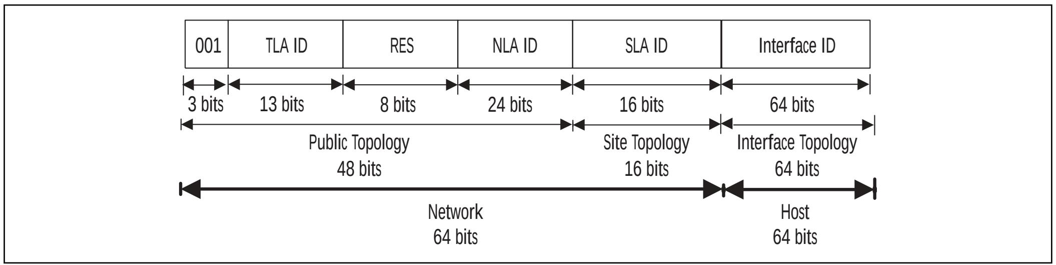
Figure 4.3 Unicast Address Hierarchical Topology



An IPv6 address can be expressed in the same manner as IPv4 addresses. An IPv4 address contains network and host bits. The network and host portions of an IPv6 address are shown in Figure 4.4.

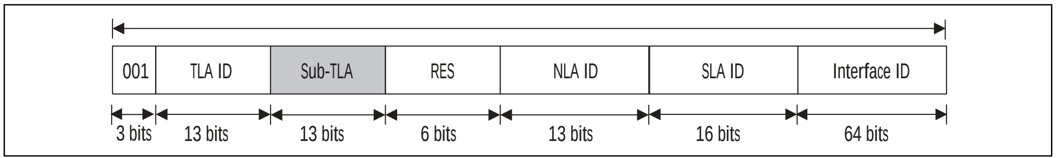
The network portion of an IPv6 address is 64 bits and the host portion is 64 bits. The expression of IPv6 addresses uses CIDR notation to divide the address into network and host portions. An address with a /48 mask represents an aggregatable network prefix assigned out of the public topology allocation. Subnetting and address assignment are discussed in detail later in this chapter.

Figure 4.4 Unicast Address Network and Host Bits



The regional IRs use a modified version of the IPv6 address TLA field for allocation of the initial address space. This requires dividing the TLA space into a sub-TLA, which allows for allocating less of the address space to the original TLAs as previously planned. The IR reserves the additional 6 bits for the TLA so that if the need arises, it can allocate the address space to the TLA. The reservation preserves the aggregation policies but allows for efficient allocation on an as-needed basis. This policy creates a modified format to the IPv6 address space. The modified version is shown in Figure 4.5.

Figure 4.5 Modified Sub-TLA Format



The address allocation policy for IPv6 is as follows:

- Regional IRs assign addresses to qualified sub-TLAs (TLA ISPs).
- TLA ISPs assign NLA addresses to NLA ISPs (TLA customers).
- NLA ISPs assign SLA addresses to their customers.

TLA providers have to be able to allocate SLA addresses to customers. TLA ISPs provide direct Internet connectivity to end users as well as NLAs. Subnetting the address space within the TLA and NLA providers' networks is left up to the discretion of the individual providers. Regardless of how providers subnet the address space within their networks, aggregation of the address space falls on the prefix boundaries shown in Table 4.3.

Table 4.3 Aggregation Prefix Boundaries

ID	Longest Prefix	Length in Bits
TLA	/16	13
Sub-TLA	/29	13
Reserved		
N:LA	/48	13
SLA	/64	16

The Loopback Address

The loopback address is a part of the IPv6 unicast address space. The loopback interface is not a physical interface and has no hardware associated with it. It is a software interface that is always reachable regardless of the physical interface status. RFC 2373 defines the loopback interface address as 0:0:0:0:0:0:0:1, or ::1 in condensed form. An IPv6 node uses the loopback interface to send packets to itself and can never be used as a source address for IPv6 packets sent outside a single node (router). The standard prohibits a node from forwarding a packet with a destination address of a loopback.

The Unspecified Address

The unspecified address is another part of the IPv6 unicast address space. The unspecified address is never assigned to any node. In addition, the unspecified address may never be used as a destination address in an IP packet nor can it be used in routing headers. Unspecified addresses are used during the auto configuration process. Auto configuration is discussed in Chapter 2.

Interface Identifiers

Interface identifiers are used to uniquely identify interfaces on a link. IPv6 utilizes a modified EUI-64 format to construct interface identifiers. EUI-64 addresses are a derivation of the 24-bit company_id value assigned by the IEEE registration authority. Conversion to the EUI-64 format is accomplished by modifying the 48-bit Media Access Control (MAC) address assigned to the hardware interface. Other interfaces such as tunnel, Frame-Relay, and loopback derive the

EUI-64 identifier via MAC addresses from the router. Cisco routers contain pools of MAC addresses that are assigned for internal identifiers. The exceptions to these rules are tunnels that are used with IPv6 overlay tunnels. For these types of interfaces the Interface Identifier is the IPv4 address with zeroes in the high-order 32 bits.

To create EUI-64 addresses from vendor MAC addresses, perform the following steps:

1. Insert **ff-ee** between the third and fourth bytes of the MAC address.
2. Complement the "Universal/Local" (U/L) bit. (The U/L bit is the 'u' bit in IEEE EUI-64 terminology. Complementing this bit means to invert the bit from the original value and set it to either a 0 or 1. Setting the bit to 0 means that the interface identifier is locally administered; setting it to 1 indicates a globally unique interface identifier.)

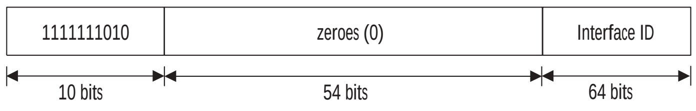
Local-use Unicast Addresses

Two types of local-use addresses exist within the IPv6 address space:

- Link-local addresses (used on a single link)
- Site-local addresses (used within a site)

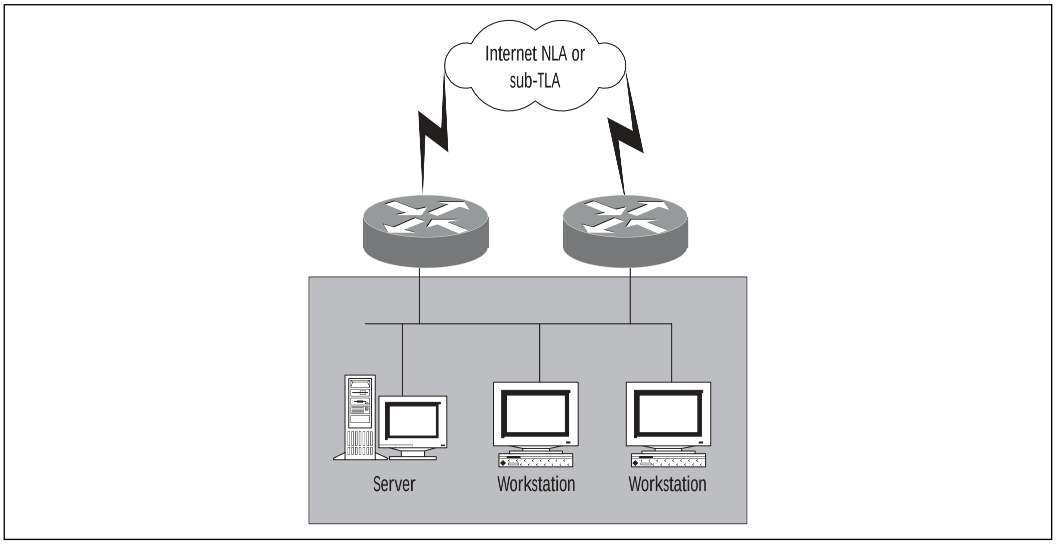
Link-local addresses are significant only on the physical medium connected to the router (that is, Ethernet, Token Ring, or WAN segment). Only the hosts and router interface connected to the same segment are aware of the link-local addresses for that segment. The router does not advertise link-local addresses, which considerably simplifies renumbering. After the addresses have been assigned (either manually or via auto configuration), the global unicast address space can change without having to revisit and reconfigure the link-local addresses. In addition, link-local addressing is used in neighbor discovery and internal routing. Figure 4.6 shows the format for the link-local address.

Figure 4.6 Link-Local Unicast Address Format



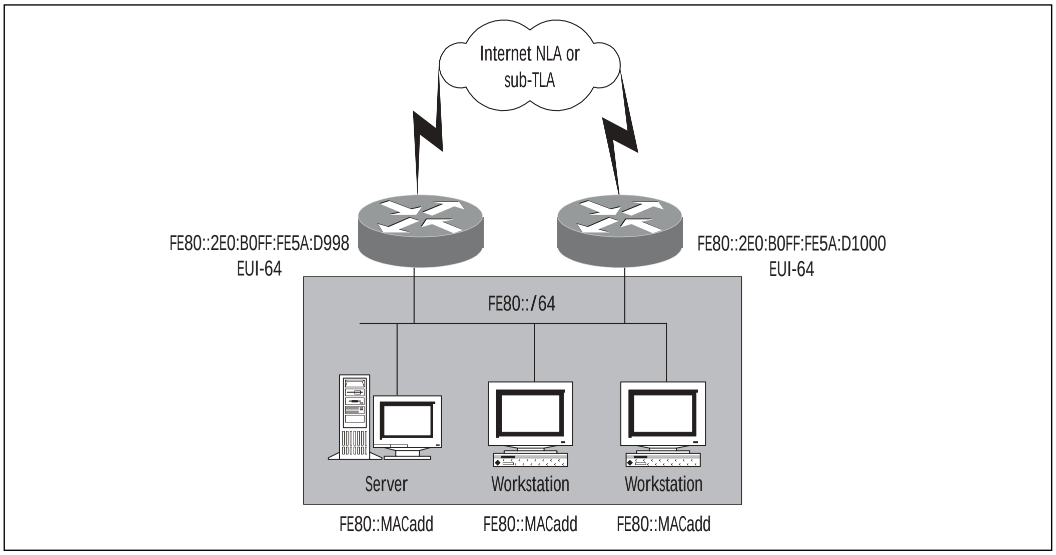
The left-most 10 bits shown in Figure 4.6 translate into the prefix FE80::/64. Appending the Interface ID to the FE80::/64 prefix derives link-local addresses. Figure 4.7 depicts a segment utilizing link-local addressing.

Figure 4.7 Link-Local Address Space



Link-local devices are configured using a MAC address or the EUI-64 format prepended by FE80. Figure 4.8 depicts the address assignments for the devices in link format.

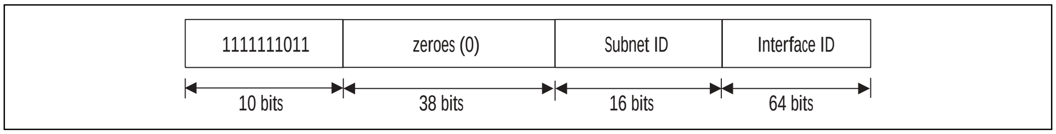
Figure 4.8 Link-Local Assigned Addresses



In this example, the routers are utilizing EUI-64 format for the Interface ID; the server and workstations are using MAC addresses. These addresses are never visible outside the local segment on which they are configured. If the devices need to be reached from hosts outside the local segment, site-local unicast addresses are assigned.

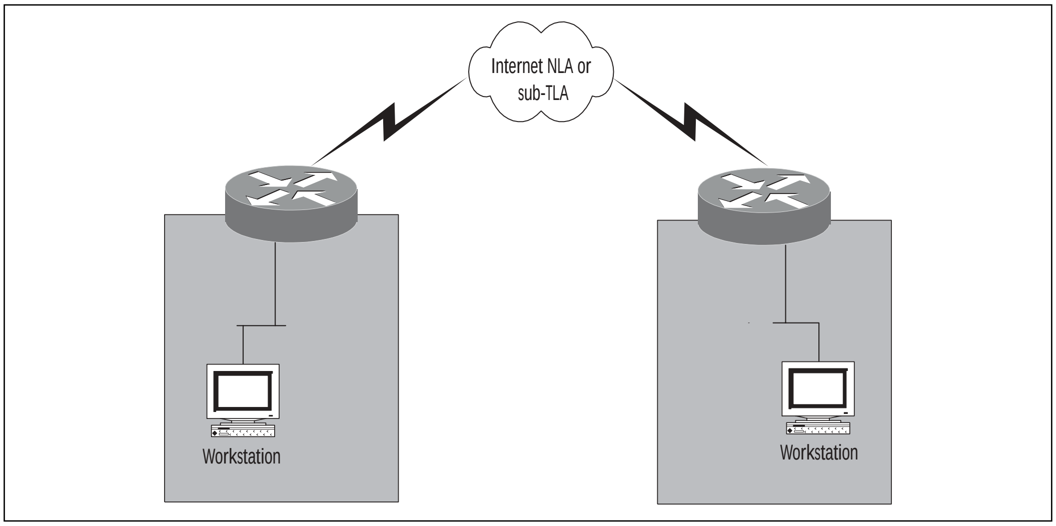
A *site-local address* is an address that is routable within a site only. This means that hosts that are configured with a site-local address can communicate with other hosts within the same environment but are not globally routable. These addresses are assigned without a global unicast prefix. Site-local addresses are used in the same fashion as RFC 1918 addresses in the IPv4 world. Site-local addresses are considered private and form the addressing structure for an internal domain. Figure 4.9 shows the format for the site-local address.

Figure 4.9 Site-Local Address Format



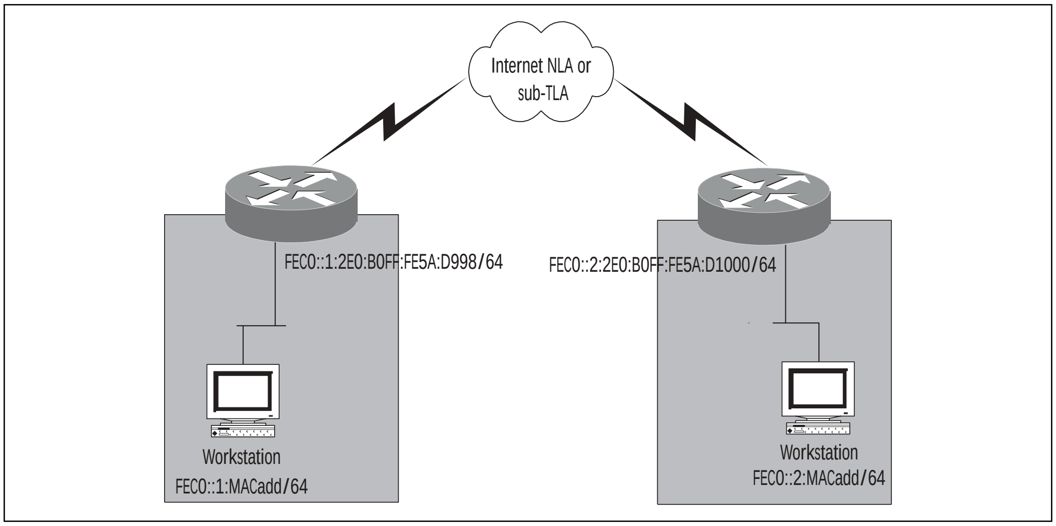
The left-most 10 bits shown in Figure 4.9 translate into the prefix FEC0::/10. Concatenating the 16-bit SLA field (subnet ID) with the interface ID derives the site-local address. Figure 4.10 shows a topology that utilizes site-local addresses.

Figure 4.10 Site-Local Address Space



Site-local devices are configured using a MAC address or the EUI-64 format prepended by FEC0. Figure 4.11 depicts the address assignments for the devices in site-local format.

Figure 4.11 Site-Local Assigned Addresses



In this example, the routers utilize site-local addresses and interface IDs in the EUI-64 format. The workstations are using the MAC address as the interface ID. The numbers 1 and 2 before the interface ID represent the subnet ID. Workstations on subnet 1 can communicate with workstations on subnet 2 and vice versa. These addresses are never advertised outside the internal domain to the global Internet. If future Internet connectivity is required, it makes sense to initially configure the host with a Globally Routable Unicast address. Again, this introduces the issue of renumbering in the event that the customer changes providers.

NSAP Addresses

IPv6 allows for the mapping of NSAP addresses into IPv6 addresses. NSAP addresses are part of the OSI protocol suite developed by the International Organization for Standardization (ISO). OSI was the predecessor of IPv4 routing. The OSI model incorporates the End System-to-Intermediate System (ES-IS), Intermediate System-to-Intermediate System (IS-IS), and Interdomain Routing Protocol (IDRP) suites. Because providers have deployed Internet backbones utilizing these protocol suites, IPv6 allows for the mapping of the NSAP addresses into IPv6 addresses. It is recommended that providers that have deployed networks with an OSI NSAP addressing plan migrate to an IPv6 plan. RFC 188 provides information on the mapping of NSAP to IPv6 addresses. This standard is still under development.

IPX Addresses

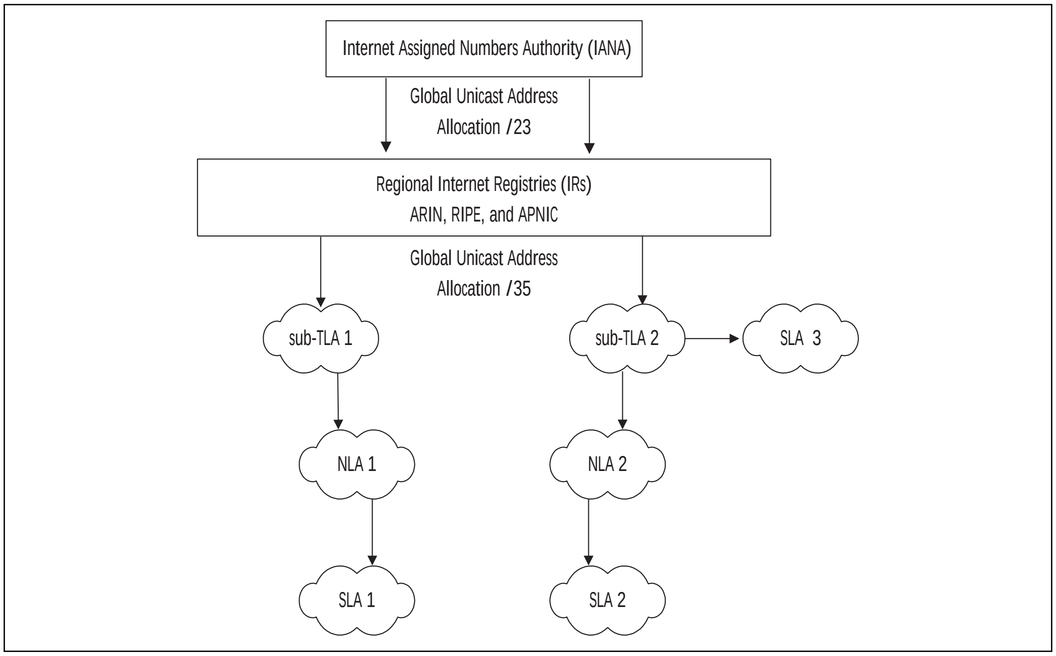
The standard for mapping IPX addresses into IPv6 addresses is also under development.

Subnetting and Prefixes for IPv6 Aggregation

As stated, IPv6 provides a much larger address space than IPv4. IPv6 was developed with a structure and format that is conducive to high levels of aggregation. In the previous section, you learned that IPv6 addresses are broken up into prefixes of TLA, sub-TLA, NLA, and SLA allocations. Providers allocate addresses in a hierarchical fashion that provides aggregation from the lower-level prefixes to the higher-level TLAs. RFC 2373 provides a clear method for the allocation of the IPv6 address space.

ARIN, RIPE, and APNIC allocate sub-TLA address space to the TLA providers. These providers in turn allocate NLA address space to downstream providers who in turn allocate to SLAs. Remember that TLAs can also allocate SLAs. Figure 4.12 depicts the hierarchy of global unicast address allocation.

Figure 4.12 Global Unicast Address Allocation


Internet Assigned Numbers Authority (IANA)
Global Unicast Address Allocation /23
Regional Internet Registries (IRs) ARIN, RIPE, and APNIC
Global Unicast Address Allocation /35
sub-TLA 1
sub-TLA 2
SLA 3
NLA 1
NLA 2
SLA 1
SLA 2

The following IPv6 address assignments have been allocated to the three IRs:

2001:0400::/23 – ARIN

2001:0200::/23 – APNIC

2001:0600::/23 – RIPE

We will use ARIN as an example to demonstrate the aggregation capabilities of IPv6. ARIN has adopted assignment guidelines for IPv6 address allocation from RFC 3177. These guidelines provide the following recommendations for IPv6 address space allocation:

- Allocate /35 addresses to TLAs—this is a sub-TLA address space.
- TLAs allocating address space to NLAs are responsible for prefix/ network length determination.
- As a general rule, allocate a /48 to all SLAs (except for very large SLAs).
- Allocate a /64 to SLAs that have only one subnet.
- Allocate a /128 to SLAs with only one subscriber.

These policies relate more to SLA end customers, but they provide efficient guidelines that facilitate aggregation. ARIN has the responsibility for assigning the slow-start sub-TLA addresses out of the 2001:0400::/23 block to providers. Let's say that ARIN allocates the following addresses:

2001:0420::/35 - sub-TLA1

2001:0428::/35 - sub-TLA2

This allocation gives each of these sub-TLA providers the capability to subnet this space and allocate to the NLAs. So rules govern the subnetting of the NLA address space. This subnetting should be done at the NLA provider's discretion. The TLAs can allocate the following address space to the downstream NLAs:

- Sub-TLA1 provides the following prefixes to its NLA customers:

  2001:0420:0001::/48

  2001:0420:0002::/48

  2001:0420:0002::/48 etc
- Sub-TLA2 provides the following prefixes to its NLA customers

  2001:0428:0001::/48

2001:0428:0002::/48

2001:0428:0002::/48 etc

Each NLA provider can subnet and allocate the address space delegated to them by the sub-TLAs. Using the guidelines established in RFC 3177, the NLAs should allocate /48s to their downstream SLA customers. For example, an NLA provider with the address space of 2001:0420:0001::/48 can allocate the following address space:

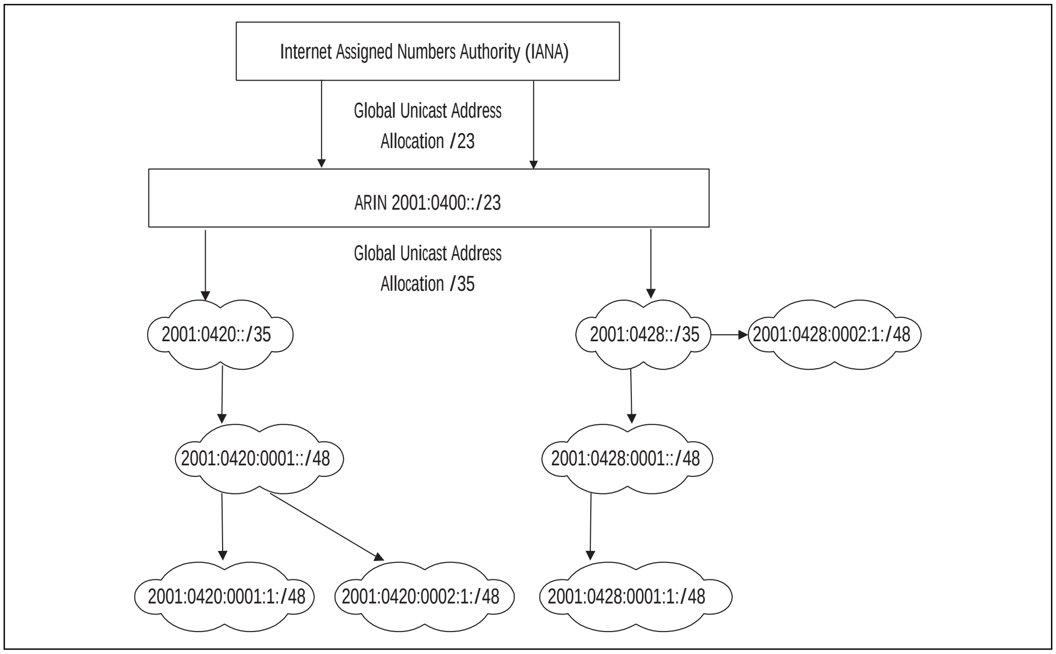
2001:0420:0001:1:/48

2001:0420:0001:2:/48

2001:0420:0001:3:/48 etc

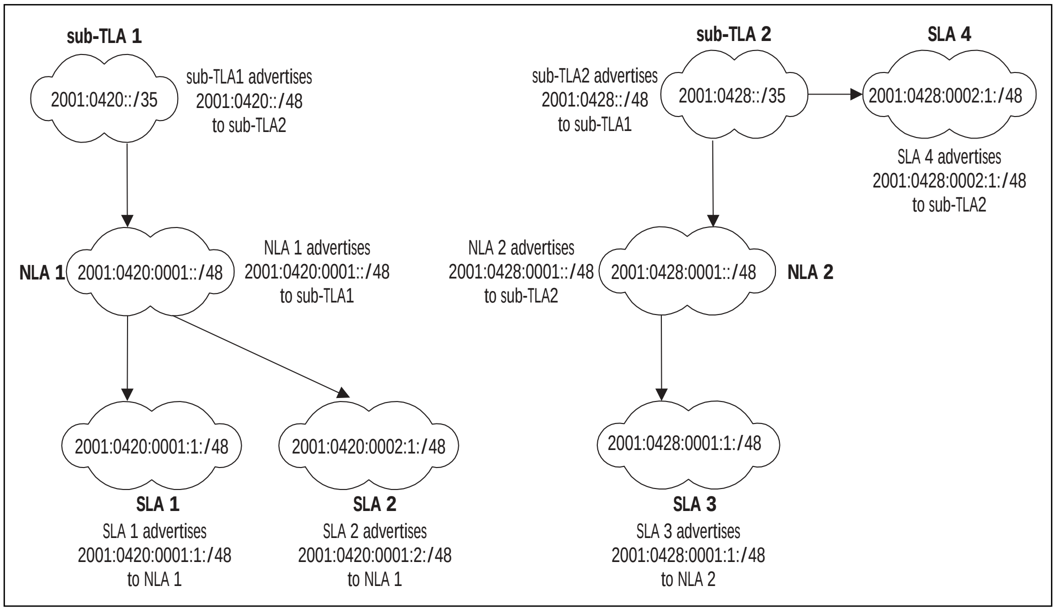
The allocation policies described previously provide the aggregation desired for a hierarchical IPv6 infrastructure. Figure 4.13 shows the distribution of the ARIN-allocated IPv6 addresses to the NLA providers and SLA customers.

Figure 4.13 ARIN



Internet providers use BGPv4 to propagate addresses throughout the Internet. In today's Internet, addresses are aggregated and advertised to the upstream provider when possible. The inefficiencies of IPv4 allocation and advertisement policies have created an abundance of routes with a small prefix. With the IPv6 aggregation capabilities, the SLA can advertise the entire prefix assigned to them to the NLA. All SLA space is a subset of the NLA that the customer is connected to. This enables the NLA to generate one prefix announcement to the sub-TLA for all SLA customers. The same holds true for the NLA space. The NLA allocates all addresses out of the sub-TLA assignment, enabling the sub-TLA to generate one prefix advertisement to its peers. Figure 4.14 depicts the aggregate announcements.

Figure 4.14 Aggregate Route Announcements



sub-TLA 1
2001:0420::/35
sub-TLA1 advertises 2001:0420::/48 to sub-TLA2
sub-TLA 2
2001:0428::/35
sub-TLA2 advertises 2001:0428::/48 to sub-TLA1
SLA 4
2001:0428:0002:1:/48
SLA 4 advertises 2001:0428:0002:1:/48 to sub-TLA2
NLA 1
2001:0420:0001::/48
NLA 1 advertises 2001:0420:0001::/48 to sub-TLA1
NLA 2 advertises 2001:0428:0001::/48 to sub-TLA2
2001:0428:0001::/48
NLA 2
2001:0420:0001:1:/48
2001:0420:0002:1:/48
2001:0428:0001:1:/48
SLA 1
SLA 1 advertises 2001:0420:0001:1:/48 to NLA 1
SLA 2
SLA 2 advertises 2001:0420:0001:2:/48 to NLA 1
SLA 3
SLA 3 advertises 2001:0428:0001:1:/48 to NLA 2

Using this hierarchical aggregation model substantially reduces the size of the Internet routing table. The routing table for sub-TLA1 comprises only NLA prefixes and prefixes learned from other sub-TLAs (or TLAs). Address space assigned to TLAs is not portable. This means that the addresses have to be advertised to the TLA that allocated the address space, which cuts down significantly on the de-aggregation of addresses as is seen in the current IPv4 allocations. Non-portability does have its drawbacks. If a customer is unsatisfied with the level of service provided by her upstream provider and switches providers, the entire network has to be renumbered out of the new provider's address space.

Switching providers creates issues not only with addressing, but also with the subject of dual homing. IPv6 address space is not intended to be portable; in other words, a secondary provider does not accept prefix advertisements from other providers, which prevents customers from dual homing to multiple providers. This topic is discussed in detail in "The Need For Further Development" section later in this chapter.

Configuring & Implementing…

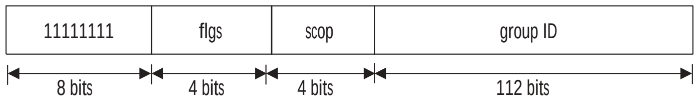
IP Renumbering

Readdressing an environment to a new IP addressing scheme is not a trivial exercise. It takes careful planning and execution to renumber without impacting normal business operations. This exercise has been simplified by the widespread use of DNS, but all environments have their share of static IP addresses. Hosts with static IP addresses must be physically reconfigured. In addition, many applications have static IP addressing embedded. These factors *must* be taken into consideration when considering renumbering.

Multicast Address

IPv6 multicast addresses are used to identify groups of interfaces. Packets are sent from a single host to multiple receivers as defined by the multicast address. A router is not limited to only one multicast group and may belong to several multicast groups. A multicast address is identified by the presence of eight 1 bits at the start of the IPv6 address. Figure 4.15 shows the structure of the IPv6 multicast address.

Figure 4.15 Multicast Address



The *flgs* field is a set of 4 bits with the first three reserved and the fourth used to flag whether the multicast address is permanently or non-permanently assigned. A 0 indicates permanent and a 1 indicates non-permanent. The global Internet numbering authority assigns permanent multicast addresses.

The *scop* field is another set of 4 bits used to specify the scope of the multicast group. Table 4.4 outlines the values for the *scop* fields.

Table 4.4 Multicast Scope Field

Scope Value	Scope
0	Reserved
1	Node-local scope
2	Link-local scope
3	Unassigned
4	Unassigned
5	Site-local scope
6	Unassigned
7	Unassigned
8	Organization-local scope
9	Unassigned
A	Unassigned
B	Unassigned
C	Unassigned
D	Unassigned
E	Global scope
F	Reserved

The *group ID* field identifies multicast addresses as permanent or non-permanent within a scope. It is important to distinguish between permanent addresses within a scope and permanent addresses across a scope. Permanent addresses within a scope are those addresses that are reserved for multicast functions for that particular scope. For example, FF01 indicates a node-local scope. Addresses FF01:0:0:0:0:0:0:1 and FF01:0:0:0:0:0:0:2 are reserved within the FF01 scope. Permanent addresses across a scope are those addresses that have a reserved group ID value for any scope range. For example, the group ID of 100 is reserved for the VMTP Managers Group (RFC 1045). This address is a permanent address across any of the scope values shown in Table 4.4. Table 4.5 defines reserved multicast groups for both fixed scope and variable scope allocations.

Table 4.5 Reserved Multicast Addresses

Node-Local Scope

Reserved For	Multicast-Address
All Nodes Address	FF01:0:0:0:0:0:0:1
All Routers Address	FF01:0:0:0:0:0:0:2

Link-Local Scope

Reserved For	Multicast-Address
All Nodes Address	FF02:0:0:0:0:0:0:1
All Routers Address	FF02:0:0:0:0:0:0:2
Unassigned	FF02:0:0:0:0:0:0:3
DVMRP Routers	FF02:0:0:0:0:0:0:4
OSPFIGP	FF02:0:0:0:0:0:0:5
OSPFIGP Designated Routers	FF02:0:0:0:0:0:0:6
ST Routers	FF02:0:0:0:0:0:0:7
ST Hosts	FF02:0:0:0:0:0:0:8
RIP Routers	FF02:0:0:0:0:0:0:9
EIGRP Routers	FF02:0:0:0:0:0:0:A
Mobile-Agents	FF02:0:0:0:0:0:0:B
SSDP	FF02:0:0:0:0:0:0:C
All PIM Routers	FF02:0:0:0:0:0:0:D
RSVP-ENCAPSULATION	FF02:0:0:0:0:0:0:E
Link Name	FF02:0:0:0:0:0:1:1
All-dhcp-agents	FF02:0:0:0:0:0:1:2
Solicited-Node Address	FF02:0:0:0:0:1:FFXX:XXXX

Site-Local Scope

Reserved For	Multicast-Address
All Routers Address	FF05:0:0:0:0:0:0:2
All-dhcp-servers	FF05:0:0:0:0:0:1:3
All-dhcp-relays	FF05:0:0:0:0:0:1:4
Service Location, Version 2	FF0X:0:0:0:0:0:1:1000- FF0X:0:0:0:0:0:1:13FF

Continued

Table 4.5 Continued

Variable Scope	
Reserved For	**Multicast-Address**
Reserved	FF0X:0:0:0:0:0:0:0
VMTP	FF0X:0:0:0:0:0:0:100
Network	FF0X:0:0:0:0:0:0:101
SGI-Dogfight	FF0X:0:0:0:0:0:0:102
Rwhod	FF0X:0:0:0:0:0:0:103
VNP	FF0X:0:0:0:0:0:0:104
Artificial	FF0X:0:0:0:0:0:0:105
NSS	FF0X:0:0:0:0:0:0:106
AUDIONEWS	FF0X:0:0:0:0:0:0:107
SUN NIS+	FF0X:0:0:0:0:0:0:108
MTP	FF0X:0:0:0:0:0:0:109
IETF-1-LOW-AUDIO	FF0X:0:0:0:0:0:0:10A
IETF-1-AUDIO	FF0X:0:0:0:0:0:0:10B
IETF-1-VIDEO	FF0X:0:0:0:0:0:0:10C
IETF-2-LOW-AUDIO	FF0X:0:0:0:0:0:0:10D
IETF-2-AUDIO	FF0X:0:0:0:0:0:0:10E
IETF-2-VIDEO	FF0X:0:0:0:0:0:0:10F
MUSIC-SERVICE	FF0X:0:0:0:0:0:0:110
SEANET-TELEMETRY	FF0X:0:0:0:0:0:0:111
SEANET-IMAGE	FF0X:0:0:0:0:0:0:112
MLOADD	FF0X:0:0:0:0:0:0:113
any private experiment	FF0X:0:0:0:0:0:0:114
DVMRP	FF0X:0:0:0:0:0:0:115
SVRLOC	FF0X:0:0:0:0:0:0:116
XINGTV	FF0X:0:0:0:0:0:0:117
microsoft-ds	FF0X:0:0:0:0:0:0:118
nbc-pro	FF0X:0:0:0:0:0:0:119
nbc-pfn	FF0X:0:0:0:0:0:0:11A
lmsc-calren-1	FF0X:0:0:0:0:0:0:11B
lmsc-calren-2	FF0X:0:0:0:0:0:0:11C

Continued

Table 4.5 Continued

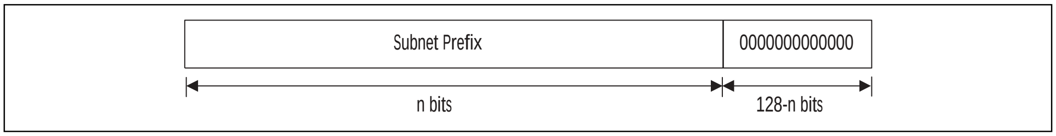
Reserved For	Multicast-Address
lmsc-calren-3	FF0X:0:0:0:0:0:0:11D
lmsc-calren-4	FF0X:0:0:0:0:0:0:11E
ampr-info	FF0X:0:0:0:0:0:0:11F
Mtrace	FF0X:0:0:0:0:0:0:120
RSVP-encap-1	FF0X:0:0:0:0:0:0:121
RSVP-encap-2	FF0X:0:0:0:0:0:0:122
SVRLOC-DA	FF0X:0:0:0:0:0:0:123
rln-server	FF0X:0:0:0:0:0:0:124
proshare-mc	FF0X:0:0:0:0:0:0:125
Dantz	FF0X:0:0:0:0:0:0:126
cisco-rp-announce	FF0X:0:0:0:0:0:0:127
cisco-rp-discovery	FF0X:0:0:0:0:0:0:128
Gatekeeper	FF0X:0:0:0:0:0:0:129
Iberiagames	FF0X:0:0:0:0:0:0:12A
Rwho	FF0X:0:0:0:0:0:0:201
SUN RPC	FF0X:0:0:0:0:0:0:202
Mbus/Ipv6	FF0X:0:0:0:0:0:0:300
Multimedia	FF0X:0:0:0:0:0:2:0000-FF0X:0:0:0:0:0:2:7FFD
SAPv1	FF0X:0:0:0:0:0:2:7FFE
SAPv0	FF0X:0:0:0:0:0:2:7FFF
SAP	FF0X:0:0:0:0:0:2:8000-FF0X:0:0:0:0:0:2:FFFF

RFC 2373 also provides a method for mapping multicast addresses into MAC addresses. The low-order 32 bits of the multicast address generate the MAC address. This is only feasible for group IDs that have 32 bits or fewer. IPv6 multicast address assignment should be done so that the group ID is always in the low-order 32 bits.

Anycast Address

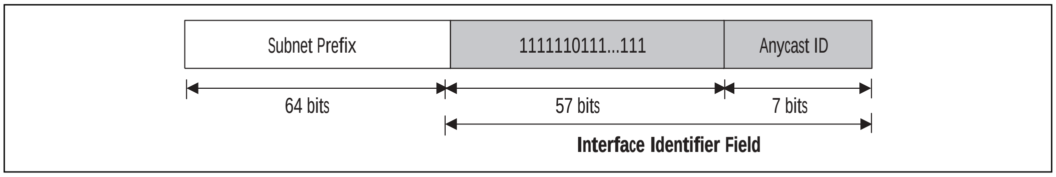
Any address assigned to more than one interface is considered an IPv6 anycast address. Packets sent to an anycast address are routed dynamically to the nearest interface configured with the anycast address. Routing protocols determine the "nearness" of the closest (in terms of routing distance) anycast interface. Anycast addresses are indistinguishable from a unicast address. Routers must be configured to handle anycast packets. Anycast addresses are beneficial because they can be used to augment dynamic routing protocols when choosing the shortest path to a destination. For example, anycast addresses can be configured on a set of Internet router interfaces to provide one address for routing to the Internet. Dynamic routing forwards along the best path to the anycast address. An additional benefit of anycast addressing is the capability to load share to multiple hosts. An example of this is if a user wants to use a service provided on multiple servers. The user initiates a connection to the anycast address and the packet is routed to the nearest server based on the dynamic route. The user does not have to specify which server to connect to. This requires each server to be configured with the anycast address. The anycast address for routers is called the *subnet-router* anycast address. All routers are required to support this address for the interfaces that are configured as subnets. Figure 4.16 depicts the format of the subnet-router anycast address.

Figure 4.16 Anycast Address



In addition to the subnet-router anycast address, there are two reserved anycast addresses. The format of these addresses is dependent on the type of IPv6 address configured. The format is determined by examining the format prefix. A simple rule to remember is this: If the leading bits of an address equals 000, then the interface ID can have any width; if the leading bits of an address is not equal to 000, then the interface ID is 64 bits wide. The only exception is the multicast address discussed earlier. Figure 4.17 depicts the anycast address structure for the types of addresses that require the 64-bit interface ID.

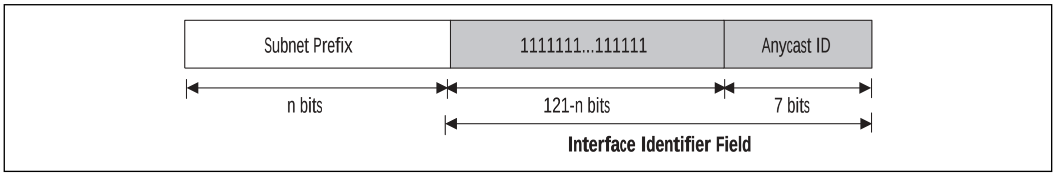
Figure 4.17 Anycast Addresses Requiring EUI-64 Format



The universal/local bit must be set to 0 for this type of anycast address. The second reserved anycast address covers all addresses whose format prefix equals

000. These are addresses whose interface ID is not in EUI-64 format. The Interface ID length is dependent upon the subnet prefix. Figure 4.18 depicts this type of address.

Figure 4.18 Anycast Address Non–EUI-64 Format



Anycast addresses in this format ARE NOT to be assigned as unicast addresses on any interface. Additional anycast identifiers are anticipated in the near future. Currently three reserve subnet anycast identifiers exist:

- 7E = 126 Decimal (used for Mobile IPv6 Home-Agents anycast)
- 7F = 127 Decimal
- 00 = 0 Decimal

If we were to write the mobile IPv6 home agents IPv6 EUI-64 interface ID address, it would be represented as follows:

```
1111110111111111     1111111111111111    1111111111111111   1111111111111110
```

The 0 in the first 16 bits is the universal local bit and means this address is local. The last eight bits of the address represent 7E. 11111110 = 7E in hex.

Nodes-Required Address

IPv6 requires that certain addresses be configured to route IPv6 packets. To activate IPv6 routing on a Cisco router, a site-local or global unicast address must be configured. This automatically configures the link-local address for that interface. In addition, the router automatically joins the required multicast groups. The following listing shows an example of this:

```
6Router-1#sh conf
interface Ethernet0
 ip address 192.168.123.50 255.255.255.0
 no ip route-cache
 no ip mroute-cache
<<<<remaining output omitted>>>>
```

```
6Router-1#sh int e 0
Ethernet0 is up, line protocol is up
Hardware is Lance, address is 00e0.b05a.d998 (bia 00e0.b05a.d998)
 Internet address is 192.168.123.50/24
<<<<remaining output omitted>>>>
```

The following listing shows the output after an IPv6 address is configured on the router:

```
6Router-1#sh ipv6 int e 0
Ethernet0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::2E0:B0FF:FE5A:D998
  Global unicast address(es):
    3FFE:4200:1:1::1, subnet is 3FFE:4200:1:1::/64
  Joined group address(es):
    FF02::1:FF00:1
    FF02::1
    FF02::2
    FF02::1:FF5A:D998
  MTU is 1500 bytes
<<<<remaining output omitted>>>>
```

As can be seen from this output, the router automatically created the following addresses:

```
FE80::2E0:B0FF:FE5A:D998     Link-local address
3FFE:4200:1:1::1              Global unicast address
3FFE:4200:1:1::/64            Subnet router anycast address
FF02::1:FF00:1                Solicited-node multicast group
FF02::1                        All-nodes multicast group
FF02::2                        All-routers multicast group
FF02::1:FF5A:D998             Solicited-node multicast range (in
                                combination with solicited node group)
```

As can be seen from this output, IPv6 is significantly different from IPv4 addressing in terms of the affect it has on each interface. The following list is a summarization of the required addresses for IPv6 to operate between a host and a router.

A host deployed with IPv6 must recognize the following addresses:

- Unicast address
- Loopback address
- Link-local address
- All-nodes multicast address
- Solicited-node multicast address (for each unicast and anycast address configured)
- Multicast address for all groups that the host belongs to

Routers must recognize the same addresses as the host plus the following:

- Subnet router anycast address for each subnet that the router advertises
- All routers multicast address
- Anycast address

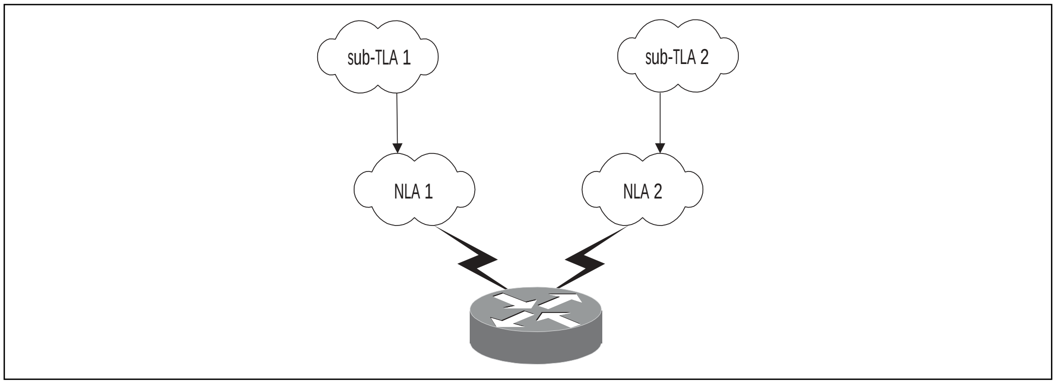
The Need for Further Development

Although IPv6 does in fact present many new and useful ideas aimed at increased efficiency and ease of routing and configuration, the work that is needed prior to deploying IPv6 natively on the Internet is not done. In this section, we will look at one current issue that is gaining increased attention by the IETF's IPv6 working gruop formerly called the Internet Protocol Next-Generation Working Group (IPNGWG). See http://playground.sun.com/pub/ipng/html/png-main.html for information on the working group's charter and the current status of their goals.

The Multihoming Problem

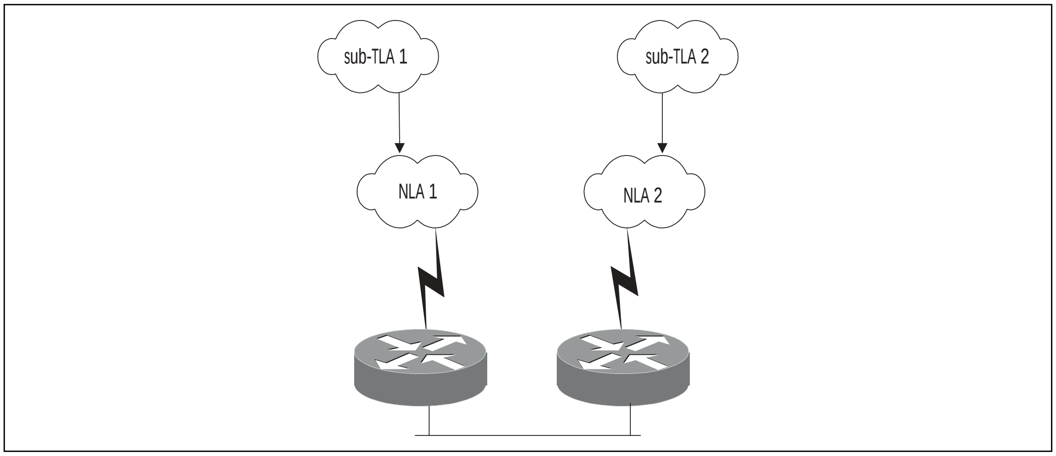
Multihoming is an issue that is mentioned throughout this chapter as a limitation of the IPv6 architecture. You have seen how IPv6 provides an addressing architecture that is hierarchical, scalable, and efficient in terms of address allocation, aggregation, and structure. However, these benefits do not apply to environments that wish to implement dual-homing solutions to multiple service providers. Dual homing is a necessity for ensuring Internet connectivity in the event that the primary link to a service provider fails. Dual homing to the same service provider is not an issue if a link fails (unless it becomes a congestion issue), but if the service provider experiences a failure on its backbone, it impacts Internet connectivity regardless of how many redundant links you have. Industry best practices recommend that end customers dual home to multiple service providers, especially in the environments where Internet connectivity is an integral part of the day-to-day business of an organization. One of the basic premises of IPv6 address space is that it is non-portable. In other words, one provider cannot advertise address space that belongs to an aggregate of another. Two types of dual homing exist from a customer SLA edge: via a single router or via multiple routers. The dual-homing problem is also evident between providers, but our discussion focuses on the end SLA customers. Figure 4.19 depicts an SLA customer who is multi-homed to two service providers via a single router.

