Cisco Press Technology Folio

Cisco Press
201 West 103rd Street
Indianapolis, IN 46290 USA

Cisco Press Technology Folio

Published by:
Cisco Press
201 West 103rd Street
Indianapolis, IN 46290 USA

Printed in the United States of America 1 2 3 4 5 6 7 8 9 0

ISBN: 1-58705-057-9

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an "as is" basis. The authors, Cisco Press, and Cisco Systems, Inc., shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the authors and are not necessarily those of Cisco Systems, Inc.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Reader feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Publisher	John Wait
Executive Editor	John Kane
Cisco Systems Program Manager	Bob Anstey
	Bill Warren
Managing Editor	Patrick Kanouse
Production Editor	Marc Fowler
Compositor	Steve Gifford

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems Europe
11 Rue Camille Desmoulins
92782 Issy-les-Moulineaux
Cedex 9
France
http://www-europe.cisco.com
Tel: 33 1 58 04 60 00
Fax: 33 1 58 04 61 00

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems Australia,
Pty., Ltd
Level 17, 99 Walker Street
North Sydney
NSW 2059 Australia
http://www.cisco.com
Tel: +61 2 8448 7100
Fax: +61 2 9957 4350

Cisco Systems has more than 200 offices in the following countries. Addresses, phone numbers, and fax numbers are listed on the Cisco Web site at www.cisco.com/go/offices

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China • Colombia • Costa Rica • Croatia • Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong • Hungary • India • Indonesia • Ireland Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines Poland • Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

Contents at a Glance

Contents

Excerpt from Routing TCP/IP, Volume II

Excerpt from Cisco Voice over Frame Relay, ATM and IP

Excerpt from Managing Cisco Network Security

Excerpt from Cisco BGP-4 Command and Configuration

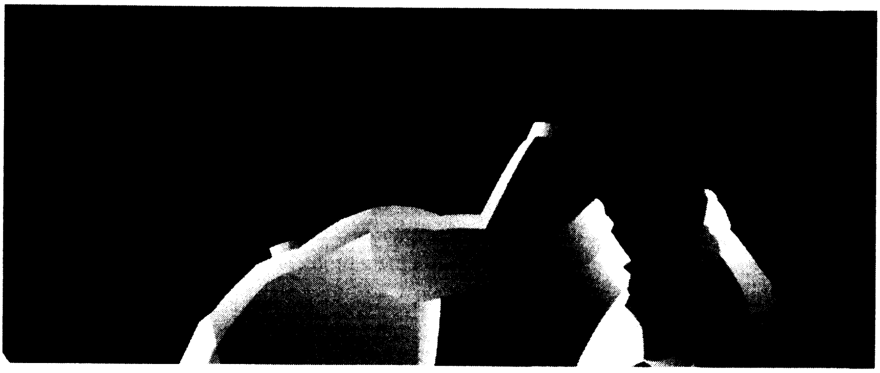
Excerpt from IP Quality of Service

Excerpt from High Availability Networking Fundamentals

Excerpt from Cisco Secure Internet Security Solutions

Excerpt from Integrating Voice and Data Networks

Excerpt from MPLS and VPN Architectures



from Routing TCP/IP, Volume II

by Jeff Doyle
and Jennifer DeHaven Carroll

(1-57870-089-2)

Cisco Press

About the Authors

Jeff Doyle, CCIE #1919, is a Professional Services Consultant with Juniper Networks, Inc. in Denver, Colorado. Specializing in IP routing protocols and MPLS Traffic Engineering, Jeff has helped design and implement large-scale Internet service provider networks throughout North America, Europe, and Asia. Jeff has also lectured on advanced networking technologies at service provider forums such as the North American Network Operators' Group (NANOG) and the Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT). Prior to joining Juniper Networks, Jeff was a Senior Network Systems Consultant with International Network Services. Jeff can be contacted at jeff@juniper.net.

Jennifer DeHaven Carroll, is a principal consultant with Lucent technologies and is a Cisco Certified Internetwork Expert (CCIE # 1402). She has planned, designed, and implemented many large networks over the past 13 years. She has also developed and taught theory and Cisco implementation classes on all IP routing protocols. Jenny can be reached at jennifer.carroll@ieee.org.

Contents at a Glance

Bold chapters are elements included in this folio.

This chapter covers the following key topics:

- **The Origins of EGP**—This section discusses the history of the development of the Exterior Gateway Protocol, presented in RFC 827 (1982).
- **Operation of EGP**—This section explores the fundamental mechanics of EGP with a focus on EGP topology issues, EGP functions, and EGP message formats.
- **Shortcomings of EGP**—This section explores some of the reasons why EGP is no longer pursued as a viable external gateway protocol solution.
- **Configuring EGP**—This section presents four separate case studies—EGP stub gateway, EGP core gateway, indirect neighbors, and default routes—to demonstrate different types of EGP configuration.
- **Troubleshooting EGP**—This section examines how to interpret an EGP neighbor table and presents a case study on the slow convergence speed of an EGP network to show why EGP is no longer a popular option.

CHAPTER 1

Exterior Gateway Protocol

The first question knowledgeable readers will (and should) ask is "Why kill a few trees publishing a chapter about an obsolete protocol such as the Exterior Gateway Protocol (EGP)?" After all, EGP has been almost universally replaced by the Border Gateway Protocol (BGP). This question has two answers.

First, although EGP is rarely used these days, it is still occasionally encountered. As of this writing, for instance, you can still find EGP in a few U.S. military internetworks. As a CCIE, you should understand EGP for such rare encounters.

Second, this chapter serves as something of a history lesson. Examining the motives for developing an external gateway protocol and the shortcomings of the original external protocol provides a prologue for the following two chapters. BGP will make more sense to you if you are familiar with the roots from which it evolved.

The Origins of EGP

In the early 1980s, the routers (gateways) that made up the ARPANET (predecessor of the modern Internet) ran a distance vector routing protocol known as the *Gateway-to-Gateway Protocol* (GGP). Every gateway knew a route to every reachable network, at a distance measured in gateway hops. As the ARPANET grew, its architects foresaw the same problem that administrators of many growing internetworks encounter today: Their routing protocol did not scale well.

Eric Rosen, in RFC 827[1], chronicles the scalability problems:

- With all gateways knowing all routes, "the overhead of the routing algorithm becomes excessively large." Whenever a topology change occurs, the likelihood of which increases with the size of the internetwork, all gateways have to exchange routing information and recalculate their tables. Even when the internetwork is in a steady state, the size of the routing tables and routing updates becomes an increasing burden.
- As the number of GGP software implementations increases, and the hardware platforms on which they are implemented become more diverse, "it becomes impossible to regard the Internet as an integrated communications system." Specifically, maintenance and troubleshooting become "nearly impossible."

- As the number of gateways grows, so does the number of gateway administrators. As a result, resistance to software upgrades increases: "[A]ny proposed change must be made in too many different places by too many different people."