Figure 4.19 Multihomed Single Router



In this scenario, the SLA customer has dual connections to different providers on a single router. Figure 4.20 depicts an SLA customer who is multihomed via multiple routers.

Figure 4.20 Multihomed Multiple Routers



In this scenario, the SLA customer has single connections to different providers on separate routers. The Ethernet connection is for example purposes only. The routers do not have to be on the same LAN and may in fact be in separate geographical locations.

Multiple solutions are being investigated that enable these types of peer-to-peer connections to take place. Some of the proposed solutions are as follows:

- Dual IPv6 addressing on a host
- Portability of address space
- Dual IPv6 address prefixes allocated within the environment
- Independently assigned address space

Dual IPv6 Addressing on a Host

The solution of dual IPv6 addressing on a host requires that host machines internal to the SLA be configured with IPv6 addresses from each provider. The host makes a decision of which IPv6 source address is used based on the availability of a viable path to an outside host. When a broken path is found, the router that is aware of it signals the host machines. IPv6 addresses are configured with a *lifetime* attribute. Lifetime has two values: *preferred* and *valid*. The lifetime value determines the time that an IPv6 address is valid. Preferred lifetime must be equal to or less than the valid lifetime value. The preferred lifetime attribute can be set to zero to inform a host that the IPv6 source address for the broken link is invalid. This attribute is encoded in the prefix that the router(s) advertise within the SLA environment. In our example, if the link between the SLA customer and the NLA1 or NLA2 provider goes down, the hosts within the environment automatically adjust the source IPv6 address to the valid prefix needed for routing. The implication here is that the host machines are capable of supporting this feature. Alternatively, the external host must be aware of the change to the destination IPv6 address when it sends a response to the internal SLA host. This can only be accomplished with DNS, because no signaling is done between the internal SLA host and the external host. Several efforts are under way to address the DNS architecture needed to support this solution. RFC 1886 outlines IPv6 DNS solutions. In addition to the inefficient allocation of address space, the solution proposed also has issues with TCP. TCP does not allow for address changing in the middle of a TCP session. The only solution to this is to adjust TCP to allow for this, which in itself seems easy, but the ramifications are far more than meet the eye. Most TCP applications are built in such a way that modifications of

TCP would require modifications of the application. This could mean drastic reworking of current network software. Also, current operating systems themselves may need overhauling in order to switch source addresses dynamically when a network becomes unreachable. How does a source know that a network failure has occurred in the right place? What if the destination had a problem? How would the source know if switching IPv6 source addresses would fix the problem?

Portability of Address Space

Portable address space means that upstream providers announce prefixes for SLA customers that are not part of their assigned aggregated prefix. Again, the use of portable address space introduces smaller prefixes that need to be advertised globally on the Internet. This is the same issue created in the IPv4 space that has generated the enormous routing tables seen in today's Internet backbones. While this solution breaks down the aggregation advancements achieved by IPv6, it might be a necessary evil. For example, an upstream provider NLA1 assigns its SLA customer an address prefix. The SLA has a redundant connection to NLA2 that serves as a backup to the NLA1 connection. In the event that the connection to NLA1 fails, NLA2 is required to announce the prefix assigned by NLA1. This breaks the aggregation of addresses advertised to the upstream sub-TLA by NLA2. Sub-TLA-2 now has an entry in its routing table that is carved out of the sub-TLA1 address space in addition to the aggregate advertised by sub-TLA1.

Dual IPv6 Address Prefixes Allocated within the Environment

The solution of allocating dual IPv6 address prefixes within the environment calls for IPv6 addresses from each provider to be distributed throughout the SLA environment. Hosts communicating to NLA1 utilize NLA1 address prefixes and hosts communicating to NLA2 utilize NLA2 address prefixes. This solution doesn't address the redundancy requirement for customer networks. Even though the customer has dual internet connection, a host can use only one connection at a time. Portable address space is the only way to overcome this issue. For example, if a customer loses a link to NLA2, NLA1 must advertise a prefix belonging to NLA2. This generates two entries in NLA1's routing table. One is the global aggregate route of the sub-TLA2 that provides the path to NLA2 and the second is the SLA prefix injected by the customer. This scenario is similar to the inefficiencies seen in today's IPv4 routing tables.

Independently Assigned Address Space

Another solution under review is the assignment of independent address space to the SLAs. Addresses assigned independently are not part of the aggregate prefixes assigned to the sub-TLAs. This means that every sub-TLA would have a globally unique address allocation that is a summarized advertisement to the upstream providers. The addresses assigned are portable and accepted as valid prefixes by all service providers. Due to the scope of the IPv6 address space, this appears to be the most viable solution.

A tremendous amount of focus is on the multihoming issue. Resolution of this issue and the adoption of a standard practice is a major obstacle to the adoption of the IPv6 address space within the Internet community.

The 6Bone

Now that you have the basis for understanding elementary IPv6 addressing and routing, let us look into current IPv6 deployment and the successes and shortcomings of them. The primary example of IPv6 deployment is the IETF Next-Generation Transition Working Group (NGTRANSWG) 6Bone. The 6Bone is a network of IPv6-speaking entities interconnected over the classical IPv4 Internet. It consists of both native networks (where IPv6 is running without being tunneled through another Layer 3 protocol) and IPv4 tunnels between different IPv6 speaking entities. The purpose of this network is twofold. The first reason for the 6Bone is to provide implementers a means to test their IPv6 implementations in a large network where other vendors have deployed their own version of IPv6 implementations. By enabling this, we can ensure that IPv6 implementations are interoperable. This way, protocol developers can make sure that the protocol specifications are specific enough to allow for implementers to develop IPv6-speaking machines without ambiguity. The second reason for the 6Bone is to give networks operators a chance to design networks and get their feet wet with the new protocol. Also, it enables operators to uncover any problems with the IPv6 protocol (such as the previous multihoming problems) that may have been missed or underappreciated during the protocol's conception. Although the fathers of the IPv6 protocol were extremely meticulous in the protocol's design, it never hurts to get the new technology running on a live network somewhere prior to implementation on a grand scale. The 6Bone helps to work out all the details and test new features, prior to deployment, in a cooperative, multinational fashion. The 6Bone is not only a tremendous asset for dampening the learning curve for entities wishing to deploy IPv6, but is also a way to partially meet the criteria for obtaining IPv6 address allocations. ARIN and the other IRs have established a temporary bootstrap phase that established requirements for obtaining an initial allocation of IPv6 addresses. The bootstrap phase is temporary until the widespread allocation of IPv6 addresses enables the IRs to move into the general phase. Please refer to the IR's Web pages for the policies governing IPv6 address allocations. The Web pages are as follows:

- **ARIN** www.arin.net
- **RIPE** www.ripe.net
- **APNIC** www.apnic.net

The 6Bone utilizes routing practices as defined by the IETF. For the most current IPv6 routing practices on the 6Bone, see RFC 2772 (www.ieft.org/rtc/rfc2772.txt). For more information on the 6Bone, please see www.6bone.net.

Summary

IPv6 provides for many of the much-needed improvements in the Internet that we will need in the near future. Not only does it solve the address depletion problems of today's IPv4, but it also makes for a more scalable Internet core, which can help improve routing efficiency of the Internet as a whole. By allowing for 128 bits of addresses, we can see that adequate address space is available for the future. By then aggregating this address space in an efficient manner, we may establish a firm upper limit on routing table size in the Internet core. This, in turn, can help us to build an Internet to take us into the future.

Although the two primary problems of the Internet can be solved with IPv6, the protocol improvements do not stop there. Also built into the IPv6 protocol are means for hop-by-hop routing, authentication of packets, encrypted packets, tag switching, QoS, and other things to make the protocol more versatile than its grandfather, IPv4. Furthermore, IPv6 has built into it the capability to use multicast and unicast routing in a manner such that boundaries easily can be scoped to ensure that data does not get to places where it is not allowed. IPv6 also introduces the use of an anycast address, for applications that may be serviced by multiple machines, but the need for distribution of these services in a scalable manner is required.

IPv6 is already in testing, and is starting into production in some areas, connected via the 6Bone. This virtual backbone will provide for testing of both IPv6 implementations and the protocol itself. Clearly, the IPv6 is getting more and more attention and is looking like a promising tool for the future of the Internet.

Solutions Fast Track

The Basics of IPv6 Addressing

- ☑ IPv6 uses a 128-bit format that represents the next generation of Internet protocol that can meet both the current addressing requirements and those generated by emerging markets.
- ☑ IPv6 was not developed solely as a mechanism for addressing address depletion. It provides enhancements to aggregation, QoS, security, and routing.

- ☑ Converting to IPv6 is not a trivial task; it is an entirely new addressing structure that introduces complexities in the deployment and management of the address space.

IPv6 Addressing Scheme Characteristics

- ☑ A format for the deployment of highly structured, hierarchical address space is introduced in the global unicast address space.
- ☑ Due to the variety of address types used in the IPv6 address architecture, routers now have multiple addresses per physical interface. This can include the global unicast address, link-local address, site-local address, and multicast addresses on a per-interface basis.
- ☑ IPv6 provides three mechanisms for representing the interface identifier of the host address portion of an IPv6 address. These mechanisms are the MAC address, the converted EUI-64 address, and an IPv4 address.

The Need for Further Development

- ☑ Multihoming to multiple service providers can defeat the highly structured aggregation policies defined in the IPv6 address architecture. A tremendous amount of focus exists in the area of multihoming to alleviate these issues.
- ☑ Modifications to the TCP stack and to DNS architectures are required to fully realize the benefits of the IPv6 addressing.
- ☑ The 6Bone provides a test bed for the development of IPv6 infrastructures and is constantly evolving to meet the requirements generated by ongoing deployments of IPv6.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: How do I subnet my address space if I am an SLA?

A: Subnetting of the IPv6 address space assigned to you by an NLA or sub-TLA provider is up to you. The first thing to do is to completely understand how the address space is broken up. This means that you need to understand the IPv6 address SLA structure completely. The 16 SLA bits provide 65,535 subnets and the 64 interface identifier bits provide 4,294,967,295 hosts per subnet. For subnetting purposes, I would recommend starting with subnet one and allocating as needed from there. The host addresses are assigned automatically with either MAC addresses of the host or EUI-64 addresses converted from MAC addresses. The exception to interface ID rule is if you want to translate the current IPv4 address into an IPv6 address. Your first subnet could be 0000000000000001:Interface IDs; your second could be 0000000000000002:Interface IDs, and so on. The first part of the address has been omitted, because these are dependent on whether the address is site local or a global unicast address.

Q: Will IPv6 ever see widespread deployment?

A: This is a question that everyone will have to ask sooner rather than later. Although limitations to using IPv6 in a dual-homed environment exist and complexities are associated with the deployment, IPv6 will be deployed. IPv4 address space has a limited lifespan and estimates have put its exhaustion between the years 2008-2015.

Q: How can I get onto the 6Bone?

A: The 6Bone has a mailing list where operational issues associated with it, as well as new proposals for transition strategies, are discussed and collaborated upon. This list can be joined by sending e-mail to majordomo@isi.edu, with "subscribe 6bone" as the contents of the message. It is encouraged that all mailing list members of the 6Bone actually become IPv6 speakers on the 6Bone as well. Refer to the Web page www.6bone.net for information on how to get connected and to find the nearest upstream provider to obtain a tunnel.

Q: Where do I get an IPv6 address?

A: The providers who have them delegate IPv6 addresses. When you join the 6Bone or any other IPv6 network, your upstream provider is responsible for providing you with adequate IPv6 address space to take care of your needs.

Q: Where do I get the IPv6 protocol specifications for more details?

A: All protocol specifications are in the form of IETF RFCs. These can be found at www.ietf.org, where a search engine helps you pull up all current IPv6 RFC and Internet drafts.

Chapter 5

Configuring IPv6 Addressing

Solutions in this chapter:

- **Configuring IPv6 Addressing**
- **Verifying Addressing Configuration**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

Introduction

This chapter will explain the commands required to configure IPv6 addresses and associated parameters on the Cisco IOS, and should enable you to effectively configure IPv6 on either your own network or a customer's network. This chapter is geared toward those who want to learn the exact command syntax necessary to fully configure the Cisco router to support IPv6.

The second half of the chapter is dedicated to verifying the router configuration. We will cover the commands that are necessary to quickly view key information about the IPv6 configuration. Each command will be discussed in detail to ensure that you have all the tools necessary to configure and verify IPv6 on the IOS-based Cisco router.

Configuring IPv6 Addressing

The first step in configuring IPv6 on your router is making sure that the correct version of IOS is installed on the router. IPv6 was first introduced in IOS version 12.2(1)T Technology release, so that is the earliest version that can be used. The Cisco IOS version can be checked using the *show version* command, a sample output of which is seen below. For a thorough explanation of IOS version numbering, please visit Cisco's Web site.

```
6Router-1#show version
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-IS-L), Version 12.2(8)T,  RELEASE
    SOFTWARE (fc2)
TAC Support: http://www.cisco.com/tac
Copyright (c) 1986-2002 by cisco Systems, Inc.
Compiled Wed 13-Feb-02 21:11 by ccai
Image text-base: 0x0306DA78, data-base: 0x00001000
```

As you can see from this output, the version of code being used on the router is Cisco IOS 12.2(8)T, which at the time of the writing of this book is the latest release. We will look next at some of the commands that can be configured on the router for IPv6. These will be broken down into LAN and WAN configurations. Some of the commands from the router are listed below to give a quick overview of the various commands that can be configured just for IPv6. The first mode shown is the global configuration mode. The second list shown is one from an interface, in this case an Ethernet interface.

```
6Router-1(config)#ipv6 ?
  access-list      Configure access lists
  hop-limit        Configure hop count limit
  host             Configure static hostnames
  icmp             Configure ICMP parameters
  neighbor         Neighbor
  prefix-list      Build a prefix list
  route            Configure static routes
  router           Enable an IPv6 routing process
  unicast-routing  Enable unicast routing

6Router-1(config-if)#ipv6 ?
IPv6 interface subcommands:
  address          Configure IPv6 address on interface
  enable           Enable IPv6 on interface
  mtu              Set IPv6 Maximum Transmission Unit
  nd               IPv6 interface Neighbor Discovery subcommands
  redirects        Enable sending of ICMP Redirect messages
  rip              Configure RIP routing protocol
  traffic-filter   Access control list for packets
  unnumbered       Configure IPv6 interface as unnumbered
```

Once you have verified that the Cisco IOS version you are using supports IPv6, the next step is to enable IPv6 globally on the router. This is done while in the *configuration* mode with the command is *ipv6 unicast-routing*. This command enables IPv6 for the entire router. If this command is not enabled globally, the rest of the commands on the interfaces will not operate. The command to enable IPv6 on the router is shown below.

```
6Router-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 unicast-routing
6Router-1(config)#
```

Enabling IPv6 globally does not do much good until IPv6 is configured on some individual interfaces, so the next step is to enable IPv6 on the desired interfaces. The following sections will cover how to configure the IPv6 address on LAN and WAN interfaces, as well as how to configure some of the options available for tuning IPv6.

Configuring LAN Addresses

There are a few steps involved in configuring the LAN address. Assuming that the IPv6 global routing has already been configured, the first step is to configure the actual interface. In most cases this will be an Ethernet interface, although it is currently possible to configure IPv6 on Token Ring and Fiber Distributed Data Interface (FDDI) as well. This section will focus on Ethernet interfaces.

The process of configuring an IPv6 address on an interface is pretty straightforward. There are three types of addresses that can be assigned on LAN addresses. The three types are *link-local*, *site-local* and the *global* addresses. The global and site-local addresses are assigned at the same time. If a global address is already assigned by the architecture of your network, then the full address will be typed in during configuration. If only the first 64 bits are specified, then the *Extended Unique Identifier (EUI)* command at the end of the global address will have an interface identifier assigned for the global address. To enter the address in the router, go into **configuration** mode and then select the desired interface. Each of the commands can be seen below, the first with the full address and the second using the *EUI* parameter at the end of the command to have the router assign the last 64 bits of the address.

The EUI works similarly to the link-local addresses that will be discussed later in the chapter. If the EUI is used, then only the first 64 bits of the address need to be specified; the rest of the address will be filled in automatically using the MAC address of the router. If there are multiple interfaces using the *EUI* parameter, you will notice that all of the interfaces will have addresses with the same last 64 bits.

```
Router configuration for predetermined global address
6Router-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#int e0
6Router-1(config-if)#ipv6 address 2000:1:1::1/64
6Router-1(config-if)#

Router configuration for global address to be assigned interface identifier
6Router-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#int e0
6Router-1(config-if)#ipv6 address 2000:1:1:1::/64 eui-64
6Router-1(config-if)#
```

When the *EUI* parameter is used, the remaining 64 bits of the address are automatically filled in by the router. The address produced by the command above can be seen below. Notice that only the first 64 bits were defined above. Also notice that the link-local address has the same last 64 bits as the global address.

```
6Router-1#sh ipv6 int e0
Serial1 is down, line protocol is down
IPv6 is enabled, link-local address is FE80::2E0:B0FF:FE5A:D998
Global unicast address(es):
2001:1:1:1:2E0:B0FF:FE5A:D998, subnet is 2001:1:1:1::/64
```

Designing & Planning…

EUI-64

The Extended Unique Identifier (EUI) can be used when assigning addresses. This is something that you may want to consider using when designing your network—it may use up a little more address space, but it can make deploying the network a bit simpler. One of the benefits of using the EUI is that you don't have to know the last 64 bits of the address, as the router will figure that out based on the 48-bit identifier that is used for the interface. The other 16 bits are randomly generated. (If more than one interface of a router is configured using the *EUI* parameter, then all of the interfaces will share the same last 64 bits. This can make monitoring a little easier: if you know the MAC address of the router, then you can recognize as a router address any IPv6 address that has that MAC address in it.

This can also be a problem in the eyes of the administrator because now, instead of setting the interface IPv6 address, the router is assigning the value, which can be difficult to manage. Also, if the EUI is used and an end user machine needs to be statically defined, then the end user won't know what the router interface address is (although sometimes that can be a good thing). The traditional way to set up a router interface is to use an address at the beginning or the end of the address space.

When the IPv6 address has been assigned to the interface, by default IPv6 is also enabled on the interface itself. Having IPv6 enabled on the interface assigns the interface a link-local address. This address (as discussed in Chapter 4) is a local-only address that is not propagated outside of the local link. The router automatically assigns a link-local address, and will typically use the EUI identification of the router for the last 64 bits of the address. If the architecture of your network requires that the local links have specific addresses, you can assign an address as link-local by simply typing **link-local** after the IPv6 address in the configuration. Remember that for link-local to be enabled the address must be a valid one—it must be between FE80 and FEBF. The command can be seen below, followed by an example of an invalid address attempt:

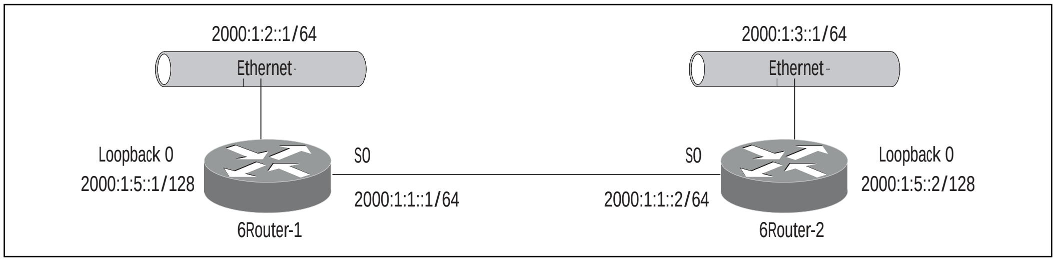
```
6Router-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#int e0
6Router-1(config-if)#ipv6 address fe80::1:1:1:1 link-local
6Router-1(config-if)#

6Router-1(config-if)#ipv6 addr 2001::1 link-local
Invalid link-local address
6Router-1(config-if)#
```

Sometimes an interface may not require an IPv6 interface, as is the case when subinterfaces are used for tunneling. The configuration of an unnumbered interface is similar to the equivalent IPv4 configuration. Simply type the command **ipv6 unnumbered** and the interface will have no IPv6 address assigned to it, although it will be associated with the interface specified at the end of the command. An unnumbered interface can also be used for segments that will use only link-local addresses. As with a regular IPv6 address, IPv6 is enabled when an unnumbered address is configured. If no global address is needed, then a link-local address will be configured just by enabling IPv6 on the interface. The command for enabling IPv6 on an interface while maintaining an unnumbered interface is shown here:

```
6Router-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#int s1
6Router-1(config-if)#ipv6 unnumbered loopback0
6Router-1(config-if)#ipv6 enable
```

If desired, multiple IPv6 addresses can be assigned to an interface. Cisco IOS release 12.2.(4)T or later is required for this operation. Unlike IPv4, the *secondary* command is not necessary when adding a second IPv6 address; simply add another IPv6 address on the desired interface and the interface will now have multiple IPv6 addresses. Note that at present only one link-local address is supported. This will probably not change, because there should not be a demand for multiple local addresses. When the IPv6 address is assigned on an interface, the interface by default will join several multicast groups for that interface. The multicast address joins the *all-nodes* multicast group, the *all-routers* multicast group and the *solicited-node* multicast group. Figure 5.1 is a quick diagram of the network as configured above.

Figure 5.1 LAN Diagram



Configuring Duplicate Address Detection

The purpose of duplicate address detection is to verify that a new IPv6 address is unique to the router. The router will check its other interfaces using neighbor solicitation messages to verify that the IPv6 address is unique; if it is not, then an error will be returned stating that the IPv6 address is a duplicate and indicating what interface the address resides on. This feature is on the router by default, and thus requires no configuration, but there is a parameter that can be configured for the detection process: the number of solicitation messages that are sent out by an interface can be changed in the configuration (the default is one). It must fall in the range from 0 to 600. The command is as follows:

```
6Router-1(config-if)#ipv6 nd dad attempts 2
```

The duplicate address detection can be turned off by setting the value of *attempts* to zero. This will shut off the detection for that interface only. If you want to turn off duplicate address detection for an entire router, every interface on the router must have the attempt number set to zero. Note that the command *no ipv6 nd dad attempts* will *not* disable duplicate address detection on that router interface, but will instead reset the number of attempts to the default of one. This is counterintuitive to the usual procedure of Cisco IOS, but that is the way it works.

Configuring & Implementing…

Duplicate Address Detection

It should be noted that when duplicate address detection is used, there will only be an error message if the duplicate IPv6 address is on the same router. When a duplicate IPv6 address is tried on an Ethernet interface and that address is already assigned to a serial interface, an error message is displayed. But if you are configuring two neighbor routers and, for example, you mistype the IPv6 address to be the same as the router on the other end, you will not get an error message! Duplicate address detection is used mostly for Ethernet interfaces because most serial interfaces are point-to-point, so the duplicate address detection does not provide much functionality. However, if you use the command **show ipv6 interface XXX**, the IPv6 address will show up as a duplicate because it is still discovered. In addition, if duplicate address detection was turned off on an interface and then turned on, it will not give an error message for already configured addresses.

Note that duplicate address detection is for unicast addresses only—it must not be performed on anycast addresses. Duplicate address detection is most useful for link-local addresses, because they are usually automatically assigned. Most global and site-local address are already assigned and there should not be any nodes that are assigned the same address as the router, so the likelihood of having a duplicate address is already reduced.

Configuring DNS

Configuring the Domain Name System (DNS) for IPv6 is not unlike configuring it for IPv4. DNS is enabled by default, but having DNS enabled on your router does not enable it for the interfaces. Only the router itself uses the DNS information, meaning that specifying a DNS server on the router will not send that information out through the Ethernet interface to the clients of the interface. DNS is often shut off on the router because the DNS server will sometimes try to attach itself to mistyped commands, which wastes time. If DNS has been shut off and needs to be turned back on, use the command shown here:

```
6Router-1(config)#ip domain-lookup
```

With DNS enabled, there are a few additional features that become available, very similar to the ones available with IPv4. A name server and default domain name can be specified. The commands for entering a domain name or domain list are, not surprisingly, the same as IPv4, because only names are being specified. However, when configuring the name server, the command does not use *IPv6* in the command but must distinguish between IPv4 and IPv6. The command can have either an IPv4 or IPv6 address entered; the example below shows IPv6:

```
6Router-1(config)#ip name-server 1000:1000:2ad::2000:2000:2
```

WARNING

The command shown above is not supported by Cisco IOS code 12.2(4)T1. It is supported by Cisco IOS 12.2(8)T and later.

As with IPv4, you can also use host-name-to-address mappings. This is especially convenient given the length of some of the IPv6 addresses, as it enables you to type just a name instead of the whole address. Host names are often used with routers that have asynchronous connections to other routers and switch console ports. A host name mapping is the easiest way to configure those routers. Several examples of host name mapping are shown below. Notice how the *ipv6* command is used to specify that the address is going to be IPv6. The first example shows how to set up a direct host name for the IPv6 address; the second shows how to set the host name to the local router and out of the *Aux* port (which could be connected to another *6router-2* console port). In the second command, remember that 2001 is the port number that the host name maps to (in this case the *Aux* port), and not part of the IPv6 address.

```
6Router-1(config)# ipv6 host 6Router-2 2000:1:1::2


6Router-1(config)# ipv6 host backup  2001 2000:1:5::1
```

Configuring WAN Addresses

Now let's take a look at configuring the WAN addresses with respect to IPv6, with a focus on Asynchronous Transfer Mode (ATM) and Frame-Relay. Just as with the LAN addresses, there are some differences in the commands. Most of the commands have to do with the mapping of permanent virtual circuits (PVCs), switched virtual circuits (SVCs), and data-link connection identifiers (DLCIs) with either the ATM or Frame-Relay network.

Configuring ATM

The configuration of ATM using IPv6 is not very different from the configuration for IPv4. IPv6 will have some different commands for the configuration, but the theory behind ATM is the same, and IPv6 does not change the way ATM works or offer any improvements to the ATM platform. As we have seen in previous sections, the *ipv6 address* command, followed by the IPv6 address, is used to enter an IPv6 address for an ATM interface.

If there are several PVCs that are mapped to one interface (a point-to-multipoint interface), the command *protocol ipv6* must be entered on the interface for the particular PVC. This command is not needed if there are not multiple connections on one interface, because in that case the interface would just be a point-to-point interface and no mapping would be required. For a point-to-point interface, only an IPv6 address would be required to configure the ATM interface. When point-to-multipoint is used, ATM mapping must be used. Examples of some configurations are shown here. The first configuration is just point-to-point:

```
Point-to-Point
6Router-1(config-if-atm-vc)# ipv6 address 2000:1:20::1/64

6Router-2(config-if-atm-vc)# ipv6 address 2000:1:20::2/64

Point-to-Multipoint
6Router-1(config-if-atm-vc) protocol ipv6 2000:1:20::2
6Router-1(config-if-atm-vc) protocol ipv6 fe80::1:1:20:2
6Router-1(config-if-atm-vc) ipv6 address 2000:1:20::1

6Router-2(config-if-atm-vc) protocol ipv6 2000:1:20::1
6Router-2(config-if-atm-vc) protocol ipv6 fe80::1:1:20:1
6Router-2(config-if-atm-vc) ipv6 address 2000:1:20::2
```

NOTE

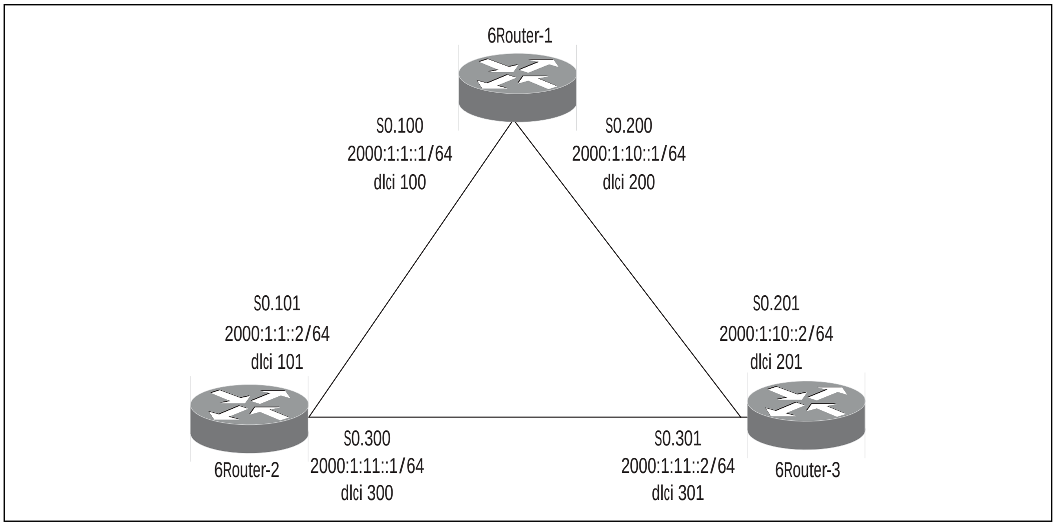
In IPv4, Inverse ARP was used to find the network layer address of the node at the other end. When using IPv6, the inverse neighbor discovery protocol will be used to by the router to discover the far end node's global IPv6 address. The problem is that inverse neighbor discovery is not currently supported by Cisco IOS.

Configuring Frame-Relay

As with ATM, the configuration of Frame-Relay for IPv6 is not very different from that of IPv4, but some of the commands are a little different. Since it provides no major enhancements, we'll just explain the different commands that are necessary when IPv6 is implemented on Frame-Relay.

Frame-Relay, like ATM, has only a few commands that need to be entered to get the interface up and running. Also like ATM, Frame-Relay has several different scenarios for point-to-point and point-to-multipoint interfaces. The first scenario is a fully meshed topology between three routers. Each router has a PVC connection to the other two routers through point-to-point connections. Because the connections are point-to-point, no mappings are needed. One IPv6 address is required for each PVC, just as it would be with IPv4. The second scenario also uses three routers, but this time there are no subinterfaces used on the Frame-Relay connection. Because these interfaces are point-to-multipoint, they will require mappings to know how to get to the other routers. The commands needed for this are very similar to those of IPv4, with the exception of the **ipv6** designation. Some sample configurations for both point-to-point and point-to-multipoint interfaces are shown below. Notice that, just like in IPv4, map statements are only needed in point-to-multipoint instances. Also notice that the link-local address is mapped as well as the global IPv6 address—this is one extra command that is needed when using IPv6. Diagrams of the two frame-relay scenarios are shown in Figures 5.2 and 5.3.

Figure 5.2 Frame-Relay Point-to-Point

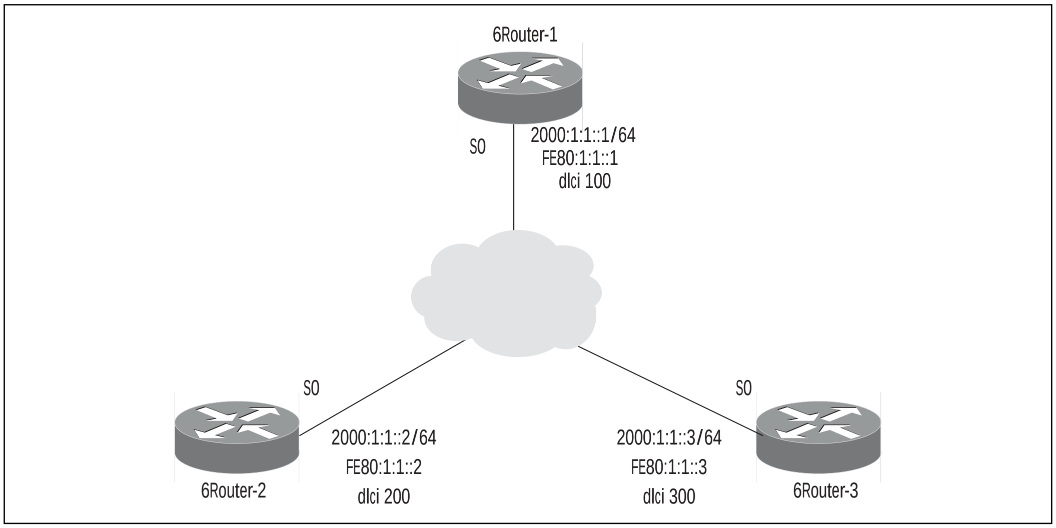


```
Point-to-Point
6Router-1(config)#int s0
6Router-1(config-if)#encapsulation frame-relay
6Router-1(config)#int s0.100 point-to-point
6Router-1(config-subif)#ipv6 address 2000:1:1::1/64
6Router-1(config-subif)#frame-relay interface-dlci 101
6Router-1(config)#int s0.200 point-to-point
6Router-1(config-subif)#ipv6 address 2000:1:10::1/64
6Router-1(config-subif)#frame-relay interface-dlci 201



6Router-2(config)#int s0
6Router-2(config-if)#encapsulation frame-relay
6Router-2(config)#int s0.101 point-to-point
6Router-2(config-subif)#ipv6 address 2000:1:1::2/64
6Router-2(config-subif)#frame-relay interface-dlci 100
6Router-2(config)#int s0.300 point-to-point
6Router-2(config-subif)#ipv6 address 2000:1:11::1/64
6Router-2(config-subif)#frame-relay interface-dlci 301

6Router-3(config)#int s0
6Router-3(config-if)#encapsulation frame-relay
```

```
6Router-3(config)#int s0.201 point-to-point
6Router-3(config-subif)#ipv6 address 2000:1:10::2/64
6Router-3(config-subif)#frame-relay interface-dlci 200
6Router-3(config)#int s0.301 point-to-point
6Router-3(config-subif)#ipv6 address 2000:1:11::2/64
6Router-3(config-subif)#frame-relay interface-dlci 300
```

Notice that the commands are almost identical to the commands that would be used for configuring IPv4. The command used to define the DLCI is exactly the same one used with IPv4.

Figure 5.3 Frame-Relay Point-to-Multipoint



```
Point-to-Multipoint
6Router-1(config)#int s0
6Router-1(config-if)#encapsulation frame-relay
6Router-1(config-if)#ipv6 address 2000:1:1::1/64
6Router-1(config-if)#ipv6 address fe80:1:1::1 link-local
6Router-1(config-if)#frame-relay map ipv6 2000:1:1::2 200
6Router-1(config-if)#frame-relay map ipv6 2000:1:1::3 300
6Router-1(config-if)#frame-relay map ipv6 fe80:1:1::2 200
6Router-1(config-if)#frame-relay map ipv6 fe80:1:1::3 300

6Router-2(config)#int s0
```

```
6Router-2(config-if)#encapsulation frame-relay
6Router-2(config-if)#ipv6 address 2000:1:1::2/64
6Router-2(config-if)#ipv6 address fe80:1:1::2 link-local
6Router-2(config-if)#frame-relay map ipv6 2000:1:1::1 100
6Router-2(config-if)#frame-relay map ipv6 2000:1:1::3 300
6Router-2(config-if)#frame-relay map ipv6 fe80:1:1::1 100
6Router-2(config-if)#frame-relay map ipv6 fe80:1:1::3 300

6Router-3(config)#int s0
6Router-3(config-if)#encapsulation frame-relay
6Router-3(config-if)#ipv6 address 2000:1:1::3/64
6Router-1(config-if)#ipv6 address fe80:1:1::3 link-local
6Router-3(config-if)#frame-relay map ipv6 2000:1:1::1 100
6Router-3(config-if)#frame-relay map ipv6 2000:1:1::2 200
6Router-3(config-if)#frame-relay map ipv6 fe80:1:1::1 100
6Router-3(config-if)#frame-relay map ipv6 fe80:1:1::1 100
```

Again, the commands are almost identical to the ones used for configuring IPv4. The big difference is that the link-local address is also specified, although it does not need to be. The map statement must also explicitly have the *ipv6* parameter in the command.

Configuring ICMPv6 and Neighbor Discovery

Configuring ICMPv6 primarily involves just tweaking the settings, not actually performing setup on the router. The tweaking is necessary because some functions of ICMP can be affected if the tweaking is not performed. One of the functions that can fail is *traceroute*, which we all know is an invaluable tool if you are trying to troubleshoot your network. The tweaking adjusts the ICMP error interval and the bucket size, or number of tokens in the bucket.

NOTE

Once again, pay attention to the version of Cisco IOS that you have on your router. The earliest version needed to adjust the ICMPv6 rate limit is Cisco IOS 12.2(4)T, but you cannot specify the number of tokens for the bucket size unless you have Cisco IOS 12.2(8)T or later.

The two parameters that can be configured for ICMPv6 are primarily for the router CPU utilization and for certain functions like *traceroute* to function properly. The problem with traceroute is that if the rate limit is not set low enough, it can require rapid requests faster than the rate limit, causing the function to fail. The *bucketsize* option lets you tell the router how many tokens can be taken out of the bucket until the process is stopped. For example, if a client were continually sending unrecognizable packets, the router would have to continually send back error messages, potentially hurting the router's CPU performance. This way, when the bucket is empty, the router will stop sending error messages, conserving CPU performance. The command to set the ICMPv6 rate limit can be seen here. Note that the value of 100 is the error interval in milliseconds, and 10 is the number of tokens that are in the bucket:

```
6Router-1(config)# ipv6 icmp error-interval 100 10
```

Neighbor discovery in IPv6 is equivalent to IP Address Resolution Protocol (ARP) in IPv4. The neighbor discovery process will go out and find the 48-bit hardware address (usually referred to as the *MAC address*) that is associated with a particular IPv6 address. The router will keep a record of that address for a specified period time. If there is no activity from the node, then the entry in the neighbor discovery table will time out and have to be rediscovered the next time the router needs to communicate with that node. In some instances, a static cache entry may have to be inserted into the neighbor discovery cache. For example, if a router needed to talk to a server, but the server was over a threshold of utilization, it would stop responding to neighbor discovery packets. A static entry would be needed because we still need to talk to the server even though it is busy. Static entries are only allowed on LAN interfaces (Ethernet, FDDI, and Token Ring). Like ARP, neighbor discovery does not go out WAN interfaces (Serial etc.) because those are typically point-to-point interfaces with no nodes on them. An example of the command to configure a static entry for the neighbor discovery cache can be seen below. Also shown is an example of the command being attempted on a WAN interface:

```
6Router-1(config)#ipv6 neighbor 2000:1:2::10 ethernet0 0000.1234.5678
6Router-1(config)#

6Router-1(config)#ipv6 neighbor 2000:1:1::10 serial0 0000.1111.2222
% Static Neighbor Cache entries not supported on this interface type
6Router-1(config)#
```

Configuring & Implementing…

Upgrading Cisco IOS

As we have seen throughout the configuration section of the chapter, there are several features that require you to take notice of what version of Cisco IOS is running on the router. You should also take care when upgrading the Cisco IOS code: be sure to back up your configuration before upgrading the code in case there are any problems. You should also have network diagrams of your existing network as another means of going back to your previous network state in the event that anything goes wrong during the upgrade. I noticed when I upgraded my Cisco IOS from 12.2(4)T1 to 12.2(8)T that I lost all of the IPv6 configurations. I had to use my backup configurations to restore the IPv6 portion of the configuration.

Verifying Addressing Configuration

It is crucial to verify that all of your configurations have in fact worked. This section lists many of the commands that are needed to verify that the configurations have taken place and are working the way you want them to. Obviously, the *show* command will be used extensively in this section to show different parts of the configuration. The output of the *show* command will be displayed, along with a detailed description of what the important part of the output is telling you. The following list displays the available *show* commands that are specific to IPv6. (Note that this is an output for a router using Cisco IOS 12.2(8)T—if you are using a different version of the Cisco IOS, your output may look slightly different):

```
6Router-1#show ipv6 ?
  access-list   Summary of access lists
  interface     IPv6 interface status and configuration
  mtu           MTU per destination cache
  neighbors     Show IPv6 neighbor cache entries
  prefix-list   List IPv6 prefix lists
  protocols     IPv6 Routing Protocols
  rip           RIP routing protocol status
  route         Show IPv6 route table entries
  routers       Show local IPv6 routers
  traffic       IPv6 protocol statistics
  tunnel        Summary of IPv6 tunnels

6Router-1#clear ipv6 ?
  neighbors     Clear IPv6 ND Entry Cache
  prefix-list   Prefix-list
  route         Clear IPv6 route table entries
  traffic       Clear traffic counters

6Router-1#debug ipv6 ?
  icmp          ICMPv6 debugging
  nd            IPv6 Neighbor Discovery debugging
  packet        IPv6 packet debugging
  rip           RIP Routing Protocol debugging
  routing       IPv6 routing table debugging
```

The first command shows whether IPv6 is configured to perform a *show running-config*. There is no way to tell whether IPv6 has been enabled other than using this command. There are some *show* commands that will indicate that the global command has not been configured on the router, but the only sure way to know is to check the actual configuration. The command to check the configuration of the router is as follows:

```
6Router-1#show running-config
Building configuration...

Current configuration : 1487 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
service password-encryption
service tcp-small-servers
!
hostname 6Router-1
!
ipv6 unicast-routing
```

```
!
interface Loopback0
 no ip address
 no ip route-cache
 no ip mroute-cache
```

As shown above, the command *ipv6 unicast-routing* is used in the configuration of the router. This is the global command that is required for IPv6 to operate on the router.

The next command we will look at is similar to one used in IPv4: *show ipv6 interface brief*. This command will give a brief overview of all the interfaces that have IPv6 enabled, along with their respective global IPv6 addresses. The status of the interface itself—that is, whether it is in an operational state—is also listed. Notice that only the global address is listed, not the link-local address. Those addresses can be seen using other commands that we will discuss later. The output of the *show ipv6 interface brief* command is shown here:

```
6Router-1#show ipv6 interface brief
Ethernet0                    [up/up]
    2000:1:2::1
Loopback0                    [up/up]
    2000:1:5::1
Serial0                      [up/up]
    2000:1:1::1
Serial1                      [down/down]
    unassigned
```

Verifying LAN Addressing

To verify that the LAN addresses are configured correctly, perform a *show ipv6 interfaces <type>* command. This command is very useful for gathering some of the basic information about an interface, as it will display a detailed description of what is configured for the interface. The global and link-local addresses will be shown, as well as different multicast groups that have been joined on the interface. There is also information on duplicate address detection and the rate limit at which ICMP error messages are being transmitted. The following is an example of the output for an Ethernet interface:

```
6Router-1#show ipv6 int ethernet0
Ethernet0 is up, line protocol is up
```

```
  IPv6 is enabled, link-local address is FE80::2E0:B0FF:FE5A:D998
  Global unicast address(es):
    2000:1:2::1, subnet is 2000:1:2::/64
  Joined group address(es):
    FF02::1
    FF02::1:FF5A:D998
    FF02::1:FF00:1
    FF02::2
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
```

Notice that both the global address and the link-local address are shown. The multicast addresses are also shown in the lines that follow.

Verifying WAN Addressing

To verify that your ATM and Frame-Relay have been set up correctly for IPv6, you will use several commands, similar to those used with IPv4. The first commands we will look at are the mapping commands for both ATM and Frame-Relay. These two commands are actually the same commands used with IPv4 and will output both IPv4 and IPv6 mappings, as can be seen in the Frame-Relay output:

```
6Router-1#show atm map
Map list ATM0pvc1:  PERMANENT
Ipv6 FE80::1:1 maps to VC 1,  VPI 1, VCI 32, ATM0,
Broadcast
Ipv6 2000:1:1::1 maps to VC 1, VPI 1, VCI 32, ATM0

Frame-Relay
```

```
6Router-1#show frame-relay map
Serial1 (up): ip 10.10.10.2 dlci 200(0xC8,0x3080), static,
               CISCO, status defined, active
Serial1 (up): ipv6 2000:1:1::2 dlci 200(0xC8,0x3080), static,
               CISCO, status defined, active
Serial1 (up): ipv6 2000:1:1::3 dlci 300(0x12C,0x48C0), static,
               CISCO, status defined, active
```

Verifying ICMPv6 and Neighbor Discovery Configuration

To look at the neighbor discovery cache, use the command *show ipv6 neighbors Ethernet0*. The output will show all of the discovered neighbors that the router has in its cache. Both discovered and statically configured entries will be shown, as well as the age of each entry. A hyphen (–) in the age field indicates that the entry is a static one. A sample output of the command is shown here:

```
6Router-1#show ipv6 neighbors ethernet0
IPv6 Address                          Age Link-layer Addr State Interface
2000:1:2::10                           - 0000.1234.5678  REACH Ethernet0
2000:1:2::15                            0 0000.2345.5678  REACH Ethernet0
2000:1:2::17                            1 0000.2222.5678  REACH Ethernet0
```

To view ICMPv6 traffic and other general traffic on the router, issue the command *show ipv6 traffic*. This will give the IPv6 statistics, broken down into general, ICMP, and User Datagram Protocol (UDP) statistics. Take note of the general statistic *not a router*, which will have a value if the router is not globally configured for IPv6. Although this is a good indicator that the router doesn't have IPv6 enabled, it is not a fool-proof way to check, unless the counters were just cleared. The best way to see if IPv6 is globally configured, as mentioned before, is to check the running configuration.

The statistics listed below are the ICMP statistics. In the *Sent* portion you can see how many of the packets were rate-limited. You can also see how many errors there were and whether you need to change your rate-limit and bucket size. These outputs are similar to the outputs from an IPv4 command.

```
6Router-1#show ipv6 traffic
IPv6 statistics:
  Rcvd:  4903 total, 4892 local destination
```

```
           0 format errors, 0 hop count exceeded
           0 bad header, 0 unknown option, 0 bad source
           0 unknown protocol, 0 not a router
           0 fragments, 0 total reassembled
           0 reassembly timeouts, 0 reassembly failures
  Sent:  27330 generated, 0 forwarded
           0 fragmented into 0 fragments
           1 encapsulation failed, 11 no route, 0 too big
  Mcast: 0 received, 0 sent

ICMP statistics:
  Rcvd: 36 input, 0 checksum errors, 0 too short
          0 unknown info type, 0 unknown error type
          unreach: 0 routing, 0 admin, 0 neighbor, 0 address, 0 port
          parameter: 0 error, 0 header, 0 option
          0 hopcount expired, 0 reassembly timeout, 0 too big
          15 echo request, 10 echo reply
          0 group query, 0 group report, 0 group reduce
          0 router solicit, 0 router advert, 0 redirects
          2 neighbor solicit, 9 neighbor advert
  Sent: 2561 output, 0 rate-limited
          unreach: 0 routing, 0 admin, 0 neighbor, 0 address, 0 port
          parameter: 0 error, 0 header, 0 option
          0 hopcount expired, 0 reassembly timeout,0 too big
          15 echo request, 15 echo reply
          0 group query, 0 group report, 0 group reduce
          0 router solicit, 2480 router advert, 0 redirects
          25 neighbor solicit, 26 neighbor advert

UDP statistics:
  Rcvd: 4797 input, 0 checksum errors, 0 length errors
          0 no port, 0 dropped
  Sent: 24701 output
```

Summary

Configuring IPv6 is not unlike configuring IPv4—you still need an IP address on your Ethernet interface in order for that interface to operate using IP, and whether an IPv4 or IPv6 address is used does not affect the way that Ethernet works or how the router will actually route the packet. The major differences are in the syntax used to enter the address. Most of the commands that involve IPv6 specifically have *ipv6* somewhere in the syntax of the command, whereas IPv4 just uses the syntax *IP*.

In this chapter we learned how to configure IPv6 addresses on LAN interfaces. We also learned that there are three types of addresses that can be configured on a LAN address: the *global*, *site-local* and the *link-local* addresses. The global address is a well-known address that can be routed throughout the Internet. The site-local address is one that can be routed anywhere within the site but cannot be routed to the Internet. The link-local address is an address that can only be used for the specific interface on which it is configured.

These three types of addresses are configured using the *ipv6 address* command followed by certain parameters that distinguish the address as one of the three types. You can assign specific IPv6 addresses to these interfaces, or you can use the EUI-64 command to let the router use its identifier to fill in the last 64 bits of the address. (This is not recommended for the global address, but should be fine for the link-local address.) Unnumbered addresses are assigned in a manner very similar to IPv4.

We discussed several parameters and ways to tweak them for your network. Duplicate address detection was discussed, and we saw how to alter the number of neighbor solicitation messages that are sent out. We saw that, contrary to typical Cisco IOS protocol, entering the *no* command for duplicate address detection does not disable it—rather, it resets it to its default parameters. We saw that the process of configuring DNS is almost identical to that of IPv4 as well as with statically mapping host names to IPv6 address. Once again, the only difference was that *ipv6* was specifically typed into the syntax to differentiate between IPv6 and IPv4 addresses. The *rate limit* for ICMPv6 is a new parameter that can be changed—some basic functions like *traceroute* can have problems if this limit is not altered, and not having traceroute at your disposal can make troubleshooting much more difficult.

We looked at configuring for WAN interfaces, focusing on the Frame-Relay and ATM connections. Once again, we saw that there was not much difference between IPv4 and IPv6 except for the syntax of the commands. A few examples where given to show how ATM can be mapped to certain PVCs and how Frame-Relay can be mapped to different DLCIs.

We also looked at ways to verify that all of our configurations went through correctly. We learned how to verify that the global command has been configured on the router for IPv6, as well as how to identify what interfaces IPv6 has been configured on and what the IPv6 address of each interface is. The different IPv6 parameters were checked with various *show* commands. These commands were, once again, similar to IPv4 except in syntax. The output of the *show* commands was also very similar to the output in IPv4. Because the outputs are similar, we can read them faster and more efficiently.

One thing that we noticed throughout the configuration and verification processes was that IPv6 is still growing and being tweaked in the Cisco IOS. There are minor changes in each revision of the Cisco IOS that can cause slight differences in the way various commands will work. This will probably be the case for some time because IPv6 is still growing and becoming more popular as time goes on. You should always be aware of what version of Cisco IOS you are running on your router, and be sure to check Cisco CCO to how IPv6 has been updated with the different releases.

Solutions Fast Track

Configuring IPv6 Addressing

☑ When configuring your interfaces, remember that *IPv6* must be included in the syntax to distinguish between IPv4 and IPv6 addresses.

☑ There are three types of addresses that can be assigned on an interface: global, site-local and link-local. These different types of addresses are defined on the interface with different configuration parameters. These addresses must be within their respective address spaces, as defined earlier in the book.

☑ If you are using the *EUI-64* command when entering an IPv6 address, remember that the router uses its identifier for the last 64 bits of the IPv6 address, therefore if you want to find out the address, you have to go back and get the full IPv6 address.

☑ With the configuration of duplicate address detection, you can specify the number of solicitation messages sent out. If you want to disable this feature, set the number of solicitation messages to zero. Entering the **no** command will restore the number of solicitation messages to the default of one.

- ☑ The ability to configuring the rate limit for ICMPv6 is a new feature that was not available in IPv4. This feature allows you to set the rate at which error messages are sent out, as well as to configure a bucket size for the number of error messages that can be sent.
- ☑ WAN connections, like Frame-Relay and ATM, function the same way with IPv6 as with IPv4. The only differences are that you are using an IPv6 address and that you can enter a link-local address.
- ☑ Most of the commands that involve IPv6 have *IPv6* in the actual syntax used to configure the command on the router. If *IP*v6 is not specified an IPv4 address will be assumed, and you will receive an error from your router because it will not know how to read the address. One exception is entering the DNS information; that command uses just *IP* in the syntax, but will accept either an IPv6 or IPv4 address.

Verifying IPv6 Addressing

- ☑ There is no Address Resolution Protocol with IPv6; it has been replaced with the *neighbor discovery* protocol. Showing the neighbor discovery will give the link-layer address that is associated with the IPv6 address.
- ☑ When using *show* commands to view IPv6 configurations, then *ipv6* is usually in the syntax of the command. If just *IP* is used, typically only IPv4 results will be produced.
- ☑ Remember to always know what version of Cisco IOS you are running, because IPv6 is still an emerging technology and there will be changes is almost every new revision of Cisco IOS that becomes available.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: Why is the router giving me an error when I try to configure an IPv6 link-local address?

A: The most likely reason is that you are entering an invalid link-local address. Remember from previous chapters that the first 10 bits define the type of address. The start of a link-local address must fall in the range of FE80 to FEBF.

Q: Can I assign multiple addresses on my interfaces?

A: Yes, you can, as long as you have the correct version of Cisco IOS. In order to have multiple addresses with the same prefix length, you must have Cisco IOS version 12.2(4)T or later; with prior versions of code you will receive an error message. Keep in mind that you can only have multiple addresses for global and site-local address—multiple link-local addresses are not permitted.

Q: How do I know what version of Cisco IOS I need to run on my router?

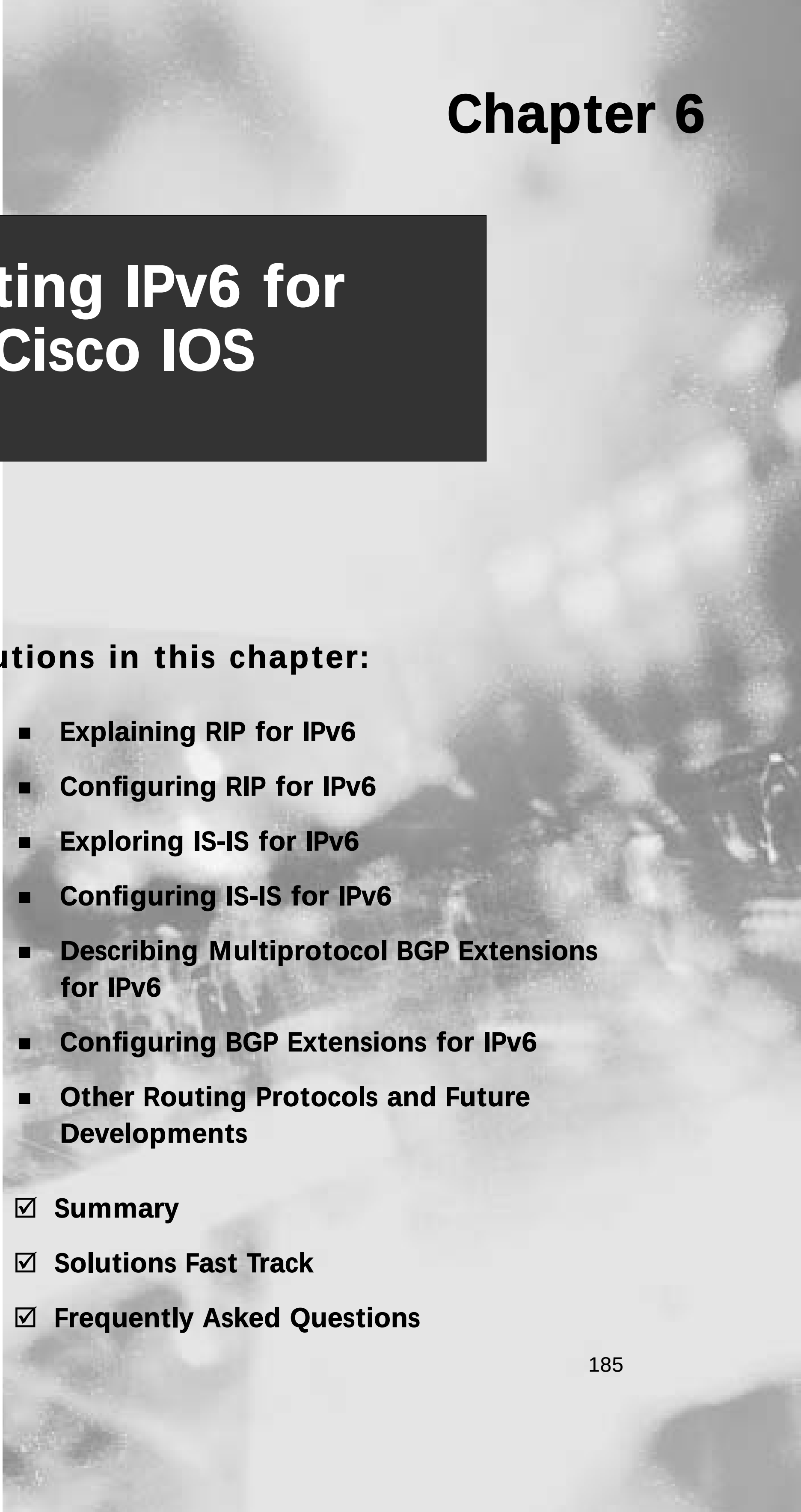
A: You will need at least Cisco IOS version 12.0(21)ST or 12.2(2)T to run IPv6. I would recommend one of the latest copies that Cisco has released, because IPv6 is constantly evolving and there will be changes to IPv6 in each revision. Of course, be sure to check CCO for the latest caveats for each release.

Q: Can I have an IPv6 address and an IPv4 address on the same interface?

A: IPv6 and IPv4 act independently of each other, therefore it is possible to have both versions on one interface. However, there is no way to check both IPv6 and IPv4 addresses with the same command.

Q: What is neighbor discovery?

A: Neighbor discovery is the equivalent of ARP in IPv4. Neighbor discovery will resolve the MAC address to the IPv6 address. This process is the same as ARP: it will hold the MAC address in its table for a specified amount of time, and if there is no activity from that MAC address the table will flush the entry out.



Chapter 6

Routing IPv6 for the Cisco IOS

Solutions in this chapter:

Introduction

The routing of IPv6 traffic has changed significantly. New versions of routing protocols have been developed to support dynamic routing on IPv6 networks, because there have been only a few routing protocols built into the Cisco IOS to this point. The Routing Information Protocol (RIP), Intermediate System-to-Intermediate System (IS-IS), and the Border Gateway Protocol (BGP) have been updated and built into the Cisco IOS already, and new versions of Open Shortest Path First (OSPF) and Enhanced Interior Gateway Routing Protocol (EIGRP) will be supported with future releases of the IOS.

Routing IPv6 remains as complex as it was with IPv4, but it has been enhanced to support the enhanced addressing characteristics of IPv6. Configuring a network for a dynamic routing protocol has long been a task required of network engineers and designers and it will remain a valued skill with IPv6. This chapter outlines the necessary steps to configure each supported routing protocol and explores the details of how each protocol operates on the network.

Explaining RIP for IPv6

To discuss RIP—also referred to as RIPng (Next Generation)—for IPv6, a brief discussion of its predecessors, IPv4 RIP v1 and v2, will be necessary. Version 2 represents several enhancements to the original RIP standard. The most notable enhancement is the support for classless routing (also known as variable-length subnet masking, or VLSM), as well as authentication for RIP routing updates. Although two different versions exist, they are essentially the same protocol and share the same common characteristics, advantages, and flaws.

The Routing Information Protocol is one of the oldest and most stable dynamic routing protocols in use today. RIP's ancestry dates back as far as 1969 with its involvement in the original DARPANet project. RIP is a true open-standard, distance-vector protocol. Distance-vector protocols represent the first generation of dynamic-routing protocols.

Distance-vector protocols are based primarily on the Bellman-Ford algorithm, outlining many of the primary functions of such protocols. The Bellman-Ford algorithm gets its name from the individuals who developed the concept of distance-vector route calculation. This algorithm bases its metric calculation on a single-path tree concept, using the parameter of weight, and using a relaxation technique progressively throughout a path. This technique enables the protocol to select from several paths to a destination network, using a Boolean expression to determine the weight of a particular path and to select the path with the best metric. This is a very brief discussion of this process; a detailed discussion of the Bellman-Ford algorithm is outside the scope of this chapter. The term *distance vector* comes from the function of the protocols. Protocols use a vector, or list, of distances or hop counts to determine the optimal routes to a given destination network. All distance-vector routing protocols share several key elements, which we'll discuss in the following sections: periodic updates, routing loops, triggered updates, split horizon, counting to infinity, poison reverse, hold-down timers, and RIP timers.

Periodic Updates

Routers use routing updates to propagate topological and reachablity information throughout the network. RIP networks rely heavily on such updates to ensure full connectivity to the network around them as well as to ensure that neighboring routers are still online and functioning properly. All this is done to obtain a network state known as convergence, which is when all the routers within a network agree on the network topology.

RIP represents the first standardized routing protocols developed, and, due to this and to its ease of configuration, it still represents a very large portion of the installed base of routing protocols today. The manner in which distance-vector protocols transmit their routing updates is one of the reasons that link-state routing protocols, such as OSPF, are now considered a better alternative than distance-vector routing protocols.

Distance-vector protocols transmit their entire routing table to all directly connected neighbors at predefined intervals. The update timer within distance-vector protocols governs the timing and sequence of these updates. The default value for this timing is 30 seconds for RIP. This means that, by default, every router in the network will push an entire routing table to every directly connected neighbor every 30 seconds regardless of any changes in the network. In other words, in a stable environment, every neighbor of a given router will receive that router's entire routing table every 30 seconds to tell it things about the network that it most likely already knew. Seems like a lot of overhead, doesn't it? This has been one of the major reasons that distance-vector protocols have lost much of their popularity to the more advanced link-state protocols.

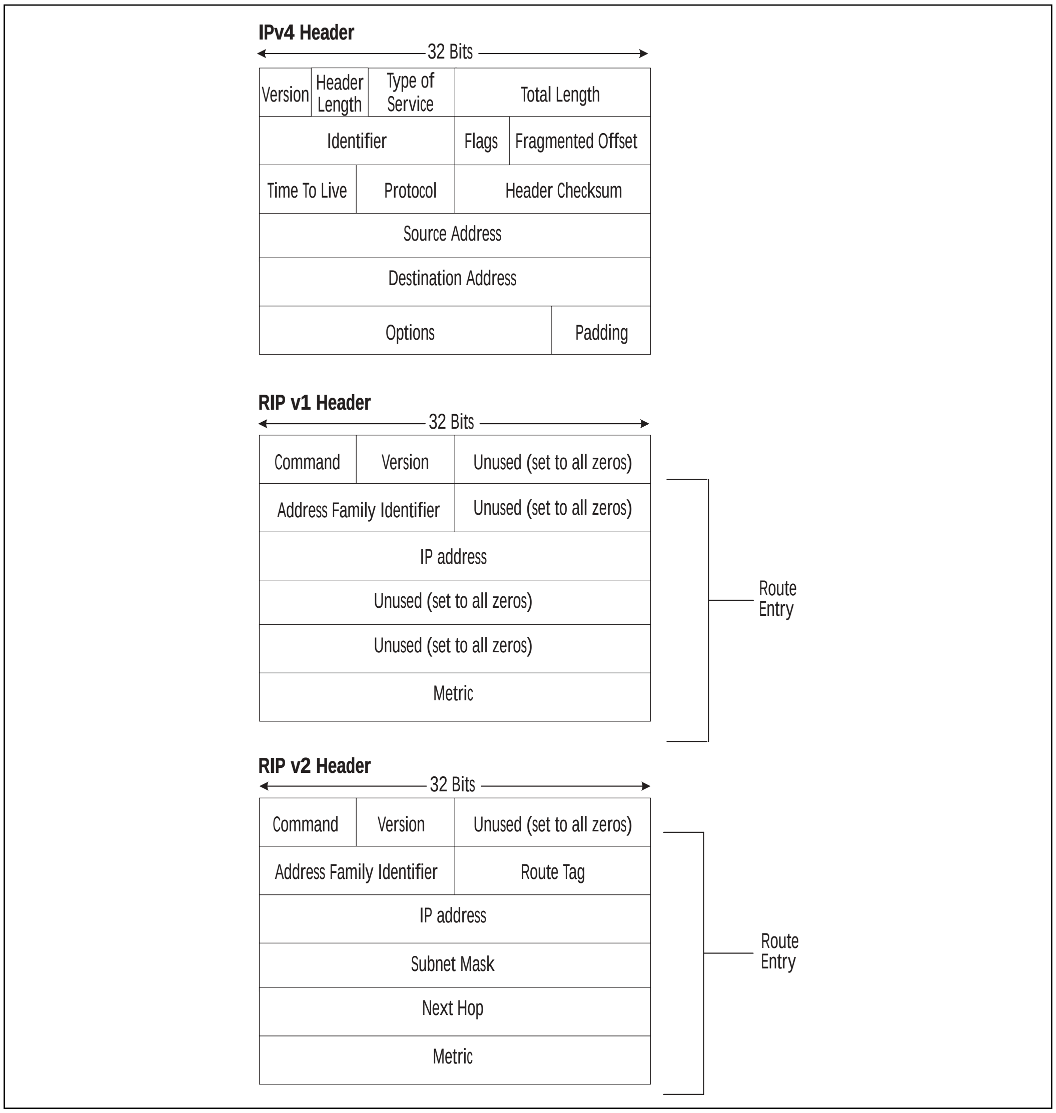
Routers take a very localized view of the things surrounding them, both in calculating path cost to networks from their perspective as well as in broadcasting and advertising network reachability with regard to what is immediately adjacent to them. In other words, all the routers know about the network only from their own perspective. No single router can know about the entire network topology firsthand; instead, every router must rely on third-party information provided by its neighbors just as its neighbors must rely on information that they received from their neighbors. This leads to a problem that almost every distance-vector routing protocol faces: *routing by rumor.*

Routing by rumor is a serious problem for distance-vector protocols. Because of the manner in which distance-vector protocols view the network, from a localized perspective, it is not possible for every router to have first-hand information for the entire network. Instead, each router must rely on the periodic updates transmitted to it from its directly connected neighbors. So, if incorrect routing information is transmitted at any point, that information will propagate throughout the network unchecked, because every router trusts its neighbor routers to provide correct information. Many compare this process to a childhood game called "telephone," in which children sit in a circle and the first child is given a sentence to whisper to the next child , who then whispers that same sentence to the next child , and so on until it gets all the way around the circle. The funny part being that the message that started at the beginning rarely is the message received at the other end of the circle. This is, unfortunately, a very good comparison. The nature of distance-vector protocols is to pass along "facts" about the network, of which they have no direct knowledge. This process causes a high margin of error within distance-vector routing protocols. As the size and diameter of a distance-vector–routed network increases, the probability and severity of this type of problem also increases.

Like all other distance-vector protocols, RIP sends its entire routing update to all directly connected neighbors through the use of periodic updates, every 30 seconds in this case, using UDP port 520 for this broadcast. This update comes regardless of network change and or link failure. A maximum of 25 networks can be contained within a single RIP packet. The protocol uses the update not only to communicate routing update changes, but also to verify neighbor connectivity and activity. All of these factors make RIP a good solution for small– to mid-sized networks only. Figure 6.1 shows examples of the RIP version 1 and version 2 packets. Notice that the size and number/size of fields do not change between the two protocols; however, RIP v2 packet has added the Route Tag, subnet mask, and next hop field. These additional fields enable RIP v2 to support variable-length subnet masking (VLSM) as well as authentication when making routing updates. Also included is an example of the IPv4 header. A single RIP routing entry can contain as many as 25 networks, making the size of a single RIP packet as large as

512 bytes. Add that to the corresponding 20 to 24 bytes of the IPv4 header attached and you have a large amount of traffic traversing your network!

Figure 6.1 RIP v1, RIP v2 Packets and IPv4 Header



Routing Loops

From our discussion in the previous section, you know that distance-vector protocols take a very localized view of the network, considering only how an individual router will reach a given destination network and the metrics for reaching that network. When transmitting its routing table to directly connected neighbors, this is also the information that it will send out; that is, how it will get to a destination network. Due to these factors, as well as the periodic updates used to transmit network and topology changes in a distance-vector routing environment, routing loops can be a real threat to a distance-vector network.

Routing loops are a state in which, due to incorrect information being propagated through a network, packets will continually encircle a network unable to reach the given destination network. Routing loops pose a great danger and are more common in distance-vector routing networks because of the delay in the propagation of network information as well as the problem of "routing by rumor" discussed in the previous section.

In a routing loop, a network has become unreachable due to a link failure or some other unforeseen circumstance. The directly affected router will transmit this change to its directly connected neighbors, informing them of the inability to reach this network; however, this update may not make it to every router within the network, and another router may transmit its ability to reach the network in the meantime, using the link through the first router that saw the link was down. So now all the routers, including the one that was first affected by the link failure, will update their routing tables to use the new router and route packets to this router, which in turn will forward them back to the original router, which will forward them to the transmitting router, so it can reach the network. A loop has now occurred.

Routing loops can be detrimental to the speed and performance of a production network. As more packets are sent to the inaccessible network, more and more packets will continue to encircle the network, screeching network traffic to a standstill. Fortunately, the designers of RIP foresaw this problem early on and built several countermeasures into this distance-vector routing protocol, such as triggered updates, split horizon, counting to infinity, poison reverse, and hold downs.

Triggered Updates

As you already know, distance-vector protocols use periodic updates to propagate network information to neighboring routers and to ensure connectivity. But, in many instances this is not enough to ensure connectivity. In such cases, triggered updates serve as a critical element to ensure that connectivity is maintained at all times and that link changes are transmitted on a real-time basis.

Triggered updates are a mechanism used by routing protocols to deal with sudden network changes such as link failure. Upon detection of a network change, the affected router(s) will send updates informing its neighbors of the new conditions; the neighbors will in turn forward this information to their neighbors, thus helping to prevent conditions such as routing loops and decreasing convergence time. Triggered updates augment periodic updates to ensure that routers would not have to wait until the next periodic update to know about a network change.

When sending a periodic update, routers transmit their entire routing table, regardless of change. Triggered updates, on the other hand, send only information regarding the change in the link or network status that has occurred. This fact alone makes triggered updates much more efficient than periodic updates. This also helps distance-vector protocols to converge at a much faster rate, allowing for data transfer to resume much faster.

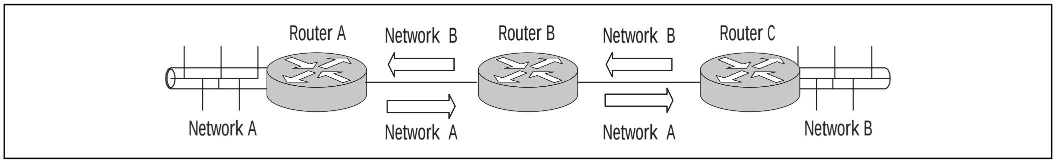
Split Horizon

Loops in a production environment can be a major service-affecting, if not service-halting, problem. Split horizon is a mechanism employed by distance-vector routing protocols used to help limit routing loops. This is done by a very simple rule stating that a network will not be advertised out of the same interface through which it was learned. Routers will omit these routes from their routing tables when sending their routing tables back out of the interface through which the route was learned. In the examples of sample networks we have discussed thus far, split horizon could effectively stop routing loops from occurring. However, split horizon cannot prevent routing loops in every situation.

Many people dealing with split horizon encounter a problem with non-broadcast multi-access mediums such as frame-relay or ATM. Many implementations of NBMA functionality work in a basic hub and spoke topology, a central router, or hub router, with several spoke routers hanging off it. The problem arises when all of these spoke routers terminate into one single interface on the hub router (a multi-point Frame Relay interface, for example). According to the rules of split horizon, an interface cannot advertise a network out of the same interface through which the network was learned and, unfortunately, the hub router cannot, therefore, send the update to the other routers also attached to that interface.

This has been recognized as a problem by Cisco and split horizon is disabled by default on all Frame Relay physical interfaces. However, split horizon is enabled on all point-to-point sub-interfaces and all point-to-multipoint sub-interfaces. When using point-to-point sub-interfaces, each link is treated like a separate point-to-point link (so there is no conflict with the rule of split horizon); whereas, with point-to-multipoint sub-interfaces, all sub-interfaces are treated as a shared medium and must be in the same subnet to communicate. Obviously, this creates a problem with the rule of split horizon. Disabling split horizon leaves the possibility of routing loops, so you must be very careful in this type of environment. Figure 6.2 demonstrates split horizon operation.

Figure 6.2 Split Horizon



Counting to Infinity

As we discussed earlier, some distance-vector protocols calculate their routing metric based on the number of hops or routers that must be traversed to reach a given destination network. Within these routing protocols the maximum number of hops, or infinity, is a number set specifying the maximum distance or diameter of a network that a packet can cross to reach a given destination network before it is considered to be unreachable. RIP, our prime example of a hop–count-based, distance-vector routing protocol, has an infinity number set at 16 hops. If a destination network is 16 or more hops away, it will be considered unreachable and will not be considered for packet forwarding by the router. The concept of counting to infinity is part of RFC 1058, the RFC that specifies the operation of RIP v1.

Counting to infinity is incorporated into distance-vector routing protocols as a means of eliminating the possibility of packets traversing the network aimlessly for an eternity due to a downed link and the routing loop that could ensue thereafter. In such a situation, the packet would be permitted to traverse the network for a maximum of 16 hops before it would be removed from the network. As the packet bounces from router to router in a situation such as this, the Time To Live (TTL) field is incremented by one. When the counter reaches 16, the packet is removed from the network by the next router it encounters.

Many feel that the hop count limitation placed on RIP of 16 is too small for many of today's networks and organizations, but when you consider 16 hops of aimlessly circling a network, 16 becomes a very large number. Imagine the lost traffic and delay associated with a packet that is counting to infinity and think of what even double that amount, 32, could cause.

Counting to infinity is not by itself a loop-prevention mechanism; rather, it should be considered a last line of defense for dealing with routing loops. The concept of infinity, however, is incorporated into a loop-prevention mechanism known as poison reverse.

Poison Reverse

In the previous section, we introduced the concept of counting to infinity, or the maximum number of hops that a packet can traverse before the network is considered unreachable. Poison reverse works with this concept as well as with split horizon to help prevent loops. RFC 1058 outlines poison reverse (this is the same RFC that discusse the operation of RIP v1). It is used as a solution to the problem of counting to infinity, or packets circling the network until they have reached the maximum number of hops—16, in the case of RIP.

In a standard application of split horizon, a rule states that a network cannot be advertised out of the interface through which the network was learned. Using this as a basis, poison reverse takes this a step further. When using poison reverse and split horizon in concert, networks learned via a certain interface are advertised back out the same interface, with one major difference: the route is given a metric of unreachable, or infinity, so that the receiving router will not consider, under any circumstances, the use of that route (because it will be seen as unreachable). In this case, the route is said to have been *poisoned*. Routers that receive this route will not consider it as an alternative path and will not add it to their routing tables, thus preventing a routing loop in this situation.

Poison reverse is sometimes considered a disadvantage because it increases the size of routing updates. In simple split horizon operation, routes learned via an interface would not be transmitted out that interface, thus reducing the size of the routing update. However, when using poison reverse, the routes are included, but with the infinity metric. This causes larger updates and an increase in bandwidth utilization. However, this is generally seen as an acceptable tradeoff for the increased protection against routing loops.

Hold-Down Timers

As we have discussed through several of the preceding sections, link failure can cause several problems throughout a production network. We have also discussed several of the different mechanisms, such as triggered updates and split horizon, that help to deal with the effects of link failure, especially loop avoidance. Now we will turn our attention to the final of these safety mechanisms, hold downs.

Hold downs are safety mechanisms that prevent regular routing update messages from re-instating a route that is down due to link failure or other unforeseen circumstances. In normal routing operation, when a router goes down or a link to a network fails, the neighboring routers re-calculate their routing metrics and find or choose a new path to reach the down network. However, during this time, a non-directly connected router could possibly re-transmit a routing update specifying that the link is up, before triggered updates can reach it to inform it that the route is in fact down. This router could potentially update all the devices that had just been notified that the link was down, and cause them to re-instate the route.

Hold downs specifically address this issue, by specifying an amount of time, in seconds, that the router should wait before accepting any changes to the status of the route. This helps to prevent the scenario discussed in the previous paragraph. Hold downs are generally calculated to be greater than the period needed to update every router in the network of the failed link.

Distance-vector routing protocols use *hold-down timers*, which are mechanisms that specify the amount of time, in seconds, that the router will suppress better path information for a route for which the router has received information that the path is down. During this time, the route will be marked as inaccessible and is advertised as unreachable; however, it will still be used to forward traffic. The default hold-down timer is 180 seconds for RIP. When the hold timer expires, routes advertised by other sources will be accepted for the given network.

RIP Timers

A set of three timers is incorporated into RIP; they govern the *update interval*, the *hold-down interval*, and the *flush timer*. The update interval is the amount of time that a router will wait before sending its entire routing table to all its neighbors. The default value for this is 30 seconds. Many believe that this is too often, so it is a configurable parameter. Keep in mind, though, that the update interval should be uniform across your RIP network to avoid your routers being out of sync.

The next value is the hold-down timer. If you recall from our earlier discussion, when a route is placed in the hold-down state, no routing updates are received for that route, and that route will not be advertised. The hold-down timer is the amount of time that the router will wait without receiving updates before putting a route into hold-down state. The default value for this is 180 seconds; this value is also configurable.

The final timer is the flush timer. This timer specifies the amount of time in seconds that the router will wait before completely removing a route from its routing table, after which a new route to the destination network is sought. The default value for this is six times the update interval, or 240 seconds. This value is also configurable.

Administrative Distance

Cisco routers incorporate the concept of *administrative distance* when making routing decisions. Administrative distance is a scale from 0 to 255 that specifies the reliability of a given route. 0 is the most reliable and 255 is the least reliable—actually it is considered unreachable. Each of the routing protocols that Cisco routers support is given a specified default administrative distance; however, these values can be manipulated when necessary. RIP in all its forms, IPv4 v1and v2 as well as IPv6, are assigned the default value of 120. You can change this value for RIP in RIP configuration mode by use of the *distance* command. Table 6.1 provides the administrative distances of routing protocols on Cisco routers.

Table 6.1 Administrative Distances

Routing Protocol	Administrative Distance
Connected Interface	0
Static Route	1
EIGRP Summary Route	5
External BGP	20
Internal EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP v1, v2, IPv6 RIP	120
EGP	140
External EIGRP	170
Internal BGP	200
Unknown (Unreachable)	255

Configuring RIP for IPv6

The RIP standard for IPv6, or RIPng, is built upon the original RIP standard, which is why it is important to have a thorough understanding of the original protocol. However, RIPng does offer a few enhancements to the original RIP standard; these include, of course, full support for the IPv6 standard and prefixes as well as support for an all-routers RIP multicast address group for routing update messages to all RIPng routers. This offers an alternative to the broadcast messaging solution used by the previous implementations of RIP.

Basic IPv6 RIP Configuration

Although IPv6 RIP is very similar to its predecessor RIP, some striking differences exist between the two protocols when it comes down to their actual configuration. The first of these differences is apparent when enabling the RIP routing process. In the previous release, you started the RIP process by giving the command *Router rip* in global configuration mode followed by specifying the networks that would be included in the RIP routing domain by typing the *network* command. The configuration for IPv6 RIP is similar, but not identical. To enable the RIP routing process for IPv6, the *ipv6 router rip <word>* command is used. The *word* command is actually a process identifier that is a user-defined series of numbers or letters that identify the specific RIP process. This enables the router to run multiple segregated RIP processes on the same router. To add specific networks to the RIPng routing domain, the *ipv6 rip <word> enable* command is used. This command places that specific IPv6 enabled interface into the specified IPv6 RIP routing domain. The following listing shows an example of this process on the router:

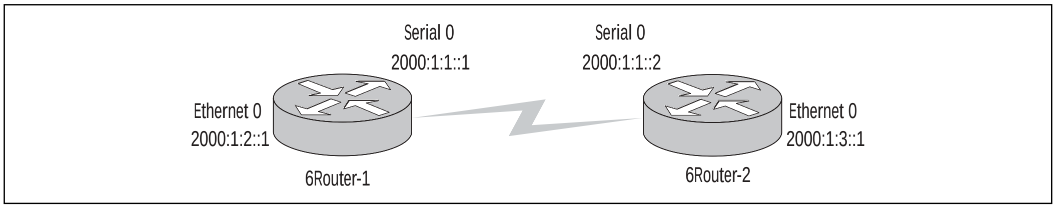
```
6Router-1>enable
Password:
6Router-1#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 router rip cisco
6Router-1(config-rtr)#exit
6Router-1(config)#interface ethernet 0
6Router-1(config-if)#ipv6 rip cisco enable
6Router-1(config-if)#exit
6Router-1(config)#interface serial 0
6Router-1(config-if)#ipv6 rip cisco enable
```

```
6Router-1(config-if)#exit
6Router-1(config)#
```

In the preceding example, we entered global configuration mode by issuing the *config terminal* command and enabled the Ipv6 RIP routing process by using the *ipv6 router rip cisco* command. We then included both the Ethernet 0 and Serial 0 interfaces into the Ipv6 RIP routing process by issuing the *ipv6 rip cisco enable* command on each interface in interface configuration mode.

Figure 6.3 shows an example of a simple IPv6 RIP network.

Figure 6.3 Simple IPv6 RIP Network



In the example shown in Figure 6.3, we have created a RIP routing domain with three separate subnets, 2000:1:1::/64, which is the network that runs between the two router's serial links, and we have two networks off of the Ethernet 0 routers of each of the routers 2000:1:2::/64 and 2000:1:3::/64 respectively. By issuing *show ipv6 protocol*, we can see all the IPv6 routing protocols currently running on the router. In the following listing, we see that we have both connected networks, static routes, and of course RIP running on our 6Router-1 router:

```
6Router-1#show ipv6 protocol
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "rip cisco"
  Interfaces:
    Serial0
    Ethernet0
  Redistribution:
    Redistributing protocol rip cisco
```

By issuing the *show ipv6 route* command, you can see the routes derived for IPv6. At the top of the following code example, you can see that IPv6 can currently support seven types of routes (Connected, Local, Static, RIP, BGP, IS-IS levels one and two, as well as interarea IS-IS—we will cover IS-IS later in this chapter). Routes derived from a RIP routing process are denoted with an *R*. The following example shows that 6Router-1 has derived one route from the RIP routing process, a connection to 2000:1:3:: /64, which, of course, is the network off the Ethernet 0 interface on 6Router-2. We also see the interface that RIP uses to get to that network, in this case the Serial 0 interface of 6Router-1.

```
6Router-1#show ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
Timers: Uptime/Expires

L   2000:1:1::1/128 [0/0]
     via ::, Serial0, 1d05h/never
C   2000:1:1::/64 [0/0]
     via ::, Serial0, 1d05h/never
L   2000:1:2::1/128 [0/0]
     via ::, Ethernet0, 1d05h/never
C   2000:1:2::/64 [0/0]
     via ::, Ethernet0, 1d05h/never
R   2000:1:3::/64 [120/2]
     via FE80::2E0:B0FF:FE55:B035, Serial0, 00:00:13/00:02:46
L   FE80::/10 [0/0]
     via ::, Null0, 1d05h/never
L   FF00::/8 [0/0]
     via ::, Null0, 1d05h/never
6Router-1#
```

Default Routes in IPv6 RIP

A *default route* is, essentially, a route that a router and/or routing protocol uses to forward packets for which it does not have the actual destination network address in its routing tables. Default routes have been used widely by almost all IPv4 routing protocols. IPv6 RIP also supports the utilization and configuration of default routes.

Unlike with IPv4 RIP, however, where you configured default routes at the global level, you configure IPv6 RIP for default routes on an interface level by using the *default-information* command. This command tells the router to inject a

::/0 route into the RIP routing domain as a default route (or, as it is also known, a gateway of last resort). Follow the *default-information* command by one of two parameters: *originate* or *only* settings.

The *originate* setting tells the router to inject this ::/0 route into the RIP routing domain and to advertise this route along with all the other routes in its RIP routing advertisements. The *only* parameter instructs the router running IPv6 RIP to advertise only this default route, and to suppress advertisement of any other routes.

The following listing show an example of default route configuration in an IPv6 RIP routing process. We have configured the serial 0 interface of our 6router-1 router to originate a default route and to advertise it to the rest of the IPv6 RIP routing domain.

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#interface serial0
6Router-1(config-if)#ipv6 rip cisco default-information ?
  only        Advertise only the default route
  originate  Originate the default route

6Router-1(config-if)#ipv6 rip cisco default-information originate
6Router-1(config-if)#exit
```

In the preceding example, we entered interface configuration mode and entered the *ipv6 rip cisco default-information originate* command, thus telling the router to use the network attached to the serial 0 interface as the default route.

If you look at the routing table of the neighboring 6router-2, you will see that 6router-1 is indeed injecting that default route into your RIP routing domain. If you examine the preceding routing table, you will see an additional RIP entry for the ::/0 network via serial 0. Note that entries for RIP routes have an "R" next to them. This is your default route:

```
6Router-2#show ipv6 route
IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
Timers: Uptime/Expires

L   2000:1:1::2/128 [0/0]
      via ::, Serial0, 5d14h/never
```

```
C   2000:1:1::/64 [0/0]
     via ::, Serial0, 5d14h/never
R   2000:1:2::/64 [120/2]
     via FE80::2E0:B0FF:FE5A:D998, Serial0, 00:00:09/00:02:50
L   2000:1:3::1/128 [0/0]
     via ::, Ethernet0, 5d14h/never
C   2000:1:3::/64 [0/0]
     via ::, Ethernet0, 5d14h/never
L   FE80::/10 [0/0]
     via ::, Null0, 5d14h/never
L   FF00::/8 [0/0]
     via ::, Null0, 5d14h/never
R   ::/0 [120/2]
     via FE80::2E0:B0FF:FE5A:D998, Serial0, 00:00:09/00:02:50
6Router-2#exit
```

IPv6 RIP Route Redistribution

Route redistribution is a very useful mechanism. Redistribution enables routes created and calculated by another routing process to be injected into your routing domain, in this instance, our IPv6 RIP routing domain. Redistribution, as a practice, offers several benefits, such as enabling full connectivity throughout your network without having to run one standard protocol throughout. This enables administrators to leverage the best that each of the different types of routing protocols has to offer for individual parts of the overall network. Network administrators commonly incorporate redistribution between their core and edge networks. Administrators will often select one routing protocol, such as OSPF, to act as a network backbone protocol, and utilize a less resource-intensive protocol, such as RIP to run at the network edge, thus enabling the network to segment core traffic from edge traffic, while enabling full connectivity throughout the network.

Route redistribution for IPv6 RIP operates on these principles as well; however, the scope of redistribution is limited to other IPv6 protocols. This list currently includes IS-IS and BGP. RIP also has the capability to redistribute other RIP routing processes as well as static and connected routes.

You perform the IPv6 RIP route redistribution at the RIP configuration level with the *redistribute* command. Follow this command with the particular protocol that you wish to redistribute. Each one of these protocols has specific configuration steps that will follow. Start with static and connected routes. These routes, aside from another RIP process, are the easiest to configure for redistribution. Upon selecting one of these two choices, you can complete configuration by simply pressing **Enter**; however, you can perform further configuration by using the *metric* and *route-map* parameters. The *metric* command specifies a metric that IPv6 RIP assigns to use when calculating the best path to take to reach a given network. Remember that RIP uses hop-count as its primary metric; when distributing protocols into RIP, it will automatically assign a value of 1 for redistributed routes. If this is not preferable, the *metric* command can change this to the desired metric. You can use the *route-map* command, followed by the name of a route map to use, to incorporate policy-based routing for this route. *Policy-based routing* is a mechanism that enables routers to make routing and forwarding decisions based on numerous factors such as the type of traffic and the interface that the packet arrived on. It enables administrators to have granular control over traffic decisions in their network. The following listing is an example of static route redistribution in our RIP network. We will redistribute static routes into the RIP routing domain and assign a metric value of 4 to the redistributed static routes:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 router rip cisco
6Router-1(config-rtr)#redistribute ?
  bgp        Border Gateway Protocol (BGP)
  connected  Connected Routes
  isis       ISO IS-IS
  rip        IPv6 Routing Information Protocol (RIPv6)
  static     Static Routes

6Router-1(config-rtr)#redistribute static ?
  metric     Metric for redistributed routes
  route-map  Route map reference
  <cr>

6Router-1(config-rtr)#redistribute static metric 4
6Router-1(config-rtr)#exit
6Router-1(config)#
```

You can verify redistribution configuration by looking at your IPv6 routing protocols by issuing the *show ipv6 protocols* command. The redistribution portion of the following command output shows that this router is redistributing static routes with a metric of 4. By specifying a metric of 4, you are telling the router that the network is 4 hops away. Remember, the maximum number of hops is 15, with 16 being unreachable. When manually assigning metrics to routes, remember to assign the appropriate metric for a given network. Not doing so can lead to sub-optimal routing performance.

```
6Router-1#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "rip cisco"
  Interfaces:
    Serial0
    Ethernet0
  Redistribution:
    Redistributing protocol static with metric 4
6Router-1#
```

Redistribution of another RIP routing process is very similar to the redistribution of static or connected routes. To accomplish this, use the *redistribute rip <word>* command. The <*word*> parameter specifies the RIP routing process that you wish to redistribute. You can also incorporate the optional commands of *metric* and *route map* into this redistribution. One important note to make, however, is that when redistributing RIP into RIP, the metrics will be retained, and you need to manipulate the metric settings only if you do not want to use the standard RIP routing metrics for redistributed routes. The following listing shows an example of RIP into RIP route redistribution. We are configuring the Cisco RIP process to redistribute the Cisco2 RIP process:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 router rip cisco
6Router-1(config-rtr)#redistribute ?
  bgp         Border Gateway Protocol (BGP)
  connected   Connected Routes
  isis        ISO IS-IS
  rip         IPv6 Routing Information Protocol (RIPv6)
  static      Static Routes
```

```
6Router-1(config-rtr)#redistribute rip ?
  WORD  User selected string identifying this process
6Router-1(config-rtr)#redistribute rip cisco2
6Router-1(config-rtr)# exit
```

You can verify this redistribution by using the *show ipv6 protocols* command again:

```
6Router-1#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "rip cisco"
  Interfaces:
    Serial0
    Ethernet0
  Redistribution:
    Redistributing protocol static with metric 4
    Redistributing protocol rip cisco2
```

Redistribution of BGP routes is carried out in much the same way as redistribution of other RIP processes. The primary difference is that BGP uses an autonomous system number to specify the routing process in question. When specifying redistribution for BGP within a RIP routing domain, this autonomous system number must be given. As with the previous discussions about static and RIP routes, you can also utilize the *metric* and *route-map* commands to manipulate the redistributed BGP routes. The following listing shows an example of BGP redistribution. We have redistributed the BGP AS 2 and have assigned a metric value of 4 for the redistributed BGP routes:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 router rip cisco
6Router-1(config-rtr)#redistribute ?
  bgp         Border Gateway Protocol (BGP)
  connected   Connected Routes
  isis        ISO IS-IS
  rip         IPv6 Routing Information Protocol (RIPv6)
  static      Static Routes
```

```
6Router-1(config-rtr)#redistribute bgp ?
  <1-65535>  Autonomous system number

6Router-1(config-rtr)#redistribute bgp 2 ?
  metric     Metric for redistributed routes
  route-map  Route map reference
  <cr>

6Router-1(config-rtr)#redistribute bgp 2 metric 4
6Router-1(config-rtr)#
```

Redistribution of IS-IS routes is slightly more involved than it is for the previous routing protocols. (We discuss the details of IS-IS in the next section.) IS-IS operates slightly differently than other routing protocols do in that it specifies two types of routes, level 1 and level 2. Level 1 areas are areas that do not act as a backbone area for the network and level 2 areas are backbone areas. RIP has the capability to redistribute IS-IS routes by their level. RIP can redistribute either just level 1 routes, level 2 routes, or both level 1 and level 2 routes.

You perform redistribution for IS-IS in much the same way you configure the redistribution for the other routing protocols. First, you use the *redistribute isis* command to tell the RIP routing process to redistribute IS-IS. After entering this command, you have the capability to specify the route tag, or identifier, that IS-IS is using to identify the routing process. You can then specify if you want to redistribute level 1, level 2, or both level 1 and level 2 routes into the RIP routing process. You can then specify a route-map to use or manipulate the metrics. The following listing shows an example of IS-IS redistribution into IPv6 RIP. We are redistributing an IS-IS process called Cisco. We will redistribute both level 1 and level 2 routes as well as assign the redistributed routes a metric of 4:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 router rip cisco
6Router-1(config-rtr)#redistribute ?
  bgp        Border Gateway Protocol (BGP)
  connected  Connected Routes
  isis       ISO IS-IS
  rip        IPv6 Routing Information Protocol (RIPv6)
  static     Static Routes
```

```
6Router-1(config-rtr)#redistribute isis ?
  WORD       IPv6 process name
  level-1    IS-IS level-1 routes only
  level-1-2  IS-IS level-1 and level-2 routes
  level-2    IS-IS level-2 routes only
  metric     Metric for redistributed routes
  route-map  Route map reference
  <cr>

6Router-1(config-rtr)#redistribute isis cisco ?
  level-1    IS-IS level-1 routes only
  level-1-2  IS-IS level-1 and level-2 routes
  level-2    IS-IS level-2 routes only
  metric     Metric for redistributed routes
  route-map  Route map reference
  <cr>

6Router-1(config-rtr)#redistribute isis cisco level-1-2 ?
  metric     Metric for redistributed routes
  route-map  Route map reference
  <cr>

6Router-1(config-rtr)#redistribute isis cisco level-1-2 metric 4
```

Filtering IPv6 RIP Routing

Numerous instances exist in which you do not want all your routing table information to be transmitted from one router to all its neighbors. Conversely, there are also occasions when you do not wish a router to receive and update its routing table entries with all the information received from its neighbors. To address this concern, IPv6 RIP has the capability to utilize distribute-lists along with prefix-lists to filter incoming and outgoing routing table entries. This is a two-step process. The first step in this process is the creation of the prefix list. You perform this configuration at the global configuration level by using the *ipv6 prefix-list* command. The prefix-list is very similar to an access list in that both use a permit/deny system to allow or disallow traffic. Configuration of the IPv6 prefix list follows with specifying a name for the prefix-list as well as specifying the permit and/or deny statement(s). Remember that, as with standard access-lists, an implicit deny-all is at the end of every prefix list. The following listing shows an example of a prefix-list:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 prefix-list?
prefix-list

6Router-1(config)#ipv6 prefix-list ?
  WORD             Name of a prefix list
  sequence-number  Include/exclude sequence numbers in NVGEN

6Router-1(config)#ipv6 prefix-list cisco ?
  deny         Specify packets to reject
  description  Prefix-list specific description
  permit       Specify packets to forward
  seq          sequence number of an entry

6Router-1(config)#ipv6 prefix-list cisco permit ?
  X:X:X:X::X/<0-128>  IPv6 prefix <network>/<length>

6Router-1(config)#ipv6 prefix-list cisco permit 2000:1:1::/64 ?
  ge  Minimum prefix length to be matched
  le  Maximum prefix length to be matched
  <cr>

6Router-1(config)#ipv6 prefix-list cisco permit 2000:1:1::/64
6Router-1(config)#
```

In the preceding example, we created a prefix-list to enable 2000:1:1::/64. We also had some other configuration options as well. The *ge* option enables you to specify a minimum length that an address must match to qualify for the prefix-list and the *le* option enables you to set a maximum length of matching. These two options are useful if you want a single entry to be used to represent more than one address. As it stands now, we are specifying that an exact match is necessary to qualify for the prefix list.