The solution proposed in RFC 827 was that the ARPANET be migrated from a single internetwork to a system of interconnected, autonomously controlled internetworks. Within each internetwork, known as an autonomous system (AS), the administrative authority for that AS is free to manage the internetwork as it chooses. In effect, the concept of autonomous systems broadens the scope of internetworking and adds a new layer of hierarchy. Where there was a single internetwork—a network of networks—there is now a network of autonomous systems, each of which is itself an internetwork. And just as a network is identified by an IP address, an AS is identified by an autonomous system number. An AS number is a 16-bit number assigned by the same addressing authority that assigns IP addresses.

NOTE Also like IP addresses, some AS numbers are reserved for private use. These numbers range from 64512 to 65535. See RFC 1930 (www.isi.edu/in-notes/rfc1930.txt) for more information.

Chief among the choices the administrative authority of each AS is free to make is the routing protocol that its gateways run. Because the gateways are interior to the AS, their routing protocols are known as interior gateway protocols (IGPs). Because GGP was the routing protocol of the ARPANET, it became by default the first IGP. However, interest in the more modern (and simpler) Routing Information Protocol (RIP) was building in 1982, and it was expected that this and other as-yet-unplanned protocols would be used in many autonomous systems. These days, GGP has been completely replaced by RIP, RIP-2, Interior Gateway Routing Protocol (IGRP), Enhanced IGRP (EIGRP), Open Shortest Path First (OSPF), and Integrated Intermediate System-to-Intermediate System (IS-IS).

Each AS is connected to other autonomous systems via one or more exterior gateways. RFC 827 proposed that the exterior gateways share routing information between each other by means of a protocol known as the EGP. Contrary to popular belief, although EGP is a distance vector protocol, it is not a routing protocol. It has no algorithm for choosing an optimal path between networks; rather, it is a common language that exterior gateways use to exchange reachability information with other exterior gateways. That reachability information is a simple list of major network addresses (no subnets) and the gateways by which they can be reached.

Operation of EGP

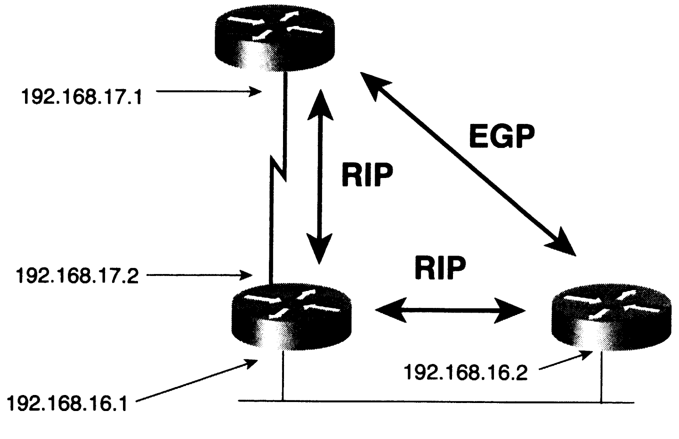
Version 1 of EGP was proposed in RFC 827. Version 2, slightly modified from version 1, was proposed in RFC 888[2], and the formal specification of EGPv2 is given in RFC 904[3].

EGP Topology Issues

EGP messages are exchanged between EGP neighbors, or *peers*. If the neighbors are in the same AS, they are *interior neighbors*. If they are in different autonomous systems, they are *exterior neighbors*. EGP has no function that automatically discovers its neighbors; the addresses of the neighbors are manually configured, and the messages they exchange are unicast to the configured addresses.

RFC 888 suggests that the time-to-live (TTL) of EGP messages be set to a low number, because an EGP message should never travel farther than to a single neighbor. However, nothing in the EGP functionality requires EGP neighbors to share a common data link. For example, Figure 1-1 shows two EGP neighbors separated by a router that speaks only RIP. Because EGP messages are unicast to neighbors, they can cross router boundaries. Therefore, Cisco routers set the TTL of EGP packets to 255.

Figure 1-1 *EGP Neighbors Do Not Have to Be Connected to the Same Network*

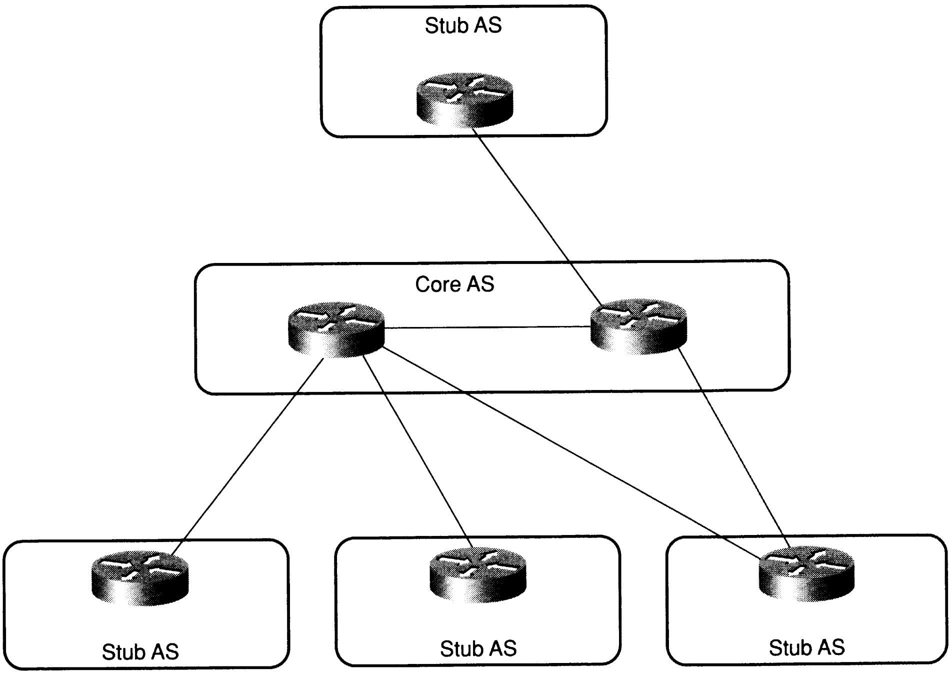


EGP gateways are either core gateways or stub gateways. Both gateway types can accept information about networks in other autonomous systems, but a stub gateway can send only information about networks in its own AS. Only core gateways can send information they have learned about networks in autonomous systems other than their own.

To understand why EGP defines core and stub gateways, it is necessary to understand the architectural limitations of EGP. As previously mentioned, EGP is not a routing protocol. Its updates list only reachable networks, without including enough information to determine shortest paths or to prevent routing loops. Therefore, the EGP topology must be built with no loops.

Figure 1-2 shows an EGP topology. There is a single core AS to which all other autonomous systems (stub autonomous systems) must attach. This two-level tree topology is very similar to the two-level topology requirements of OSPF, and its purpose is the same. Recall from *Routing TCP/IP, Volume I* that interarea OSPF routing is essentially distance vector, and therefore vulnerable to routing loops. Requiring all traffic between nonbackbone OSPF areas to traverse the backbone area reduces the potential for routing loops by forcing a loop-free interarea topology. Likewise, requiring all EGP reachability information between stub autonomous systems to traverse the core AS reduces the potential for routing loops in the EGP topology.

Figure 1-2 *To Prevent Routing Loops, Only Core Gateways Can Send Information Learned from One AS to Another AS*



EGP Functions

EGP consists of the following three mechanisms:

- Neighbor Acquisition Protocol
- Neighbor Reachability Protocol
- Network Reachability Protocol

These three mechanisms use ten message types to establish a neighbor relationship, maintain the neighbor relationship, exchange network reachability information with the neighbor, and notify the neighbor of procedural or formatting errors. Table 1-1 lists all of the EGP message types and the mechanism that uses each message type.

Table 1-1 *EGP Message Types*

Message Type	**Mechanism**
Neighbor Acquisition Request	Neighbor Acquisition
Neighbor Acquisition Confirm	Neighbor Acquisition
Neighbor Acquisition Refuse	Neighbor Acquisition
Neighbor Cease	Neighbor Acquisition
Neighbor Cease Acknowledgment	Neighbor Acquisition
Hello	Neighbor Reachability
I-Heard-You	Neighbor Reachability
Poll	Network Reachability
Update	Network Reachability
Error	All functions

The following sections discuss the details of each of the three EGP mechanisms; the section "EGP Message Formats" in this chapter covers the specific details of the messages.

Neighbor Acquisition Protocol

Before EGP neighbors can exchange reachability information, they must establish that they are compatible. This function is performed by a simple two-way handshake in which one neighbor sends a Neighbor Acquisition Request message, and the other neighbor responds with a Neighbor Acquisition Confirm message.

None of the RFCs specify how two EGP neighbors initially discover each other. In practice, an EGP gateway learns of its neighbor by manual configuration of the neighbor's IP address. The gateway then unicasts an Acquisition Request message to the configured neighbor. The message states a *Hello interval*, the minimum interval between Hello messages that the gateway is willing to accept from the neighbor, and a *Poll interval*, the minimum interval that the gateway is willing to be polled by the neighbor for routing updates. The neighbor's responding Acquisition Confirm message will contain its own values for the same two intervals. If the neighbors agree on the values, they are ready to exchange network reachability information.

When a gateway first learns of a neighbor, it considers the neighbor to be in the Idle state. Before sending the first Acquisition Request, the gateway transitions the neighbor to the *Acquire* state; when the gateway receives an Acquisition Confirm, it transitions the neighbor to the Down state.

NOTE See RFC 904 for a complete explanation of the EGP finite state machine.

A gateway can refuse to accept a neighbor by responding with a Neighbor Acquisition Refuse message rather than an Acquisition Confirm message. The Refuse message can include a reason for the refusal, such as a lack of table space, or it can refuse for an unspecified reason.

A gateway can also break an established neighbor relationship by sending a Neighbor Cease message. As with the Refuse message, the originating gateway has the option of including a reason for the Cease or leaving the reason unspecified. A neighbor receiving a Neighbor Cease message responds with a Neighbor Cease Acknowledgment.

The last case of a Neighbor Acquisition procedure is a case in which a gateway sends an Acquisition Request but the neighbor does not respond. RFC 888 suggests retransmitting the Acquisition message "at a reasonable rate, perhaps every 30 seconds or so." Cisco's EGP implementation does not just repeat unacknowledged messages over a constant period. Rather, it retransmits an unacknowledged Acquisition message 30 seconds after the original transmission. It then waits 60 seconds before the next transmission. If no response is received within 30 seconds of the third transmission, the gateway transitions the neighbor state from Acquire to Idle (see Example 1-1). The gateway remains in the Idle state for 300 seconds (5 minutes) and then transitions to Acquire and starts the process all over.

Notice in Example 1-1 that each EGP message has a sequence number. The sequence number allows EGP message pairs (such as Neighbor Acquisition Request/Confirm, Request/Refusal, and Cease/Cease-Ack pairs) to be identified. The next section, "Network Reachability Protocol," details how the sequence numbers are used.

When two EGP gateways become neighbors, one is the *active* neighbor and one is the *passive* neighbor. Active gateways always initiate the neighbor relationship by sending Neighbor Acquisition Requests. Passive gateways do not send Acquisition Requests; they only respond to them. The same is true for Hello/I-Heard-You message pairs, described in the following section: The active neighbor sends the Hello, and the passive neighbor responds with an I-Heard-You (I-H-U). A passive gateway can initiate a Neighbor Cease message, however, to which the active gateway must reply with a Cease Acknowledgement message.

Example 1-1 **debug ip egp transactions** *Command Output Displays EGP State Transitions*

```
Shemp#debug ip egp transactions
EGP debugging is on
Shemp#
EGP: 192.168.16.2 going from IDLE to ACQUIRE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: 192.168.16.2 going from ACQUIRE to IDLE
EGP: 192.168.16.2 going from IDLE to ACQUIRE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: 192.168.16.2 going from ACQUIRE to IDLE
```

A core gateway, which can be a neighbor of routers in several other autonomous systems, might be the active gateway of one neighbor adjacency and the passive gateway of another neighbor adjacency. Cisco's EGP implementation uses the AS numbers as the determining factor: The neighbor whose AS number is lower will be the active neighbor.

Neighbor Reachability Protocol

After a gateway has acquired a neighbor, it maintains the neighbor relationship by sending periodic Hello messages. The neighbor responds to each Hello with an I-H-U message. RFC 904 does not specify a standard period between Hellos; Cisco uses a default period of 60 seconds, which can be changed with the command **timers egp**.

When three Hello/I-H-U message pairs have been exchanged, the neighbor state changes from Down to Up (see Example 1-2). The neighbors can then exchange network reachability information, as described in the next section.

If an active neighbor sends three sequential messages without receiving a response, the neighbor state transitions to Down. The gateway sends three more Hellos at the normal Hello interval; if there is still no response, the state changes to Cease. The gateway sends three Neighbor Cease messages at 60-second intervals. If the neighbor responds to any of the messages with a Cease Acknowledgment, or does not respond at all, the gateway transitions the neighbor state to Idle and waits 5 minutes before transitioning back to Acquire and attempting to reacquire the neighbor. Example 1-3 shows this sequence of events.

Example 1-2 **debug ip egp transactions** *Command Output Displays Two-Way Handshake Success and EGP State Transitions*

```
EGP: 192.168.16.2 going from IDLE to ACQUIRE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=ACQUIRE, Code=REQUEST, Status=1 (ACTIVE-MODE), Hello=60, Poll=180
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=ACQUIRE, Code=CONFIRM, Status=2 (PASSIVE-MODE), Hello=60, Poll=180
EGP: 192.168.16.2 going from ACQUIRE to DOWN
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=2 (DOWN)
EGP: 192.168.16.2 going from DOWN to UP
```

Example 1-3 *The Neighbor at 192.168.16.2 Has Stopped Responding. The Interval Between Each of the Unacknowledged EGP Messages Is 60 Seconds*

```
Shemp#
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: 192.168.16.2 going from UP to DOWN
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: 192.168.16.2 going from DOWN to CEASE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=5 (HALTING)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=1 (ACTIVE-MODE)
```

Example 1-4 shows another example of a dead neighbor, except this time a core gateway (192.168.16.2) in the passive mode is discovering the dead neighbor (192.168.16.1).