You configure the second part of distribute-lists at the RIP configuration level by using the *distribute-list prefix-list* command. Follow this with the name of the list and specify whether this will be an in-bound or out-bound distribution list. The final part of the configuration is specifying an interface on which to apply the distribute-list. Although this is an optional step, if it is not performed, the distribute-list will be applied to all IPv6 interfaces on the router. The following listing shows a configuration example of this. In this example, we have configured the router to apply the Cisco prefix list as an outbound distribute list on the Ethernet 0 interface:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#ipv6 router rip cisco

6Router-1(config-rtr)#distribute-list ?
  prefix-list  Filter connections based on an IPv6 prefix-list

6Router-1(config-rtr)#distribute-list prefix-list ?
  WORD  Prefix-list name

6Router-1(config-rtr)#distribute-list prefix-list cisco ?
  in
  out

6Router-1(config-rtr)#distribute-list prefix-list cisco out ?
  Async               Async interface
  BVI                 Bridge-Group Virtual Interface
  CTunnel             CTunnel interface
  Dialer              Dialer interface
  Ethernet            IEEE 802.3
  Lex                 Lex interface
  Loopback            Loopback interface
  MFR                 Multilink Frame Relay bundle interface
  Multilink           Multilink-group interface
  Null                Null interface
  Serial              Serial
  Tunnel              Tunnel interface
  Vif                 PGM Multicast Host interface
  Virtual-Template   Virtual Template interface
  Virtual-TokenRing  Virtual TokenRing
  <cr>

6Router-1(config-rtr)#distribute-list prefix-list cisco out Ethernet 0
6Router-1(config-rtr)#
```

Verifying IPv6 RIP Operation

You have a few commands to use to verify that IPv6 RIP is working correctly. The first of these is *show ipv6 rip*. This command gives you all the specific information relating to the operation of IPv6 RIP, including the following:

- Timer Information
- Port Information
- Update Frequency
- Update Types
- Default Route Information

This command is a very useful tool for verifying that RIP is functioning correctly. The following listing shows an example of a command output from the *show ipv6 rip* command:

```
6Router-1#show ipv6 rip
RIP process "cisco", port 521, multicast-group FF02::9, pid 71
     Administrative distance is 120.  Routing table is 0
     Updates every 30 seconds, expire after 180
     Holddown lasts 180 seconds, garbage collect after 120
     Split horizon is on; poison reverse is off
     Default routes are generated
     Periodic updates 24667, trigger updates 2
6Router-1#
```

show ipv6 protocols is useful for showing all the IPv6 protocols, including RIP, that are running on a router. Information included from this output includes the following:

- Routing protocols in use
- Interfaces used by routing protocols

- Redistribution information

The following listing show an example of this command output:

```
6Router-1#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "rip cisco"
  Interfaces:
    Serial0
    Ethernet0
  Redistribution:
    Redistributing protocol static with metric 4
    Redistributing protocol rip cisco
```

Although we have already used this command throughout our discussion of RIP numerous times, the *show ipv6 route* command is also important in verifying RIP operation. This command output shows all the IPv6 routes that have been learned by the router thus far, as well as the protocol used to obtain the routes in question. The following listing shows an example of this command output:

```
6Router-1#show ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
Timers: Uptime/Expires

L   2000:1:1::1/128 [0/0]
     via ::, Serial0, 5d17h/never
C   2000:1:1::/64 [0/0]
     via ::, Serial0, 5d17h/never
L   2000:1:2::1/128 [0/0]
     via ::, Ethernet0, 1w2d/never
C   2000:1:2::/64 [0/0]
     via ::, Ethernet0, 1w2d/never
R   2000:1:3::/64 [120/2]
     via FE80::2E0:B0FF:FE55:B035, Serial0, 02:45:40/00:02:32
L   FE80::/10 [0/0]
     via ::, Null0, 1w2d/never
```

```
L    FF00::/8 [0/0]
      via ::, Null0, 1w2d/never
6Router-1#
```

Configuring & Implementing…

A Whole New Protocol

With a whole new protocol in RIPng, wouldn't it make much more sense to transition completely over to this new standard when implementing IPv6 in a network? The answer to this is: *It depends*. Although RIPng does show a lot of promise as a protocol, don't forget that we are discussing RIP, the often lamented and downplayed protocol of the networking world. RIP is one of the oldest and most stable routing protocols in existence. However, a lot of the reason that we now have multiple protocols is because of the limitations of RIP and its distance-vector routing protocol brothers. Two other factors should be considered in this discussion as well: your existing network and your plans for growth. It is hard to imagine that any organization considering IPv6 does not have an existing network—probably a large one if they are looking to IPv6 as an alternative.

The assumption that should be made is that if you would put RIP on this existing IPv4 network, then you would probably be safe in putting IPv6 on it. If not, then you should seriously consider putting RIP for IPv6 on this network. If there are plans for network growth, will this protocol scale to meet them? Will it need legacy support for IPv4? These, among hundreds of other questions, should be considered before choosing any protocol for your network. It's important to note that two of the most widely deployed protocols in Cisco networks, OSPF and EIGRP, are also due to be supported on the existing standard, allowing for support for both IPv4 and IPv6.

Exploring IS-IS for IPv6

IS-IS is one of the first *link-state routing protocols*. In the previous sections our discussion has focused on RIP, which is a distance-vector routing protocol. As you know from our discussion, distance-vector routing protocols use the distance, or number of hops, to determine the metric for reaching a given destination. Link-state routing protocols, such as IS-IS, take a much more overall picture of the network and base their metrics on a more accurate set of metrics and calculations. We'll discuss some of the overall aspects that all link-state routing protocols share here.

Link-state routing protocols offer much more scalability and many more features than their distance-vector predecessors. Link-state protocols also offer a system of hierarchy within a routed domain that affords the user much greater control and scalability than would be possible using distance-vector routing protocols. These protocols also offer support for VLSM, enabling a network to support multiple subnet masks, making much better utilization of available IP address space. It is for these reasons, along with several others, that link-state protocols are the protocols of choice within large enterprise organizations as well as within the networks of major service providers.

Link-state protocols derive their name from the manner in which they view the network. They take the state, or conditions, of the path into account when making a routing determination. Routers examine not only the state of the specific link, but also the link and its relation to the neighboring router. Information such as bandwidth of the links and delay are taken into consideration as is information pertaining to interface types, IP addresses, and subnet mask, which are all calculated in the link-state database to make the best path determination at any given time. This provides a much better system of path determination than does the distance-vector routing system that generally takes into account hop count as its primary metric.

Link-state protocols are typically more difficult to configure than distance-vector protocols. Also, the CPU and memory utilization on routers running link-state routing protocols is typically greater than the load when running distance-vector routing protocols, so you must ensure that you router has the capability to support link-state protocols. It would not be wise to completely write off distance-vector protocols either. Even though networks continue to grow at accelerated rates and link-state protocols can scale to meet these needs, distance-vector protocols, due to their simplicity and ease of configuration will most likely be around for some time to come.

Link-State Advertisements

Link-state advertisements (LSAs) are the primary means of communication for link-state protocols. When a router comes online, it floods the area around it with LSAs to announce its networks. When other routers flood the network with

LSAs, routers discover the network topology, build neighbor relationships, and form adjacencies.

This initial flooding of packets is very important, because it ensures that all routers in a link-state environment know about all the other routes so that an accurate picture of the network is available to every router. Routers also use LSAs to inform other routers of network topology changes. Every time a change occurs, a router or routers send LSAs out with specific information. If the receiving router does not know the updated information already (from another source, perhaps), it requests a link-state update (LSU) containing the new information. LSAs are coded with sequence numbers; if a router receives an LSA with a number older than the last one it received, it knows to discard it. LSAs also have an aging mechanism (the default is 30 minutes), which will make an LSA invalid and require a new one to be sent.

Neighbors

Neighbor relationships are formed in link-state protocols to facilitate network discovery. When a new router comes online it floods the network with LSAs to obtain neighbor reachability information with routers adjacent to it. Soon every router knows about every other router within a single area. Because of their membership in a given area, routers will also learn about the given areas within a single network, but they may not know about every route within those areas. This helps to limit the size of routing tables without limiting network reachability.

With that information in hand, routers can make the optimal path selection and determination for each network. Neighbor relationships are handled differently by each link-state protocol; for instance, in an OSPF network, each router builds a table of its adjacent neighbors, allowing for quick and efficient access to neighbor information.

The establishment and communication with neighbors help to form relationships known as adjacencies. Adjacencies are relationships formed between routers for the purpose of exchanging routing information. Adjacencies will be formed with different types of routers in different situations. In the case of a broadcast or non-broadcast multi-access media used for interconnecting routers, a designated router and backup designated router will be elected. The routers on this segment will establish adjacencies with these routers and only accept routing updates from them. In the case of point-to-point links, routers will establish adjacencies with each router to which it is directly connected in order to communicate and exchange routing information.

Link-State Database

The *link-state database* is a collection of all the information a router has obtained through the receiving of LSAs. Every time a new LSA is received, the information contained within is compared to the router's link-state database and if the sequence number is newer the information is added to the database. This information could include network reachability information, new routers, and so on. A router maintains a link-state database for every area of which it is a member.

Areas

Link-state routing protocols use *areas* to establish hierarchy within a routed network. Areas are logical, not physical, groupings of routers that are used to control route propagation and control traffic. Every router within an area has the same link-state database. Routers belonging to more than one area will maintain a link-state database for each area that they belong to.

IS-IS Specifics

Now that we have discussed some of the characteristics that all link-state routing protocols share, we'll discuss some of the specifics of the protocol. IS-IS is quite similar in some of its behaviors to OSPF. This is no accident, as the developers borrowed and improved on IS-IS when developing OSPF. Although IS-IS was developed by the ISO to route Connectionless Network Service (CLNS), it has been modified to route IP. Its modular architecture means that it can be further adapted to route other protocols such as IPX, should anyone need or choose to develop that particular aspect of it. This fact is also why it is one of the first protocols to support IPv6.

As defined in the original ISO standard (ISO 10589 and RFC 1142), IS-IS was developed to route only CLNS. There was no other support for any other network protocols such as IP. One of the driving factors behind CLNS was the expected depletion of IP. IS-IS was to become the routing protocol of choice after CLNS gained widespread acceptance.

CLNS and IS-IS were created and groomed to take up the mantle of network addressing upon the retirement or demise of IP and its merry band of routing protocols. As anyone who has ever used "the Internet" knows, this did not happen, but it wasn't from a lack of effort. Organizations implemented a tighter grip on IP address allocation, and various ways to extend IP's life were created and deployed—NAT and creative uses of private address space among them.

IS-IS is a link-state routing protocol with a similar concept of areas as OSPF. Its area 0 (the backbone areas that all areas must transit to reach each other) is called the level 2 (L2) area. All other areas are classified as level 1 (L1) areas. IS-IS also has its own interpretation of the NSAP address as discussed in the NSAP addressing section; please review before proceeding.

IS-IS routes by area within the autonomous system (L2 routing) and by the system ID within an area (L1 routing). That is, IS-IS uses the area address to determine how to reach the area, and the system ID to reach a particular device after it gets to the destination area. Therefore, it can be said that IS-IS routes on two levels: area and station.

CLNS Addressing

As we mentioned earlier, IS-IS was designed with the routing of CLNS in mind. These addresses, commonly referred to as *NSAP addresses* as well, are 80 bytes in length and are read from right to left as opposed to left to right like common English, or the way we read IP addresses. This is a very important distinction to make. Also, CLNS addresses are not assigned to an interface; rather, they are assigned to the router or network entity as a whole. This is why CLNS addresses are often commonly referred to as NET addresses. The format for this is broken up into four fields. The first of these fields is the authority and format identifier, and is one octet in length. The second is the area ID; this section is three octets in length and specifies the area that the particular network entity belongs to. The third section is the system ID. This six-octet portion of the address specifies the exact system identifier and must be unique within the network. The last portion of the address is the selector portion, which is also one octet. This is used to specify the actual port or interface of the network entity. Routing for CLNS addresses is enabled by use of the *clns routing* command in global configuration mode, followed by the *router isis* command and specification of a *net* address.

Integrated IS-IS

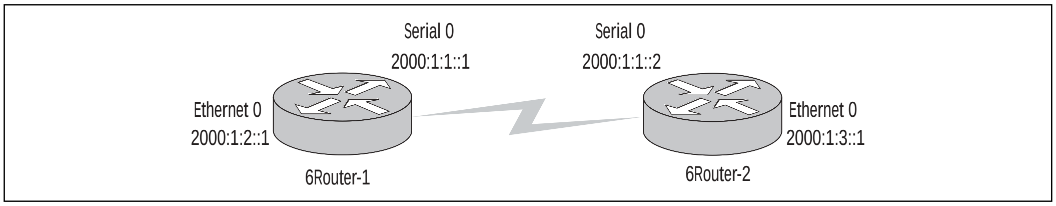
Support for IP is not inherent in IS-IS, as we discussed before—it was designed for CLNS-only routing. You need to make only a few configuration changes to your CLNS routing configuration to be able to use IS-IS. The first of these is the *ip router isis* command. If IP addresses are not configured, you must configure them on the interface at this time. You must then also assign the interfaces to an area by using the *area* command. This enables the IS-IS process to advertise the IP address to other IS-IS routers. This is a brief introduction to a fairly complex topic, but we will cover more IS-IS configuration in our discussion of IPv6 IS-IS.

Configuring IS-IS for IPv6

Unlike with RIP, a specific version of IS-IS was not written to support IPv6; rather, extensions were written into the protocol to provide this support. This is because of the modular architecture of IS-IS. So the configuration for IS-IS is somewhat different from the configuration for RIP.

You perform initial configuration for IPv6 IS-IS the same as you do the configuration for standard IPv4 IS-IS: through the use of the *router isis* command. You should then directly follow this command with the *net* command, specifying the network entity address for the router. IPv6 support comes into play at the interface level when assigning an IPv6 address to the router by using the *ipv6 address* command followed by the *ipv6 router isis <area>* command. Figure 6.4 shows an example of a basic IS-IS network.

Figure 6.4 Basic IS-IS Network



In the following configuration example we are using the two routers shown in Figure 6.4, 6Router-1 and 6Router-2 respectively.

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)# clns routing
6Router-1(config)# router isis cisco
6Router-1(config-router)# net 49.aaaa.bbbb.cccc.dddd.0000.2222.2222.2222.00
6Router-1(config-router)# exit
6Router-1(config)# interface ethernet 0
6Router-1(config-if)# ipv6 address 2000:1:2::1/64
6Router-1(config-if)# ipv6 router isis cisco
6Router-1(config-if)# interface serial 0
6Router-1(config-if)# ipv6 address 2000:1:1::1/64
```

```
6Router-1(config-if)# ipv6 router isis cisco
6Router-1(config-if)# exit

6Router-2#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-2(config)# clns routing
6Router-2(config)# router isis cisco
6Router-2(config-router)# net 49.aaaa.bbbb.cccc.dddd.0000.2222.2222.2223.01
6Router-2(config-router)# exit
6Router-2(config)# interface ethernet 0
6Router-2(config-if)# ipv6 address 2000:1:3::1/64
6Router-2(config-if)# ipv6 router isis cisco
6Router-2(config-if)# interface serial 0
6Router-2(config-if)# ipv6 address 2000:1:1::2/64
6Router-2(config-if)# ipv6 router isis cisco
6Router-2(config-if)# exit
```

In the preceding examples we enabled IS-IS routing by issuing the *clns routing* command on both routers in global configuration mode, after which we enable integrated IS-IS by issuing the *router isis cisco* command. We then assigned the NSAP address of 49.aaaa.bbbb.cccc.dddd.0000.2222.2222.2223.00 and 49.aaaa.bbbb.cccc.dddd.0000.2222.2222.2223.01 respectively by using the *net* command. After which we assigned Ipv6 addresses to both the Ethernet and serial interfaces of each router by using the *ipv6 address* command on both routers in interface configuration mode. We then assigned the interfaces to the IS-IS routing process by issuing the i*pv6 router isis cisco* command on each interface.

We can verify proper IS-IS operation by performing *show Iipv6 route* (note the I1 address, a level-1 IS-IS route running between our two test routers):

```
6Router-1#show ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
Timers: Uptime/Expires

L   2000:1:1::1/128 [0/0]
     via ::, Serial0, 5d20h/never
```

```
C    2000:1:1::/64 [0/0]
      via ::, Serial0, 5d20h/never
L    2000:1:2::1/128 [0/0]
      via ::, Ethernet0, 1w2d/never
C    2000:1:2::/64 [0/0]
      via ::, Ethernet0, 1w2d/never
I1   2000:1:3::/64 [120/2]
      via FE80::2E0:B0FF:FE55:B035, Serial0, 05:57:49/00:02:43
L    FE80::/10 [0/0]
      via ::, Null0, 1w2d/never
L    FF00::/8 [0/0]
      via ::, Null0, 1w2d/never
```

IS-IS Default Routes

Like RIP, IS-IS uses default routes to reach destination networks for which routers do not have routes in their routing tables. However, entering default route information for IS-IS IPv6 is somewhat different from that for RIP. Configuration for a default route for IPv6 in IS-IS is a two-step process. The first step is to enter address family configuration mode in IS-IS configuration. This is accomplished by using the *address-family ipv6* command in IS-IS configuration mode. Use the address family identifier to specify which group of addresses, either IPv4 or IPv6, is to be configured. Follow this command with the *default-information originate* command. This command also has optional parameters of specifying a *route-map* to use when advertising the default route. We exit IPv6 IS-IS configuration by entering the *exit-address-family* command. If a route-map is not used, the default route will be advertised to the rest of the routing domain as a Layer 2 route. The following listing shows an example of this configuration:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)# router isis cisco
6Router-1(config-router)# address-family ipv6
6Router-1(config-router-af)# default-information originate
6Router-1(config-router-af)# exit-address-family
```

Maximum Paths for IS-IS

IS-IS automatically supports load balancing for up to two paths. However, this value can be increased by up to four by manipulating the *maximum paths* value. This can be a very useful and effective configuration, allowing for faster transmission of data across multiple paths to a given destination. You manipulate the *maximum paths* command at the IS-IS configuration level. Once again, this is a two-step process; the first step is to specify the appropriate address family that you wish to change by using the *address-family ipv6* command. Follow this with the *maximum-paths <number-paths>* command. The *number* parameter specifies the number of maximum paths that you wish the router to support. The maximum for this number is four. The following listing shows an example of this configuration:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)# router isis cisco
6Router-1(config-router)# address-family ipv6
6Router-1(config-router-af)# maximum-paths 4
6Router-1(config-router-af)# exit-address-family
```

IS-IS Route Redistribution

Route redistribution in IS-IS is similar, but not identical, to route redistribution in RIP networks, so we will not go into the detail on this topic that we did on RIP. We should discuss, however, some redistribution configuration issues specific to IS-IS. We will begin with a discussion of how to perform route redistribution in IS-IS networks.

You perform IS-IS redistribution at the IS-IS router configuration level. First, enter the IS-IS configuration mode with the *router isis <area-name>* command. Then, specify the address family you wish to configure redistribution on by using *address-family ipv6*. You can then configure redistribution by using the *redistribute* command. As with RIP, IS-IS can redistribute RIP, BGP, Static, and other IS-IS processes. Also, like RIP, IS-IS has the capability to specify metric values and which route-map to use. The following listing shows an example of a basic RIP into IS-IS redistribution. In this example, we are redistributing the IPv6 RIP routing process *cisco* into our IS-IS routing domain with a metric of 100:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
6Router-1(config)# router isis cisco
6Router-1(config-router)# address-family ipv6
6Router-1(config-router-af)# redistribute rip cisco metric 100
6Router-1(config-router-af)# exit-address-family
```

IS-IS also has the capability to redistribute routes between level-1 and level-2 routers, enabling the two levels of routers to fully share connectivity information while still keeping the network segmented. Inter-level route redistribution is similar to standard redistribution; however, you must use a few more configuration commands to make it happen. Inter-level redistribution begins like standard redistribution with specifying the *isis <area-id>* and entering into the *address-family ipv6* level for configuration of IPv6 parameters. Upon entering that level, give the *redistribute* command, except now, you use the command: *isis level-1 into level 2 or level-2 into level-1*. It is important to note that level-1 routes are automatically redistributed into the level-2 routing process. With that in mind, we will configure level-2 into level-1 redistribution in the following example:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)# router isis cisco
6Router-1(config-router)# address-family ipv6
6Router-1(config-router-af)# redistribute isis level-2 into level-1
6Router-1(config-router-af)# exit-address-famil
```

Describing Muliprotocol BGP Extensions for IPv6

Up to now, our discussion has focused solely on routing protocols that operate within a single organization, or, as it would be called in the world of networking, single autonomous systems (AS). Routing protocols that operate within a single AS are referred to as interior gateway protocols (IGPs). But what about traffic that leaves the AS—what happens to it? The answer to that is *exterior gateway protocols* (EGPs). Just as IGPs operate solely within a single AS, EGPs operate between ASs, providing connectivity *between* organizations rather than *within* them.

The most well-known and by far widely deployed EGP is BGPv4. For purposes of this discussion we will just refer to it as BGP. Entire books have been written on this protocol for two basic reasons: First, it is a very complex and intricate protocol, and very few people are actually proficient in their understanding of this protocol, and second, because it is one of the most important protocols in use today, connecting the ASs of the world. With this in mind, it would be impossible to have a detailed discussion of this protocol within this chapter, so we are briefly going to discuss some of the basic concepts behind this protocol and then turn our attention to the specific IPv6 configuration tasks and considerations.

Autonomous Systems

As mentioned, BGP operates between autonomous systems (AS). What exactly is an AS? An AS is defined as a network, or networks, that are under a single administrative authority. Company networks generally fall into the category of being a single AS; however, very large multi-national firms often segment their networks into multiple autonomous systems for purposes of administration. Autonomous systems are defined by a numerical value; these numbers are generally assigned by your organization's ISP or from an organization known as ARIN. These numerical values number between 1 and 65635, a range between 64512 and 65535 have been set aside for private addressing, much like the private IP address space that many companies utilize as specified by RFC 1918. BGP uses these AS numbers, rather than just IP addresses, to route information between autonomous systems. You specify the autonomous system number for the BGP session running on a router by using the router *bgp <as number>* command.

Neighbors

Although BGP is considered to be a dynamic routing protocol, it relies on static entries, known as neighbor statements, to communicate between autonomous systems. These neighbor statements specify the neighbor's autonomous system, as well as their IP address. These neighbor statements are at the heart of the BGP protocol. Another important note to make is that an existing IP connection must exist between the two neighbors, either by static route, direct connection, or through the use of an IGP. BGP cannot dynamically find its own way to given destination networks. Neighbor relationships are configured through the use of the *neighbor <ip-address> remote-as <as number>* command.

BGP Metrics

All dynamic routing protocols incorporate the use of metrics to make calculations regarding the best path to take to make a routing decision, and BGP is no exception to this rule. Unlike other routing protocols that base routing decisions on one or a few metrics, BGP bases its routing metrics on 10 criteria. The reason for this is the advanced policy-based features that BGP has incorporated, which enable the protocol to make data-forwarding decisions based on several different factors. The metrics that BGP incorporates are as follows:

- Origin
- AS Path
- Next Hop
- Multi-Exit Discriminator (MED)
- Local Pref
- Atomic Aggregate
- Aggregator
- Community
- Originator ID
- Cluster List

Configuring BGP Extensions for IPv6

As with IS-IS, a new version of BGP was not developed specifically for the purpose of routing IPv6; rather, new multi-protocol extensions were added to the protocol to support IPv6. Enhancements made to the protocol include the support for IPv6, support for IPv6 next hop addressing information, and support for link-local IPv6 addressing.

As we discussed before, a new version of BGP was not created for IPv6, so you enter into IPv6 BGP configuration the same way that you did IPv4 configuration: through the use of the *router bgp <as number>* command. Upon entering the BGP configuration mode, you then need to enter the *no bgp default ipv4-unicast* command. This command turns off IPv4 BGP unicasts. This is a necessary step because, by default, IPv4 address information is advertised for each BGP neighbor configured on the router. IPv6 cannot work if this is taking place.

Configuring an IPv6 Neighbor Relationship

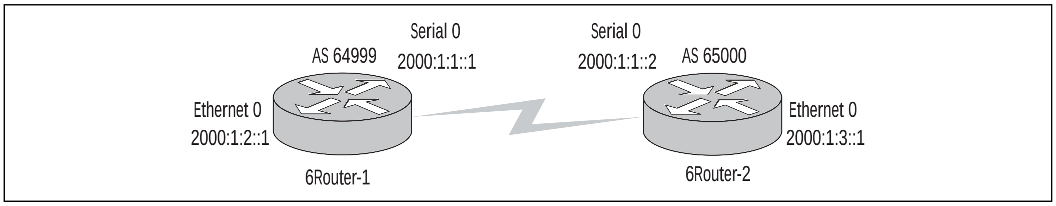
It is slightly more involved to set up IPv6 neighbor relationships than IPv4 neighbors. To set up an IPv6 neighbor in BGP, you must first specify the IPv6 address and remote-as number of the neighbor router by using the neighbor

<ipv6 address> remote-as <as number> command. Follow this command set by entering into IPv6 address family configuration and activation of the neighbor. You do this with the *ipv6 address-family* command, followed by the *neighbor <ipv6 address> activate* command. The networks that you wish that the BGP process advertise is also specified here through use of the *network* command.

Figure 6.5 shows a basic BGP network. The following listing is an example of this configuration for the network. In this example we have two routers, 6Router-1 and 6Router-2. On 6Router-1 we have created a BGP process with the autonomous system of 64999 and we have created a neighbor relationship with the router at the 2000:1:1::2 address with an AS number of 65000. We have also instructed the router to advertise the 2000:1:1::0/64 and 2000:1:2::0/64 networks.

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#router bgp 64999 (Enable BGP routing with AS 64999)
6Router-1(config-router)#no bgp default ipv4-unicast (Disable 1Pv4 unicasts)
6Router-1(config-router)#neighbor 2000:1:1::2 remote-as 65000 (Neighbor)
6Router-1(config-router)#address-family ipv6 unicast (Allow IPv6 unicasts)
6Router-1(config-router-af)#neighbor 2000:1:1::2 activate (Activate neighor
6Router-1(config-router-af)#network ?
6Router-1(config-router-af)#network 2000:1:1::0/64 (Advertise network)
6Router-1(config-router-af)#network 2000:1:2::0/64 (Advertise network)
6Router-1(config-router-af)#exit
6Router-1(config-router)#exit
6Router-1(config)#
```

Figure 6.5 Basic BGP Network



On 6Router-2 we created a BGP process with an autonomous system of 65000 and we created a neighbor relationship with 2000:1:1::1 with an autonomous system ID of 64999. We also instructed the router to advertise the 2000:1:1::0/64 and 2000:1:3::0/64 networks.

```
6Router-2#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-2(config)#router bgp 65000 (Enable BGP routing with AS 65000)
6Router-2(config-router)#no bgp defaul ipv4-unicast (Disable 1Pv4 unicasts)
6Router-2(config-router)#neighbor 2000:1:1::1 remote-as 64999 (Neighbor)
6Router-2(config-router)#address-family ipv6 unicast (Allow Ipv6 unicasts)
6Router-2(config-router-af)#neighbor 2000:1:1::1 activate (Activate neighor)
6Router-2(config-router-af)#network 2000:1:1::0/64 (Advertise Network)
6Router-2(config-router-af)#network 2000:1:3::0/64  (Advertise Network)
6Router-2(config-router-af)#exit
6Router-2(config-router)#exit
6Router-2(config)#exit
```

We can verify the success of our BGP configuration by using the *show BGP neighbors* command on 6Router-1. The following listing shows that we have indeed established a neighbor relationship with 2000:1:1::2 and shows that IPv6 unicast traffic is going between the two routers. The more important information to be gained from this output would be the first line of the config, verifying that we do in fact have a neighbor relationship between the two routers as well as the information contained in the following bolded lines concerning version, router ID, BGP state, and up time. This is followed by the information concerning the IPv6-specific information about the BGP table and neighbor version as well as connected/dropped connection information:

```
6Router-1#show bgp neighbors
BGP neighbor is 2000:1:1::2,  remote AS 65000, external link
  BGP version 4, remote router ID 172.16.8.33
  BGP state = Established, up for 00:13:43
  Last read 00:00:43, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv6 Unicast: advertised and received
  Received 18 messages, 0 notifications, 0 in queue
  Sent 18 messages, 0 notifications, 0 in queue
  Default minimum time between advertisement runs is 30 seconds

 For address family: IPv6 Unicast
  BGP table version 3, neighbor version 3
  Index 1, Offset 0, Mask 0x2
  Route refresh request: received 0, sent 0
  1 accepted prefixes consume 68 bytes
  Prefix advertised 2, suppressed 0, withdrawn 0

  Connections established 1; dropped 0
  Last reset never
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 2000:1:1::1, Local port: 179
Foreign host: 2000:1:1::2, Foreign port: 11000

Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)

Event Timers (current time is 0x33538748):
Timer           Starts    Wakeups             Next
Retrans             17          0               0x0
TimeWait             0          0               0x0
AckHold             17         10               0x0
SendWnd              0          0               0x0
KeepAlive            0          0               0x0
GiveUp               0          0               0x0
PmtuAger             0          0               0x0
DeadWait             0          0               0x0

iss: 1756584196  snduna: 1756584638  sndnxt: 1756584638     sndwnd:  15943
irs: 2281490682  rcvnxt: 2281491115  rcvwnd:      15952  delrcvwnd:    432

SRTT: 269 ms, RTTO: 516 ms, RTV: 247 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: passive open, nagle, gen tcbs

Datagrams (max data segment is 1440 bytes):
Rcvd: 25 (out of order: 0), with data: 17, total data bytes: 432
Sent: 27 (retransmit: 0, fastretransmit: 0), with data: 27, total data
bytes: 19
6Router-1#
```

You can also verify that IPv6 BGP is functioning by using the *show ipv6 route* command. The following *show ipv6 route* command output shows that the 6Router-1 has learned of the 2000:1:3::/64 network by way of the BGP routing process:

```
6Router-1#show ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
Timers: Uptime/Expires

L   2000:1:1::1/128 [0/0]
     via ::, Serial0, 5d23h/never
C   2000:1:1::/64 [0/0]
     via ::, Serial0, 5d23h/never
L   2000:1:2::1/128 [0/0]
     via ::, Ethernet0, 1w2d/never
C   2000:1:2::/64 [0/0]
     via ::, Ethernet0, 1w2d/never
B   2000:1:3::/64 [20/1]
     via 2000:1:1::2, Serial0, 00:00:47/never
L   FE80::/10 [0/0]
     via ::, Null0, 1w2d/never
L   FF00::/8 [0/0]
     via ::, Null0, 1w2d/never
6Router-1#
```

Configuring a BGP Router ID

BGP uses router IDs to identify BGP speaking routers. In IPv4, this address was automatically set to the loopback interface of the router. If this address did not exist, then the address was set as the highest IPv4 address on the router. BGP does not, however, do this for a router running only IPv6 BGP; therefore, you must manually configure this value. Although you are not running BGP for IPv4 traffic, you will need to configure the Router ID as an IPv4 address. To do this, you use the *bgp router-id <ip address>* command in BGP configuration mode. The following listing shows an example of this configuration:

```
6Router-1(config)#router bgp 64999
6Router-1(config-router)#bgp router-id 172.16.0.1
6Router-1(config-router)#
```

Configuring BGP Peer Groups

Peer groups are a management mechanism that Cisco routers can use to make policy changes to multiple routers. This functionality is very advantageous for administrators who have to replicate policy changes to multiple BGP-speaking routers. You define peer groups on routers with a name and a set of routing policies. You configure a peer group for IPv6 BGP by using the *<name> peer-group* command. Then, create a neighbor relationship, as we discussed earlier, and then activate the peer group by using the *neighbor <peer group name>* activate command in address family configuration mode. Finally, add the neighbor's IPv6 address to the peer group by using the neighbor *<ipv6 address> peer group <peer group name>* command. The following listing shows an example of this configuration:

```
6Router-1(config)#router bgp 64999
6Router-1(config-router)#neighbor cisco peer-group (Establish peer group)
6Router-1(config-router)#neighbor 2000:1:1::2 remote-as 65000 (neighbor est)
6Router-1(config-router)#address-family ipv6 unicast (Enable IPV6 unicast)
6Router-1(config-router-af)#neighbor cisco activate (activate neighbor)
6Router-1(config-router-af)#neighbor 2000:1:1::2 peer-group cisco (add to
    peer group)
6Router-1(config-router-af)#exit
6Router-1(config-router)#
```

Configuring Link-Local Addressing

You use link-local addresses for routers on a single link. They are an inherent part of the IPv6 protocol. Each router assigns each of its active IPv6 interfaces a link-local address. Therefore, a unique address, specific to the link running between the two routers, can be used. BGP has the capability to use this address for neighbor addressing to automatically assign a link-local address to peer routers. You configure link-local addressing in the BGP configuration mode by using the *update-source <interface type>* command in neighbor configuration mode. This command specifies that the router should use the address configured on the interface specified for the neighbor relationship. After this, you should enter into address-family configuration mode and activate the neighbor relationship. At this point you also have the option of also configuring a route map. The following example shows this configuration:

```
6Router-1(config)#router bgp 64999
6Router-1(config-router)#neighbor 2000:1:1::2 update-source ? (Specify update
    source)
  Async                Async interface
  BVI                  Bridge-Group Virtual Interface
  CTunnel              CTunnel interface
  Dialer               Dialer interface
  Ethernet             IEEE 802.3
  Lex                  Lex interface
  Loopback             Loopback interface
  MFR                  Multilink Frame Relay bundle interface
  Multilink            Multilink-group interface
  Null                 Null interface
  Serial               Serial
  Tunnel               Tunnel interface
  Vif                  PGM Multicast Host interface
  Virtual-Template     Virtual Template interface
  Virtual-TokenRing    Virtual TokenRing

6Router-1(config-router)#neighbor 2000:1:1::2 update-source serial ?
  <0-1>  Serial interface number

6Router-1(config-router)#neighbor 2000:1:1::2 update-source serial 0?
.  :  <0-1>

6Router-1(config-router)#neighbor 2000:1:1::2 update-source serial 0
6Router-1(config-router)#neighbor 2000:1:1::2 update-source serial 0
6Router-1(config-router)#address-family ipv6
6Router-1(config-router-af)#?
Router Address Family configuration commands:
  aggregate-address     Configure BGP aggregate entries
  bgp                   BGP specific commands
  default               Set a command to its defaults
  default-information   Control distribution of default information
  default-metric      Set metric of redistributed routes
  distribute-list     Filter networks in routing updates
  exit-address-family Exit from Address Family configuration mode
  help                Description of the interactive help system
  neighbor            Specify a neighbor router
  network             Specify a network to announce via BGP
  no                  Negate a command or set its defaults
  redistribute        Redistribute IPv6 prefixes from another routing
                      protocol
  synchronization     Perform IGP synchronization
  table-map           Map external entry attributes into routing table

6Router-1(config-router-af)#neighbor 2000:1:1::2 activate (Activate neighbor)
6Router-1(config-router-af)#exit-address-family
6Router-1(config-router)#
```

BGP Redistribution

Like both RIP and IS-IS, BGP also has the capability to redistribute routes from other routing protocols and from static routes on the local router. You configure redistribution from the *ipv6 address-family* configuration level in BGP configuration mode by using the *redistribute* command. After you enter the *redistribute* command, a default protocol is specified, but you have the capability to specify a different metric other than the default and to specify a route map statement.

WARNING

BGP uses a ten-step process in calculating its metric. Redistributed routes do not go through this process, unless of course you are redistributing another BGP process; therefore, you should manually configure the redistributed protocols to appropriately fit into the BGP routing process.

Because this configuration is almost identical to the redistribution of the other protocols, we will not go into the process in detail. The following listing shows an example of BGP RIP redistribution:

```
6Router-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
6Router-1(config)#router bgp 64999
6Router-1(config-router)#address-family ipv6 unicast
6Router-1(config-router-af)#redistribute ?
  bgp         Border Gateway Protocol (BGP)
  connected   Connected Routes
  isis        ISO IS-IS
  rip         IPv6 Routing Information Protocol (RIPv6)
  static      Static Route

6Router-1(config-router-af)#redistribute rip ?
  WORD  User selected string identifying this process

6Router-1(config-router-af)#redistribute rip cisco ?
  metric      Metric for redistributed routes
  route-map   Route map reference
  <cr>
6Router-1(config-router-af)#redistribute rip cisco metric ?
  <0-4294967295>  Default metric

6Router-1(config-router-af)#redistribute rip cisco metric 100
6Router-1(config-router-af)#exit
6Router-1(config-router-af)#exit-address-family
6Router-1(config-router)#
```

We can verify the successful redistribution of RIP into BGP by issuing the *show IPV6 protocols* command:

```
6Router-1#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "bgp 64999"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn RoutemapOut
    2000:1:1::2
```

```
IPv6 Routing Protocol is "rip cisco"
  Interfaces:
    None
  Redistribution:
    Redistributing protocol rip cisco
6Router-1#
```

Designing & Planning…

Why Redistribute?

Redistribution is a very powerful tool that affords a network full connectivity both within itself and to outside networks, without the cost of the high processor utilization of running multiple routing protocols on the same router, or choosing a single uniform routing protocol to go across an entire network, if even possible. Rather, redistribution enables multiple routing protocols and processes to work in harmony, sharing the information. This enables the network administrator to fully leverage the capabilities of the routing protocols in question, while oftentimes averting its pitfalls.

Let's examine the three routing protocols we discuss in the chapter to see how we could fully utilize their strengths in a network, starting with RIP. RIP is an excellent protocol for small, stable networks. With its hop count limitation of 15 routers and periodic broadcasting of its entire routing table, it has also proven to be a bane to large, complex networks, but remains an excellent protocol for small networks. IS-IS, meanwhile, was traditionally intended for large CLNS networks. As a distance-vector routing protocol, it makes its path and metric calculation based on several factors, generally making better routing decisions than RIP. It is also a far more scalable solution than RIP, utilized across the backbone networks of many major ISPs. However, IS-IS is also very processor intensive, not even supported on many low-end routers. The third protocol in question is BGP, which is an exterior routing protocol—it works between organizations to provide connectivity. But how does it communicate within a network? Although a version of BGP exists to serve this function (IBGP), it still requires either static routes or an IGP to provide connectivity.

Continued

So you have three protocols that have desirable characteristics for a particular network, but none is perfect for your network. Redistribution enables you to leverage these strengths in such a way that the weaknesses are completely forgotten. In your network alone with just these three protocols, you could leverage redistribution to optimize your network by using RIP at the edge network and redistributing it into an IS-IS session, which would serve as a backbone protocol for your network. This IS-IS process would then in turn interface with BGP to provide internal connectivity for BGP and receive external connectivity for the network.

Although this is a somewhat over-simplification, it demonstrates the effectiveness and scalability that redistribution can afford a network.

Verifying BGP Operation

Several commands are available to ensure that your IPv6 BGP configuration is running correctly. We review some of these commands, their applications, and the command syntax in the following sections.

Using the *show bgp* Command

You can verify your BGP operation by using the *show bgp ipv6* command. This command is very useful in diagnosing the overall health of the IPv6 BGP process, as well as verifying that the protocol is advertising the appropriate networks. This output, displayed in the following listing, shows the local router ID, networks being advertised and learned via BGP, BGP table version, as well as path and weights for the networks.

```
6Router-1#show bgp ipv6
BGP table version is 13, local router ID is 172.16.0.1 (Table version and
local router ID)
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure
Origin codes: i - IGP, e - EGP, ? – incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  2000:1:1::/64    2000:1:1::2                            0 65000 i
*>                   ::                                32768 i
*> 2000:1:2::/64    ::                                       32768 i
*> 2000:1:3::/64    2000:1:1::2                                  0 65000 i
(Advertised Networks)
6Router-1#
```

Using the *show bgp ipv6 summary* Command

The *show bgp ipv6 summary* command also gives an overall picture of the IPv6 BGP routing process; however, it does it in a different way than the *show bgp ipv6* command did. As the following example illustrates, this command shows the BGP router identifier, local AS number, and BGP table version. Instead of listing the individual network entries, as does the *show bgp ipv6* command, it shows the number of network entries with the available number of paths. Neighbor information is listed at the bottom of this command output. The command output also displays memory utilization as well as route-map and filter-list usage.