Example 1-4 *Neighbor 192.168.16.1 Has Stopped Responding. The* **debug** *Messages Are Taken from 192.168.16.2, a Gateway in Passive Mode*

```
Moe#
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=1
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=1
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=1
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: 192.168.16.1 going from UP to DOWN
[redacted]
     Type=ACQUIRE, Code=CEASE, Status=5 (HALTING)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=2 (PASSIVE-MODE)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=2 (PASSIVE-MODE)
[redacted]
```

When the gateway does not receive a Hello within the 60-second Hello interval, it tries to "wake up" its neighbor. Because a gateway in passive mode cannot send Hellos, it sends a Poll message. The gateway then waits for one Poll interval. (Cisco's default Poll interval is 180 seconds, or 3 minutes.) If no response is received, it sends another Poll and waits another Poll interval. If there still is no response, the gateway changes the neighbor state to Down and then immediately to Cease. As in Example 1-3, three Cease messages are sent and the neighbor state is changed to Idle.

Network Reachability Protocol

When the neighbor state is Up, the EGP neighbors can begin exchanging reachability information. Each gateway periodically sends a Poll message to its neighbor, containing some sequence number. The neighbor responds with an Update message that contains the same sequence number and a list of reachable networks. Example 1-5 shows how Cisco's IOS Software uses the sequence numbers.

Example 1-5 *EGP Neighbors Poll Each Other Periodically for Network Reachability Updates*

```
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=120
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=120
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=120
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=120
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=120
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=120
     Type=UPDATE, Code=0, Status=1 (UP), IntGW=2, ExtGW=1, Net=192.168.16.0
     Network 172.17.0.0 via 192.168.16.2 in 0 hops
     Network 192.168.17.0 via 192.168.16.2 in 0 hops
     Network 10.0.0.0 via 192.168.16.2 in 3 hops
     Network 172.20.0.0 via 192.168.16.4 in 0 hops
     Network 192.168.18.0 via 192.168.16.3(e) in 3 hops
     Network 172.16.0.0 via 192.168.16.3(e) in 3 hops
     Network 172.18.0.0 via 192.168.16.3(e) in 3 hops
EGP: 192.168.16.2 updated 7 routes
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=3
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=UPDATE, Code=0, Status=1 (UP), IntGW=1, ExtGW=0, Net=192.168.16.0
     Network 172.19.0.0 via 192.168.16.1 in 0 hops
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=121
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=121
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
```

Every Hello/I-H-U pair exchanged between neighbors contains the same sequence number until a Poll is sent. The Poll/Update pair also uses the same sequence number. After the Update has been received, the active neighbor increments the sequence number. In Example 1-5, the sequence number is 120 through the Poll/Update, and it then is incremented to 121. Notice that both neighbors send a Poll; in this example, the Poll from the passive neighbor (192.168.16.2) has an entirely different sequence number (3). A neighbor always responds with an Update containing the same sequence number as the Poll.

The default polling interval used by Cisco's IOS Software is 180 seconds and can be changed with the command **timers egp**. Normally, a gateway sends an Update only when it is polled; however, this means a topology change might go unannounced for up to 3 minutes. EGP provides for this eventuality by allowing a gateway to send one *unsolicited* Update—that is, an Update that is not in response to a Poll—each Poll interval. Cisco, however, does not support unsolicited Updates.

NOTE The **timers egp** command is also used to change the Hello interval. The format of the command is **timers egp** *hello polltime*.

Both the Poll and the Update messages include the address of a source network. For example, the Poll and Update messages in Example 1-5 show a source network of 192.168.16.0. The source network is the network from which all reachability information is measured—that is, all networks requested or advertised can be reached via a router attached to the source network. Although this network is usually the network to which the two neighbors are both attached, it is more accurately the network about which the Poll is requesting information, and the network about which the Update is supplying information. EGP is a purely classful protocol, and the source network—as well as the network addresses listed in the Updates—are always major class network addresses, and never subnets.

Following the source network address is a list of one or more routers and the networks that can be reached via those routers. The common characteristic of the routers on the list is that they are all attached to the source network. If a router on the list is not the EGP gateway that originated the Update, the router is an *indirect* or *third-party* neighbor.

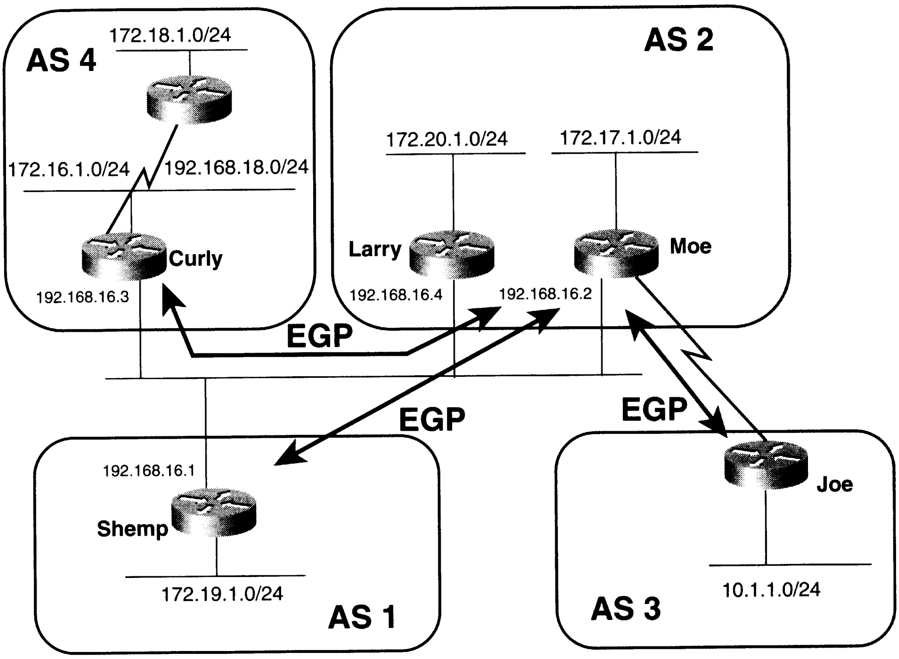
Figure 1-3 illustrates the concept of indirect EGP neighbors. One router, Moe, is a core gateway and is peered with three other gateways.

The debug messages in Example 1-5 are taken from Shemp, the router in AS1. Notice in the Update originated by Moe (192.168.16.2) that three networks are listed as reachable via Moe, but also, four networks are listed as reachable via Larry (192.168.16.4) and Curly (192.168.16.3). These two routers are Shemp's indirect neighbors, via Moe. Joe, in AS3, is not an indirect neighbor, because it is not attached to the source network. Its networks are merely advertised as being reachable via Moe.

The advertisement of indirect neighbors saves bandwidth on a common link, but more importantly, indirect neighbors increase efficiency by eliminating an unnecessary router hop. In Figure 1-3, for example, Shemp is not peered with any router other than Moe. In fact, Larry is not even speaking EGP, but is advertising its networks to Moe via RIP. Moe is performing a sort of "preemptive redirect" by informing Shemp of better next-hop routers than itself.

In fact, it is possible for an EGP Update to contain indirect neighbors only—that is, the originator might not include itself as a next hop to any network. In this scenario, the originator is a *route server*. It has learned reachability information from an IGP or from static routes, and it advertises this information to EGP neighbors without itself performing any packet-forwarding functions.

Figure 1-3 *Indirect EGP Neighbors*



From the perspective of an EGP gateway, a neighbor is either an *interior gateway* or an *exterior gateway.* A neighbor is an interior gateway if it is in the same AS, and it is an exterior gateway if it is in a different AS. In Figure 1-3, all the EGP gateways see all their neighbors as external gateways. If Larry were speaking EGP and peered with Moe, those two routers would see each other as interior gateways.

An EGP Update message includes two fields for describing whether the routers in its list are interior or exterior gateways (see the following section, "EGP Message Formats"). Looking at the first Update message in Example 1-5, you can see these fields just before the source network: IntGW=2 and ExtGW=1. The sum of these two fields tells how many routers are listed in the Update. All the interior gateways specified are listed first; therefore, if IntGW=2 and ExtGW=1, the first two routers listed are interior gateways and the last router listed is an exterior gateway. If you compare the Update message from 192.168.16.2 in Example 1-5 with Figure 1-3, you will see that the three networks reachable via Curly are listed last in the Update and are marked as exterior—that is, they are reachable via a gateway exterior to Moe. Because stub gateways cannot advertise networks outside of their own AS, only Updates from core gateways can include exterior gateways.

The EGP Update message associates a distance with each network it lists. The distance field is 8 bits, so the distance can range from 0 to 255. RFC 904 does not specify how the distance is to be interpreted, however, other than that 255 is used to indicate unreachable networks. Nor does the RFC define an algorithm for using the distance to calculate shortest inter-AS paths. Cisco chooses to interpret the distance as hops, as shown in Example 1-5. The default rules are very basic:

- A gateway advertises all networks within its own AS as having a distance of 0.
- A gateway advertises all networks within an AS other than its own as having a distance of 3.
- A gateway indicates that a network has become unreachable by giving it a distance of 255.

For example, you can see in Example 1-5 and Figure 1-3 that although network 172.20.0.0 is one router hop away from Moe, Moe is advertising the network with a distance of 0—the same distance as network 172.17.0.0, which is directly attached. Network 10.0.0.0 is also one router hop away, and network 172.18.0.0 is two hops away, but both are in different autonomous systems and are therefore advertised with a distance of 3. The point is that the distance used by EGP is virtually useless for determining the best path to a network.

Example 1-6 shows the routing table of Shemp and the route entries resulting from the Update in Example 1-5.