```
6Router-1#show bgp ipv6 summary
BGP router identifier 172.16.0.1, local AS number 64999 (Local BGP info)
BGP table version is 13, main routing table version 13 (Table version)
3 network entries and 4 paths using 659 bytes of memory (Network entries)
2 BGP path attribute entries using 120 bytes of memory (Path attribute info)
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP activity 10/23 prefixes, 14/10 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
2000:1:1::2     4 65000    1482    1489       13    0    0 00:05:04     2
6Router-1#
```

Other Routing Protocols and Future Developments

As you have probably already figured out, IPv6 support is still quite limited, with only three dynamic routing protocols being supported; however, this fact will change in the not-too-distant future. Cisco is currently developing its third phase of IPv6 support. The most exciting of these enhancements will be the support for two of the most widely deployed protocols in Cisco networks: OSPF and EIGRP. We'll briefly discuss these protocols in the following sections.

IPv6 OSPF

Open Shortest Path First has been called the premier routing protocol of its time. This distinction was made because of the feature-rich, scalable, and open-standards platform that this protocol offers. OSPF is a hierarchical, link-state protocol nature that can efficiently scale to meet the needs of any network from single sites to large multi-national corporations. Because of its open standard nature, OSPF can be used to interconnect multiple vendors' equipment, and has therefore become the protocol of choice for thousands of organizations.

OSPF IPv6 support is outlined in RFC 2740. Like IS-IS and BGP, enhancements to the existing protocol, rather than an entirely new protocol, were decided for the support for IPv6. Like with BGP and IS-IS, the basic function of the protocol; that is, designated router selection, path selection, and support for areas, remains unchanged. The support for IPv6 is made possible by changes to the addressing formats, to support the new IPv6 address formats, changes to basic LSAs, and removal of authentication functions from the OSPF packet. The ladder of these changes is due to the fact that IPv6 has an inherent security feature built in.

As we mentioned, OSPF support is part of the phase-three deployment of IPv6 services. This deployment is currently scheduled to take place in mid-2002.

IPv6 EIGRP

Enhanced Interior Gateway Routing Protocol is another protocol widely deployed throughout Cisco networks. EIGRP is known as a hybrid protocol, incorporating mechanisms of both distance-vector and link-state protocols. EIGRP offers a scalable and effective solution for any Cisco network, often a preferential protocol to OSPF; however, EIGRP is a Cisco proprietary protocol, and it cannot be used to interconnect with other vendor's equipment. As with OSPF, Cisco plans to maintain the current structure of EIGRP and will write extensions into the protocol to support IPv6. This support will also be part of the third-phase deployment and will come in mid-2002.

Summary

IPv6 routing protocols offer the dynamic routing protocol solution for IPv6. Although the support of IPv6 will certainly spread to all routing protocols in time, currently only three routing protocols support IPv6 addressing. These protocols are RIP, IS-IS, and BGP. Although the RIP implementation was an entirely new standard, both IS-IS and BGP were updated to support IPv6.

RIP is a traditional distance-vector protocol whose roots go as far back as 1982. RIP is also an open-standard routing protocol, and therefore can be used to interconnect different vendors' equipment. As a distance-vector routing protocol, RIP calculates its metric based on hop count and considers a network more than 15 hops away to be unreachable. The IPv6 implementation of RIP shares all the traditional mechanisms of RIP, because it performs metric calculation in the same fashion. IPv6 RIP also supports default routes, route redistribution, and route filtering

IS-IS is a traditional link-state routing protocol. Originally written to support CLNS routing, extensions were written into it to support IPv4 addressing and now IPv6 addressing. As a link-state routing protocol, IS-IS takes a more overall look of the network when determining the path to use to reach a destination network and will generally make a better routing decision because it is not concerned with hop count as much as it is with items such as bandwidth. IS-IS treats IPv6 support as an extension of the existing protocol instead of writing an entirely new protocol as was done in the case of RIP. Like RIP, IS-IS supports default routes and redistribution; IS-IS has the capability to configure a maximum number of paths that can be used to reach a destination network.

BGP is an exterior gateway protocol. IS-IS and RIP operate within a single network or under a single entity of control, often called an autonomous system. BGP operates between these autonomous systems to provide connectivity. Support for IPv6 also came in the form of new extensions being written into the protocol stack. IPv6 features supported by BGP include IPv6 routing, redistribution, manual configuration of a Router ID, link-local addressing, and peer groups.

OSPF and EIGRP support for IPv6 will be part of Cisco's Phase III support for IPv6. Phase III is currently scheduled to be released in mid-2002. OSPF is an industry standard link-state routing protocol that has become the protocol of choice for several large organizations. OSPF support for IPv6 will also come in the form of extensions written for the existing protocol rather than a new protocol entirely. EIGRP is a Cisco-proprietary protocol. EIGRP is also considered to be a hybrid routing protocol, which comes from its use of both link-state and distance-vector routing protocol mechanisms. EIGRP support for IPv6 will also come in the form of extensions written into the existing protocol.

Solutions Fast Track

Explaining RIP for IPv6

☑ The Routing Information Protocol (RIP) is a distance-vector routing protocol. It calculates its routing metrics based solely on hop-count.

☑ A new version of RIP, often referred to as RIPng, was written to support IPv6. This new version of RIP shares the IPv4 limitations and mechanisms.

☑ IPv6 RIP supports default routes, route redistribution, and route filtering.

Configuring RIP for IPv6

☑ You enable RIP with the *Ipv6 router rip <word>* command. The *<word>* specifies the RIP routing process to be used.

☑ Include networks in the IPv6 RIP routing process at the interface level with the *ipv6 rip <word> enable* command. Once again, the *<word>* specifies the RIP process ID.

☑ Configure default routes at the interface layer with the *ipv6 rip <word>default-information originate* command.

Exploring IS-IS for IPv6

☑ Integrated System-to-Integrated System (IS-IS) is a link-state routing protocol that was originally designed for CLNS routing. Extensions were written in to support IP and now IPv6.

☑ Because it is a link-state routing protocol, its metric calculation is based on numerous factors, not just hop-count. It can therefore make better routing decisions than typical distance-vector routing protocols can.

- ☑ IS-IS supports all the same functions as the IPv6 version of RIP and also supports specification of more than one path to use for data transmission through use of the *maximum-paths* command.

Configuring IS-IS for IPv6

- ☑ You enable IS-IS on a router with the *CLNS routing* command. Enable integrated IS-IS by issuing the *router ISIS <word>* command. Again, *<word>* refers to the process ID.
- ☑ Advertise networks via IS-IS at the interface level with the *ipv6 router isis <word>* command.
- ☑ Redistribute other routing protocols into IS-IS with the *redistribution* command in the *address family* configuration mode.

Describing MultiProtocol BGP Extensions for IPv6

- ☑ Border Gateway Protocol (BGP) is an exterior routing protocol used to route between autonomous systems.
- ☑ Like IS-IS, extensions were added into the existing standard to support IPv6.
- ☑ IPv6 BGP supports IPv6 routing, redistribution, manual configuration of a Router ID, link-local addressing, and peer groups.

Configuring BGP Extensions for IPv6

- ☑ Start basic BGP configuration with the *router bgp <process id>* command.
- ☑ To support IPv6 unicast traffic, you must disable IPv4 unicasts with the *no bgp default ipv4-unicast* command.
- ☑ Form router peering relationships between BGP routers with the *neighbor* command. You must form a neighbor relationship between every router that you wish your BGP router to exchange information with.

Other Protocols and Future Developments

- ☑ Open Shortest Path First (OSPF) is a link-state routing protocol used in many large organizations. IPv6 support will come in the form of extensions written into the existing standard
- ☑ Enhanced Interior Gateway Routing Protocol (EIGRP) is a hybrid protocol incorporating mechanisms of both distance-vector and link-state routing protocols. Like OSPF, IPv6 support will come in the form of extensions written into the existing standard
- ☑ Both OSPF and EIGRP support for IPv6 will come with the Phase III deployment of IPv6 from Cisco, which is currently scheduled for mid-2002.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: What version of the IOS supports IPv6 addressing?

A: Support for IPv6 began with IOS 12.2(4) on limited platforms. You should consult the Cisco Web site to ensure that your specific router platform supports IPv6.

Q: Why is support for IPv6 restricted to just RIP, IS-IS, and BGP?

A: As with any new technology, some growing pains are involved in rolling out new technology. Although this is speculation, the protocols that they chose to write extensions were probably picked because they are open standards that can be used to interoperate with multiple vendors' equipment. Support for EIGRP and OSPF will come shortly.

Q: In previous versions of RIP you could only specify RIP to run as a single process on the router. With the naming of the RIP process IDs in IPv6, does this mean that multiple RIP processes can be run on the same router?

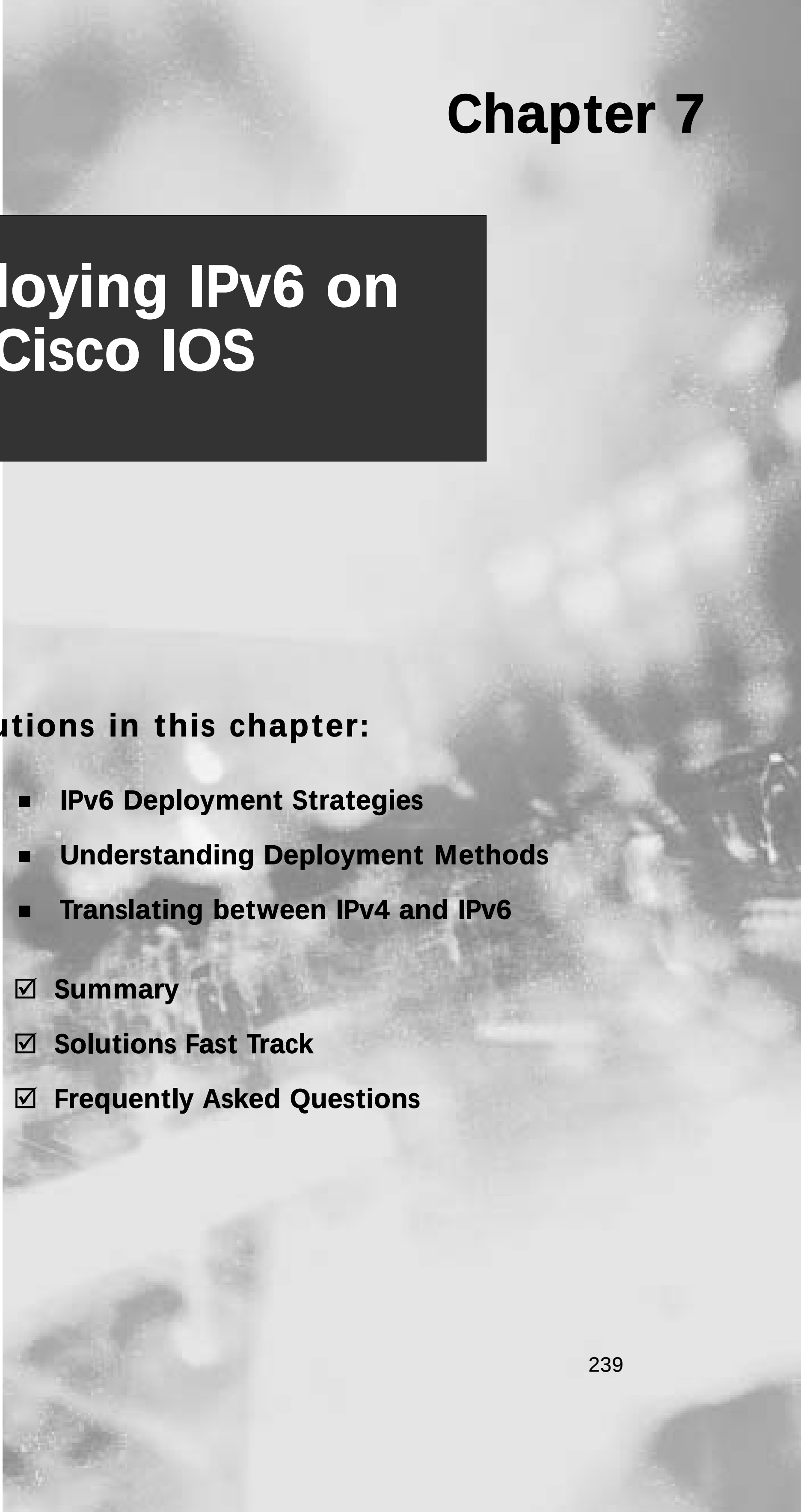
A: Yes, the new standard of RIP does support multiple RIP processes. However, you must keep in mind the processor and resource utilization that multiple RIP processors will incur and make resources available as necessary.

Q: In many examples I noted a double colon (::) when expressing an IPv6 address. What does this mean?

A: When dealing with an IPv6 address that has multiple octets that have only a zero value in them, you can compress them by using the double colon.

Q: Can an IPv6 BGP process also advertise IPv4 addresses?

A: Unfortunately, no. Because of the unicast address properties of IPv4 addressing, you must disable IPv4 unicast traffic for the entire BGP process in order for IPv6 addressing and neighbor relationships to be formed.



Chapter 7

Deploying IPv6 on the Cisco IOS

Solutions in this chapter:

- **IPv6 Deployment Strategies**
- **Understanding Deployment Methods**
- **Translating between IPv4 and IPv6**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

Introduction

With all the changes from IPv4 to IPv6, the task of deploying the new protocol can appear insurmountable. This chapter will explain methods of deployment of IPv6, the co-existence with IPv4, and the migration from IPv4. Strategies for deploying IPv6 have been discussed alongside the protocol development since its inception. This has been a carefully considered topic of the designers of IPv6, because a smooth transition from IPv4 to IPv6 is vital to the protocol's success. An entire IETF working group has been assigned the task of figuring out how to transition the Internet from IPv4 to IPv6. We will talk about some of the deployment scenarios that have been thought of so far.

Although there are various methods of deploying IPv6 on a network, a well-thought-out project plan and a solid execution are necessary for any project of this magnitude. Successful conversion and migration of each network attached to the Internet is critical to the overall transition to IPv6, so the future of the Internet is in the hands of the collective group of network engineers, administrators, and designers that work for not only tier-1 providers and ISPs, but also for individual organizations. This chapter will provide the necessary information to make an informed decision as to which path to take when deploying IPv6.

IPv6 Deployment Strategies

We will discuss many deployment strategies in this chapter for deploying IPv6 into an operational IPv4 network, or at the least running Ipv6 alongside IPv4. IPv6 and IPv4 can co-exist until the time it is decided that IPv4 is no longer needed in the network. There are pros and cons to each method—one method is not best in all cases. Each network is unique and the following sections will help determine which strategy is best for your given environment and requirements. At a high level, the basic strategies are to build tunnels or Virtual Private Networks (VPNs) for running IPv6 and IPv4 protocols concurrently at the routers and hosts. These strategies apply to both the service provider and the enterprise customer. For the service provider, these may be a result of any of the following situations:

- Internet services/applications such as integrated telephony or Internet-enabled PDAs
- Customers requesting IPv6 access to the 6Bone or between their host site to remote locations

- A need to interconnect IPv6 with other providers
- Requirement to route IPv6 traffic on the backbone versus building tunnels

Enterprise users may have needs for any of the following:

- Application(s) requiring IPv6
- IP telephony deployment requiring additional end-to-end addressing
- End-to-end security not available with existing NAT operations

For each router that IPv6 will be implemented, the IOS will need to be at 12.0 ST or 12.2 T or later. Refer to Cisco's Web page for memory requirements and individual device limitations. Cisco has more than 3000 listed minimum requirements based on product platform and feature set to determine the minimum DRAM and Flash memory requirements. To verify the platform support and list minimum memory requirements for the intended deployment, follow these steps:

1. Go to www.cisco.com/kobayashi/support/tac/software.shtml.
2. Select either **Software Advisor | Search by Release**, or **Software Upgrade Planner** for Cisco IOS selection.

Understanding Deployment Methods

It is assumed that IPv4 and IPv6 will have requirements to co-exist in the near future. IPv6 has not caught up with IPv4 robustness in routing protocols and forwarding acceleration methods, which may delay the decision to fully migrate to IPv6. However, isolated segments or departments may require IPv6 now, so to accommodate this are four basic deployment methods:

- Manually configure tunnels across an existing IPv4 network for encapsulating IPv6 traffic.
- Implement mechanisms that can automatically build and select tunnels based on the IPv6 header.
- Provision dedicated virtual links such as ATM or Frame Relay PVCs or MPLS VPNs.
- Deploy a dual-stack network where IPv4 and IPv6 are fully redundant of the other protocol.

Each of these deployment methods requires upgrading some of the routers to support both IPv4 and IPv6 protocols, which we will refer to as dual-stack routers.

Regardless of the deployment method, in a dual IPv6/IPv4 network, it will be the Domain Name Server (DNS) that directs the requesting host to which protocol to use. The API exists for DNS to support both IPv6 and IPv4 protocols on the same server. A new "A6" record now exists. "A6" records are the equivalent of the "A" record used by IPv4. DNS will probably play a major role in simplifying the deployment of IPv6. As IPv6 routes are added, the DNS can be updated to the new IPv6 addresses for hosts that have limited users still using IPv4. The DNS can be configured with the IPv6 address as the primary and the IPv4 address as the secondary.

NOTE

Cisco routers do not currently route IPv6 by default as they do IPv4. You must enter *ipv6 unicast-routing* in global configuration mode to enable the forwarding of IPv6 unicast packets.

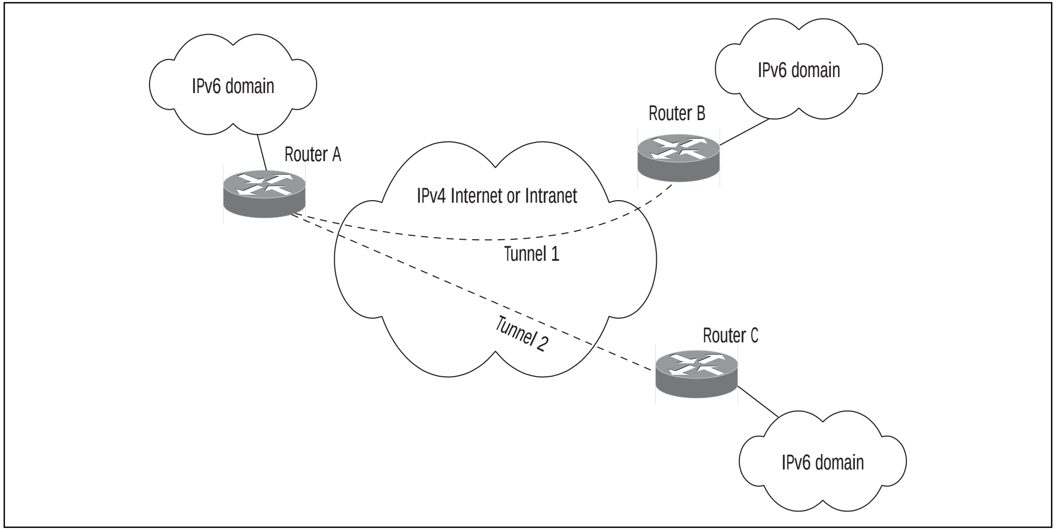
Configuring IPv6 over IPv4 Tunnels

This is a very quick and easy approach to deploying IPv6 to limited locations. IPv6 appears only at the access points supporting IPv6 hosts. The interface supporting the IPv6 segment is configured for both IPv4 and IPv6, or IPv6 only. The IPv6 traffic is then encapsulated into IPv4 packets for transport through the network. Only the routers supporting the ingress and egress tunnel interfaces require a dual stack (running both IPv6 and IPv4). Each of the tunnel interfaces will treat IPv6 packets the same as a physical interface. The IPv6 hosts will use the tunnel's address as their default gateway function and a routing protocol such as IPv6 RIP will be required to forward packets (route) between tunnels.

IPv6 Manually Configured Tunnel

IPv6 packets are encapsulated inside IPv4 packets in the payload section. A dual-stack router is required at the end-points of the tunnel. The tunnel appears as an LLC point-to-point connection to the IPv6 domains. With the entire IPv6 packet encapsulated inside the IPv4 payload section of the frame, the original IPv6 headers are not modified. If there are a limited number of IPv6 domains to interconnect, this may be a viable solution. However, you have to manually configure a tunnel for each connection, which will impose scaling and management issues with large numbers of IPv6 domains interconnecting with manual tunnels. Figure 7.1 illustrates this type of configuration. If Router B sends a packet to Router C, the packet has to travel across the tunnel to Router A before it can enter the second tunnel to reach Router C and vice versa. The packets are routed across the IPv4 network based on the destination address of the tunnel. No routing decisions are made based on the IPv6 destination address; only the tunnel destination address is routed upon, while IPv6 is encapsulated within the tunnel.

Figure 7.1 Manually Configured Tunnel



The minimum configuration commands required to configure the tunnel are listed in this section. For the service provider, this may mean building a tunnel between customer sites or possibly the 6Bone. For the enterprise, this would enable communication between two IPv6 domains across the existing IPv4 infrastructure.

The following is the required minimum configuration needed to set up a manual IPv6 tunnel on a Cisco router. The following commands will create a tunnel and then specify both the ingress and egress of the tunnel.

The first command creates the tunnel or virtual interface for IPv6.

```
Router6(config)#interface tunnel tunnel-number
```

The following commands are entered at the (tunnel) interface level. An IPv6 address is assigned to the tunnel interface. The tunnel interface processes packets just like a physical interface.

```
Router6(config-if)#ipv6 address ipv6-prefix/prefix-length [eui-64]
```

Next, create the local Tunnel End Point (TEP) associated with this tunnel interface.

```
Router6(config-if)#tunnel source {ip-address | interface-type interface-
number}
```

Then you will, create the remote TEP by specifying the remote TEP IP address.

```
Router6(config-if)#tunnel destination ip-address
```

The last command that you need to complete the tunnel mode is to select the encapsulation method. There are two types: *ipv6ip* and Cisco's protocol-independent Generic Route Encapsulation (GRE) *gre ip*. If you use a GRE tunnel, simply replace the *ipv6ip* in the command with *gre ip*

```
Router6(config-if)#tunnel mode ipv6ip
```

IPv6 over IPv4 GRE Tunnel

Cisco's standard GRE tunnels provide the necessary services to encapsulate traffic in a point-to-point configuration. GRE tunnels are not tied to any protocol or transport, which is a benefit over the preceding tunnel technique. As a result of this protocol independence, GRE is the only tunnel method that can distinguish Integrated IS-IS from other protocols and transport that traffic on the same tunnel. The only configuration difference from the IPv6 manually configured tunnel is the last step, which specifies the tunnel type. Substituting GRE for the tunnel mode will create a GRE tunnel. IPv6 traffic will have the same forwarding characteristics as the IPv6 manually configured tunnel.

Configuring & Implementing…

QoS on Manual Tunnels

Queuing methods used on standard interfaces may not apply to virtual interfaces. Priority Queuing, Weighted Fair Queuing, Distributed Weighted Fair Queuing, and Committed Access Rate methods are NOT supported on tunnel interfaces. For Quality of Service on tunnel interfaces, Generic Traffic Shaping will enable the manual configuration of the bit-rate, burst-size, and excess-burst-size for the virtual interface. If

Continued

the current traffic requires a set amount of bandwidth, this will ensure the tunneled traffic won't exceed a set bandwidth and compromise the traffic assigned to the physical interface.

Automatic IPv6 Tunnel

Automatic tunneling of IPv6 over IPv4 was an early technique where the low order 32 bits of the IPv6 address were replaced with the 32-bit IPv4 address and the leading 96 bits were 0:0:0:0:0:0. This would produce an IPv6 address of 0:0:0:0:0:0:A.B.C.D, where A.B.C.D would be the IPv4 address. Although this gives an automatic way of building tunnels, the benefits of the increased address space of IPv6 cannot be realized, because each host must have an IPv4 address and the same constraints remain for which IPv6 was created to resolve. This technique doesn't scale well in large environments; it appears ISATAP is an extension of this format allowing aggregatable IPv6 addressing. ISATAP is still in draft form and is explained later in this chapter.

The following is the required minimum configuration needed to set up an automatic IPv6 tunnel on a Cisco router. The following commands will create the tunnel, associate the interface with the tunnel source, and specify the tunnel type.

This first command creates the tunnel or virtual interface for IPv6.

```
Router6(config)# interface tunnel {tunnel number}
```

There is no need for an ip address on the tunnel interface.

```
IPv6.Router6(config-if)# no ip address
```

This is different from the manual tunnel where an IP address is assigned to the tunnel. An IPv4 and IPv6 address are both assigned to the physical interface. With the automatic tunnel, the TEP is directed to the interface only. With the tunnels being built automatically, there is no need to pre-configure the remote TEP.

```
Router6(config-if)# tunnel source {interface-type interface-number}
```

The following command defines the type of tunnel.

```
Router6(config-if)# tunnel mode ipv6ip auto-tunnel
```

Automatic 6to4 Tunnel

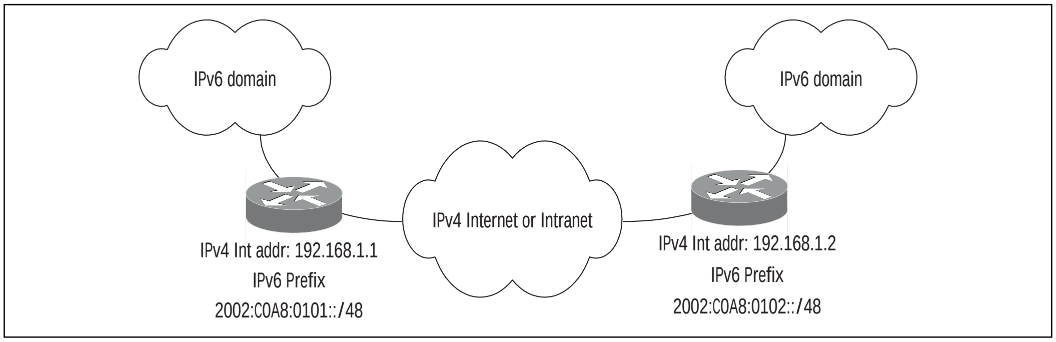
Cisco currently supports automatic 6to4 tunnels (RFC 3056). You need to do only minimal manual configuration for this technique. The IPv4 network, either an intranet or the Internet, is treated as a unicast point-to-point link layer. The intent of using 6to4 tunnels is to connect IPv6 domains and not for individual hosts, albeit technically it would work but could create scaling issues. Although IPv6 packets are encapsulated inside IPv4 packets, predefined tunnels are not defined. The Internet Assigned Number Authority (IANA) has assigned a unique routing prefix for 6to4 tunnels. The unique 13-bit TLA is 0x0002, or referred to as 2002::/16. The following 32 bits (the high level NLA) is the unique global IPv4 address of the 6to4 tunnel routers interface (See Table 7.1). When a 6to4 router receives an IPv6 packet that is not part of its own domain with an FP/TLA of 2002 it will encapsulate the packet inside an IPv4 packet and set the protocol field to 41. The router will then use the high-level NLA (32-bit IPv4 address) as the destination address and route the packet across the IPv4 network. If the TLA is not of a value of 2002, then the packet will be encapsulated in the same manner and routed (to a default route destination) to another automatic 6to4-configured router (see Figure 7.2) or relay router, possibly a (dual-stack) router connected to the 6Bone for de-capsulation and forwarding.

If multiple subnets exist in the IPv6 domain, an IPv6 IGP will need to be used but no IPv6 EGP is needed, because the IPv4 exterior routing will perform the necessary routing.

Table 7.1 6to4 Address

Bits	3	13	32	16	64
Field	FP 001	TLA 0x0002	V4ADDR	SLA ID	Interface ID

Figure 7.2 Automatic 6to4 Tunnel Interconnection



The following commands are the minimum commands required to set up the local router for 6to4 tunnels. First, use the following command to create the tunnel interface for IPv6:

```
Router6(config)#interface tunnel {tunnel-number}
```

The tunnel will use the specified interface for its source IPv6 address. The specified interface must have a valid global IPv6 address.

```
Router6(config-if)#ipv6 unnumbered {interface-type interface-number}
```

The following command defines the local TEP. This must be the same interface used for the source address in the previous command.

```
Router6(config-if)#tunnel source {interface-type interface-number
```

This next command specifies 6to4 as the tunnel method:

```
Router6(config-if)#tunnel mode ipv6ip 6to4
```

The next command exits the interface mode and returns to global configuration:

```
Router6(config-if)#exit
```

Finally, the last command creates a static route for all 6to4 addresses to the tunnel interface:

```
Router6(config)#ipv6 route 2002::/16 tunnel {tunnel-number}
```

ISATAP Tunnel

Future support of Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) is planned from Cisco at a later date. This is very similar to the 6to4 technique with a major difference being where the IPv4 address is. 6to4 imbeds the address in the NLA ID field and ISATAP uses a specified TYPE of "FE" and inserts the IPv4 address into the combined TSE and TSD fields of the IPv6 packet (see Table 7.2.) As the name implies, ISATAP is meant as a means of connecting intranets versus connecting inter-domains. This approach offers advantages such as the following:

- Aggregatable global unicast IPv6 addresses can be created that can be either routed across the IPv6 network or tunneled locally.
- Greater aggregation at the borders routers can be realized using the upper 64 bits as defined, versus imbedding an IPv4 address as 6to4.

- Both registered and private addresses can be used in ISATAP tunnels with no effect.

Table 7.2 ISATAP Address Format

Bits	3	13	8	24	16	8	8	8	8	32-bits
Field	FP	RES	TLA ID	NLA ID	SLA ID	0x00	0x00	0x5E	0xFE	IPv4 Address of Endpoint

6over4 Tunnels

Cisco has no plans to implement 6over 4 and recommends using ISATAP instead. This is another automatic tunneling method, which uses the IPv4 address as the IPv6 lower 32-bit interface address. The IPv6 packets are encapsulated in IPv4 packets with the protocol type of 41. The IPv6 frame would be represented as FE80::A.B.C.D where A.B.C.D is the IPv4 address. 6over4 uses Internet Group Membership Protocol (IGMP) to notify local IPv4 routers of their intent to use 6over4. The requirement of a multicast infrastructure has limited its deployment.

Tunnel Broker

Tunnel Broker (TB) is not a Cisco function. The broker, most likely an ISP, is responsible for providing access to destined IPv6 hosts or networks. TB is usually a server serving many clients, or tunnels, and providing connection to either IPv6 services or the 6Bone. The customer, with either a dedicated IPv6 or dual-stack router, creates a connection to the ISP TB, most likely a dual-stack server, and the ISP assumes responsibility of creating the tunnels to the end networks or hosts using 6over4 tunnels. Tunnels can be created between hosts or routers. The Web site at www.ipv6.org should have a list of well-known tunnel brokers. A search of the Internet will result in a number of free tunnel brokers, which your host can connect to. The only requirement on the host is a valid IPv6 stack.

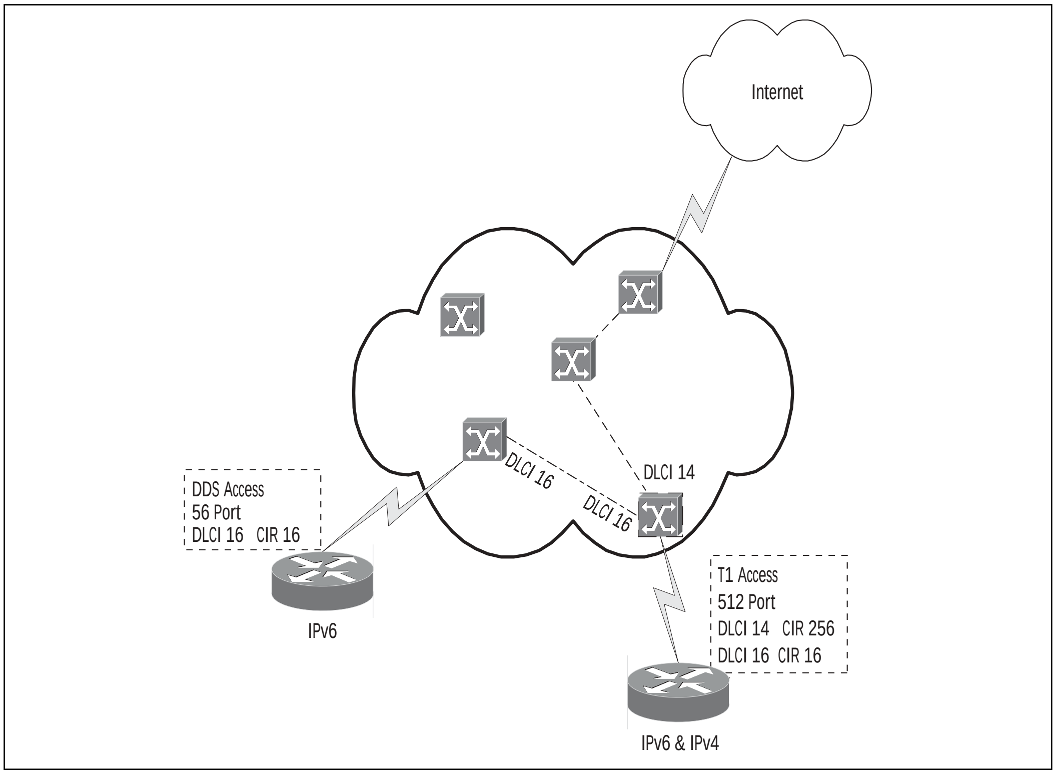
TB tunnels use 6over4 tunnels; because Cisco doesn't support 6over4 tunnels a Cisco router cannot be used to create a tunnel to a TB.

Configuring IPv6 over Dedicated Links

Both physical and virtual links can be used to segments or isolate IPv6 traffic from IPv4 traffic. A dedicated WAN physical link provides separation of IPv6 traffic, but these links are typically costly and slow to be provisioned. Three common types of WAN connectivity support virtual links, which the administrator can use to control traffic flows and provide QoS. Frame Relay, ATM, and now with the emerging optical networking, lambdas can be provisioned on a single interface.

For Frame Relay (Figure 7.3), you can add a PVC to an existing port and provide dedicated CIR to the PVC. You can think of ATM in much the same way, except the added levels of QoS that can be provisioned to a single PVC is much more granular. With optical networking, if you use multiple lambdas, you can configure dedicated lightstreams for separation of traffic types. In all three of these examples, it is possible to use a separate virtual interface and not require the addition of additional interfaces or "local loops" from their service provider. For both the end customer and the service provider, this enables both IPv4 and IPv6 traffic to share resources while at the same time enabling enforcement of SLAs and QoS.

Figure 7.3 Frame relay



Pros:

- Link utilization is unaffected. Existing SLAs can be maintained.
- Capability to utilize existing local loops and router investment while keeping IP backbone traffic separated.

Cons:

- IPv6 routing isn't as robust as that of IPv4. OSPF and EIGRP aren't available yet and hardware acceleration is only available on one or two models currently.

Figures 7.4 (A & B) are display commands of the configuration of the Frame Relay serial port from the *show interface serial 0* command.

Figure 7.4 (a) Frame Relay Configuration Example: Hub Cisco Configuration

```
interface Serial0
 encapsulation frame-relay
!
interface Serial0.14 point-to-point
 description to Internet
 ip address 1.2.3.4 255.255.255.0
 frame-relay interface-dlci 14
!
interface Serial0.16 point-to-point
 description to IPv6 remote domain
 ipv6 address 3ffe:1111:2222:1122::62/64
 frame-relay interface-dlci 16
```

Figure 7.4 (b) Frame Relay Configuration Example: Cisco IPv6 Configuration

```
interface Serial0
 encapsulation frame-relay
!
interface Serial0.16 point-to-point
 description to Hub Cisco
 ipv6 address 3ffe:1111:2222:1122::63/64
 frame-relay interface-dlci 16
!
```

NOTE

As with IPv4 addresses, IPv6 addresses can be mapped to DLCIs on a single interface versus using sub-interfaces.

```
frame-relay map IPv6 IPv6-address dlci [broadcast] [cisco]
[ietf] [payload-compression {packet-by-packet |
frf9 stac [hardware-options] | data-stream stac
[hardware-options]}]
```

Deploying IPv6 over an MPLS Backbone

Very little change is needed for an MPLS network when deploying IPv6. Because MPLS forwards based on labels and not IP headers, the fact the IPv6 traffic is being transported is transparent to MPLS. Three approaches exist for transporting IPv6 over an MPLS network. The first two approaches are supported currently and don't introduce new technologies to the P or PE routers; the third approach supports IPv6 at the PE and is currently under development in the draft stages within the IETF. The first two approaches are very attractive to the service provide running MPLS, because they require no upgrades to hardware or software to the network.

IPv6 Using Tunnels on the Customer Edge Routers

The approach of IPv6 using tunnels on the Customer Edge (CE) routers is much like the first examples of building tunnels across a routed network. You configure the CE router as a dual-stack router and configure for IPv4-compatible or 6to4 tunnels. All traffic to the PE routers remains IPv4 and therefore the IPv6 is transparent to MPLS. For the enterprise user utilizing MPLS, this enables the use of the existing MPLS infrastructure for IPv6 tunneled within IPv4.

IPv6 over a Circuit Transport over MPLS

IPv6 over a circuit transport over MPLS is another way to move IPv6 traffic across the MPLS network with no changes needed to the P or PE routers. Layer 2 tunnels such as Any Transport over MPLS (AToM) can be created. Because these tunnels are based on layer two technologies, the MPLS network doesn't care if the upper-layer traffic is IPv6 or IPv4. AToM can provide both leased line "like" services as well as layer 2 services for Frame Relay, ATM, PPP, HDLC, or Ethernet across the switched MPLS backbone. Like in the manual tunnel examples earlier, the ingress and egress are defined for the traffic flows and would behave as point-to-point circuits more than routed traffic. Another limitation is that IPv4 and IPv6 cannot co-exist on the same circuit. For the service provider this is an attractive way to provide end-to-end connections.

IPv6 on the Provider Edge Routers

IPv6 on the Provider Edge (PE) Router is under development as IETF Internet-Draft draft-ietf-ngtrans-bgp-ngtrans-bgp-tunnel-03 specification. PE routers must be configured as dual-stack routers to provide both IPv6 and IPv4 to the CE routers. Native IPv6 is exchanged between the 6PE and CE router. No changes are needed to the P routers, because they are forwarding based on labels, not IP headers. IPv4 is run between the PE and P routers to establish BTP sessions for propagating IPv6 reachability via MP-iBGP. The MPLS network appears to be providing native IPv6 services with no changes to the core and native IPv6 service to the CE routers. The dual-stack PE routers are referred to as 6PE routers and will exchange reachability information with other 6PE routers in the domain using multiprotocol BGP and sharing a common IPv4 routing protocol with the other P and PE routers. This is the only MPLS solution where reachability information is exchanged between the 6PE routers and shares a common IPv4 routing protocol. MP-BGP provides the extensions necessary to advertise next hop reachability using IPv6. IPv6 traffic will be encapsulated with two levels of MPLS labels. The top label is the label distribution protocol (LDP), which carries the traffic between the 6PE routers using IPv4 routing protocols. The lower label is associated with the destination IPv6 address and uses multiprotocol BGP-4. Please refer to Figure 7.5.

Using a Dual-Stack Backbone

Taken to the extreme, a dual-stack network would have all applications, hosts, and routers as dual-stack devices. This is a valid approach to IPv6 migration if prepared to undertake the required tasks. The upside is that there would be full native routing and communications throughout the network, regardless of the protocol, without building tunnels or placing the appropriate translations in front of specific networks or hosts. The network could be transitioned in two waves (see Figure 7.6). The first wave would be adding IPv6 to the network, then when the need for

IPv4 is no longer required, the second wave would be the removal of IPv4 from the network. The effort to fully implement a dual-stack network may outweigh the benefits. To fully implement a dual-stack network, administration and management requirements will theoretically be doubled. A second addressing schema as well as routing plan/protocol will be implemented, all applications and servers would require the addition and testing of IPv6 and all the client hosts and applications would be upgraded with IPv6. What may be the biggest issue with proceeding with this approach is the lack of comparative routing protocols and features developed for IPv6 at this time. IPv4 has mature QoS and routing robustness, which hasn't been implemented in IPv6 as of this writing.

Figure 7.5 IPv6 on the Provider Edge Routers

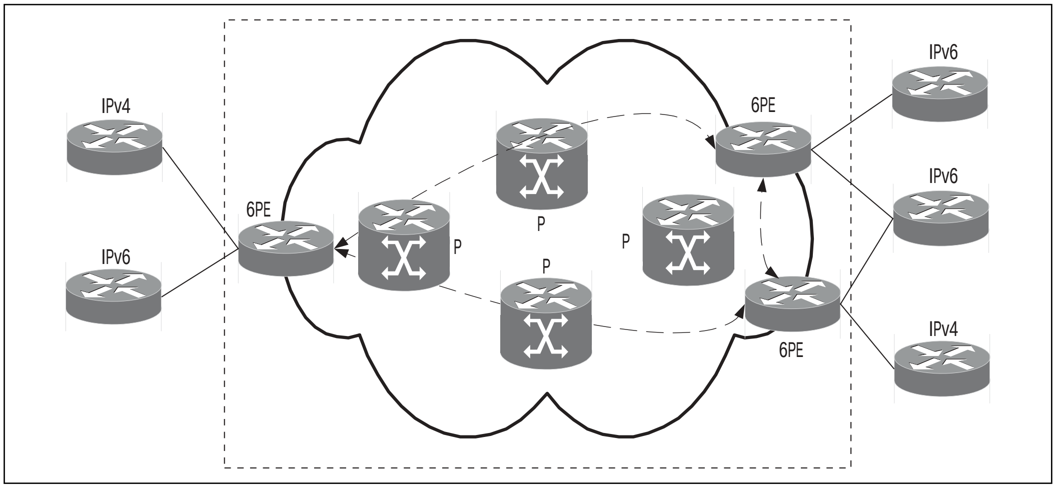
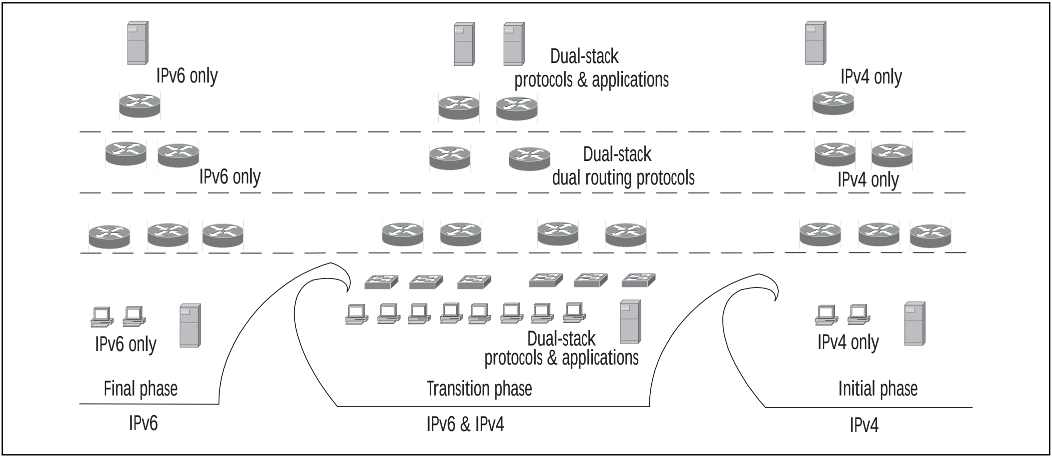


Figure 7.6 Migration to IPv6 from IPv4 Using Dual-Stack Strategy



Designing & Planning…

Planning Methodology

The following items should be inputs to any network design or upgrade. IPv6 is no exception.

- How many hosts: What are the effects on the addressing plan?
- What are the current network utilizations? Is the LAN saturated; is the WAN saturated? Is there available bandwidth to support the new additions?
- What are the traffic patterns? How much traffic is destined for other domains?
- Is there sufficient budget? This applies to hardware, software, training, and personnel.
- CPU and memory utilization: Can the current equipment support the additional protocols and traffic?
- Training: Is training needed for design, support, or management of proposed deployment?
- Staffing: Can the current staff complete the proposed changes with the current responsibilities?
- Security: Will the new changes fit into the current security policies?
- Time Line: Establish a time line with defined milestones that are realistic.
- Management requirements: Needs for fault, performance, and security need to be addressed.
- Response time: Ensure SLAs are considered with the design.
- Give thorough consideration to the chosen routing protocol to ensure it will scale with the network; do not make a short-sighted decision based on immediate requirements.
- Creating an aggregateable addressing plan will simplify the routing plan and minimize the utilization placed on the router.

The following input will be needed for decision of deployment of IPv6 in addition to the preceding considerations.

Continued

- Is connection to the 6Bone desired? Requirement for globally unique address.
- Is interoperability between IPv6 hosts and IPv4 required? Will the new IPv6 domain be an isolated test network or are the users expected to access the same applications as other users of the network?
- Will existing apps support IPv6? Is there a need for protocol translation?
- Will existing SLA prevent encapsulating IPv6 across the existing network? Will protocols need to be routed on separate circuits?
- Troubleshooting tools: Are the NMS system and engineers equipped to support the new protocol?
- Regardless of the deployment method chosen, basic IP design issues will remain. Determining the proper addressing and how many addresses are needed for each domain hasn't changed.

Each network will have unique requirements, but it is common to underestimate the requirements for network projects and as a result cause complications. Ensuring all the effected groups have input into the planning and design phase can help ensure a successful deployment.

Translating between IPv4 and IPv6

As with the network, there will be a transition period where IPv6 hosts will need to communicate with IPv4 hosts. In the first part of the chapter we talked about how to transport protocols across the network, but we didn't talk about how they would talk to the remote host if it hadn't been upgraded to IPv6 support. Building a tunnel across the network won't do any good if the host at the far end cannot communicate IPv6, hence the need to have an interim solution to translate IPv6 data to IPv4 data. Most of these solutions introduce a single point of failure in the network and are intended as a transition tool, not as a permanent network solution, but they do allow for the communication between IPv6 and IPv4 hosts.

Protocol Translation Mechanisms

As with the protocols, various mechanisms exist for translating IPv6 to IPv4, depending on the requirements. Protocols can be translated on a per-host basis, application basis, or all traffic entering and exiting a domain can be translated. Translations for all traffic entering or leaving a domain at Layer 3 is accomplished with NAT-PT or translated at Layer 4 with TCP-UDP. Data can be translated after it is received on the NIC and before it gets to the protocol stack with BIS; DSTM and Socks require a client on the IPv6 workstations that terminates the IPv6 session at the translation box, which then initiates an IPv4 session and passes traffic between the two sessions. The benefit of using a protocol translation is that there is no need to modify the existing applications. The network migration can begin without the need to upgrade all the applications at the same time.

NAT-PT

NAT-PT is defined in RFC2766.Cisco had an early release of NAT-PT in 11.3(x)T EFT, but due to problems has removed the feature from IOS 12.2(1)T; they plan to re-introduce the feature in a future release. The term NAT refers to Network Address Translation, as in IPv4, but instead of converting an IPv4 address to another IPv4 address, the translation is from a valid IPv4 packet to a valid IPv6 packet. The PT refers to Protocol Translation. Because IPv4 and IPv6 are different protocols, more is involved than with IPv4 NAT. Three flavors of NAT-PT are defined: Basic NAT-PT, NAT-PT, and Bi-Directional-NAT-PT. The following is a description of each:

- **Basic NAT-PT** Traditional NAT-PT is set up to enable an IPv6 domain to initiate sessions and communicate with hosts in an IPv4 network. The router providing the connection between the IPv4 and IPv6 networks would have a pool of IPv4 addresses for mapping sessions. When an IPv6 host originates a session outside the IPv6 domain, the router will map the IPv6 address to one of the available IPv4 addresses in the pool and translate the IP, TCP, UDP, and ICMP header checksums.
- **NAPT-PT** Adds port translation as well as address translation. This provides the same functionality as *NAT overload* used with Cisco's current IPv4 routing. A single IPv4 address can be used to map multiple IPv6 addresses to. The NAPT-PT translates the TCP and UDP port

numbers and ICMP query identifiers of multiple IPv6 addresses to a single IPv4 address.

- **Bi-Directional-NAT-PT** There is an additional requirement for hosts in an IPv4 network to initiate a session to an IPv6 domain. A DNS Application Level Gateway (ALG) must be used to translate V6 addresses in DNS Queries and responses.

TCP-UDP Relay

TCP-UDP Relay is defined in RFC 3142 as "An IPv6-to-IPv4 Transport Relay Translator." The assumption is that a new IPv6 host has a need to establish communication with an IPv4 host. The primary difference between TCP-UDP Relay and NAT-PT is that NAT-PT operates at the network layer and TCP-UDP works at the transport layer. This is technology that has been used in firewall-related products in the past and has been adapted to fill the requirement of IPv6 hosts communicating with legacy IPv4 hosts. Although the Transport Relay Translator (TRT) is a stateful device and DNS entries will need to map the IPv4 hosts to the TRT, no modification is required on any of the hosts or routers. This has been proven to work well with protocols such as HTTP, SMTP, SSH, and most of the day-to-day protocols. This is a very simplistic method for isolated IPv6 domains to communicate to IPv4 networks.

- Supports bi-directional traffic only
- May not work with NAT and certain protocols such as IPSec
- MTU size isn't an issue, because the translation happens at the transport layer and not the network layer.
- Most likely server router based.
- No changes needed on routers or hosts

Bump in the Stack Method

The Bump In the Stack (BIS) method is defined in RFC2767. BIS inserts modules in dual-stack hosts between the network Interface driver modules and the IPv4 modules. These modules snoop data flowing between the modules and translate IPv4 into IPv6 and vice versa. This enables applications that have not or cannot be modified to communicate with IPv6 to support IPv6 traffic transparently. All the functions of NAT-PT, including defining a pool of IPv4 addresses, are performed on the host. The difference is, instead of performing NAT-PT at the IPv6 domain edge, the translation is performed at the host. This is a valid solution in the initial stages to address applications waiting to be modified to run in the IPv6 domain or for applications that cannot be upgraded.

Dual-Stack Transition Mechanism

Dual-Stack Transition Mechanism (DSTM) has not been finalized as of this writing. The current IETF draft is draft-ietf-ngtrans-dstm-07.txt. Cisco has not made a public commitment to support DSTM as of this writing. Unlike the other translation mechanisms, DSTM includes tunneling. DSTM is designed as a solution for intranets that have totally converted the network to IPv6 and still have requirements to access IPv4 hosts or have applications in the network that still require IPv4. The assumptions for DSTM is that the hosts will be dual-stack hosts with a DSTM client installed, which will sit between the IP stacks and the interface. A DSTM TEP (tunnel end point) and DSTM server are also required. No IPv4 address is defined to the hosts when using DSTM. A server (DSTM server) has a pool of addresses or ports that are allocated on a per-session basis. When the need to initiate an IPv4 session is initiated, the host will contact the DSTM server for a valid IPv4 host address to use during the session. The server will provide a valid IPv4 address from its address pool to the requesting host to use for the session. The sending host will then encapsulate the IPv4, using the provided address from the DSTM server, packet inside an IPv6 packet using IPv4 over IPv6 (also referred to as 4over6 tunnel) and send the packet to the DSTM TEP, which will de-capsulate the IPv4 packet from IPv6 and forward the IPv4 frame.

Socks-Based IPv6/IPv4 Gateway

The Socks-based IPv6/IPv4 gateway method is based on version 5 of the SOCKS protocol (SOCKSv5) and is defined in RFC3089. This method is unique in that the translation between IPv4 and IPv6 is done at the application layer and not at the networking layers. A "Socks Lib" module is required at the client site; this module resides between the application layer and the socket layer. This is referred to as socksifying the client. The other requirement is that a dual-stack gateway (enhanced SOCKS server) is required to connect the IPv4 and IPv6 network domains. When the client wants to establish a session to a host with a different protocol, the client creates and terminates the session to the gateway. This session is entirely IPv4 or entirely IPv6. The gateway, which is also connected to the dissimilar network, then creates a session to the remote host using the second protocol stack and relays the packets between the two hosts. The Socks protocol can run on both Windows and UNIX operating systems. Although this method does not require dual-stack workstations, it does require socksifying all the hosts. The two major differences with Socks are that the sessions terminate at the gateway and there is end-to-end session security between the gateway and the client and the gateway and the destination host. The network headers are not modified in any way that eliminates issues with fragmentation, properly translating protocol features, and DNS translations.

Summary

No matter how you look at it, there will be some pain in implementing IPv6. However, there is no reason that the pain has to be great or that the network has to be upgraded overnight. People are reluctant to change and may think that the way things are now aren't broken and don't need to be fixed. It appears that "if" has been replaced with "when" IPv6 will become the dominant protocol. This is evident by a quick glance at who is shipping IPv6 support now. Just to name a few are the desktop and server companies such as Compaq, Microsoft, IBM, Sun, Solaris, and Linux. In the networking field, Cisco, 3Com, Toshiba, Nortel, Lucent, Nokia, and many others are all shipping IPv6 now. You may already have devices in your home with IPv6, such as a 3G phone or Sony PlayStation 2, without even being aware of it.

With standards still being tweaked and the lack of hardware implementation, Cisco currently supports IPv6 in software; you may not want to rush out and perform an entire IPv6 upgrade on your network, such as creating an entire dual-stack network, yet. This chapter covered various techniques to deploy IPv6 into an existing IPv4 network, and/or to enable communications between IPv6 domains across an existing IPv4 infrastructure, or to ensure the capability to continue assessing legacy IPv4 servers. The approach that you choose will determine the number of upgrades to routers for IPv6 support and IPv6 stack installs that will be required on hosts. Over a dozen different techniques have been discussed in this chapter for integrating IPv6 with IPv4. All these approaches fall into four basic categories: complete integration of IPv6 (dual stack), using dedicated links, tunneling, or protocol translation. These four approaches can even be combined if needed; for example, a dual-stack network could use dedicated links to ensure SLAs are met across the wide area network.

For isolated IPv6 domains that need to communicate between other IPv6 domains, both manual and automatic tunnels can carry IPv6 traffic across existing IPv4 networks. Manual tunnels encapsulate IPv6 packets inside IPv4 packets for delivery to pre-defined TEPs. These TEPs will be required to run both IPv4 and IPv6. IPv6 manually configured tunnels and IPv6 over IPv4 GRE tunnels are manual tunnels. Automatic tunnels also encapsulate IPv6 inside IPv4, but the tunnels don't have to be pre-defined. Instead, the edge router uses the destination address of the IPv6 packet to determine the end TEP. These approaches require much less setup time and each of the approaches use different addressing schemes for embedding IPv4 addresses inside the IPv6 address to facilitate the formation of automatic tunnels. There are a number of automatic tunnel techniques:

Automatic IPv6 Tunnel, Automatic 6to4 Tunnel, ISATAP Tunnel, 6over4 Tunnels, and Tunnel Brokers. Tunnels can also be created in MPLS networks by using the CE routers as the TEPs.

If the existing links that would normally be used for tunneling is of a critical nature or can't be jeopardized to perform outside set SLAs, using dedicated links for IPv6 is a better solution than tunneling (encapsulating) IPv6 inside IPv4. Three solutions exist for using dedicated links; the first is to use dedicated interfaces and dedicated physical links. Although dedicated transport and interfaces guarantee separation of traffic, it is costly to duplicate ports and circuits. If Frame Relay or ATM is deployed, a virtual circuit can be created with the proper bandwidth guarantees and appropriate QoS. Dedicated Layer 2 links can also be deployed if MPLS is deployed with the use of AToM with the appropriate QoS.

For native IPv6 support across the entire network, only one solution is available today. MPLS will be able to offer native service with the use of 6PE routers, but this approach is still in the draft stages in the IETF. For native support, all the routers would be upgraded to run both IPv4 and IPv6 concurrently—this is referred to as a dual-stack router. This creates the greatest overhead with the need for dual-addressing and dual-routing protocols running in the network. There are two current drawbacks with using this approach now. The first is that Cisco has implemented IPv6 in software and as such there is no hardware acceleration at this time. The second is that RIP and IS-IS are the only available routing protocols; OSPFv3 or EIGRP haven't been released yet for IPv6.

Regardless of the deployment method that you use to introduce IPv6 to your networks, you will need to communicate to IPv4 hosts. Even if all the servers in the network were to support IPv6, there are still IPv4 hosts on the Internet, which will require protocols translation to facilitate communication. Again, there is more than one way to perform IPv6 to IPv4 translations. NAT-PT and TCP-UDP use a server to pass traffic between the two protocol domains with protocol translation happening at the network layer and transport layer respectively. BIS is a solution that is deployed on the destination server. A module is installed that sits between the network card and the IPv4 protocol stack. When an IPv6 packet arrives at the host, it is translated to IPv4 and delivered to the IPv4 protocol stack. DSTM and Socks-based gateways use gateways that communicate between the protocol domains. Both require a client on the IPv6 hosts that terminates the session at the translation server and the server then creates a session to the end IPv4 host. Only Socks is available at this time; DSTM is still in the IETF draft mode.

Solutions Fast Track

IPv6 Deployment Strategies

☑ Creating tunnels across an existing IPv4 network is a tried-and-true method of creating connectivity between IPv6 domains. Point-to-point can be created manually with IPv6 manually configured tunnels and Cisco's GRE tunnels. Tunnels can be created automatically based on the address using Automatic IPv6 Tunnels, Automatic 6to4 Tunnels, and ISATAP Tunnels. If MPLS is deployed, tunnels can be created at the CE router.

☑ For QoS or SLA requirements, the use of dedicated links will provide the needed control over the different protocols. Separate links can be provisioned with total separation as the most costly and secure method. Using virtual links with transports such as Frame Relay, ATM, or Optical LAMBDAs. Circuit transport over MPLS can also provide QoS while separating the protocols.

☑ When isolating IPv4 from IPv6 isn't a concern or there is a need to have both IPv4 and IPv6 co-exist in the domains both hosts and routers can run in a dual-stack mode which basically running dual protocol stacks in both the hosts and routers. The routers will also be required to run dual routing protocols.

Understanding Deployment Methods

☑ **Tunnels** A technique to encapsulate IPv6 packets inside IPv4 packets for transport across an existing IPv4 infrastructure.

- **IPv6 Manually Configured Tunnel** TEPs are defined across an IPv4 network. All IPv6 traffic enters and exits the network at the TEPs.

  Pros: Ease of deployment, low cost, tried-and-tested technology.

  Cons: Doesn't scale, tunnels are pre-defined point-to-point links, high management overhead, troubleshooting is difficult.

- **IPv6 over IPv4 GRE Tunnel** Enhanced features of the IPv6 manually configured tunnel. Added feature of protocol independence of traffic configured for the tunnel.
- **Automatic IPv6 Tunnel** Tunnels are created automatically based on the lower 32 bits of the address. The IPv6 address is created by taking an IPv4 address and padding 0's into the upper 96 bits.

  Pros: Ease of deployment, low cost, tunnels are created automatically.

  Cons: No realization of the increased address space of IPv6, doesn't scale, troubleshooting is difficult.
- **Automatic 6to4 Tunnel** Tunnels are created automatically based on the IPv6 address. An IPv4 address is used in the upper 32 bits of the NLA, which allows another 16 bits of SLA for segmentation within the IPv6 domain.

  Pros: Ease of deployment, low cost, tunnels are created automatically, use of valid IPv6 frames.

  Cons: Troubleshooting is difficult,
- **ISATAP Tunnel** This tunnel technique is still in draft form. Cisco has plans to support ISATAP at a later date. ISATAP uses an IPv4 address to create the tunnel, but places the 32 bit IPv4 address in the low 32 bits of the interface and uses a valid 64 bit IPv6 network address.

  Pros: Ease of deployment, low cost, tunnels are created automatically, full use of valid IPv6 network addresses.

  Cons: Troubleshooting is difficult.
- **6over4 Tunnels** This tunnel technique uses multicast routing as a virtual link layer. Cisco does not and does not plan to support 6over4 tunnels. Like the Automatic IPv6 tunnel, an IPv4 address is used in the lower 32 bits with a fixed FE80::/96 prefix.

  Pros: Ease of deployment, low cost, tunnels are created automatically.

  Cons: Not supported on Cisco products, no realization of the increased address space of IPv6, requires multicast network, doesn't scale, troubleshooting is difficult.

☑ **IPv6 over Dedicated Links** Using existing transport infrastructure, IPv6 is directed to dedicated links or virtual links. IPv6 has no impact on IPv4 traffic.

Pros: QoS and SLAs can be managed for IPv4 and IPv6 separately.

Cons: IPv6 hardware acceleration and routing isn't as robust as that of IPv4.

☑ **IPv6 over an MPLS Backbone** Tunneling, dedicated links, and dual-stack techniques work with MPLS as they do with IPv4-routed backbones. The same restraints and benefits apply to each of the techniques; the only difference is the benefits realized by having MPLS on the backbone.

- **IPv6 Using Tunnels on the Customer Edge Routers** This has the same characteristics as the manually configured tunnels, with the difference being that the encapsulated packets are switched across the MPLS backbone versus being routed across an IPv4 backbone.
- **IPv6 over a Circuit Transport over MPLS** Same characteristics as using dedicated links. MPLS VPNs are created with the appropriate QoS assigned to each VPN.
- **IPv6 on the Provider Edge Router** This is not commercially available and is still in draft form with the IETF. Using MP-BGP, MPLS is able to advertise next-hop reachability using IPv6 address, which will enable native IPv6 service to be offered to the CE routers from the PE.

☑ **Using a Dual-Stack Backbone** Requires fully redundant configurations of IPv4 and IPv6 with each router supporting both IPv4 and IPv6 and their routing protocols.

Pros: Ease of deployment, ability to use protocol analyzer for IPv6 issues, full management of IPv6.

Cons: IPv6 hardware acceleration and routing protocols not as robust as IPv4, depending on the size of the network, the overhead to upgrade all the routers (and hosts) for IPv6 support could be large; possible hardware and memory upgrades to support IPv6.

Translating between IPv4 and IPv6

☑ Translation techniques have three basic approaches:

- All traffic passes through a single "translation device," such as NAT-PT or TCP-UDP Relay.
- Sniffing packets at the host and making the translation from IPv6 to IPv4 before the NIC passes the data to the network layer of the host. The BIS method requires a client (or module) to be installed on each of the workstations.
- Using a server to terminate both the IPv6 session as well as the IPv4 session. Both DSTM and Socks-based IPv6/IPv4 gateways relay data between the two sessions. This requires the installation of a client to communicate with the server.

☑ Using a server to translate all traffic creates a single point of failure, because all traffic passes through the translator. Heavy traffic could tax the server, requiring more server power or additional memory to support the volume of packets. The upside is that this approach eases management—because all traffic passes through a single device, all changes and troubleshooting would be at a central location and not dispersed across the network.

☑ Installing clients on each workstation can be an administration and management burden for large deployments. For large deployments the amount of effort to install clients on all the workstations and manage host compatibility and possible upgrades should be factored in before deploying these techniques.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: Is IPv6 routing enabled in Cisco IOS?

A: IPv6 is not enabled by default and must be enabled in global configuration of the router.