Example 1-6 *Shemp's Routing Table*

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

Gateway of last resort is not set

[redacted]
C    192.168.16.0 is directly connected, Ethernet0
[redacted]
     172.19.0.0 255.255.255.0 is subnetted, 1 subnets
C       172.19.1.0 is directly connected, Loopback0
Shemp#
```

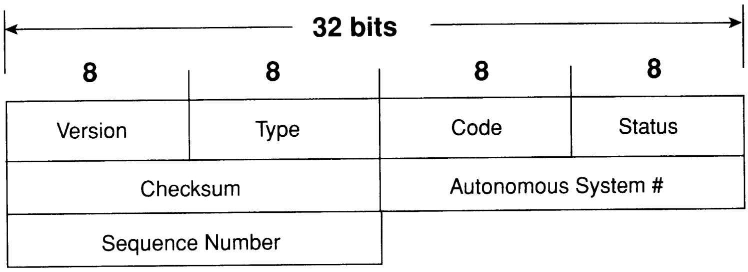
There are two points of interest in the routing table. First, notice that the EGP entries have an administrative distance of 140. This is higher than the administrative distance of any IGP (with the exception of External EIGRP), so a router will always choose an IGP route over an EGP advertisement of the same network.

Second, notice that the distances to each of the EGP-advertised networks are one higher than the distances shown in the Update of Example 1-5. Cisco's EGP process increments the distance by one, just as a RIP routing algorithm does.

EGP Message Formats

EGP uses five different formats to encode the ten message types shown in Table 1-1. All the messages have a common header, as shown in Figure 1-4.

Figure 1-4 *EGP Message Header*



The fields in the EGP message header are defined as follows:

- **Version**—Specifies the current EGP version number. If this number in a received message does not agree with the receiver's version number, the message is rejected. The version number of all current EGP implementations is 2.
- **Type**—Specifies which of the five message formats follows the header. Table 1-2 (which appears after this list) shows the ten EGP message types and the type number used by each.
- **Code**—Specifies the subtype. For example, if type = 5, the code specifies whether the message is a Hello or an I-Heard-You.
- **Status**—Varies according to the message type (as with the Code field). For example, a Neighbor Acquisition message can use the status to indicate whether it is active or passive, whereas a Neighbor Reachability message can use the Status field to indicate an Up or Down state.
- **Checksum**—The one's complement of the one's complement sum of the EGP message. This is the same error-checking algorithm used by IP, TCP, and UDP.
- **Autonomous System Number**—Specifies the AS of the message's originator.
- **Sequence Number**—Synchronizes message pairs (as described previously in this chapter). For example, an Update should always contain the same sequence number as the Poll to which it is responding.

Table 1-2 *EGP Message Types*

Type	Message
3	Neighbor Acquisition Request
3	Neighbor Acquisition Confirm
3	Neighbor Acquisition Refuse
3	Neighbor Cease
3	Neighbor Cease Acknowledgment
5	Hello
5	I-Heard-You
2	Poll
1	Update
8	Error

The Neighbor Acquisition Message (EGP Message Type 3)

Neighbor Acquisition messages are EGP message type 3. Table 1-3 shows the codes used to indicate the EGP message. Table 1.4, taken from RFC 904, shows the possible values of the Status field and the reasons a particular status might be used.

Table 1-3 *Codes Used with Message Type 3*

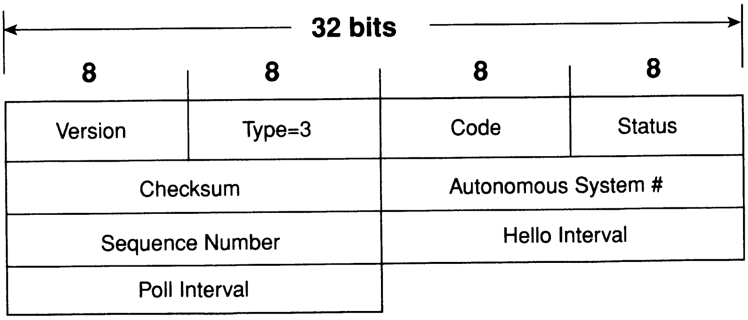
Code	Message
0	Neighbor Acquisition Request
1	Neighbor Acquisition Confirm
2	Neighbor Acquisition Refuse
3	Neighbor Cease
4	Neighbor Cease Acknowledgment

Figure 1-5 shows the format of the Neighbor Acquisition message. The Hello Interval and Poll Interval fields are present only in the Neighbor Acquisition Request (code 0) and Neighbor Acquisition Confirm (code 1) messages. All other Neighbor Acquisition messages are identical to the message header, with no other fields included.

Table 1-4 *Status Numbers Used with Message Type 3*

Status	Description	Use
0	Unspecified	When nothing else fits
1	Active mode	Request/Confirm only
2	Passive mode	Request/Confirm only
3	Insufficient resources	1. Out of table space 2. Out of system resources
4	Administratively prohibited	1. Unknown autonomous system 2. Use another gateway
5	Going down	1. Operator initiated stop 2. Abort timeout
6	Perimeter problem	1. Nonsense polling parameters 2. Unable to assume compatible mode
7	Protocol violation	Invalid command or response received in this state

Figure 1-5 *The Neighbor Acquisition Message*



- **Hello interval**—The minimum interval, in seconds, between Hellos that the originator is willing to accept. The Cisco default Hello interval is 60 seconds and can be changed with the command **timers egp**.
- **Poll interval**—The minimum interval, in seconds, between Polls that the originator is willing to accept. The Cisco default Poll interval is 180 seconds and can be changed with the command **timers egp**.

The Neighbor Reachability Message (EGP Message Type 5)

The Neighbor Reachability message (see Figure 1-6) is the EGP header, with Type = 5. No additional fields are included, because all necessary information is carried in the Code (see Table 1-5) and Status (see Table 1-6) fields.

Figure 1-6 *The Neighbor Reachability Message*

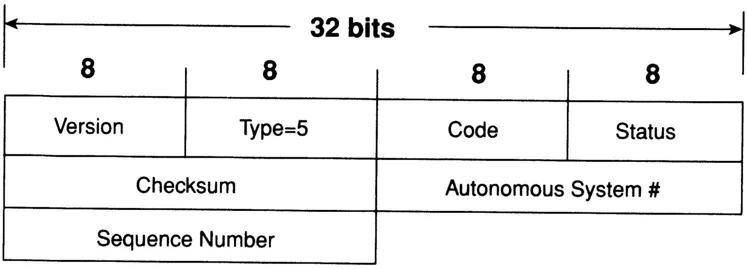


Table 1-5 *Codes Used with Message Type 5*

Code	Message
0	Hello
1	I-Heard-You

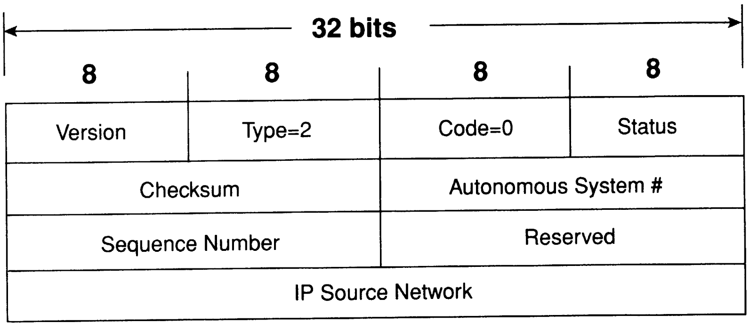
Table 1-6 *Status Numbers Used with Message Types 5 and 2*

Status	Description
0	Indeterminate
1	Up state
2	Down state

The Poll Message (EGP Message Type 2)

The only field that is added to the EGP header to create the Poll message (see Figure 1-7) is the IP Source Network, the network about which reachability information is being requested. The IP address encoded in this field is always a major Class A, B, or C network. The Code field is always 0, and the Status numbers used are the same as those described in Table 1-6. (RFC 888 shows the Status field as unused in the Poll and Error messages.)

Figure 1-7 *The Poll Message*



The Update Message (EGP Message Type 1)

As with the Poll message, the Code field of the Update is always 0. Table 1-7 shows the possible values of the Status field, which is the same as the values of Table 1-6 with the exception of the Unsolicited value.

Table 1-7 *Status Numbers Used with Message Type 1*

Status	Description
0	Indeterminate
1	Up state
2	Down state
128	Unsolicited

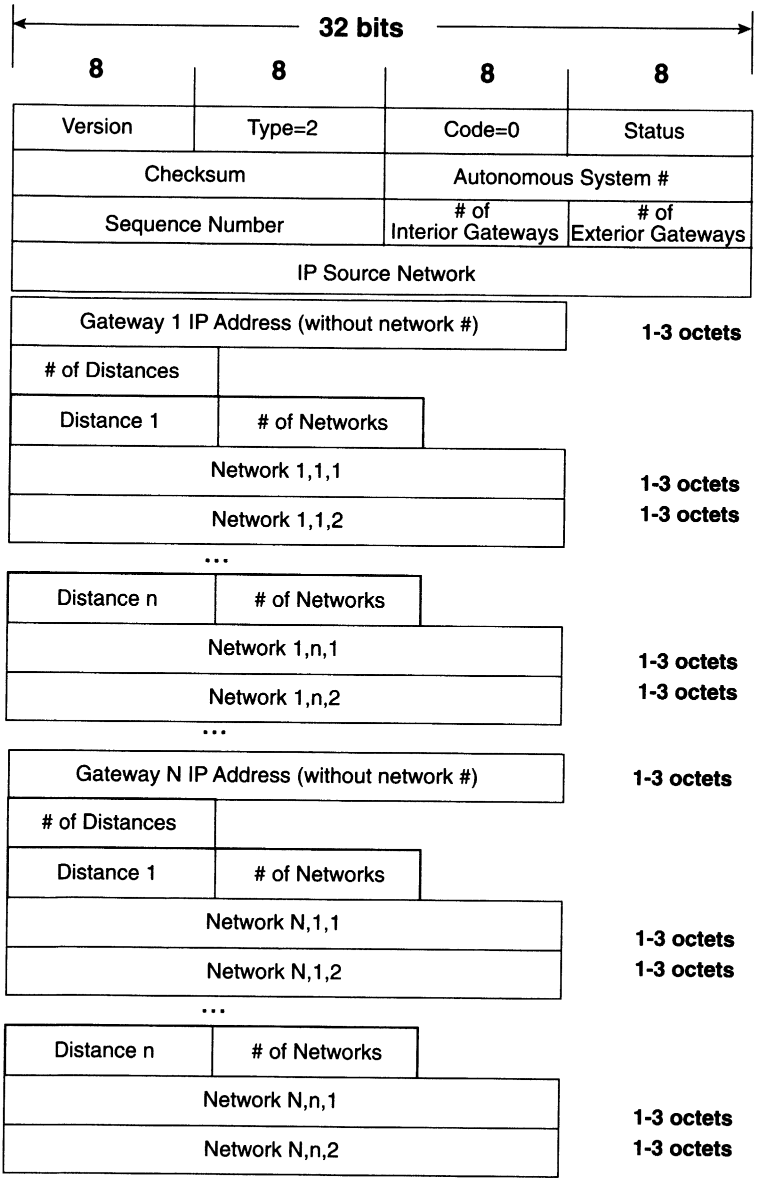
The most significant bit of the Status field is the Unsolicited bit; if the bit is set (giving the field a value of 128), the Update is unsolicited. The Unsolicited bit can be used in combination with any of the other Status values.

The Update message includes a four-level hierarchy of lists. Figure 1-8 shows the format of the Update message and how the hierarchy of lists is organized.

At the highest level of the hierarchy is a list of all the routers that are directly attached to the source network. The number of gateways on the list is specified by the sum of the # of Interior Gateways and the # of Exterior Gateways fields.

At the next level, interior gateways are distinguished from exterior gateways. All interior gateways, including the originator, are listed first. If there are any exterior gateways, they are listed after the interior gateways.

Figure 1-8 *The Update Message*



At the third layer of the hierarchy, each listed gateway has a list of distances. As with the interior and exterior gateways, a field specifies the number of distances on the list.