```
Router6(config)# IPv6 unicast-routing
```

Q: What IOS is required to support IPv6?

A: IOS will need to be at 12.0 ST or 12.2 T or later.

Q: Can IPv4 and IPv6 co-exist on the same networks?

A: Yes, they can. There is no need to segment the two protocols, and co-existence allows for a phased deployment of IPv6 onto the network.

Q: My production servers are deployed with IPv4; how could I justify deploying IPv6?

A: IPv6 could be loaded on the server and the applications could create sessions on both protocol stacks, or, if there are barriers to this, there are many protocol translations discussed in this chapter to translate IPv6 into IPv4 packets so that the server would only see IPv4 traffic and would have no knowledge of IPv6.

Q: How much equipment must be added to the network to deploy IPv6?

A: Existing Cisco routers can support both IPv4 and IPv6 with the current 12.2T release.

Q: My current IP traffic is mission critical and is subject to strict SLAs; introducing a new protocol to these links would be hard to justify.

A: IPv6 can be isolated with the use of virtual links while maintaining bandwidth, and QoS guarantees your existing traffic. Frame Relay and ATM both offer virtual circuits, MPLS offers AtoM, and optical LAMBDAs can all provide independents traffic prioritization across a common transport.

Chapter 8

IPv6 Security

Solutions in this chapter:

- IPSec Overview
- Understanding the Building Blocks of IPSec
- Combining IPSec's Cryptographic Mechanisms
- Applying Perimeter Security

☑ Summary

☑ Solutions Fast Track

☑ Frequently Asked Questions

Introduction

In the open, hostile environment of today's Internet, clearly the security of information must be the top priority of any organization. To best protect an organization's critical electronic information, security should be implanted at many layers, but the introduction of security at the network layer has grown significantly in recent years. IP Security (IPSec) specifications have been implemented widely with IPv4 between communicating hosts on local networks or on *virtual private networks* (VPNs), in which the Internet serves as the communication channel between two private networks.

IPv6 has integrated the IPSec framework into the protocol specifications and allows for easy implementation of security at the network layer. This chapter focuses on the IPSec framework, explaining how each component works in IPv6; it also explores the topic of perimeter security, another critical component of a good security implementation, and explains how to apply this type of security for IPv6 using Cisco IOS.

IPSec Overview

Security risks specific to IP include data theft, data tampering, and peer impersonation. *Data theft* is typically accomplished by eavesdropping on all the traffic traversing a particular network segment using a device called a *sniffer*. *Data tampering* involves actually intercepting traffic (not just eavesdropping) that is not destined for a particular machine, modifying the data, and sending the modified traffic along to the ultimate destination. *Peer impersonation* involves pretending to be another node by sending traffic to a particular destination node, but specifying another IP address as the source address instead of your IP address. This impersonation is called *IP spoofing* and can be used in denial of service (DoS) attacks and in more sophisticated attacks that exploit a trust relationship.

Through the inclusion of IP Security (IPSec) extension headers, IPv6 includes security features that provide cryptographic security services at the network layer:

- **Authentication** The verification of the identity of a peer—a countermeasure against IP spoofing.
- **Integrity** The verification that data has not been modified in transit—a countermeasure against data tampering.

- **Confidentiality** The protection of data in transit—a countermeasure against data theft.
- **Access control** The limitation of communication to only authorized peers—can be a countermeasure against many attacks.

IPSec is a standard defined by a suite of Requests for Comments (RFCs) from the Internet Engineering Task Force (IETF) that can be implemented in both IPv4 and IPv6. There are many deployments of IPSec within the existing IPv4 world used to establish VPNs. A VPN provides you with the means to communicate over a public network while maintaining the privacy of the communications and can provide you with a cost-effective means of leveraging an Internet connection for communication to multiple enterprise sites, partners, and clients through a VPN gateway at the network perimeter. With IPv6, you will have an opportunity to utilize IPSec to provide true end-to-end security by protecting data along the entire path between the source and destination nodes, not just between gateways.

Designing & Planning…

Comparing IPSec and SSL

After reading about the complexity of IPSec, you may ask the question: Why would anyone use IPSec when Secure Sockets Layer (SSL) is available and is so easy to implement? The answer is that there are situations where it may make sense to use SSL instead of IPSec, but the use of IPSec instead of SSL makes sense in other situations. The basic differences can be described in terms of ease of use, implementation layer, and scope of protection.

Because of its wide implementation in Web servers and Web browsers, SSL is a commonly used protection that can be configured relatively easily. Many Web servers implement SSL to provide encrypted and authenticated communications (depending on the configuration) to an unknown community of potential browsers. The actual use of SSL during a session is often transparent to the user. On the other hand, IPSec has not been widely implemented or deployed on user workstations, and it is typically more complex to configure and implement. The most common use of IPSec today is in both remote access and site-to-site VPNs.

Continued

In terms of the OSI model, you implement SSL at the transport layer, and it provides protection to TCP services. No protection is available for UDP services. In addition, an application that wishes to take advantage of the protection provided by SSL needs to either be Web-enabled so that it can be used via a Web browser, or programmed to take advantage of the desired SSL services. On the other hand, IPSec is implemented at the network layer and provides protection to any transport layer protocol (including both TCP and UDP). It requires no modification to applications and can protect all traffic between two nodes.

Understanding the Building Blocks of IPSec

IPSec is a complex set of standards and protocols. You can implement IPSec in both IPv4 and IPv6. Within IPv4, IPSec services are just another “upper layer” protocol that you can specify within the Protocol field of the IPv4 header. Within IPv6, IPSec services are extension headers that you can include within an IPv6 header. In either case, the IPSec services consist of two security protocols: the Authentication Header (AH) and the Encapsulating Security Payload (ESP).

Extension Headers Overview

AH is an extension header and protocol that uses a cryptographic signature to provide both connectionless integrity and data origin authentication. *Connection integrity* ensures that the data within the packet is not modified in transit from the sender to the receiver; it is a countermeasure against data tampering. *Data origin authentication* ensures that the packet was sent by the expected sender, not someone impersonating the sender. It is a countermeasure against IP spoofing.

ESP is an extension header and protocol that can provide confidentiality, data origin authentication, connectionless integrity, replay protection, and limited traffic flow confidentiality. *Confidentiality* ensures that even if the traffic is subject to eavesdropping, the privacy of the data is ensured such that only the authorized recipient will be able to read it. The *data origin authentication* and *connectionless integrity* provided by ESP are not as extensive as those provided by AH in that AH protects as much of the IP header information as possible, and ESP does not (unless it is used in tunnel mode). *Replay protection* ensures that once a packet has been received, its session number cannot be reused by an attacker. *Limited traffic* *flow* confidentiality ensures that an eavesdropper cannot determine communicating nodes nor the frequency or volume of traffic. This is available only when using tunnel mode.

When more than one IPv6 extension header is included in a packet, the header should adhere to the following order:

1. IPv6 Header
2. Hop-by-Hop Options Header
3. Destination Options Header (for options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header)
4. Routing Header
5. Fragment Header
6. Authentication Header
7. Encapsulating Security Payload Header
8. Destination Options Header (for options to be processed only by the final destination of the packet)

NOTE

The ESP and AH formats and their use in both transport and tunnel mode are identified and described in Chapter 3.

Choosing the Mode of Operation

Both AH and ESP support two modes of use: *transport mode* and *tunnel mode*. In transport mode, the selected protocol (AH or ESP) provides protection primarily for upper layer protocols, and its extension header is created and inserted as part of the original IPv6 header. Figure 8.1 illustrates the basic premise of transport mode without getting into the specifics of either AH or ESP.

However, in tunnel mode, the selected protocol (AH or ESP) encapsulates the original packet in a new one and treats it as the payload. In other words, the extension header is created and inserted as part of a new IPv6 header that encapsulates the entire original packet as the payload. As a result, tunnel mode provides the requested services (AH or ESP) to the entire original packet. Figure 8.2 illustrates the basic premise of transport mode without getting into the specifics of either AH or ESP. Typically, you would use transport mode when utilizing IPSec directly between two communicating hosts, and you would use tunnel mode when utilizing IPSec between two gateways that provide services to one or more hosts behind each of them. Tunnel mode is commonly implemented today in order to create site-to-site VPNs for an organization.

Figure 8.1 IPSec Transport Mode

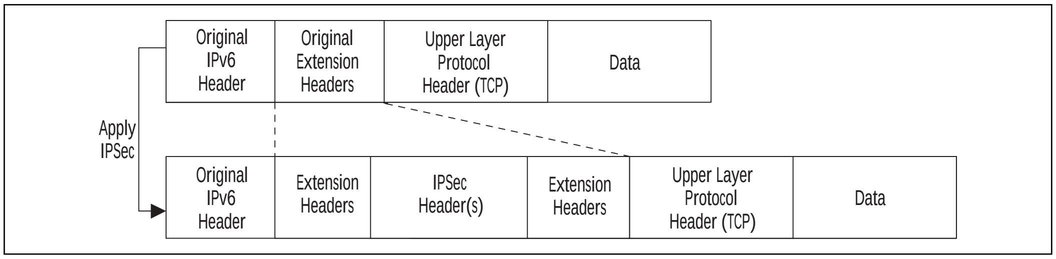
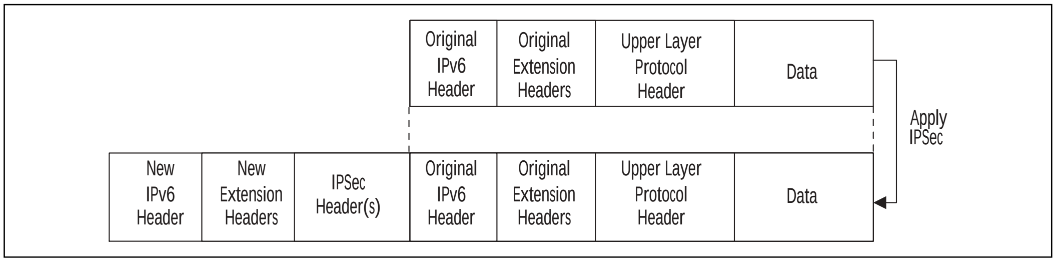


Figure 8.2 IPSec Tunnel Mode



Internet Key Exchange Overview

Although the AH and ESP protocols provide the actual cryptographic services at the network layer, you still need a mechanism that will determine which services should be applied to the different traffic flows, and to negotiate the required cryptographic keys for those services. Internet Key Exchange (IKE), defined in RFC 2409, is a secure key management protocol that essentially provides the mortar that ties everything together within IPSec. IKE negotiates a Security Association (SA) that contains the information necessary for an IPSec peer to process incoming or outgoing traffic. Because SAs are unidirectional, two SAs are required to specify the cryptographic services for bi-directional traffic between two IPSec peers. One SA is used for outbound traffic, and one SA is used for inbound traffic. In addition, the contents of a particular SA varies for each connection, based on the services that are being applied (such as AH, ESP, AH, or ESP). IKE negotiates these SAs using two primary phases:

- **Phase 1** Establishes an Internet Security Association and Key Management Protocol (ISAKMP, RFC 2408) SA, which provides a secure mechanism for negotiating the actual SA that will be used for the IPSec services (AH and ESP).
- **Phase 2** Establishes the pair of IPSec SAs that will be used to select and apply the appropriate IPSec services (AH and ESP).

The IKE protocol, phases, and their associated modes will be discussed in more detail in the section "Internet Key Exchange" later in this chapter.

Understanding the Implementation Options

Because IPSec is intended to provide network layer security services, it can be implemented on end hosts and devices. For an end host, the IPSec functionality is typically implemented in one of the following ways:

- **Native Operating System IP Stack** Many operating system vendors, including Microsoft, are now providing integrated IPSec functionality in the native IP stack of the operating system.
- **Replacement Stack** For those operating systems that do not have a native stack IPSec implementation, a complete third-party replacement stack may be available that takes the place of the entire native stack.
- **Bump in the Stack (BITS)** An alternative to a complete replacement stack, a third-party BITS implementation is inserted into the native operating system stack using a known application programming interface (API).

Although end host IPSec implementations are more available now, most actual IPSec implementations utilize IPSec gateways that provide services to one or more hosts that reside behind it. For a gateway, the IPSec functionality is typically implemented in one of the following ways:

- **Bump in the Wire (BITW)** Called a "bump in the wire" because it is largely transparent to other devices on the network, this IPSec implementation is typically a dual interface device that provides IPSec services to incoming and outgoing packets. An example of a BITW implementation is a VPN gateway.

- **Integrated with Firewall** An alternative gateway implementation that is integrated with a firewall. Because firewalls reside primarily at the network perimeter, this implementation is a relatively common implementation for remote access and site-to-site VPNs.

Designing & Planning…

Perimeter Topology

When a VPN is using IPSec, you often need to make a decision about the location of the VPN gateway with respect to a firewall. Should the VPN gateway be located behind the firewall, in front of the firewall, or parallel with the firewall? From a security perspective, the first choice should be the location that enables the firewall to inspect the decrypted traffic. This accomplishes two things:

1. Enables the firewall to apply a consistent policy to traffic traversing the network perimeter
2. Provides defense-in-depth should another site or user that is part of the VPN be compromised

To accomplish this, the VPN gateway can be located in a demilitarized zone (DMZ) and be protected by either the external border router or by the firewall itself, depending on how the DMZ is implemented.

Implementing the VPN gateway either parallel to the firewall or behind the firewall prevents the firewall from inspecting decrypted traffic. This could result in prohibited traffic traversing the perimeter through the encrypted tunnel, or it could lead to the compromise of the site if another site or user computer that is part of the VPN is compromised. Although there may be business-related or non-security related technical reasons to deviate from this recommended topology, an organization should be aware of the risks before proceeding: The firewall cannot inspect and control decrypted traffic from the VPN gateway.

Understanding the Authentication Options

Within IPSec are two primary methods of authenticating peers: *pre-shared keys* and *certificates*. In both methods, some authentication information is exchanged between the peers out of band. With pre-shared keys, two or more peers securely exchange (pre-share) a secret key prior to any IPSec negotiation. The method of exchanging the pre-shared keys is typically a manual process that involves typing them in or sharing them via a floppy diskette. Pre-shared keys are a simple and inexpensive way to authenticate peers; however, they do not scale as well as certificates. Certificates are a preferable authentication option for larger and more meshed IPSec deployments, but because they require the use of a certificate authority (CA), they are typically more complex to implement.

Cryptographic Algorithms Used in IPSec

You use different types of cryptographic algorithms to achieve different goals. The following types of algorithms are used in IPSec:

- **Keyed-Hash Message Authentication Code (HMAC)** Sometimes called a keyed hash, HMAC is used to authenticate both the source of a message and its integrity. It requires a shared secret key between the two parties and the use of a specified hash function. The HMAC function produces a value (the MAC) that is based on the message being sent and the shared secret key. The original message and the MAC are sent to the recipient, who calculates a MAC based on the received message and the shared secret key. The recipient then compares the calculated MAC to the one that was received with the message. If they match, then the sender is authenticated and the message has not been altered. If they don't match, then either the sender has been spoofed or the message has been altered. The keyed hashes required for IPSec are defined in RFC 2403 (HMAC-MD5-96), RFC 2404 (HMAC-SHA-1-96), and RFC 2857 (HMAC-RIPEMD-160-96). Other keyed hashes can be implemented but are not required.
- **Block-Oriented Encryption** The block-oriented encryption algorithm required for IPSec is DES. Stronger encryption algorithms are often implemented; these include Triple DES, Blowfish, CAST, IDEA, and RC5. In addition, the NULL encryption algorithm can be used with ESP, but does not actually provide any encryption. It would be used in situations where only ESP authentication is desired.

- **Diffie-Hellman Key Agreement** The Diffie-Hellman key agreement protocol relies on something known as the discrete algorithm problem to provide the security that enables two users to exchange a secret key over an insecure medium without the prior exchange of any secrets. It requires the use of private values by both peers and the exchange of associated public values by both peers. Diffie-Hellman is used within IPSec to generate session keys.

Combining IPSec's Cryptographic Mechanisms

Now that you are aware of the IPSec building blocks, you may be wondering how IPSec nodes determine which cryptographic services to apply and which cryptographic keys to use for specific traffic flows. It uses various mechanisms (which we'll explore in this section) to accomplish the following specific tasks:

- Determine whether any IPSec services need to be applied to the traffic.
- Negotiate the security services (AH and ESP), algorithms, and associated parameters to use for a given traffic flow. (Negotiation)
- Authenticate the node with which you are communicating. (Authentication)
- Exchange and manage keys for the cryptographic services. (Key Exchange)
- Track all agreements.

These tasks are accomplished through a combination of the behind-the-scenes mechanisms, including the Security Policy Database (SPD), the SA, and the Internet Key Exchange (IKE) protocol. The SPD solves the problem of identifying the traffic to which IPSec services should be applied. The IKE protocol solves the problem of negotiating the services, algorithms, and parameters, authenticating the other node, and exchanging and managing cryptographic keys. The SA concept solves the problem of tracking the IKE agreements with respect to services, algorithms, and parameters for particular traffic flows.

The Security Policy Database

How does an IPSec node know whether or not to apply IPSec services to a packet and which services it should apply? By using the Security Policy Database (SPD). The SPD identifies the services to be applied to IP packets, and is consulted in the processing of all traffic (inbound and outbound), including non-IPSec traffic. It requires you to provide the security policy information through the appropriate configuration mechanism (for example, command line or graphical user interface). For any packet, the SPD will identify one of three options for processing:

- **Discard** This option can be applied to traffic that is not allowed to enter or exit the host or gateway.
- **Bypass IPSec** This option can be applied to traffic that is allowed to enter or exit the host or gateway, but does not require IPSec services to be applied to it.
- **Apply IPSec** This option can be applied to traffic entering or exiting the host or gateway that requires IPSec protection. For this option, the relevant information must be specified, including the services, protocols, and algorithms.

The SPD is an ordered list of policy entries that are identified by *selectors*, which are roughly equivalent to the types of information used by a traffic-filtering device. Table 8.1 identifies typical IPSec selectors and the types of values that can be defined for each one.

Table 8.1 IPSec Selectors

Selector	Description
Destination IP Address	A single IP address, range of addresses, address + mask, or a wildcard address
Source IP Address	A single IP address, range of addresses, address + mask, or a wildcard address
Name	A userid (fully qualified user name string (DNS) – Mozart@foo.bar.com or X.500 distinguished name) or System Name (for example, FQDN or X.500 distinguished name)
Data Sensitivity Level	IPSO/CIPSO labels

Continued

Table 8.1 Continued

Selector	Description
Transport Layer Protocol	Obtained from the IPv6 “Next Header” field; may be an individual transport protocol value, a wildcard, or OPAQUE
Source Port	May be an individual UDP or TCP port values, a wildcard port value, or OPAQUE
Destination Port	May be an individual UDP or TCP port values, a wildcard port value, or OPAQUE

NOTE

The OPAQUE value referenced throughout Table 8.1 is necessary because the value for these fields may be encrypted.

You can use the selectors identified in Table 8.1 to define granular or broad policies to apply to traffic. For example, for host implementations, defining a broad policy can result in one SA being used for all traffic between a pair of hosts, regardless of the applications that are communicating. Likewise, for gateway implementations, a broad policy can result in one SA being used for all traffic between a pair of gateways, regardless of the hosts behind the gateways that are communicating. On the other hand, a granular policy on host implementations can result in different SAs being used for different communicating applications, depending on the security services required by them. Likewise, a granular policy on gateway implementations can result in different SAs being used for different communicating host pairs, depending on the security services required by them.

The Security Association

The SA is a mechanism that IPSec uses to keep track of the details of a negotiated IPSec session between two nodes. An SA is unidirectional so that a pair of SAs is actually required for communication between a pair of nodes. From one node’s perspective, one SA is used for outbound traffic and the other SA is used for inbound traffic. An SA is uniquely identified by three components: a destination IP address, a security protocol identifier (AH or ESP), and a Security Parameter Index (SPI). The SPI is a 32-bit number that is used to distinguish among different SAs terminating at the same destination IP address and using the same protocol (AH or ESP). It is generated by one node and sent to its peer for it to include in the AH or ESP headers.

NOTE

See Figures 3.12 and 3.15 for an illustration of the SPI's relationship to the AH and ESP headers.

SAs are stored and tracked in a mechanism called the Security Association Database (SAD). They are indexed based on the SA triple-identifier of destination IP address, security protocol (AH or ESP), and SPI, and include the information identified in Table 8.2.

Table 8.2 Security Association Database (SAD) Fields

SAD Field	Description
Sequence Number Counter	32-bit value used to generate the Sequence Number field in AH or ESP headers
Sequence Number Overflow	Flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent transmission of additional packets on the SA
Anti-Replay Window	32-bit counter and a bit-map used to determine whether an inbound AH or ESP packet is a replay. This is used only for inbound traffic, and only if anti-replay is enabled.
AH Authentication Information	Specifies that authentication algorithm, keys, and other parameters that AH uses
ESP Authentication Information	Specifies that authentication algorithm, keys, and other parameters that ESP uses if the authentication service of ESP is selected
ESP Encryption Information	Specifies that encryption algorithm, keys, and other parameters that ESP uses, if the encryption service of ESP is selected
SA Lifetime	Specifies the time interval after which an SA must be replaced with a new one or terminated. The interval can be expressed as a time interval, a byte count, or both.
IPSec Protocol Mode	The IPSec mode of operation (transport or tunnel) that has been applied to the traffic
Path MTU	Any observed path MTU and aging variables

Because an individual SA is linked to one of the IPSec security protocols (such as AH or ESP), a single SA cannot be used in situations where traffic requires the services of more than one security protocol. In those situations, more than one SA is required to implement the required services. Called an *SA bundle*, this grouping of one or more SAs can include SAs that terminate at different endpoints (for example, at a security gateway and a host behind the gateway). There are two basic types of bundles:

- **Transport Adjacency** A situation where one or more protocols are applied to the same IP packet without the use of tunneling.
- **Iterated Tunneling** A situation where multiple layers of security are applied to an IP packet through nested tunneling.

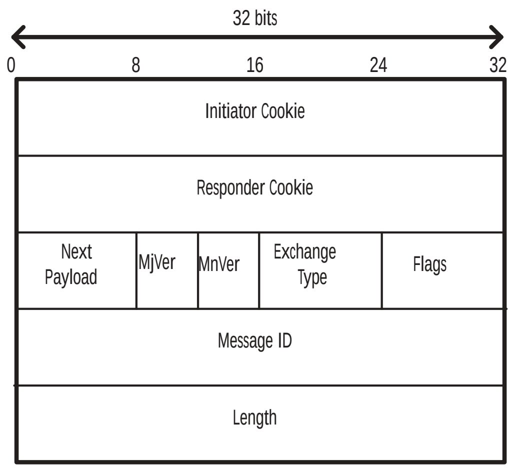
Internet Key Exchange

Internet Key Exchange (IKE) is the mechanism that determines which services should be applied to the different traffic flows and negotiates the required cryptographic keys for those services. It is the glue that binds the IPSec building blocks together. Although you can configure keys manually and statically, this is not a scalable or a secure solution. Manual key management can be practical in a small deployment, but presents significant management problems if the deployment expands. In addition, because the keys are static (unless you manually change them), as time passes and the keys are used to encrypt/decrypt more traffic, your security risk increases because the probability of the keys being cracked increases. IKE provides an automated and scalable solution to key management that can ensure that the keys are continually refreshed.

IKE is a hybrid protocol that integrates ISAKMP with subsets of the Oakley key determination protocol and the SKEME key exchange mechanism. ISAKMP is a framework for authentication and key exchange, but does not define the actual mechanisms. Oakley defines a series of key exchanges that are called modes, and SKEME describes another versatile key exchange technique. IKE uses parts of the Oakley and SKEME protocols within the ISAKMP framework by using the ISAKMP packet format, shown in Figure 8.3. IKE uses the packet format and defined ISAKMP exchanges and payloads to negotiate SAs, authentication peers, and exchange keys. It accomplishes this using the following phases and modes:

- **Phase 1** Establishes an ISAKMP SA that provides a secure mechanism for negotiating the actual SA that will be used for the IPSec services (AH and ESP). Two modes can be used in Phase 1: main mode and aggressive mode. Both modes accomplish the goal of establishing an ISAKMP SA; however, there are differences in the amount of security protection that is provided, and the quickness with which the ISAKMP SA is established.
- **Phase 2** Establishes the pair of IPSec SAs that will be used to select and apply the appropriate IPSec services (AH and ESP). This phase is accomplished using the protection services provided by the ISAKMP SA, and uses only one mode (quick mode) to negotiate the IPSec SA.

Figure 8.3 ISAKMP Header Format



Initiator Cookie	8 octet cookie of entity that initiated the SA establishment, notification, or deletion.
Responder Cookie	8 octet cookie of entity that is responding to an SA establishment request, notification, or deletion.
Next Payload	8 bits that indicates the type of the first payload in the message.
MjVer	4 bits that indicates the major version of the ISAKMP protocol in use (currently 1).
MnVer	4 bits that indicates the minor version of the ISAKMP protocol in use (currently 0).
Exchange Type	8 bits that indicates the type of exchange being used, and dictates the message and payload orderings in the ISAKMP exchange.
Flags	8 bits that indicate the specific options that are used for the ISAKMP exchange, including encryption and authentication.
Message ID	4 octet unique message identifier that is used to identify protocol state during phase 2 negotiations.
Length	4 octet value that identifies the length of the total message including the header.

The ISAKMP packet format includes cookies generated by both the initiator and the responder that together identify the applicable SA. The packets also include an exchange type that identifies the type of exchange that is occurring between the two nodes. This exchange type determines both the series of ISAKMP messages that will be exchanged and the payload types that will be included in the packets. For this discussion, you need to be aware of the exchange types identified in Table 8.3.

Table 8.3 ISAKMP Exchanges Used by IKE

ISAKMP Exchange Type	ISAKMP Value	IKE Phase and Mode
Identity Protection	2	Phase 1 Main Mode
Aggressive	4	Phase 1 Aggressive Mode

Depending on what the two IPSec nodes are attempting to accomplish, an ISAKMP packet will include one or more payloads. For this discussion, you need to be aware of the ISAKMP payload types identified in Table 8.4.

Table 8.4 ISAKMP Payload Types Used by IKE

ISAKMP Payload Type	ISAKMP Value	Description
Security Association	1	Used to negotiate security attributes and to indicate the Domain of Interpretation (DOI).
Key Exchange	4	Used to exchange data that will be used to generate a session key. The interpretation of the data depends on the DOI and supports a variety of key exchange techniques.
Identification	5	Used to exchange information that identifies the communicating IPSec nodes.
Certificate	6	Used to exchange certificates or related information if certificates are used and another distribution mechanism (for example, a directory service) is not available.
Hash	8	Contains data generated by the hash function over some part of the message and/or ISAKMP state. This payload may be used to verify the integrity of the data in an ISAKMP message or for the authentication of the negotiating entities.

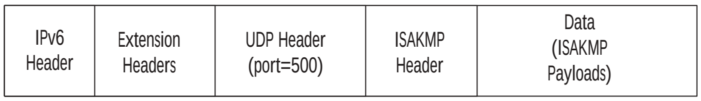
Continued

Table 8.4 Continued

ISAKMP Payload Type	ISAKMP Value	Description
Signature	9	The Signature Payload contains data generated by the digital signature function, over some part of the message and/or ISAKMP state. This payload is used to verify the integrity of the data in the ISAKMP message and may be of use for non-repudiation services.
Nonce	10	The Nonce Payload contains random data used to guarantee liveness during an exchange and protect against replay attacks.

Although ISAKMP may be implemented over any transport protocol, the Internet Assigned Numbers Authority (IANA) has reserved UDP port 500 for use by ISAKMP. Putting it all together, Figure 8.4 illustrates an ISAKMP datagram within an IPv6 packet. It includes an IPv6 header, any extension headers, a UDP header that indicates port 500, an ISAKMP header, and one or more ISAKMP payloads, as defined previously in Figure 8.3.

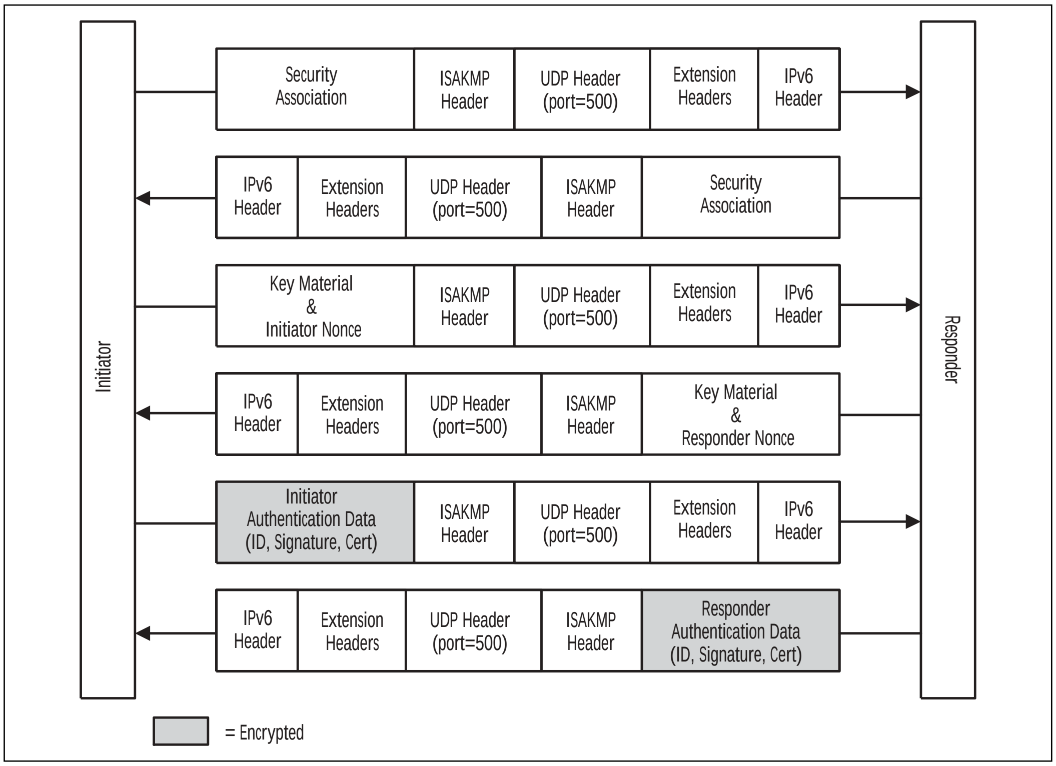
Figure 8.4 Internet Key Exchange (IKE) Packet



IKE Phase 1 Main Mode

IKE Phase 1 Main Mode is used to establish the first phase ISAKMP SA, which will provide protection for the negotiation of IPSec SAs and is illustrated in Figure 8.5. It consists of three exchanges between the initiator and the responder for a total of six exchanged packets. The first two packets that are exchanged negotiate policy with respect to basic algorithms and hashes. The next two packets are used to exchange key material that will be used to calculate a session key and nonces that will be used to perform authentication. The final two packets are used to perform mutual authentication using the negotiated mechanism. The packets include relevant identification and authentication data that are encrypted using the session key calculated from the second exchange, as you can see in Figure 8.5. Because this data is encrypted, main mode provides what is called *identity protection*. This means that any eavesdropper that may have been capturing this exchange would not be able to view the ISAKMP payload to determine the identities of the communicating parties. This is particularly effective when the communicating IPSec nodes are gateways that may be negotiating SAs for multiple clients that reside behind them.

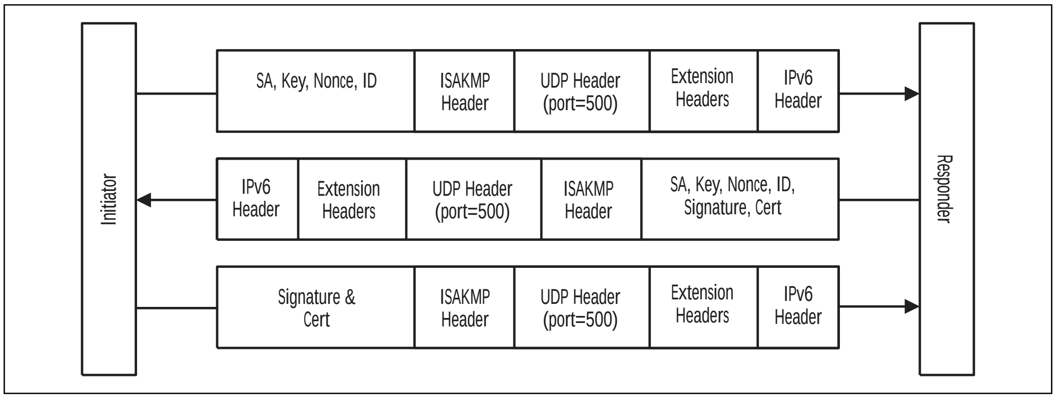
Figure 8.5 IKE Main Mode Exchange


Initiator
Responder
Security Association
ISAKMP Header
UDP Header (port=500)
Extension Headers
IPv6 Header
Key Material & Initiator Nonce
Key Material & Responder Nonce
Initiator Authentication Data (ID, Signature, Cert)
Responder Authentication Data (ID, Signature, Cert)
= Encrypted

IKE Phase 1 Aggressive Mode

IKE Phase 1 Aggressive Mode, illustrated in Figure 8.6, can also be used to establish the first phase ISAKMP SA that will provide protection for the negotiation of IPSec SAs; however, it uses a total of three packets instead of six. With aggressive mode, the initiator includes as much information as it can in the first packet, including its SA proposals, key material, nonce, and ID. The responder sends back all the information that is necessary to complete the exchange, including the selected SA, its own key material, its own nonce, its own ID, and any required authentication data. With the last packet, the initiator confirms that exchange by including its own authentication data. As you can see, aggressive mode accomplishes the ISAKMP SA negotiation, authentication, and key exchange much quicker than main mode, which can be important if the IPSec nodes are under a heavy load. However, unlike main mode, notice that none of the payloads of the packets in the aggressive mode are encrypted. This means that the identities are communicated in the clear and any eavesdropper will be able to see who the communicating end parties are. Aggressive mode does not provide identity protection.

Figure 8.6 IKE Aggressive Mode Exchange



The choice of using main mode or aggressive mode will depend on your requirements, but essentially comes down to a choice of security (identity protection) versus performance (speed).

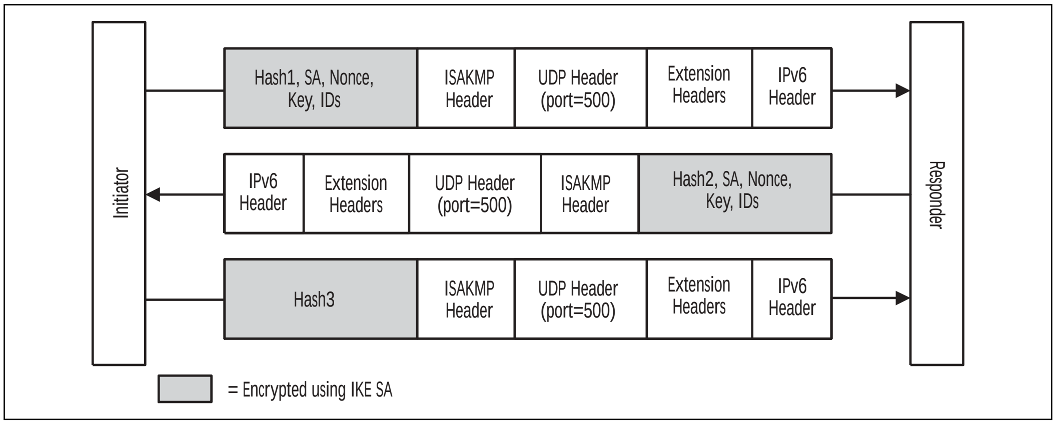
IKE Phase 2 Quick Mode

After negotiating and establishing the ISAKMP SA, the two IPSec nodes move on to Phase 2 to negotiate IPSec SAs that will be used to apply the IPSec cryptographic services to the appropriate traffic. There is only one mode for the negotiation of IPSec SAs: *quick mode*. As illustrated in Figure 8.7, quick mode, like aggressive mode, consists of a 3-packet exchange. Unlike aggressive mode, the payloads of all the packets of a quick mode exchange are encrypted using the ISAKMP SA established in Phase 1.

NOTE

The payloads of quick mode packets are encrypted using the ISAKMP SA established in Phase 1 regardless of the mode (main mode or aggressive mode) that was used in Phase 1.

Figure 8.7 IKE Phase 2 Quick Mode Exchange



As mentioned earlier, an SA is unidirectional. Because a pair of SAs is required for communication between a pair of nodes, the result of a successful quick mode exchange is two SAs (one SA is used for outbound traffic and the other SA is used for inbound traffic). In addition, quick mode is used to generate fresh keying material for an SA (ISAKMP or IPSec) that is about to expire. The key refresh can be accomplished in two ways—with or without *perfect forward secrecy* (PFS). Without PFS, the keys are refreshed using the existing keying material, but with additional hashing; whereas with PFS, a new Diffie-Hellman exchange is performed to generate brand new key material. Also, without PFS, if one session key is compromised, subsequent session keys that are based on the same key material can be compromised as well; PFS ensures that this does not occur, because the new session keys are generated from completely new key material.

After a pair of IPSec SAs have been negotiated between two nodes, the appropriate IPSec services (AH and ESP) can be applied.

Applying Perimeter Security

Access lists provide packet-filtering capabilities for Cisco routers by filtering network traffic to determine whether routed packets can be forwarded or should be blocked at the router interfaces. They can be used to control access to services on the router itself and to filter traffic passing through the router. They are an important component in the secure configuration of a router. Access lists contain one or more rules and are applied to an interface on the router. In addition, you can configure access lists to filter inbound traffic, outbound traffic, or both. The two basic types of access lists are standard and extended. *Standard access lists* can filter based on only the source IP address within a packet. *Extended access lists* can filter based on other packet parameters, including source and destination IP address, source and destination ports, and protocol. In addition, extended access lists support logging.

Cisco supports only standard access lists for IPv6 in the current release of IOS (12.2), which represents Phase 1 of their roadmap to support IPv6. Extended access lists will be supported in Phase 2 of their IPv6 roadmap. To configure standard access lists of IPv6, you need to perform the following tasks:

1. Define the IPv6 standard access list.
2. Apply the IPv6 standard access list to an interface.
3. Verify the IPv6 standard access list configuration.

Table 8.5 identifies and describes the commands you can use to accomplish these tasks.

Table 8.5 IPv6 Standard Access List Commands

Task	IOS Command	Description
1	**ipv6 access-list** *access-list-name* {**permit** | **deny**} { *source-ipv6-prefix/ prefix-length* | **any**} { *destination-ipv6-prefix/ prefix-length* | **any**} [**priority** *value*]	Defines a standard IPv6 access list and sets deny or permit conditions for the access list. The *access-list-name* argument specifies the name of the IPv6 access list. Access list names cannot contain a space or quotation mark. The **deny** keyword specifies deny conditions for the access list. The **permit** keyword specifies permit conditions for the access list. The *source-ipv6-prefix/prefix-length* and *destination-ipv6-prefix/prefix-length* arguments are mandatory. The *prefix-length* argument indicates the number of consecutive, most-significant bits that are used in the match. A slash mark must precede the decimal value. The **any** keyword, when specified instead of the *source-ipv6-prefix/prefix-length* or *destination-ipv6-prefix/prefix-length*, matches any prefix and is equivalent to the IPv6 prefix ::/0.

Continued

Table 8.5 Continued

Task	IOS Command	Description
		The **priority** keyword specifies the order in which the statement is applied in the access list. The acceptable range is from 1 to 4294967295.
2	**interface** *interface-type interface-number*	Specifies an interface type and number and places the router in interface configuration mode.
3	**ipv6 traffic-filter** *access-list-name* {**in** | **out**}	Applies the specified IPv6 access list to the interface specified in the previous step. The **in** keyword filters incoming IPv6 traffic on the specified interface. The **out** keyword filters outgoing IPv6 traffic on the specified interface.
4	**show ipv6 access-list**	Verifies that the IPv6 standard access lists are configured correctly.

The following example configures the access list named list2 and applies the access list to outbound traffic on Ethernet interface 0. Specifically, the first access list entry keeps all packets from the network fec0:0:0:2::/64 (packets that have the site-local prefix fec0:0:0:2 as the first 64 bits of their source IPv6 address) from exiting Ethernet interface 0. The second entry in the access list permits all other traffic to exit Ethernet interface 0. The second entry is necessary because an implicit deny all condition is at the end of each IPv6 access list.