Finally, for each listed distance there is a list of networks that can be reached at that distance and via that gateway. A field is included to specify the number of networks on the list.

The complete descriptions for the fields of the Update message format are as follows:

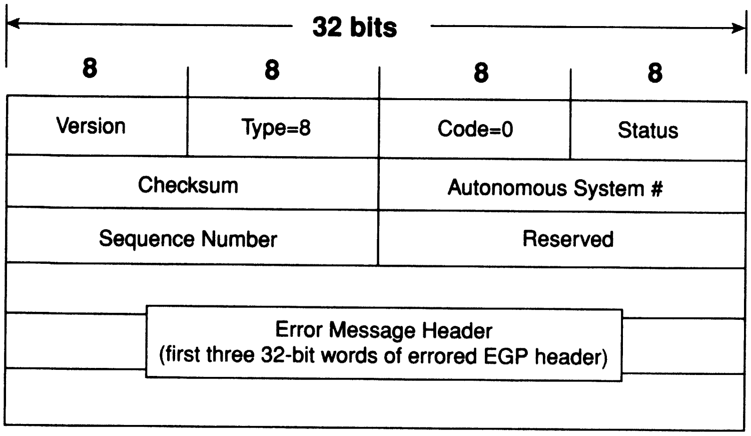
- **# of Interior Gateways**—Specifies the number of interior gateways on the list.
- **# of Exterior Gateways**—Specifies the number of exterior gateways following the list of interior gateways. The sum of this field and the # of Interior Gateways, shown as N in Figure 1-8, is the total number of gateways listed in the Update.
- **IP Source Network**—Specifies the network about which reachability information is being supplied. That is, all networks listed in the Update are reachable via a gateway attached to this network. The IP address encoded in this field is always a major Class A, B, or C network.
- **Gateway IP Address**—Specifies the address of a gateway attached to the source network. Only the host portion of the major Class A, B, or C address is listed; as a result, the length of the field is variable from 1 octet for a Class C address to 3 octets for a Class A address. The network portion of the address is already known from the IP Source Network field.
- **# of Distances**—Specifies the total number of distances being advertised under the listed gateway.
- **Distance**—Specifies a particular distance advertised under the listed gateway.
- **# of Networks**—Specifies the total number of networks advertised under the listed distance of the listed gateway.
- **Network**—Specifies the IP address of the network being advertised. In Figure 1-8, each network is shown as belonging to a particular gateway, a particular distance, and a particular order in the network list. Like the Gateway IP Address field, the Network field is variable. Unlike the Gateway IP Address field, the Network field lists the network portion rather than the host portion of a major Class A, B, or C address.

The Error Message (EGP Message Type 8)

A gateway can send an Error message (see Figure 1-9) at any time to notify a sender of a bad EGP message or an invalid field value. The Code field of the error message is always 0, and the Status is one of the values described in Table 1-7.

NOTE RFC 888 shows the Status field in the Error message (like in the Poll message) as unused. RFC 904 specifies the uses shown in Table 1-7.

Figure 1-9 *The Error Message*



The originator of the Error message can use an arbitrary value as the sequence number. Table 1-8, which is taken from RFC 904, describes the possible values of the Reason field. The Error message header is the first 12 octets of the EGP message that prompted the Error message.

Table 1-8 *Values of the Reason Field of the Error Message*

Reason Field Value	Description	Use
0	Unspecified	When nothing else fits.
1	Bad EGP header format	1. Bad message length. 2. Invalid Type, Code, or Status field.
2	Bad EGP Data field format	1. Nonsense polling rates (Request/Confirm). 2. Invalid Update message format. 3. Response IP Network Address field does not match command (Update).
3	Reachability info unavailable	No information available on the network specified in the IP Network Address field (Poll).
4	Excessive polling rate	1. Two or more Hello messages received within the Hello interval. 2. Two or more Poll messages received within the Poll interval. 3. Two or more Request messages received within some (reasonably short) interval.
5	No response	No Update received for the Poll within some (reasonably long) interval.

Shortcomings of EGP

The fundamental problem with EGP is its inability to detect routing loops. Because there is an upper boundary on the distance EGP uses (255), you might be tempted to say that counting to infinity is at least a rudimentary loop-detection mechanism. It is, but the high limit combined with the typical Poll interval makes counting to infinity useless. Given a default Poll interval of 180 seconds, EGP peers could take almost 13 hours to count to infinity.

As a result, EGP must be run on an engineered loop-free topology. Although that was not a problem in 1983, when EGP was intended merely to connect stub gateways to the ARPANET backbone, the creators of EGP already foresaw that such a limited topology would soon become inadequate. The autonomous systems making up the Internet would need to evolve into a less structured mesh, in which many autonomous systems could serve as transit systems for many other autonomous systems.

With the advent of the NSFnet, the limitations of EGP became more pronounced. Not only were there now multiple backbones, but there were acceptable use policies concerning what traffic could traverse what backbone. Because EGP cannot support sophisticated policy-based routing, interim solutions had to be engineered[4].

Another major problem with EGP is its inability to adequately interact with IGPs to determine a shortest route to a network in another AS. For example, EGP distances do not reliably translate into RIP hop counts. If the EGP distance causes the hop count to exceed 15, RIP declares the network unreachable. Other shortcomings of EGP include its susceptibility to failures when attempting to convey information on a large number of networks, and its vulnerability to intentionally or unintentionally inaccurate network information.

Last but certainly not least, EGP can be mind-numbingly slow to advertise a network change. The section "Troubleshooting EGP" includes an example in which a network in an EGP-connected AS becomes unreachable. As the example demonstrates, almost an hour passes before a gateway four hops away determines that the network has gone down.

Several attempts were made to create an EGPv3, but none were successful. In the end, EGP was abandoned in favor of an entirely new inter-AS protocol, BGP. As a result, Exterior Gateway Protocol is now not only the name of a protocol, but the name of a class of protocols, giving rise to the notion of an EGP named EGP. Nonetheless, the legacy of EGP is still with us today in the form of autonomous systems and inter-AS routing.

Configuring EGP

You can configure EGP on a router in four basic steps:

Step 1 Specify the router's AS with the command **autonomous-system**.

Step 2 Start the EGP process and specify the neighbor's AS with the command **router egp**.

Step 3 Specify the EGP neighbors with the **neighbor** command.

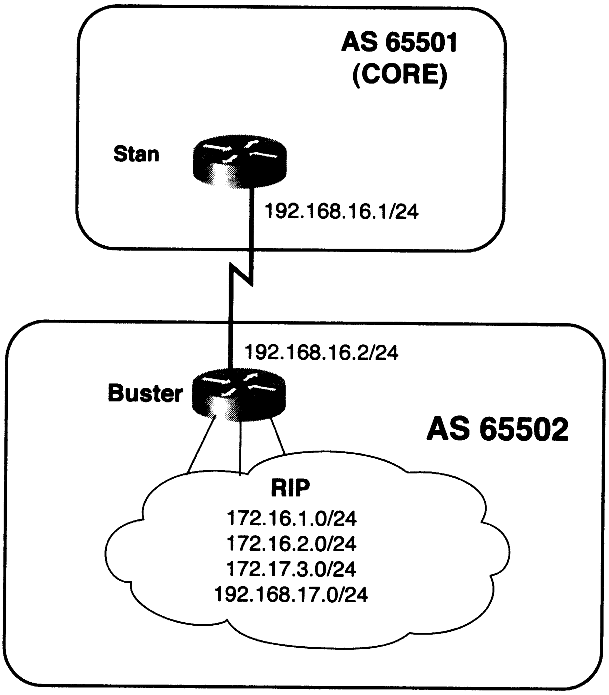
Step 4 Specify what networks are to be advertised by EGP.

The first three steps are demonstrated in the first case study, along with several approaches to Step 4.

Case Study: An EGP Stub Gateway

Figure 1-10 shows an EGP stub gateway in AS 65502, connected to a core gateway in AS 65501. The IGP of the stub AS is RIP.

Figure 1-10 *EGP Stub Gateway Advertises the Interior Networks of AS 65502 to the Core Gateway*



Example 1-7 shows the initial configuration of the stub gateway.

Example 1-7 *Stub Gateway Configuration for Figure 1-10*