```
ipv6 access-list list2 deny fec0:0:0:2::/64 any
ipv6 access-list list2 permit any any
interface ethernet 0
ipv6 traffic-filter dublin out
```

Summary

Through the inclusion of IP Security (IPSec) extension headers, security features have been included in IPv6 that provide cryptographic security services at the network layer.

Within IPv6, IPSec services are extension headers that can be included within an IPv6 header. In either case, the IPSec services consist of two security protocols: the Authentication Header (AH) and the Encapsulating Security Payload (ESP). AH is an extension header and protocol that uses a cryptographic signature to provide both connectionless integrity and data origin authentication. ESP is an extension header and protocol that can provide confidentiality, data origin authentication, connectionless integrity, replay protection, and limited traffic flow confidentiality.

Although the AH and ESP protocols provide the actual cryptographic services at the network layer, you still need mechanisms that will determine which services should be applied to the different traffic flows, and to negotiate the required cryptographic keys for those services.

These tasks are accomplished through a combination of mechanisms, including the Security Policy Database (SPD), the Security Association (SA), and the Internet Key Exchange (IKE) protocol. The SPD identifies the services to be applied to IP packets, and is consulted in the processing of all traffic (inbound and outbound), including non-IPSec traffic. It is an ordered list of policy entries that are identified by "selectors," which are roughly equivalent to the types of information used by a traffic-filtering device. For any packet, the SPD will identify one of three options for processing: discard, bypass IPSec, or apply IPSec.

The SA is a mechanism that IPSec uses to keep track of the details of a negotiated IPSec session between two nodes. It solves the problem of tracking the IKE agreements with respect to services, algorithms, and parameters for particular traffic flows.

Internet Key Exchange (IKE) is the mechanism that determines which services should be applied to the different traffic flows, and to negotiate the required cryptographic keys for those services. It is the glue the binds the IPSec building blocks together. IKE has two phases. Phase 1 is used to establish a secure channel (ISAKAMP SA) through which IPSec cryptographic services and algorithms can be negotiated. Phase 2 is the actual negotiation of the IPSec cryptographic services and algorithms through the secure channel established in Phase 1.

Solutions Fast Track

IPSec Overview

☑ Security risks specific to IP include data theft, data tampering, and peer impersonation.

☑ Security features included in IPv6 via IP Security (IPSec) extension headers provide authentication, integrity, confidentiality, and access control.

☑ IPSec is a standard defined by a suite of Internet Engineering Task Force (IETF) Requests for Comments (RFCs) that can be implemented in both IPv4 and IPv6.

Understanding the Building Blocks of IPSec

☑ IPSec services are extension headers that can be included within an IPv6 header. The services consist of two security protocols: the Authentication Header (AH) and the Encapsulating Security Payload (ESP).

☑ AH is an extension header and protocol that uses a cryptographic signature to provide both connectionless integrity and data origin authentication. You should use AH when it's important to be certain that you are communicating with correct node and when you want to ensure that data is not modified in transit.

☑ ESP is an extension header and protocol that can provide confidentiality, data origin authentication, connectionless integrity, replay protection, and limited traffic flow confidentiality.

☑ Both AH and ESP support two modes of use: transport mode and tunnel mode. In transport mode, the selected protocol (AH or ESP) provides protection primarily for upper layer protocols, and its extension header is created and inserted as part of the original IPv6 header. However, in tunnel mode, the extension header is created and inserted as part of a new IPv6 header that encapsulates the entire original packet as the payload.

☑ Because IPSec is intended to provide network layer security services, it can be implemented on end hosts and devices.

☑ Within IPsec there are two primary methods of authenticating peers in which authentication information is exchanged between the peers out of band: pre-shared keys and certificates. With pre-shared keys, two or more peers securely exchange (pre-share) a secret key prior to any IPSec negotiation—the process is simple and inexpensive but does not scale well. Certificates scale well but are typically more complex to implement.

Combining IPSec's Cryptographic Mechanisms

☑ The Security Policy Database (SPD) identifies the services to be applied to IP packets and is consulted in the processing of all traffic. For any packet, the SPD will identify one of three options for processing: discard, bypass IPSec, or apply IPSec.

☑ The SA is a mechanism that IPSec uses to keep track of the details of a negotiated IPSec session between two nodes. A pair of SAs is required for communication between a pair of nodes. An SA is uniquely identified by a destination IP address, a security protocol identifier (AH or ESP), and a Security Parameter Index (SPI).

☑ Internet Key Exchange (IKE) is the mechanism that determines which services should be applied to the different traffic flows and negotiates the required cryptographic keys for those services. IKE has two phases. Phase 1 is used to establish a secure channel (ISAKAMP SA) through which IPSec cryptographic services and algorithms can be negotiated. Phase 2 is the actual negotiation of the IPSec cryptographic services and algorithms through the secure channel established in Phase 1.

☑ Within IKE Phase 1, two possible modes can be used to establish the ISAKMP SA: main mode and aggressive mode. Main Mode consists of three exchanges between the initiator and the responder for a total of six packets exchanged. Because this data is encrypted, main mode provides what is called *identity protection*. Aggressive Mode can also be used to establish the ISAKMP SA; however, it uses a total of three packets instead of six. Aggressive mode does not provide identity protection.

Applying Perimeter Security

☑ Access lists are an important component in the secure configuration of a router. They provide packet filtering capabilities for Cisco routers, controlling access to services on the router itself, and filtering traffic passing through the router.

☑ Standard access lists can filter based on only the source IP address within a packet. Extended access lists can filter based on other packet parameters, including source and destination IP address, source and destination ports, and protocol. In addition, extended access lists support logging.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: How is IPSec different in IPv6 from IPv4?

A: There are no differences or changes in IPSec from IPv4 to IPv6. The IPSec services and mechanisms, including AH, ESP, and IKE, are the same. What is different is how IPSec is referenced with the IP header. Within IPv4, IPSec services are just another "upper layer" protocol that can be specified within the Protocol field of the IPv4 header. Within IPv6, IPSec services are extension headers that can be included within an IPv6 header. In either case, the IPSec services are the same.

Q: How is IPSec used to implement IPv4 virtual private networks (VPNs)?

A: IPSec is commonly used today in IPv4 to create virtual private networks (VPNs), which provide you with the means to communicate over a public network while maintaining the privacy of the communications. IPv4 IPSec VPNs are typically created using gateways that are effectively "bumps in the wire" and utilize at least ESP to provide the necessary privacy. They may or may not utilize AH. Tunnel mode is used to encapsulate and provide services to the entire IPv4 packet.

Q: How do I configure IPSec for IPv6 in Cisco IOS?

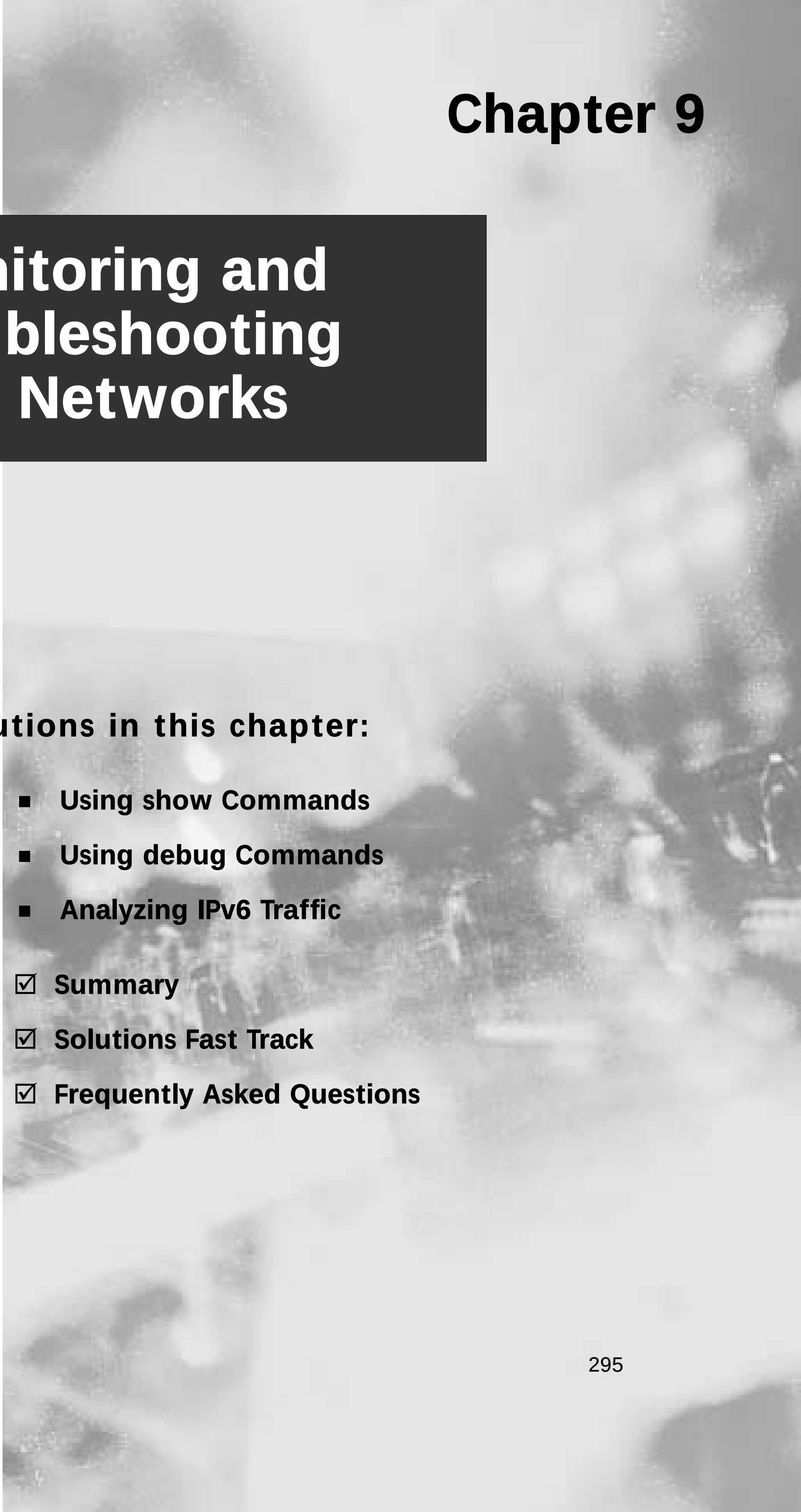
A: According to the Cisco Statement of Direction (SOD) for IPv6, IPSec will not be supported for IPv6 until Phase 3, which is scheduled for later in 2002. Within IPv4, you configure IPSec using access lists that specify the traffic to which IPSec services should be applied. These crypto access lists are then bundled with relevant IPSec parameters through something called a crypto map, which is then applied to the appropriate interface. For a complete description of how to configure IPSec for IPv4, visit the following link: www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/fipsenc/index.htm.

Q: Should I use tunnel mode or transport mode?

A: The use of tunnel mode or transport mode depends on a few factors, the first of which is whether either of the nodes involved is acting as a security gateway. If so, the IPSec RFCs require that tunnel mode be used. In addition, when you consider that gateways are typically used to create a VPN between multiple hosts, this makes sense. If one of the nodes is not acting as a security gateway, then you must identify the threat(s) that you are attempting to protect against in light of the protections offered by tunnel mode and transport mode. With tunnel mode, the IPSec services provide protection to the entire IP header. With transport mode, the IPSec services provide protection primarily to the upper layer protocols.

Q: Should I use pre-shared keys or certificates?

A: Pre-shared keys provide a quick and easy way to implement authentication and can be leveraged for a relatively small and simple IPSec implementation. While more complex, the use of certificates for authentication provides a more scalable solution for a larger or more complex IPSec implementation. If your IPSec implementation will consist of a small number of nodes in a star configuration (where a central node communicates to the rest of the nodes), then the use of pre-shared keys will help you jumpstart the implementation. If your IPSec implementation will consist of more than a small number of nodes, or will have a high degree of meshing (where many of the nodes communicate among themselves), then the use of certificates will provide a more scalable and manageable solution.



Chapter 9

Monitoring and Troubleshooting IPv6 Networks

Solutions in this chapter:

- **Using show Commands**
- **Using debug Commands**
- **Analyzing IPv6 Traffic**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

Introduction

A good network administrator is one who can effectively troubleshoot and diagnose network problems. One key skill for a network administrator is the ability to use the tools necessary to diagnose problems on the Cisco IOS. These tools consist of two main commands, *show* and *debug*. The information that can be ascertained using these commands can be used to help troubleshoot nearly all network issues. This chapter will list and explain the various *show* and *debug* commands necessary to troubleshoot an Internet Protocol Version 6 (IPv6) network running Cisco routers.

Another key skill of a network administrator is the ability to analyze protocol traffic traversing the network. Having a solid understanding of the protocol operations enables the administrator to discern the necessary information used to solve any network issue. This chapter will discuss IPv6 traffic and analyze specific traffic patterns. This information can be used as a template to troubleshoot IPv6 network problems.

Using show Commands

Cisco IOS 12.0(21)ST and 12.2(2)T introduced many new *show* commands for revealing information about the configuration of IPv6 in your Cisco network. This section will detail the new *show* commands used with the IPv6 protocol for gathering information on basic and BGP configurations.

Using Basic *show* Commands

The *show ipv6 interface* command will be your most frequently used command. It displays the usability status of interfaces configured for IPv6 and you can run it in the privileged EXEC mode or user EXEC mode. This command is the first step in determining the current status of the interface; you can often use it to determine how to proceed with troubleshooting interface/communication problems. You typically use this command when you are unable to 'ping' a remote interface or are experiencing some other connectivity related issue. It provides you with the first look at whether the problem may be related to the Layer 2 or Layer 3 portion of the IPv6 connection, which will be explained in more detail later in this chapter. The full command syntax is as follows:

```
show ipv6 interface [brief] [interface-type interface-number].
```

```
6Router-1#show ipv6 interface serial0
Serial0 is up, line protocol is up
     # denotes the status of the interface
  IPv6 is enabled, link-local address is FE80::2E0:B0FF:FE5A:D998
     # displays the status of the IPv6 on the interface and the
     # link--local address assigned
  Global unicast address(es):
    2001::1000:1000:1, subnet is 2001::/64  Joined group address(es):
     # shows the multicast groups this interface belongs to
    FF02::1
    FF02::2
    FF02::1:FF00:1
    FF02::1:FF5A:D998
  MTU is 1500 bytes
  ICMP error messages limited to one every 500 milliseconds
     # frequency of ICMP messages
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
     # neighbor discovery status
  ND reachable time is 30000 milliseconds
  Hosts use stateless autoconfig for addresses.
```

The first and most important line to review when trying to determine an IPv6 connectivity problem is the status of the interface. The interface can be in one of three states commonly referred to as up/up, up/down, or down/down, denoting the status of the Layer 2 and Layer 3 connection respectively. An up/up interface indicates that you are communicating with the interface hardware and/or the local CSU/DSU and that the Layer 3 communication, in this case IPv6, is communicating with the same Layer 3 protocol on the other side of the link, the neighbor. An interface displaying an up/down status, also referred to as line protocol down, indicates that the local device is not communicating with the neighbor across the configured Layer 3 protocol (IPv6). This problem could be the result of an incorrect IP address or subnet mask on the local or remote device, a disabled neighbor interface, a failed local or remote CSU/DSU, equipment timing, or keepalive problem. An interface displaying down/down, also referred to as interface down, indicates faulty local hardware or cabling. In summary, this output helps determine if you should be chasing a physical or logical problem. Logical problems are usually indicated with up/down and physical problems are often indicated with down/down. An interface can also be administratively down, which indicates that an administrator has disabled the interfaces.

The IPv6 interface status is derived through the use of duplicate address detection (DAD). DAD is the process of how a node determines that an address it wishes to use is not already in use by another node. If duplicate address detection has identified the link-local address of the interface as being a duplicate address, the processing of IPv6 packets is disabled on the interface and the interface is marked "stalled." If IPv6 is not enabled, the interface is marked "disabled." During the duplicate address detection process, the interface may also display DUPLICATE, TENTATIVE, or OK. The TENTATIVE status informs you that the duplicate address detection process is in progress. The other states are self-explanatory. The assigned link-local address is used for neighbor discovery and the stateless auto-configuration process and its prefix is FE80 with the remainder of the address derived from the interface identifier in modified EUI-64 format.

The joined group addresses list the multicast groups to which this interface belongs. The ICMP error messages line indicates ICMP messages are periodically sent every 500 milliseconds (default) and the rate can be modified using the *ipv6 icmp error-interval* command, which can ultimately reduce link-layer congestion.

ND DAD indicates that the Neighbor Discovery Duplicate Address Detection is enabled. The number of DAD attempts indicates the number of Neighbor Solicitation messages that were sent while the duplicate address detection process was being performed.

You can use the *show ipv6 interface brief* command to display a summary of the status of each IPv6-configured interface on the router. The command displays only the interface name, its status [up/down], and the assigned IPv6 address, just as it did for the IPv4 version of this command.

You use the *show ipv6 route* command to display the routing table and to determine the next hop corresponding to the connected network. You execute the command from the User EXEC or Privileged EXEC modes. You should use this command to troubleshoot a communication problem with a destination device. *Ping ipv6* and *traceroute ipv6* are great commands to use in addition to the *show ipv6 route* command to pinpoint the source of the traffic loss. You should use the *ping ipv6 [destination address]* command first during troubleshooting to determine if the destination device is accessible. If the ping fails, then use *traceroute ipv6 [destination address]* to determine where in the path to the destination device the connection problem may reside. The traceroute output will give you the last hop that answered the ICMP ping and should be your next focus of attention in troubleshooting the connectivity problem. The full command syntax for *show ipv6 route* command is as follows:

```
show ipv6 route [ipv6-address | ipv6-prefix/prefix-length | protocol].

6Router-1
     #show ipv6 route
IPv6 Routing Table - 9 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
     # legend of possible protocol types that may be displayed below
Timers: Uptime/Expires

L   2000:1:1::1/128 [0/0]
     via ::, Serial0, 20:45:43/never
     # neighboring network information
C   2000:1:1::/64 [0/0]
     via ::, Serial0, 20:45:46/never
L   2000:1:2::1/128 [0/0]
     via ::, Ethernet0, 20:46:12/never
C   2000:1:2::/64 [0/0]
     via ::, Ethernet0, 20:46:13/never
B   2000:1:3::/64 [20/1]
     via 2000:1:1::2, Serial0, 20:45:37/never
L   3FFE:4200:1:1::1/128 [0/0]
     via ::, Ethernet0, 1d19h/never
C   3FFE:4200:1:1::/64 [0/0]
     via ::, Ethernet0, 1d19h/never
L   FE80::/10 [0/0]
     via ::, Null0, 1w5d/never
L   FF00::/8 [0/0]
     via ::, Null0, 1w5d/never
```

The output of this command displays the routing protocol (if any) that the route is using, the prefix of the remote network [2000:1:1::1/128], the administrative distance and metric for the link [0/0], as well as the interface to forward packets through [Serial0]. The output also indicates the last time the route was updated and when the route expires [20:45:43/never] (local and connected routes never expire).

The *show ipv6 route* command enables you to specify the IPv6 address/network and prefix and/or the protocol type to enable more granularity in the output. The full command syntax is listed here:

```
show ipv6 route [ipv6-address | ipv6-prefix/prefix-length | protocol]
```

You can use the *show ipv6 route summary* command to display the number of routes per route source and each prefix length. This command lists the total number of entries in the IPv6 routing table and provides a quick look at the total number of locally connected, directly connected, statically mapped, and dynamically derived networks as well as a summary of the total number of routing table entries per given prefix length.

```
6Router-1#show ipv6 route summary
IPv6 Routing Table Summary - 9 entries
     # total routing table entries
  5 local, 3 connected, 0 static, 0 RIP, 1 BGP 0 IS-IS
     # route source
  Number of prefixes:
    /8: 1, /10: 1, /64: 4, /128: 3
     # number of accessible networks by prefix
```

The *show ipv6 neighbors* command displays the contents of the neighbor discovery cache constructed through the exchange of Router Solicitation/Advertisement, Neighbor Solicitation/Advertisement, and Redirect ICMP messages. The command can be executed from the Privileged EXEC or User EXEC modes. This command is useful in determining which if any neighbors are inaccessible, and, if accessible, the last time the neighbor was contacted. This command is useful in troubleshooting connectivity with neighboring devices. The full command syntax is as follows:

```
show ipv6 neighbors [interface-type interface-number | ipv6-address]
```

```
6Router-1#show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2000:1:2::10                                - 0000.1234.5678  REACH Ethernet0
     # list of each IPv6 neighbor
```

The command output displays the neighbor's IPv6 address [2000:1:2::10], the last time the neighbor was confirmed to be reachable (a hyphen (-) indicates a static entry), the link-layer [MAC] address of the neighbor, the state of the neighbor cache entry and the interface the neighbor can be reached through. Table 9.1 lists the various neighbor communication states. The listed states apply only to non-static neighbor cache entries.

Table 9.1 Neighbor Cache Entry States

Neighbor Cache Entry	Definition
INCMP – Incomplete	Neighbor resolution has not been completed. The Neighbor Solicitation ICMP message has been sent but the Neighbor Advertisement message has not yet been received.
REACH – Reachable	The neighbor has been confirmed as reachable within the last ReachableTime (default 30000 ms). The ReachableTIme is displayed in the show IPv6 interface output.
STALE	The neighbor has not been successfully contacted within the ReachableTime setting. No action is taken until a packet is sent.
DELAY	The DELAY state follows the STALE state and indicates a packet was sent within the last DELAY_FIRST_PROBE_TIME. If a confirmation is not received, the state will change to the PROBE state and send Neighbor Solicitation message.
PROBE	Neighbor Solicitation messages will continue to be sent at an interval specified by the neighbor discovery-related variable RetransTimer (RFC 2461, Neighbor Discovery for IP Version 6 [IPv6]), until reachability is confirmed. The RetransTimer interval is specified in milliseconds.
????	The neighbor is in an unknown state.

Use the *show ipv6 protocols* command to display the IPv6 protocols configured on the router. This command displays an at-a-glance summary providing information such as the configured route redistribution, the IPv6 neighbors, and the routing protocol configured on each interface. The command is useful for getting a snapshot of the routing configuration and can be used as a roadmap for determining the additional commands that would be useful in troubleshooting connectivity or routing problems. For example, the output from this command tells you there are directly connected, statically mapped, and BGP-derived networks accessible from this router and that route redistribution has been configured. It also tells you that the RIP routing protocol is configured on the Serial0 and Ethernet0 interfaces and that RIP is being redistributed.

```
6Router-1#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "bgp 64999"
     # the BGP network configured on this router
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                  FiltIn FiltOut Weight RoutemapIn RoutemapOut
    2000:1:1::2
IPv6 Routing Protocol is "rip cisco"
     # RIP is configured on the Serial0 and Ehternet0 interfaces
  Interfaces:
    Serial0
    Ethernet0
  Redistribution:
    Redistributing protocol rip cisco
     # RIP is being reditributed
```

The *show ipv6 protocols* command will also display if neighbor route maps or AS-filter lists have been applied to each of the interfaces. If an AS-filter list has been applied, then the neighbor weight value will also be displayed. The FiltIn, FiltOut, and Weight columns are applicable only to BGP-configured interfaces.

You can also add the *summary* keyword at the end of the command to display each configured protocol, as shown here:

```
6Router-1#show ipv6 protocols summary
Index Process Name
0      connected
1      static
5      bgp 64999
6      rip cisco
```

The *show ipv6 traffic* command provides statistics for IPv6, ICMP, and UDP packets that have been received by or originated from the IPv6-configured router. This command is useful for determining if the IPv6 packets contain errors. In addition, you can determine the overhead incurred from ICMP traffic and whether any ICMPv6 parameters should be tuned to decrease overhead.

```
6Router-1#show ipv6 traffic
IPv6 statistics:
  Rcvd:  17489 total, 14367 local destination
     # total number of IPv6 packets received by this router
         0 format errors, 0 hop count exceeded
         0 bad header, 0 unknown option, 0 bad source
         0 unknown protocol, 0 not a router
         0 fragments, 0 total reassembled
         0 reassembly timeouts, 0 reassembly failures
  Sent:  67630 generated, 0 forwarded
     # total number od IPv6 packets sent from this router
         0 fragmented into 0 fragments
         1 encapsulation failed, 3122 no route, 0 too big
  Mcast: 0 received, 0 sent

ICMP statistics:
  Rcvd: 61 input, 0 checksum errors, 0 too short
     # total number of IPv6 ICMP packets reeived by this router
        0 unknown info type, 0 unknown error type
        unreach: 0 routing, 0 admin, 0 neighbor, 0 address, 0 port
        parameter: 0 error, 0 header, 0 option
        0 hopcount expired, 0 reassembly timeout, 0 too big
        25 echo request, 25 echo reply
     # total number of ping request/replies to this router
        0 group query, 0 group report, 0 group reduce
        0 router solicit, 0 router advert, 0 redirects
        2 neighbor solicit, 9 neighbor advert
     # neigbor discovery statistics
  Sent: 6000 output, 0 rate-limited
     # total number of ICMP packets sent by this router
        unreach: 0 routing, 0 admin, 0 neighbor, 0 address, 0 port
```

```
         parameter: 0 error, 0 header, 0 option
         0 hopcount expired, 0 reassembly timeout,0 too big
         30 echo request, 25 echo reply
      # total number of ping request/replies from this router
         0 group query, 0 group report, 0 group reduce
         0 router solicit, 5880 router advert, 0 redirects
      # number of router advertisements sent by this router
         32 neighbor solicit, 33 neighbor advert
      # ICMP neighbor advertisements and solicitations
      # used in neighbor discovery sent by this router

UDP statistics:
  Rcvd: 9089 input, 0 checksum errors, 0 length errors
        0 no port, 0 dropped
  Sent: 56804 output
```

Using the *show bgp* Commands

Various commands that assist in viewing and troubleshooting Border Gateway Protocol (BGP) are included in the Cisco IOS versions supporting IPv6. The *show bgp* commands are useful in situations where you are connected to an ISP that requires the use of BGP or to other sites using BGP for routing decisions.

The *show bgp ipv6* command displays the BGP table version, the next hop address to reach the listed network along with the metric, a local preference (if configured), weight, and AS path. This command is a first step in determining the health and configuration of BGP on your router and can be used to determine how to proceed with troubleshooting. The full command syntax is as follows:

```
show bgp ipv6 [ipv6-prefix/prefix-length] [longer-prefixes]


6Router-1#show bgp ipv6
BGP table version is 13, local router ID is 172.16.0.1
      # the BGP table version number and IP address used as the router ID
Status codes: s suppressed, d damped, h history, * valid, > best, i - 
internal,r RIB-failure
Origin codes: i - IGP, e - EGP, ? – incomplete
      # indicates the origin of the entry
```

```
   Network           Next Hop            Metric LocPrf Weight Path
*  2000:1:1::/64    2000:1:1::2                             0 65000 i
     # the accessible BGP networks and the next hop address to reach them
*>                  ::                                  32768 I
     # the :: indicates the router has non-BGP routes to this network
*> 2000:1:2::/64    ::                                  32768 i
*> 2000:1:3::/64    2000:1:1::2                             0 65000 i
```

The *show bgp ipv6* command output contains similar information as the *show ipv6 route* command but displays only BGP routing information.

The *show bgp ipv6 summary* command provides an overview of the BGP configuration on the router. The command displays the local identifier, taken from the lowest IP address, as well as the BGP table version and the router memory used by the individual BGP entries.

```
6Router-1#show bgp ipv6 summary
BGP router identifier 172.16.0.1, local AS number 64999
     # the BGP router ID and AS assigned to this router
BGP table version is 13, main routing table version 13
3 network entries and 4 paths using 659 bytes of memory
     # memory used by the BGP routing protocol
2 BGP path attribute entries using 120 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP activity 10/41 prefixes, 14/10 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
2000:1:1::2     4 65000    4293    4300       13    0    0 1d22h        2
```

The last section of output from the *show bgp ipv6 summary* command displays the neighbor address, the BGP version number spoken to that neighbor, the autonomous system, the number of BGP messages received/sent by/from this neighbor, the BGP table version, the number of BGP messages waiting to be sent or processed by this router, as well as the length of time the BGP session has been ESTABLISHED/not ESTABLISHED. This output is useful in determining the health and reliability of the BGP session.

The *show bgp ipv6 neighbors* command is useful for determining the status of the BGP neighbor communications. You should use it when troubleshooting BGP connectivity or routing problems. This command provides BGP information on all IPv6 neighbors. You can add an address of an IPv6 BGP neighbor to this command to restrict the output to a single IPv6 BGP neighbor. You can further target the output from this command by adding any of the optional parameters contained in [ ]. The full syntax for the command is as follows:

```
show bgp ipv6 neighbors [ipv6-address] [received-routes | routes |
    flap-statistics | advertised-routes | paths regular-expression |
    dampened-routes].
```

The section of the output to pay particular attention to is the current BGP state and how long the peer connection has been established. The "BGP state = Established" line in the output tells you that the BGP neighbors have established a TCP connection and have agreed to communicate via the BGP protocol.

```
6Router-1#show bgp ipv6 neighbors
BGP neighbor is 2000:1:1::2,  remote AS 65000, external link
     # the BGP neighbors address and AS number
 Member of peer-group cisco for session parameters
     # update policy peer group this router belongs to
  BGP version 4, remote router ID 172.16.8.33
  BGP state = Established, up for 00:51:16
     # current state of the BGP session and how long the
     # underlying TCP connection has been established
  Last read 00:00:16, hold time is 180, keepalive interval is 60 seconds
     # BGP configuration settings
  Neighbor capabilities:
     # the BGP capabilities advertised and received from this neighbor
    Route refresh: advertised and received(old & new)
    Address family IPv6 Unicast: advertised and received
  Received 1528 messages, 0 notifications, 0 in queue
     # IPv6 unicast-specific properties of this neighbor

  Sent 1535 messages, 1 notifications, 0 in queue
  Default minimum time between advertisement runs is 30 seconds
 For address family: IPv6 Unicast    BGP table version 13, neighbor version
13
     # confirms router and neighbor are using the same BGP routing table
  Index 1, Offset 0, Mask 0x2
  cisco peer-group member
  Route refresh request: received 0, sent 0
  2 accepted prefixes consume 136 bytes
  Prefix advertised 11, suppressed 0, withdrawn 1

  Connections established 4; dropped 2
     # number of times the peers have agreed to speak BGP
     # and the how often a good connection has failed or been taken down
  Last reset 22:53:50, due to BGP Notification sent, hold time expired
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 2000:1:1::1, Local port: 179
     # peering address of the local router
Foreign host: 2000:1:1::2, Foreign port: 11631    # peering address of the
neighbor

Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)

Event Timers (current time is 0x3D6B4AF8):
Timer          Starts    Wakeups            Next
Retrans            58          2              0x0
TimeWait            0          0              0x0
AckHold            55         26              0x0
SendWnd             0          0              0x0
KeepAlive           0          0              0x0
GiveUp              0          0              0x0
PmtuAger            0          0              0x0
DeadWait            0          0              0x0

iss: 2268213783  snduna: 2268215016  sndnxt: 2268215016     sndwnd:  15152
irs:  840903895  rcvnxt:  840905059  rcvwnd:      15221  delrcvwnd:   1163

SRTT: 302 ms, RTTO: 323 ms, RTV: 21 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 424 ms, ACK hold: 200 ms
```

```
Flags: passive open, nagle, gen tcbs

Datagrams (max data segment is 1440 bytes):
Rcvd: 84 (out of order: 0), with data: 55, total data bytes: 1163
Sent: 82 (retransmit: 2, fastretransmit: 0), with data: 82, total data
bytes: 45
20
```

As you can see, the output from the *show bgp ipv6 neighbors* command is quite verbose. Additional study of the BGP protocol is required to make good use of the command output. If your network is running BGP and IPv6, you will use the preceding commands the most frequently in your day-to-day troubleshooting.

Using debug Commands

You use *debug* commands to assist with network troubleshooting. You should take care when executing *debug* commands, because they add overhead to the router. *Debug* commands read each complete packet rather than just the header, which can put enormous demands on the router's processor and memory. You should make *debug* commands as specific as possible to minimize the load placed on the router resources. Improperly used *debug* commands can cause the router to become inaccessible, requiring a potential local router restart (power on/off) to regain command-line access, which can be quite detrimental when support personnel cannot easily access the router. This section will discuss and display the results of various *debug* commands that can assist you with resolving IPv6 problems or verifying IPv6 traffic. This section covers general commands and BGP-related commands.

Here are some *debug* commands that you can use to help verify IPv6 traffic or troubleshoot IPv6 communication and neighbor relationship problems.

The *debug ipv6 packet* command displays information on the IPv6 packets received, generated, and forwarded on this router. Fast-switched packets do not generate messages. The *debug ipv6 packet* command will create substantial overhead on the router and should only be used when traffic levels are very low.

The output displays the source and destination address, the contents of the traffic class field, and whether the packet was forwarded to an upper-layer protocol or was originated from the router.

```
6Router-1#debug ipv6 packet
IPv6 unicast packet debugging is on
```

```
6Router-1#
1w6d: IPV6: source 2000:1:1::2 (Serial0)
     # the source address in the IPv6 header
1w6d:        dest 2000:1:1::1
     # the destination address in the IPv6 header
1w6d:        traffic class 192, flow 0x0, len 79+4, prot 6, hops 64, forward
             to ulp   #the contents of the traffic class, flow, length,
             protocol, and hops fields
1w6d: IPV6: source 2000:1:1::1 (local)
1w6d:        dest 2000:1:1::2 (Serial0)
1w6d:        traffic class 192, flow 0x0, len 60+0, prot 6, hops 64,
             originating
     #indicates this packet originated from this router
…
1w6d: IPV6: source FE80::2E0:B0FF:FE5A:D998 (local)
1w6d:        dest FF02::9 (Serial0)
1w6d:        traffic class 224, flow 0x0, len 112+1388, prot 17, hops 255,
             originating
1w6d: IPv6: Sending on Serial0
1w6d: IPV6: source FE80::2E0:B0FF:FE5A:D998 (local)
1w6d:        dest FF02::9 (Ethernet0)
1w6d:        traffic class 224, flow 0x0, len 112+1388, prot 17, hops 255,
             originating
1w6d: IPv6: Sending on Ethernet0
1w6d: IPV6: source FE80::2E0:B0FF:FE5A:D998 (local)
1w6d:        dest FF02::9 (Serial0)
1w6d:        traffic class 224, flow 0x0, len 112+1388, prot 17, hops 255,
             originating
1w6d: IPv6: Sending on Serial0
1w6d: IPV6: source FE80::2E0:B0FF:FE5A:D998 (local)
1w6d:        dest FF02::9 (Ethernet0)
1w6d:        traffic class 224, flow 0x0, len 112+1388, prot 17, hops 255,
             originating
1w6d: IPv6: Sending on Ethernet0
1w6d: IPV6: source 2000:1:1::2 (Serial0)
1w6d:        dest 2000:1:1::1
1w6d:        traffic class 192, flow 0x0, len 79+4, prot 6, hops 64, forward
             to ulp
     # indicates this was received by the router and forwarded
     # to an upper-layer protocol
1w6d: IPV6: source 2000:1:1::1 (local)
1w6d:       dest 2000:1:1::2 (Serial0)
1w6d:       traffic class 192, flow 0x0, len 60+12, prot 6, hops 64,
originating
```

The *debug ipv6 icmp* command is useful for troubleshooting ICMP communication on the router. The neighbor discovery process, MTU determination, and Multicast Listener Discovery (MLD) all use ICMP, although a separate *debug* command exists for troubleshooting the neighbor discovery process, which we will discuss next. The following trace is simply the ping command to give you an idea of the output generated from the *debug ipv6 icmp* command—notice the echo request and echo reply process used with the ping command.

```
6Router-1#debug ipv6 icmp
ICMP packet debugging is on
6Router-1#ping ipv6 2000:1:1::2


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2000:1:1::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/10/12 ms
6Router-1#
1w6d: ICMPv6: Sending echo request to 2000:1:1::2
     # indictaes ICMPv6 packet has been sent
1w6d: ICMPv6: Received ICMPv6 packet from 2000:1:1::2, type 129
      # ICMPv6 packet received, type 129 = echo reply
1w6d: ICMPv6: Received echo reply from 2000:1:1::2
```

The *debug ipv6 nd* command is useful for troubleshooting the neighbor discovery process where adjacencies are attained by passing ICMPv6 packets between routers to establish neighbor adjacencies. The following output displays a router advertisement (RA) message sent from the Ethernet0 interface to the two configured prefixes on the Ethernet0 interface:

```
6Router-1#debug ipv6 nd
ICMP Neighbor Discovery events debugging is on
6Router-1#
2w0d: ICMPv6-ND: Sending RA to FF02::1 on Ethernet0
     # indicates a router advertisement is being sent to
     # all-nodes multicast group
2w0d: ICMPv6-ND:     prefix = 3FFE:4200:1:1::1/64 onlink autoconfig
2w0d: ICMPv6-ND:     prefix = 2000:1:2::1/64 onlink autoconfig
     # indicates the type of autoconfiguration
```

The following *debug* output shows a more complete communication flow. Reviewing the flow will show you that the Neighbor Solicitation (NS) and Neighbor Advertisement (NA) are being passed between the FastEthernet0/0 interface and a neighbor at 2000:0:0:3::2. Also visible from the output is the Duplicate Address Detection (DAD) and its confirmation that the interfaces address is unique.

```
13:22:40:ICMPv6-ND:STALE -> DELAY:2000:0:0:3::2
     # indicates ND cache entry used to be reachable but is now stale,
     # reachability needs to be confirmed
13:22:45:ICMPv6-ND:DELAY -> PROBE:2000:0:0:3::2
     # indicates reachability being confirmed
13:22:45:ICMPv6-ND:Sending NS for 2000:0:0:3::2 on FastEthernet0/0
     # sending neighbor solicitation
13:22:45:ICMPv6-ND:Received NA for 2000:0:0:3::2 on FastEthernet0/0 from
2000:0:0:3::2
     # receiving neighbor advertisement confirming reachability
13:22:45:ICMPv6-ND:PROBE -> REACH:2000:0:0:3::2
     # entry flagged as reachable
13:22:45:ICMPv6-ND:Received NS for 2000:0:0:3::1 on FastEthernet0/0 from
FE80::203:A0FF:FED6:1400
     # received neighbor solicitation to determine link-local address
13:22:45:ICMPv6-ND:Sending NA for 2000:0:0:3::1 on FastEthernet0/0
     # sending neighbor advertisement in response to previous NS
13:23:15: ICMPv6-ND: Sending NS for FE80::1 on Ethernet0/1
13:23:16: ICMPv6-ND: DAD: FE80::1 is unique.
     # duplicate address detection was performed and address is unique
13:23:16: ICMPv6-ND: Sending NS for 2000::2 on Ethernet0/1
13:23:16: ICMPv6-ND: Sending NS for 3000::3 on Ethernet0/1
13:23:16: ICMPv6-ND: Sending NA for FE80::1 on Ethernet0/1
13:23:17: ICMPv6-ND: DAD: 2000::2 is unique.
13:23:53: ICMPv6-ND: Sending NA for 2000::2 on Ethernet0/1
13:23:53: ICMPv6-ND: DAD: 3000::3 is unique.
13:23:53: ICMPv6-ND: Sending NA for 3000::3 on Ethernet0/1
```

The *debug ipv6 routing* command displays *debug* messages for IPv6 routing table updates and route cache updates. The following output displays routes being added to the routing table:

```
13:18:43:IPv6RT0:Add 2000:0:0:1:1::/80 to table
     # specifies addition of the network to the routing table
13:18:43:IPv6RT0:Better next-hop for 2000:0:0:1:1::/80, [120/2]
     # indicates the entry was in the routing table but a lower
     # cost path was added
13:19:09:IPv6RT0:Add 2000:0:0:2::/64 to table
13:19:09:IPv6RT0:Better next-hop for 2000:0:0:2::/64, [20/1]
13:19:09:IPv6RT0:Add 2000:0:0:2:1::/80 to table
13:19:09:IPv6RT0:Better next-hop for 2000:0:0:2:1::/80, [20/1]
13:19:09:IPv6RT0:Add 2000:0:0:4::/64 to table
13:19:09:IPv6RT0:Better next-hop for 2000:0:0:4::/64, [20/1]
13:19:37:IPv6RT0:Add 2000:0:0:6::/64 to table
13:19:37:IPv6RT0:Better next-hop for 2000:0:0:6::/64, [20/2]
```

The *debug bgp ipv6* command enables the debugging of IPv6 BGP information. The full command syntax is as follows:

```
debug bgp ipv6 {dampening [neighbor-acl] | updates [neighbor-address |
    neighbor-acl | in | out]}.
```

The following output displays BGP keepalive messages.

```
6Router-1#debug bgp ipv6
BGP debugging is on
6Router-1#
2w0d: BGP: 2000:1:1::2 rcv message type 4, length (excl. header) 0
2w0d: BGP: 2000:1:1::2 send message type 4, length (incl. header) 19
```

Analyzing IPv6 Traffic

To analyze IPv6 will require new knowledge of the IPv6 packet functions. An important aspect of IPv6 is its Neighbor Discovery (ND) process, which uses ICMP packets to determine and maintain neighbor relationships and provides auto-configuration functionality similar to DHCP. In this section we will review the ICMPv6 header and the different message types in ICMPv6. This section will also discuss the neighbor discovery process and the ICMPv6 packet types used specifically for neighbor discovery and maintenance.

Designing & Planning…

Migrating to IPv6

As IPv4 address space is exhausted in the near future with the continued addition of devices to the network—many of which will be mobile—the migration to IPv6 will be absolutely necessary. It will also be costly and time consuming. Software upgrades will be required on every host and router to support IPv6 addressing; however, the cost of *not* migrating to IPv6 and simply continuing to patch and improve IPv4 will also prove costly to the consumer and to ISPs alike. It is estimated that the IPv4 address space will be exhausted by 2008, although the continued use of NAT and other methods may continue to prolong the life of IPv4.

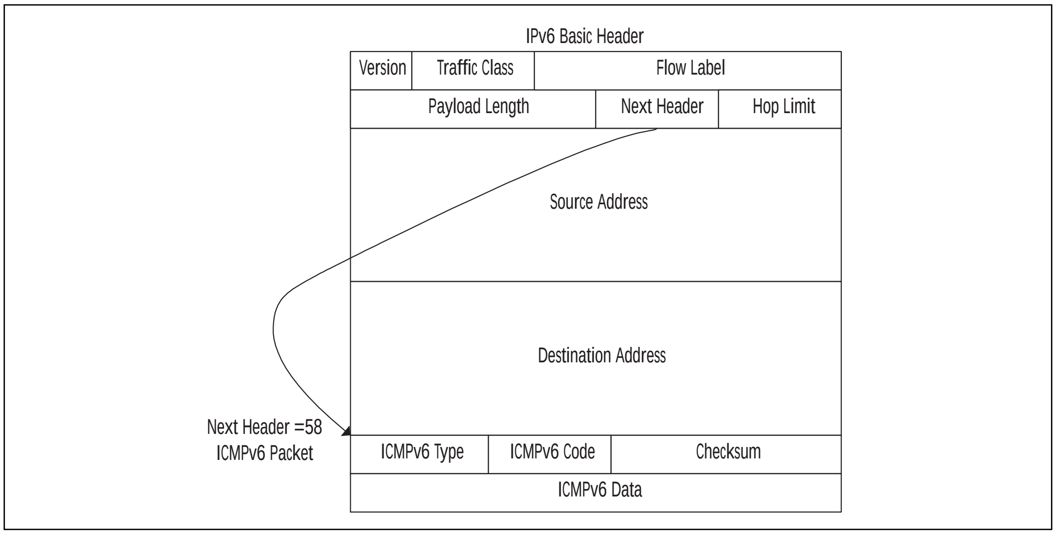
Dual-stack, *tunneling*, and *translation* methods have been engineered to assist with the migration to IPv6. The dual-stack method completely supports IPv6 by having both an IPv4 address and an IPv6 address assigned so that each protocol talks natively to its destination. Manual and automatic tunneling methods include tunnel broker, 6over4, and 6to4, providing Web-based tunnel set up, host IPv6 over IPv4 communication without explicit tunnels, and IPv6 domain communication over IPv4 using a unique IPv6 prefix to denote 6to4 communication. Translation methods provide other migration alternatives, which include stateless IP/ICMP translator, NAT-PT, Socks64, and Bump-in-the-Stack. The use of each of these methods will depend on your requirements and will require additional study and research to implement.

What's New in Internet Control Message Protocol for IPv6

Internet Control Message Protocol (ICMP) generates error and informational messages, as stated in RFC 792, in IPv6 just as it did in IPv4. ICMP is an important troubleshooting protocol for providing reachability testing and flow control at the IP layer. Ping uses ICMP echoes and replies to confirm reachability and is the most widely used ICMP-based utility. In the IPv6 protocol, ICMP packets are used in the neighbor discovery process, path Maximum Transmission Unit (MTU) discovery, and the Multicast Listener Discovery (MLD) protocol. MLD discovers multicast listeners, which are nodes that want to receive multicast packets destined for specific multicast addresses, on directly attached links on IPv6-enabled routers. MLD is based on Internet Group Management Protocol version 2 (IGMP) for IPv4.

An IPv6 ICMP packet is identified by a value of 58 in the Next Header field of the basic IPv6 packet header. Within IPv6 ICMP packets, the ICMPv6 Type and ICMPv6 Code fields identify IPv6 ICMP packet specifics, such as the ICMP message type. The value in the Checksum field is derived from the fields in the IPv6 ICMP packet and the IPv6 pseudo-header. The ICMPv6 Data field contains error or diagnostic information relevant to IP packet processing. Figure 9.1 shows the IPv6 ICMP packet header format.

Figure 9.1 ICMPv6 Packet Header Format



The ICMPv6 messages are displayed in Table 9.2. The majority of the error and informational messages are the same as ICMPv4 messages.

Table 9.2 ICMP Error Message Types

Message Type	Message Cause	Included Sub-Messages	New in IPv6
Error Message	Destination Unreachable	No route Administratively prohibited Address unreachable Port unreachable	No
	Packet too big Time exceeded Parameter problem	N/A N/A Erroneous header field Unrecognized next header type Unrecognized option	No No No
Informational Message	Echo request	N/A	No
	Echo reply	N/A	No
	Multicast listener discovery	Query Report Done	Yes (Like IGMP for IPv4)

The Neighbor Discovery Process

Neighbor Discovery (ND) is a new IPv6 protocol used by nodes (hosts and routers) to determine the Layer 2 addresses for neighbors on the same network or attached links, much like Address Resolution Protocol (ARP) in IPv4. ND is a combination of the IPv4 protocols ARP, ICMP Router Discovery (RDISC), and ICMP Redirect. The ND protocol is used to find neighboring routers that will forward packets, track the reachability status of neighbors, and detect new or changed Layer 2 addresses. You can use the neighbor discovery table to quickly determine if an alternate path exists when a path to a neighboring router fails.

The ND protocol is defined in RFC 2461 to resolve issues related to the interaction among nodes residing on the same physical link. The RFC defines all the ND interactions including router discovery, prefix discovery, address auto-configuration, duplicate address detection, neighbor unreachability detection, link-layer address resolution, next-hop determination, and redirects. Each of the ND interactions are made possible through the use of router solicitations, router advertisements, neighbor solicitations, neighbor advertisements, and redirect ICMP messages.

Router solicitations and router advertisement ICMP messages facilitate the router discovery and address auto-configuration interactions. Router discovery defines how a host locates routers that reside on an attached link; address auto-configuration defines how nodes automatically configure an address for an interface. Router solicitations and router advertisements are covered in more detail a little later in this chapter.

Neighbor solicitations and neighbor advertisements ICMP messages facilitate the prefix discovery, duplicate address detection (DAD), neighbor unreachability detection (NUD), link-layer address resolution, and next-hop determination interactions as defined in RFC 2461. Prefix discovery defines how hosts discover the set of address prefixes that define which destinations are on-link for an attached link. Prefixes are used to distinguish between destinations residing on-link from those accessible only through a router. DAD defines how a node determines that an address it wishes to use is not already in use by another node. A neighbor solicitation message is used to determine whether the IPv6 address to be assigned to the interface is unique. The status of the interface during this process is TENTATIVE and the address will be assigned to the interface if no neighbor advertisement is received in response to the neighbor solicitation or if no neighbor solicitation is received from another router attempting to use the same IPv6 address. NUD defines how nodes determine that a neighbor is no longer accessible. The default settings will determine that a neighbor is unreachable after 38 seconds. Parameters can be modified to reduce this time to as little as five seconds, although overhead is increased. Link-layer address resolution defines how nodes determine the link-layer address of a neighbor when only the destination's IP address is known. Next-hop determination defines how an IP destination address is mapped into the IP address of the neighbor to which traffic for the destination should be sent. The next-hop can be a router or the destination itself.

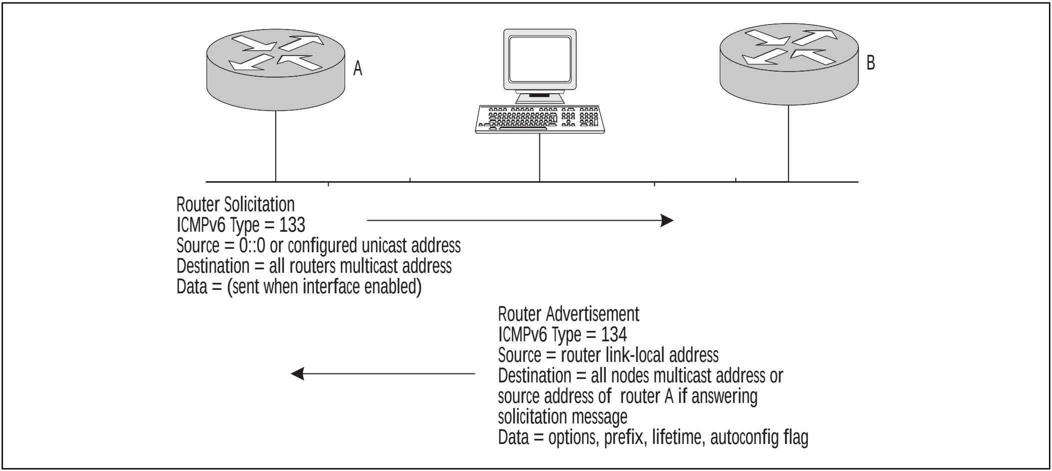
Finally, the redirect ICMP packet is used for IVMP redirects. Redirect defines how a router informs a host of a better first-hop node to reach a particular destination.

The Neighbor Discovery process, seen in Figure 9.2, uses five different ICMPv6 packet types to determine and maintain the neighbor relationships among IPv6 routers. The five packet types are router solicitation, router advertisement, neighbor solicitation, neighbor advertisement, and the redirect packet.

The router solicitation packet contains a value of 133 in the ICMP packet header and is sent to an all-routers multicast address when an IPv6 interface is enabled to request an immediate router advertisement from the neighboring routers rather than wait for their next periodic router advertisement. The router solicitation message will cause neighboring IPv6 routers to respond with a router advertisement message, which enables the host to immediately auto-configure its interface.

The router advertisement packet contains a value of 134 in the ICMP packet header and is periodically sent to an all-nodes multicast address to announce their presence or is sent in response to a router solicitation packet and is sent in response to the router solicitation message. The advertisement typically contains prefixes that local-link nodes can use to auto-configure their IPv6 addresses, the lifetime information for each advertised prefix, the flags indicating a stateless or stateful auto-configuration, whether the router sending the advertisement should be used as a default router, and host information such as hop limit and MTU.

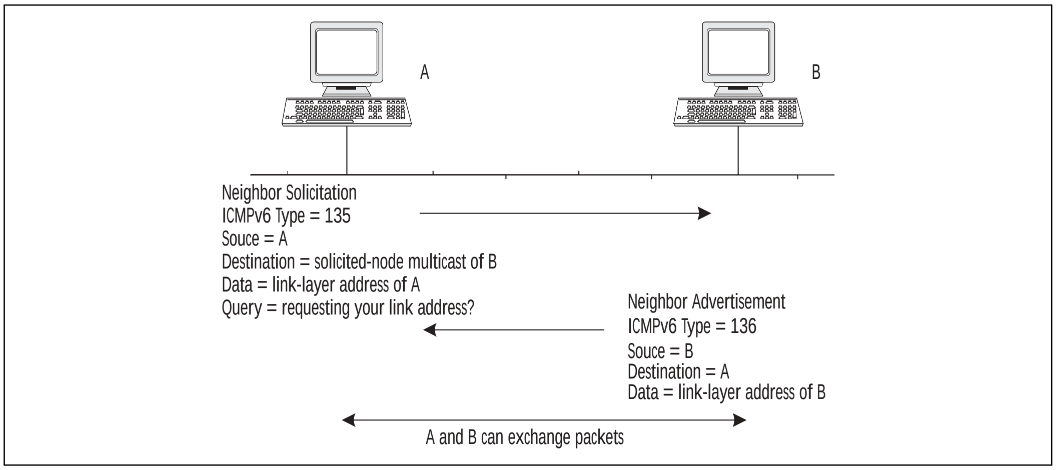
Figure 9.2 Neighbor Discovery: Router Solicitation/Advertisement Messages


A
B
Router Solicitation
ICMPv6 Type = 133
Source = 0::0 or configured unicast address
Destination = all routers multicast address
Data = (sent when interface enabled)
Router Advertisement
ICMPv6 Type = 134
Source = router link-local address
Destination = all nodes multicast address or source address of router A if answering solicitation message
Data = options, prefix, lifetime, autoconfig flag

Neighbor solicitation packets contain a value of 135 in the ICMP packet header and are sent to solicited-node multicast addresses to determine the link-layer address of a neighbor on the same local link. The neighbor solicitation can also be sent to a neighbor's unicast address to verify neighbor reachablity and is used for duplicate address detection. Neighbor reachability identifies the failure of a neighbor or the failure of the forwarding path to the neighbor. The neighbor solicitation message will cause a neighbor advertisement to be sent from the neighboring routers.

The neighbor advertisement packet contains a value of 136 in the ICMP packet header and is sent in response to a neighbor solicitation message. A neighbor advertisement message is sent with the source address of the IPv6 interface sending the neighbor advertisement. After the sender of the neighbor solicitation receives the neighbor advertisement, the two nodes can communicate. A node may also send unsolicited neighbor advertisements to announce a link-layer address change. This concept is illustrated in Figure 9.3.

Figure 9.3 Neighbor Discovery: Neighbor Solicitation/Advertisement Messages


A
B
Neighbor Solicitation
ICMPv6 Type = 135
Souce = A
Destination = solicited-node multicast of B
Data = link-layer address of A
Query = requesting your link address?
Neighbor Advertisement
ICMPv6 Type = 136
Souce = B
Destination = A
Data = link-layer address of B
A and B can exchange packets

A redirect packet contains a value of 137 in the ICMP packet header. Routers use a redirect packet to inform hosts of a better first hop for a destination. Routers also use the redirect packet when the destination address of the packet is not a multicast address, when the packet is not addressed to the router, when the packet is about to be sent out the interface it was received on, or when the source address of the packet is a global IPv6 address of a neighbor on the same link or a link-local address.

The *ipv6 icmp error-interval* command can be used to limit the rate at which all ICMP messages are sent, which ultimately can reduce link congestion.

The Neighbor discovery protocol provides many improvements over the IPv4 protocols. One of the major improvements is the incorporation of router discovery into the base IPv6 protocol rather than requiring hosts to perform discovery using information in the routing protocols. Router advertisements can carry the link-layer address without requiring additional packet exchange; thus, the address resolution overhead is reduced. Router advertisements contain the network prefix so that netmask configuration and determination are not required and not configured incorrectly. The MTU is advertised throughout the conversation path, ensuring that all nodes are using the same MTU. Router advertisements enable address auto-configuration, reducing overall configuration time. Redirect messages contain the new first hop link-layer address, thereby eliminating separate address resolution when the redirect message is received. Lastly, neighbor unreachability detection has been built into the base IPv6 protocol, improving packet delivery in the case of failed routers, links, or changed link-layer addresses.

Configuring & Implementing…

Tuning the Neighbor Discovery Process

You can tune the Neighbor Discovery process to decrease the time a neighbor failure is detected and/or reduce the overhead of the process itself. The following are some commands that can be used to tune the ND process.

- **ipv6 nd reachable-time** Configures the amount of time that a remote IPv6 node is considered reachable after some reachability confirmation event has occurred. This variable defaults to 0 ms for router advertisements and 30000 ms for neighbor discovery activity. The lower the value the more quickly unreachability status can be determined, but bandwidth and processing are increased.
- **ipv6 nd ra-interval** Configures the interval between IPv6 router advertisement transmissions on an interface. The default setting configures the router advertisement interval at 200 seconds. The interval between RA transmissions should be less than or equal to the *ipv6 nd ra-interval*.
- **ipv6 nd ra-lifetime** Configures the “router lifetime” value in IPv6 router advertisements on an interface. The default value configures the “router lifetime” at 1800 seconds. With the value set to 0, the router will not be considered as a default router.
- **ipv6 nd dad attempts** Configures the DupAddrDetectTransmits variable setting the number of consecutive neighbor solicitation messages sent on an interface while duplicate address detection is performed on the unicast IPv6 addresses of the interface. The default setting sends one neighbor solicitation message

Continued

when the interface is enabled. The variable can be set from 0 to 600.

- **ipv6 nd ns-interval** Configures the interval between IPv6 neighbor solicitation retransmissions on an interface and is set in interface configuration mode. This variable defaults to 0 for router advertisements and 1000 for neighbor discovery activity.
- **ipv6 nd prefix-advertisement** Configures which of the configured IPv6 prefixes to include in router advertisements. By default, all prefixes configured on an interface are advertised. The complete syntax of the command is as follows:

```
ipv6 nd prefix-advertisement ipv6-prefix/prefix-length
    valid-lifetime preferred-lifetime [onlink] [autoconfig].
```

- **ipv6 nd suppress-ra** Suppresses IPv6 router advertisement transmissions on a LAN interface. Router advertisements are sent on a LAN interface if unicast routing is enabled. You should use the *no* version of the command on non-LAN interfaces if you wish to send router advertisements.
- **ipv6 nd managed-config-flag** Sets the "managed address configuration" flag in IPv6 router advertisements indicating to attached hosts whether or not they should use stateful auto-configuration (DHCPv6) to obtain addresses. If set, the attached hosts should use stateful auto-configuration to obtain addresses; otherwise, stateless auto-configuration is used to obtain addresses. Hosts may use stateful and stateless address auto-configuration simultaneously.

Summary

In this chapter, we reviewed a few of the new commands that you can use to determine status and to troubleshoot IPv6 configurations and traffic flows. We reviewed several commands that provide information on the status of IPv6-configured router interfaces, show you which paths are available for routing IPv6 traffic, and enable you to determine and monitor IPv6 neighbors. New IPv6 commands also help you to determine which interfaces are configured with routing protocols and enable you to review the quantity and makeup of the IPv6 traffic originating or being received by the router. We also reviewed a few commands used to monitor the status of IPv6 BGP-configured interfaces and determine the BGP neighbors and their communication status.

We reviewed a few commands that are helpful in debugging IPv6 communication. These commands provide insight into ICMP communication, including the neighbor discovery process, IPv6 routing, and IPv6 BGP routing.

A few important aspects of IPv6 are useful to assist in understanding and analyzing IPv6 traffic flows. One important concept is the process of neighbor discovery and the process in which an unreachable neighbor is determined. ICMP packets are used to maintain the neighbor adjacencies and report when a neighbor is unreachable. By learning the contents and messages used by ICMP to maintain neighbor relationships, you will have a firm grasp on the basis of IPv6 communication.

Solutions Fast Track

Using *show* Commands

☑ **show ipv6 interface** The most frequently used command to determine the status of the IPv6-configured interfaces. This should be one of the first commands when troubleshooting link problems.

☑ **show ipv6 route** Displays the possible paths to reach other accessible IPv6 networks. The administrative distance and metric are displayed when using this command.

☑ **show ipv6 route summary** Used to display the number of routes per route source and each prefix length.

- ☑ **show ipv6 neighbors** Displays the IPv6 neighbor adjacency table and provides a listing of neighbors that have become inaccessible and if still accessible, the last time the neighbor was contacted.
- ☑ **show ipv6 protocols** Displays the IPv6 protocols configured on the router. Displays an at-a-glance summary providing information such as the configured route redistribution, the IPv6 neighbors, and the routing protocol configured on each interface.
- ☑ **show ipv6 traffic** Displays statistics about IPv6 traffic and is useful in confirming that IPv6 is operating.
- ☑ **show bgp ipv6** Displays the overall health of BGP on the router and should be one of the first commands used to troubleshoot BGP on the router.
- ☑ **show bgp ipv6 summary** Provides an overview of the BGP configuration on the router.
- ☑ **show bgp ipv6 neighbors** Useful for determining the status of the BGP neighbor communications.

Using *debug* Commands

- ☑ **debug ipv6 packet** Displays information on the packets received, generated, and forwarded on this router.
- ☑ **debug ipv6 icmp** Useful for troubleshooting ICMP communication on the router. The neighbor discovery process, MTU determination, and Multicast Listener Discovery (MLD) all use ICMP.
- ☑ **debug ipv6 nd** Useful for troubleshooting the neighbor discovery process where passing ICMPv6 packets between routers to establish neighbor adjacencies attains adjacencies.
- ☑ **debug ipv6** Displays *debug* messages for IPv6 routing table updates and route cache updates.
- ☑ **debug bgp ipv6** Enables the debugging of IPv6 BGP information.

Analyzing IPv6 Traffic

- ☑ ICMP packets in IPv6 are used in the IPv6 neighbor discovery process, path Maximum Transmission Unit (MTU) discovery, and the Multicast Listener Discovery (MLD) protocol. MLD is similar to IGMP in ICMPv4.
- ☑ Neighbor Discovery, among other things, defines the neighbor discovery process, duplicates address detection, address auto-configuration, and the neighbor unreachabilty process.
- ☑ Five ICMPv6 packet types fulfill the neighbor discovery process: Neighbor Advertisements, Neighbor Solicitations, Router Advertisements, Router Solicitations, and Redirect messages.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: Does migrating to IPv6 provide improved reliability over IPv4?

A: A migration to IPv6 does not increase the reliability over IPv4, because the communication path is made up of essentially the same hardware; however, an improvement has been made in the resiliency of the protocol itself through the implementation of the neighbor discovery protocol and its capability to detect neighbor failures. This failure detection mechanism decreases the time, if configured, to as little as five seconds.

Q: I see "6to4" often when reading about IPv6. What is 6to4 tunneling?

A: The 6to4 method is a transitional "automatic tunneling" method that enables connectivity to other IPv6 networks across the traditional IPv4 infrastructure with less required configuration than other methods. It encapsulates IPv6 traffic in an IPv4 packet and sends the traffic across the network to another 6to4 end-site. The 2002::/16 prefix is reserved for 6to4 based networks and allows for 2^{16} networks with 2^{64} nodes each.

Q: Can I still use ping and traceroute to troubleshoot and resolve IPv6 networking problems?

A: Ping and traceroute are still used to troubleshoot IPv6 networks. The syntax requires the insertion of IPv6 in the command, in addition, of course, to the destination IPv6 address. The syntax is as follows:

```
ping ipv6 [ipv6-address] and traceroute ipv6 [ipv6-address]
```

Q: What is unicast?

A: A unicast packet is the communication between a single host and a single receiver. Unicast and multicast are widely used in IPv6 rather than broadcasts. IPv4 used broadcast in many cases that are sent to all hosts on network, thereby increasing bandwidth utilization.

Q: What is a link-local address?

A: The link-local addresses are used within a local network to provide addressing on a single link, for neighbor discovery, and when no IPv6 routers are present. Routers do not forward link-local addresses.

Q: What is a site-local address?

A: The site-local addresses are used within a site when global addresses are not wanted. Site-local addresses are equivalent to using a 10.0.0.0/8 address in IPv4. Site-local addresses are not forwarded outside the site where they are used.

Appendix

Configuring IPv6 for the Cisco IOS Fast Track

This Appendix will provide you with a quick, yet comprehensive, review of the most important concepts covered in this book.

❖ Chapter 1: Introduction to the Cisco IOS

Connecting to the Router

☑ The initial connection to the router must be made through the console connector on the router. This is accomplished by use of a rolled-RJ45 cable and a 9– or 25-pin serial connector.

☑ After you have applied Internet Protocol (IP) or other logical addresses, you can access a router by Telnet sessions for configuration.

☑ Routers can also use Simple Network Management Protocol (SNMP) settings to integrate with network management systems (NMS) platforms.

Entering Commands to Configure a Cisco Router

☑ The router has two levels of access: *user mode* and *privileged mode*. User mode is a very restricted level of access and allows for only basic functions, whereas privileged mode allows for full configuration and diagnosing capabilities.

☑ Two basic types of commands are available: *configuration* commands and *show* commands. You use configuration commands to change the operation of the router and its processes; you use *show* commands to understand and diagnose the system configuration and operation.

☑ Enter a question mark (**?**) at the command prompt after a particular command to see all the available commands and their configuration parameters.

Increasing Efficiency by Using Shortcuts

☑ Commands can be abbreviated by entering only the first few letters of the command—you must enter enough characters for the IOS to be able to recognize it.

☑ When you press the **Tab** key after entering the beginning of a command, the Cisco IOS's auto-complete feature automatically fills in the rest of the command. The same rule applies as for shortcuts: you must enter enough information to make the command unique.

☑ Several keystroke shortcuts are also available throughout the IOS.

❖ Chapter 2: Introduction to IPv6 Architecture

Understanding the Benefits of IPv6

☑ The IP address size is greatly increased.

☑ A "top-down" allocation plan supports addressing hierarchies, enabling greater address aggregation.

☑ Host addressing is simplified by using the MAC address to form the host portion of the IP address.

☑ Plug and Play functionality is supported by simpler autoconfiguration of IP addresses.

☑ Multicast routing is scoped to provide greater scalability.

☑ Anycast addresses increase routing efficiency.

☑ Streamlined headers lower routing overhead.

☑ Security is built in to support IP security at a lower level in the OSI stack.

☑ Mobility is enhanced with greater scalability and ease.

☑ Performance is enhanced with greater efficiency and streamlining.

Comparing IPv6 to IPv4

☑ IPv6 enlarges the addressing structure without address translation and private address spaces.

☑ IPv6 address administration is "top-down" to allow for greater address aggregation.

☑ IPv6 headers are simpler and more efficient thanks to a fixed size and fewer fields.

☑ IPv6 features not included in IPv4 include enhanced support for multicasting, security, mobility, and discovery.

Examining IPv6 Network Architecture

☑ The impending address space depletion provided the initial impetus for developing IPv6.

☑ A streamlined architecture improves network performance.

Chapter 2 Continued

☑ Security on the IP level is built into IPv6.

☑ Plug and Play has become a reality with IPv6.

☑ Intra-subnet communications can be autoconfigured in a Plug and Play fashion.

☑ Inter-subnet communications can be set up by the stateless autoconfiguration function.

☑ Internetwork communications can be set up by stateful autoconfiguration.

Upper-Layer Protocol Issues

☑ Upper-layer checksums need to be computed over a larger IPv6 pseudoheader.

☑ The maximum packet lifetime is expressed as an IPv6 Hop Limit, not a Time To Live.

☑ Maximum upper-layer payload sizes may need revision to allow for the larger IPv6 header.

☑ The IPv6 Routing Header must be used with care to avoid the security risks of source routing.

☑ The Domain Name Service (DNS) is being enhanced to support IPv6.

☑ Application Programming Interfaces (APIs) are being developed to support the larger IPv6 addresses.

Understanding ICMPv6

☑ There are four types of IPv6 error messages:

- Destination Unreachable messages arise when a packet cannot be delivered for any reason other than congestion.
- Packet Too Big messages arise when a packet exceeds the MTU size of the outgoing interface.
- Time Exceeded messages arise when a packet exceeds its Hop Limit.
- Parameter Problem messages arise when a router cannot process an IPv6 header.

Chapter 2 Continued

☑ There are three types of Informational Messages.

- Diagnostic messages include the echo request and reply messages.
- Multicast Listener Discovery (MLD) messages are used to discover multicast listeners and the multicast addresses that are of interest to them.
- Neighbor Discovery messages support the Neighbor Discovery protocol.

Understanding Neighbor Discovery

☑ Router Solicitation and Advertisement messages are used to discover local routers and to advertise router and subnet information.

☑ Neighbor Solicitation and Advertisement messages are used to discover neighbors and to advertise their addresses.

☑ A Redirect message is used to inform a node of a better first hop to a destination.

☑ Message Options are used to include additional information in Neighbor Discovery messages.

❖ Chapter 3: The IPv6 Headers

Analyzing the IPv6 Header

☑ The IPv6 header reserves two fields (totaling 28 bits) for prioritizing and/or identifying packets and packet flows that require special handling, such as prioritization. As real-time applications such as video and voice become more prevalent, these fields should be heavily used.

☑ The *Next Header* field in IPv6 uses the same definitions used in IPv4 (and defined in RFC 1700). Thus TCP uses "6," UDP uses "17," and so on.

☑ Immediately following the destination IPv6 address is the next header—an IPv6 extension header, an upper-layer protocol header, etc.

Comparing the IPv6 and IPv4 Headers

☑ Source and destination IPv6 addresses are four times as long as IPv4 addresses (128 bits versus 32 bits)

Chapter 3 Continued

- ☑ The IPv4 header adds optional functions within the header in the *options* field. IPv6 has a completely fixed header, but adds extension headers for optional functions such as routing, fragmentation, or authentication.
- ☑ In IPv6, many IPv4 fields were maintained, a few were added, and several were dropped (or changed to extension headers)

The IPv6 Extension Headers

- ☑ Different extension headers were created based on the nodes that would need to examine and process them. For example, the Destination Header and Fragmentation Headers were created strictly for viewing by the endpoint, whereas the Routing and Hop-by-Hop Headers were created to be viewed by all routers along the path. This improves efficiency, since some headers do not need to be processed by intermediate routers.
- ☑ Authentication and Encryption are incorporated into the IPv6 standard as extension headers, supporting the need for increased security in the digital world.
- ☑ Although the Fragmentation header is part of the IPv6 standard, its use is discouraged. In reality, it will likely get very little use.

❖ Chapter 4: Explaining IPv6 Addressing

The Basics of IPv6 Addressing

- ☑ IPv6 uses a 128-bit format that represents the next generation of Internet protocol that can meet both the current addressing requirements and those generated by emerging markets.
- ☑ IPv6 was not developed solely as a mechanism for addressing address depletion. It provides enhancements to aggregation, QoS, security, and routing.
- ☑ Converting to IPv6 is not a trivial task; it is an entirely new addressing structure that introduces complexities in the deployment and management of the address space.

Chapter 4 Continued

IPv6 Addressing Scheme Characteristics

☑ A format for the deployment of highly structured, hierarchical address space is introduced in the global unicast address space.

☑ Due to the variety of address types used in the IPv6 address architecture, routers now have multiple addresses per physical interface. This can include the global unicast address, link-local address, site-local address, and multicast addresses on a per-interface basis.

☑ IPv6 provides three mechanisms for representing the interface identifier of the host address portion of an IPv6 address. These mechanisms are the MAC address, the converted EUI-64 address, and an IPv4 address.

The Need for Further Development

☑ Multihoming to multiple service providers can defeat the highly structured aggregation policies defined in the IPv6 address architecture. A tremendous amount of focus exists in the area of multihoming to alleviate these issues.

☑ Modifications to the TCP stack and to DNS architectures are required to fully realize the benefits of the IPv6 addressing.

☑ The 6Bone provides a test bed for the development of IPv6 infrastructures and is constantly evolving to meet the requirements generated by ongoing deployments of IPv6.

❖ Chapter 5: Configuring IPv6 Addressing

Configuring IPv6 Addressing

☑ When configuring your interfaces, remember that *IPv6* must be included in the syntax to distinguish between IPv4 and IPv6 addresses.

☑ There are three types of addresses that can be assigned on an interface: global, site-local and link-local. These different types of addresses are defined on the interface with different configuration parameters. These addresses must be within their respective address spaces, as defined earlier in the book.

☑ If you are using the *EUI-64* command when entering an IPv6 address, remember that the router uses its identifier for the last 64 bits of the IPv6

Chapter 5 Continued

address, therefore if you want to find out the address, you have to go back and get the full IPv6 address.

- ☑ With the configuration of duplicate address detection, you can specify the number of solicitation messages sent out. If you want to disable this feature, set the number of solicitation messages to zero. Entering the **no** command will restore the number of solicitation messages to the default of one.
- ☑ The ability to configuring the rate limit for ICMPv6 is a new feature that was not available in IPv4. This feature allows you to set the rate at which error messages are sent out, as well as to configure a bucket size for the number of error messages that can be sent.
- ☑ WAN connections, like Frame-Relay and ATM, function the same way with IPv6 as with IPv4. The only differences are that you are using an IPv6 address and that you can enter a link-local address.
- ☑ Most of the commands that involve IPv6 have *IPv6* in the actual syntax used to configure the command on the router. If *IP*v6 is not specified an IPv4 address will be assumed, and you will receive an error from your router because it will not know how to read the address. One exception is entering the DNS information; that command uses just *IP* in the syntax, but will accept either an IPv6 or IPv4 address.

Verifying IPv6 Addressing

- ☑ There is no Address Resolution Protocol with IPv6; it has been replaced with the *neighbor discovery* protocol. Showing the neighbor discovery will give the link-layer address that is associated with the IPv6 address.
- ☑ When using *show* commands to view IPv6 configurations, then *ipv6* is usually in the syntax of the command. If just *IP* is used, typically only IPv4 results will be produced.
- ☑ Remember to always know what version of Cisco IOS you are running, because IPv6 is still an emerging technology and there will be changes is almost every new revision of Cisco IOS that becomes available.

❖ Chapter 6: Routing IPv6 for the Cisco IOS

Explaining RIP for IPv6

☑ The Routing Information Protocol (RIP) is a distance-vector routing protocol. It calculates its routing metrics based solely on hop-count.

☑ A new version of RIP, often referred to as RIPng, was written to support IPv6. This new version of RIP shares the IPv4 limitations and mechanisms.

☑ IPv6 RIP supports default routes, route redistribution, and route filtering.

Configuring RIP for IPv6

☑ You enable RIP with the *Ipv6 router rip <word>* command. The *<word>* specifies the RIP routing process to be used.

☑ Include networks in the IPv6 RIP routing process at the interface level with the *ipv6 rip <word> enable* command. Once again, the *<word>* specifies the RIP process ID.

☑ Configure default routes at the interface layer with the *ipv6 rip <word>default-information originate* command.

Exploring IS-IS for IPv6

☑ Integrated System-to-Integrated System (IS-IS) is a link-state routing protocol that was originally designed for CLNS routing. Extensions were written in to support IP and now IPv6.

☑ Because it is a link-state routing protocol, its metric calculation is based on numerous factors, not just hop-count. It can therefore make better routing decisions than typical distance-vector routing protocols can.

☑ IS-IS supports all the same functions as the IPv6 version of RIP and also supports specification of more than one path to use for data transmission through use of the *maximum-paths* command.

Configuring IS-IS for IPv6

☑ You enable IS-IS on a router with the *CLNS routing* command. Enable integrated IS-IS by issuing the *router ISIS <word>* command. Again, *<word>* refers to the process ID.

Chapter 6 Continued

- ☑ Advertise networks via IS-IS at the interface level with the *ipv6 router isis* <*word*> command.
- ☑ Redistribute other routing protocols into IS-IS with the *redistribution* command in the *address family* configuration mode.

Describing MultiProtocol BGP Extensions for IPv6

- ☑ Border Gateway Protocol (BGP) is an exterior routing protocol used to route between autonomous systems.
- ☑ Like IS-IS, extensions were added into the existing standard to support IPv6.
- ☑ IPv6 BGP supports IPv6 routing, redistribution, manual configuration of a Router ID, link-local addressing, and peer groups.

Configuring BGP Extensions for IPv6

- ☑ Start basic BGP configuration with the *router bgp* <*process id*> command.
- ☑ To support IPv6 unicast traffic, you must disable IPv4 unicasts with the *no bgp default ipv4-unicast* command.
- ☑ Form router peering relationships between BGP routers with the *neighbor* command. You must form a neighbor relationship between every router that you wish your BGP router to exchange information with.

Other Protocols and Future Developments

- ☑ Open Shortest Path First (OSPF) is a link-state routing protocol used in many large organizations. IPv6 support will come in the form of extensions written into the existing standard
- ☑ Enhanced Interior Gateway Routing Protocol (EIGRP) is a hybrid protocol incorporating mechanisms of both distance-vector and link-state routing protocols. Like OSPF, IPv6 support will come in the form of extensions written into the existing standard
- ☑ Both OSPF and EIGRP support for IPv6 will come with the Phase III deployment of IPv6 from Cisco, which is currently scheduled for mid-2002.

❖ Chapter 7: Deploying IPv6 on the Cisco IOS

IPv6 Deployment Strategies

☑ Creating tunnels across an existing IPv4 network is a tried-and-true method of creating connectivity between IPv6 domains. Point-to-point can be created manually with IPv6 manually configured tunnels and Cisco's GRE tunnels. Tunnels can be created automatically based on the address using Automatic IPv6 Tunnels, Automatic 6to4 Tunnels, and ISATAP Tunnels. If MPLS is deployed, tunnels can be created at the CE router.

☑ For QoS or SLA requirements, the use of dedicated links will provide the needed control over the different protocols. Separate links can be provisioned with total separation as the most costly and secure method. Using virtual links with transports such as Frame Relay, ATM, or Optical LAMBDAs. Circuit transport over MPLS can also provide QoS while separating the protocols.

☑ When isolating IPv4 from IPv6 isn't a concern or there is a need to have both IPv4 and IPv6 co-exist in the domains both hosts and routers can run in a dual-stack mode which basically running dual protocol stacks in both the hosts and routers. The routers will also be required to run dual routing protocols.

Understanding Deployment Methods

☑ **Tunnels** A technique to encapsulate IPv6 packets inside IPv4 packets for transport across an existing IPv4 infrastructure.

- **IPv6 Manually Configured Tunnel** TEPs are defined across an IPv4 network. All IPv6 traffic enters and exits the network at the TEPs.

  Pros: Ease of deployment, low cost, tried-and-tested technology.

  Cons: Doesn't scale, tunnels are pre-defined point-to-point links, high management overhead, troubleshooting is difficult.

- **IPv6 over IPv4 GRE Tunnel** Enhanced features of the IPv6 manually configured tunnel. Added feature of protocol independence of traffic configured for the tunnel.

Chapter 7 Continued

- **Automatic IPv6 Tunnel** Tunnels are created automatically based on the lower 32 bits of the address. The IPv6 address is created by taking an IPv4 address and padding 0's into the upper 96 bits.

  Pros: Ease of deployment, low cost, tunnels are created automatically.

  Cons: No realization of the increased address space of IPv6, doesn't scale, troubleshooting is difficult.

- **Automatic 6to4 Tunnel** Tunnels are created automatically based on the IPv6 address. An IPv4 address is used in the upper 32 bits of the NLA, which allows another 16 bits of SLA for segmentation within the IPv6 domain.

  Pros: Ease of deployment, low cost, tunnels are created automatically, use of valid IPv6 frames.

  Cons: Troubleshooting is difficult,

- **ISATAP Tunnel** This tunnel technique is still in draft form. Cisco has plans to support ISATAP at a later date. ISATAP uses an IPv4 address to create the tunnel, but places the 32 bit IPv4 address in the low 32 bits of the interface and uses a valid 64 bit IPv6 network address.

  Pros: Ease of deployment, low cost, tunnels are created automatically, full use of valid IPv6 network addresses.

  Cons: Troubleshooting is difficult.

- **6over4 Tunnels** This tunnel technique uses multicast routing as a virtual link layer. Cisco does not and does not plan to support 6over4 tunnels. Like the Automatic IPv6 tunnel, an IPv4 address is used in the lower 32 bits with a fixed FE80::/96 prefix.

  Pros: Ease of deployment, low cost, tunnels are created automatically.

  Cons: Not supported on Cisco products, no realization of the increased address space of IPv6, requires multicast network, doesn't scale, troubleshooting is difficult.

☑ **IPv6 over Dedicated Links** Using existing transport infrastructure, IPv6 is directed to dedicated links or virtual links. IPv6 has no impact on IPv4 traffic.

Chapter 7 Continued

Pros: QoS and SLAs can be managed for IPv4 and IPv6 separately.

Cons: IPv6 hardware acceleration and routing isn't as robust as that of IPv4.

☑ **IPv6 over an MPLS Backbone** Tunneling, dedicated links, and dual-stack techniques work with MPLS as they do with IPv4-routed backbones. The same restraints and benefits apply to each of the techniques; the only difference is the benefits realized by having MPLS on the backbone.

- **IPv6 Using Tunnels on the Customer Edge Routers** This has the same characteristics as the manually configured tunnels, with the difference being that the encapsulated packets are switched across the MPLS backbone versus being routed across an IPv4 backbone.
- **IPv6 over a Circuit Transport over MPLS** Same characteristics as using dedicated links. MPLS VPNs are created with the appropriate QoS assigned to each VPN.
- **IPv6 on the Provider Edge Router** This is not commercially available and is still in draft form with the IETF. Using MP-BGP, MPLS is able to advertise next-hop reachability using IPv6 address, which will enable native IPv6 service to be offered to the CE routers from the PE.

☑ **Using a Dual-Stack Backbone** Requires fully redundant configurations of IPv4 and IPv6 with each router supporting both IPv4 and IPv6 and their routing protocols.

Pros: Ease of deployment, ability to use protocol analyzer for IPv6 issues, full management of IPv6.

Cons: IPv6 hardware acceleration and routing protocols not as robust as IPv4, depending on the size of the network, the overhead to upgrade all the routers (and hosts) for IPv6 support could be large; possible hardware and memory upgrades to support IPv6.

Translating between IPv4 and IPv6

☑ Translation techniques have three basic approaches:

- All traffic passes through a single "translation device," such as NAT-PT or TCP-UDP Relay.

Chapter 7 Continued

- Sniffing packets at the host and making the translation from IPv6 to IPv4 before the NIC passes the data to the network layer of the host. The BIS method requires a client (or module) to be installed on each of the workstations.
- Using a server to terminate both the IPv6 session as well as the IPv4 session. Both DSTM and Socks-based IPv6/IPv4 gateways relay data between the two sessions. This requires the installation of a client to communicate with the server.

☑ Using a server to translate all traffic creates a single point of failure, because all traffic passes through the translator. Heavy traffic could tax the server, requiring more server power or additional memory to support the volume of packets. The upside is that this approach eases management—because all traffic passes through a single device, all changes and troubleshooting would be at a central location and not dispersed across the network.

☑ Installing clients on each workstation can be an administration and management burden for large deployments. For large deployments the amount of effort to install clients on all the workstations and manage host compatibility and possible upgrades should be factored in before deploying these techniques.

❖ Chapter 8: IPv6 Security

IPSec Overview

☑ Security risks specific to IP include data theft, data tampering, and peer impersonation.

☑ Security features included in IPv6 via IP Security (IPSec) extension headers provide authentication, integrity, confidentiality, and access control.

☑ IPSec is a standard defined by a suite of Internet Engineering Task Force (IETF) Requests for Comments (RFCs) that can be implemented in both IPv4 and IPv6.

Chapter 8 Continued

Understanding the Building Blocks of IPSec

☑ IPSec services are extension headers that can be included within an IPv6 header. The services consist of two security protocols: the Authentication Header (AH) and the Encapsulating Security Payload (ESP).

☑ AH is an extension header and protocol that uses a cryptographic signature to provide both connectionless integrity and data origin authentication. You should use AH when it's important to be certain that you are communicating with correct node and when you want to ensure that data is not modified in transit.

☑ ESP is an extension header and protocol that can provide confidentiality, data origin authentication, connectionless integrity, replay protection, and limited traffic flow confidentiality.

☑ Both AH and ESP support two modes of use: transport mode and tunnel mode. In transport mode, the selected protocol (AH or ESP) provides protection primarily for upper layer protocols, and its extension header is created and inserted as part of the original IPv6 header. However, in tunnel mode, the extension header is created and inserted as part of a new IPv6 header that encapsulates the entire original packet as the payload.

☑ Because IPSec is intended to provide network layer security services, it can be implemented on end hosts and devices.

☑ Within IPsec there are two primary methods of authenticating peers in which authentication information is exchanged between the peers out of band: pre-shared keys and certificates. With pre-shared keys, two or more peers securely exchange (pre-share) a secret key prior to any IPSec negotiation—the process is simple and inexpensive but does not scale well. Certificates scale well but are typically more complex to implement.

Combining IPSec's Cryptographic Mechanisms

☑ The Security Policy Database (SPD) identifies the services to be applied to IP packets and is consulted in the processing of all traffic. For any packet, the SPD will identify one of three options for processing: discard, bypass IPSec, or apply IPSec.

Chapter 8 Continued

- ☑ The SA is a mechanism that IPSec uses to keep track of the details of a negotiated IPSec session between two nodes. A pair of SAs is required for communication between a pair of nodes. An SA is uniquely identified by a destination IP address, a security protocol identifier (AH or ESP), and a Security Parameter Index (SPI).
- ☑ Internet Key Exchange (IKE) is the mechanism that determines which services should be applied to the different traffic flows and negotiates the required cryptographic keys for those services. IKE has two phases. Phase 1 is used to establish a secure channel (ISAKAMP SA) through which IPSec cryptographic services and algorithms can be negotiated. Phase 2 is the actual negotiation of the IPSec cryptographic services and algorithms through the secure channel established in Phase 1.
- ☑ Within IKE Phase 1, two possible modes can be used to establish the ISAKMP SA: main mode and aggressive mode. Main Mode consists of three exchanges between the initiator and the responder for a total of six packets exchanged. Because this data is encrypted, main mode provides what is called *identity protection*. Aggressive Mode can also be used to establish the ISAKMP SA; however, it uses a total of three packets instead of six. Aggressive mode does not provide identity protection.

Applying Perimeter Security

- ☑ Access lists are an important component in the secure configuration of a router. They provide packet filtering capabilities for Cisco routers, controlling access to services on the router itself, and filtering traffic passing through the router.
- ☑ Standard access lists can filter based on only the source IP address within a packet. Extended access lists can filter based on other packet parameters, including source and destination IP address, source and destination ports, and protocol. In addition, extended access lists support logging.

❖ Chapter 9: Monitoring and Troubleshooting IPv6 Networks

Using *show* Commands

☑ **show ipv6 interface** The most frequently used command to determine the status of the IPv6-configured interfaces. This should be one of the first commands when troubleshooting link problems.

☑ **show ipv6 route** Displays the possible paths to reach other accessible IPv6 networks. The administrative distance and metric are displayed when using this command.

☑ **show ipv6 route summary** Used to display the number of routes per route source and each prefix length.

☑ **show ipv6 neighbors** Displays the IPv6 neighbor adjacency table and provides a listing of neighbors that have become inaccessible and if still accessible, the last time the neighbor was contacted.

☑ **show ipv6 protocols** Displays the IPv6 protocols configured on the router. Displays an at-a-glance summary providing information such as the configured route redistribution, the IPv6 neighbors, and the routing protocol configured on each interface.

☑ **show ipv6 traffic** Displays statistics about IPv6 traffic and is useful in confirming that IPv6 is operating.

☑ **show bgp ipv6** Displays the overall health of BGP on the router and should be one of the first commands used to troubleshoot BGP on the router.

☑ **show bgp ipv6 summary** Provides an overview of the BGP configuration on the router.

☑ **show bgp ipv6 neighbors** Useful for determining the status of the BGP neighbor communications.

Using *debug* Commands

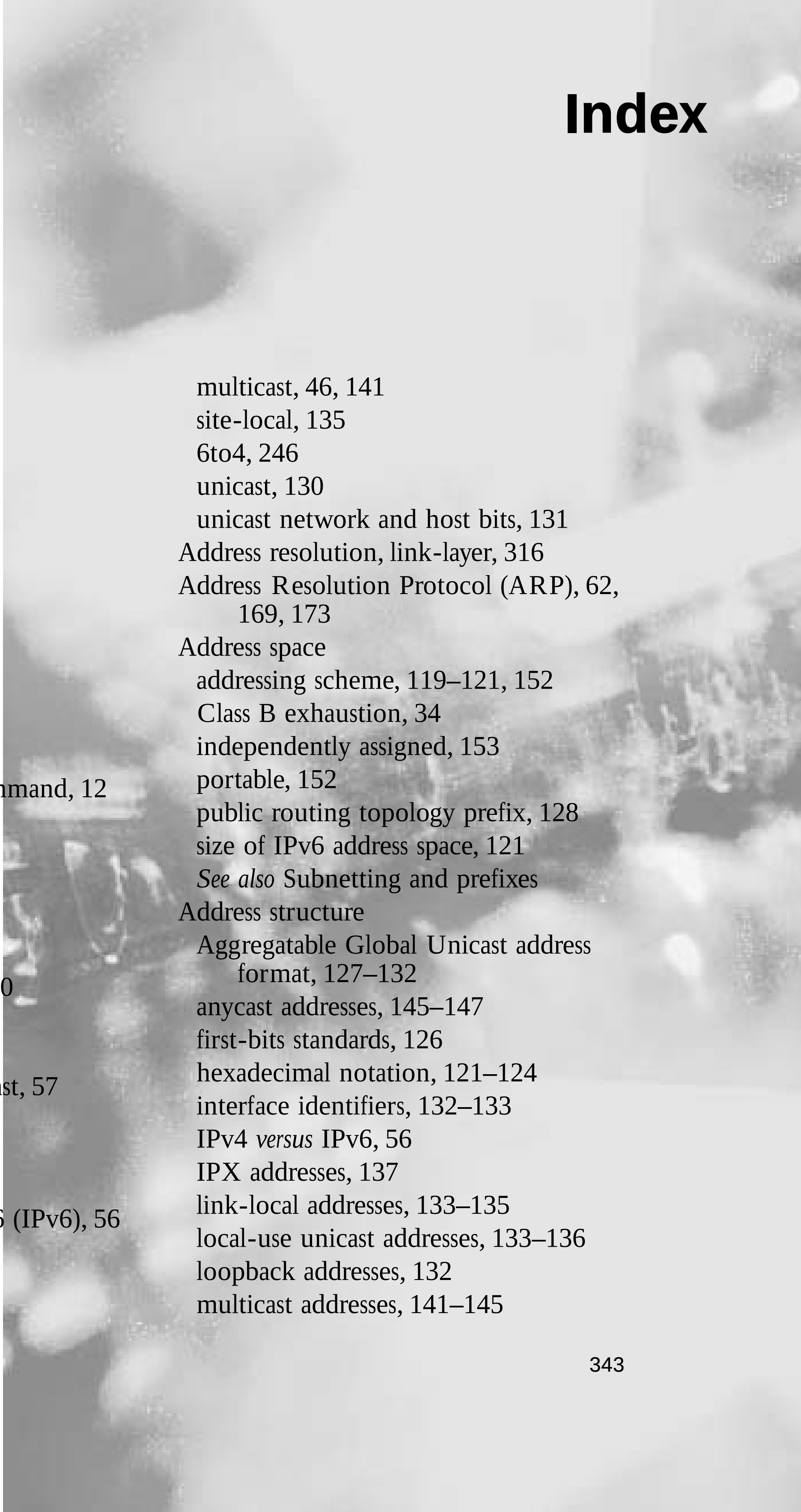
☑ **debug ipv6 packet** Displays information on the packets received, generated, and forwarded on this router.

Chapter 9 Continued

- ☑ **debug ipv6 icmp** Useful for troubleshooting ICMP communication on the router. The neighbor discovery process, MTU determination, and Multicast Listener Discovery (MLD) all use ICMP.
- ☑ **debug ipv6 nd** Useful for troubleshooting the neighbor discovery process where passing ICMPv6 packets between routers to establish neighbor adjacencies attains adjacencies.
- ☑ **debug ipv6** Displays *debug* messages for IPv6 routing table updates and route cache updates.
- ☑ **debug bgp ipv6** Enables the debugging of IPv6 BGP information.

Analyzing IPv6 Traffic

- ☑ ICMP packets in IPv6 are used in the IPv6 neighbor discovery process, path Maximum Transmission Unit (MTU) discovery, and the Multicast Listener Discovery (MLD) protocol. MLD is similar to IGMP in ICMPv4.
- ☑ Neighbor Discovery, among other things, defines the neighbor discovery process, duplicates address detection, address auto-configuration, and the neighbor unreachabilty process.
- ☑ Five ICMPv6 packet types fulfill the neighbor discovery process: Neighbor Advertisements, Neighbor Solicitations, Router Advertisements, Router Solicitations, and Redirect messages.



Index