```
autonomous-system 65502
!
router rip
 redistribute connected
 redistribute egp 65501 metric 5
 network 172.16.0.0
!
router egp 65501
 neighbor 192.168.16.1
```

Notice that the *local AS* (LAS) is specified by the **autonomous-system** statement, and the *far AS* (FAS) is specified by the **router egp** statement. An EGP process cannot be configured until the LAS is configured. The EGP process is told where to find its peer by the **neighbor** statement. Buster's routing table (see Example 1-8) contains both EGP route entries learned from the core gateway and RIP entries learned from the interior neighbors.

Example 1-8 *Buster's Routing Table Shows Entries Learned from the EGP Neighbor and from the Interior RIP Neighbors*

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

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.16.1, 00:02:12, Serial3
C    192.168.16.0 is directly connected, Serial3
R    192.168.17.0 [120/1] via 172.16.1.2, 00:00:05, Ethernet0
E    192.168.19.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
E    192.168.20.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
E    192.168.21.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
E    192.168.22.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
     172.16.0.0 255.255.255.0 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Ethernet0
R       172.16.2.0 [120/1] via 172.16.1.2, 00:00:05, Ethernet0
R    172.17.0.0 [120/1] via 172.16.1.2, 00:00:05, Ethernet0
Buster#
```

The EGP-learned routes are being redistributed into RIP with a metric of 5 (see Example 1-9).

Example 1-9 *Routing Table from a Router Interior to AS 65502 Shows the Redistributed EGP Routes*

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

Gateway of last resort is not set

R    10.0.0.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.16.0 [120/1] via 172.16.1.1, 00:00:13, Ethernet0
C    192.168.17.0 is directly connected, Ethernet3
R    192.168.19.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.20.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.21.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.22.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
     172.16.0.0 255.255.255.0 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Ethernet0
C       172.16.2.0 is directly connected, Ethernet1
     172.17.0.0 255.255.255.0 is subnetted, 1 subnets
C       172.17.3.0 is directly connected, Ethernet2
Charlie#
```

Notice that directly connected networks are also being redistributed into RIP. This configuration is necessary to advertise network 192.168.16.0 into the LAS; split horizon prevents Stan from advertising the network to Buster via EGP. An alternative configuration is to add a **network 192.168.16.0** statement to the RIP configuration, along with a **passive-interface** statement to keep RIP broadcasts off of the inter-AS link.

As Buster's EGP configuration stands so far, network information is being received from the core, but no interior networks are being advertised to the core (see Example 1-10).

Example 1-10 *Stan's Routing Table Shows That None of the Interior Networks from AS 65502 Are Being Learned from Buster*

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

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.18.2, 00:01:56, Serial1
C    192.168.16.0 is directly connected, Serial0
C    192.168.18.0 is directly connected, Serial1
E    192.168.19.0 [140/1] via 192.168.18.2, 00:01:57, Serial1
E    192.168.20.0 [140/4] via 192.168.18.2, 00:01:57, Serial1
E    192.168.21.0 [140/4] via 192.168.18.2, 00:01:57, Serial1
E    192.168.22.0 [140/1] via 192.168.18.2, 00:01:57, Serial1
Stan#
```

One option for configuring EGP to advertise the interior networks is to add a **redistribute rip** statement. However, there are hazards associated with mutual redistribution. The danger is more pronounced when there are topological loops or multiple redistribution points, but even a simple design like the one in Figure 1-10 can be vulnerable to route feedback. For safety, route filters should always be used with mutual redistribution configurations to ensure that no interior network addresses are accepted from the exterior gateway, and no exterior addresses are advertised to the exterior gateway. The problems associated with mutual redistribution are introduced in *Routing TCP/IP, Volume I* and are discussed in further detail in Chapter 2, "Introduction to Border Gateway Protocol 4," and Chapter 3, "Configuring and Troubleshooting Border Gateway Protocol 4," of this book.

A better approach to configuring EGP to advertise interior networks is to use the **network** statement. When used with EGP or BGP, the **network** statement has a different function from when used with an IGP configuration. For example, the **network 172.16.0.0** statement under Buster's RIP configuration instructs the router to enable RIP on any interface that has an IP address in the major network 172.16.0.0. When used in conjunction with an inter-AS protocol, the **network** statement tells the protocol what network addresses to advertise. Example 1-11 shows Buster's configuration to advertise all the networks in AS 65502.

Example 1-11 *Buster Configuration to Advertise All Networks in AS 65502*

```
autonomous-system 65502
!
router rip
 redistribute connected
 redistribute egp 65501 metric 5
 network 172.16.0.0
!
router egp 65501
 network 172.16.0.0
 network 172.17.0.0
 network 192.168.17.0
 neighbor 192.168.16.1
```

Example 1-12 shows Stan's routing table after the **network** statements have been added to Buster's EGP configuration.

The advantage of using the **network** statement under EGP rather than redistribution is somewhat akin to the advantage of using static routes rather than a dynamic routing protocol: Both allow precise control over network reachability. In the case of EGP, the precision is limited by EGP's classfulness. Although you can keep a major network "private" by not specifying it in a **network** statement, the same cannot be said of individual subnets. Refer back to Example 1-8, which shows that Buster's routing table contains subnets 172.16.1.0/24 and 172.16.2.0/24. Reexamining the EGP Update message format in Figure 1-8, you will recall that the Update carries only the major class portion of the IP network: the first octet of a Class A network, the first two octets of a Class B network, and the first three octets of a Class C network. Therefore, the **network** statement under EGP can specify only major networks.

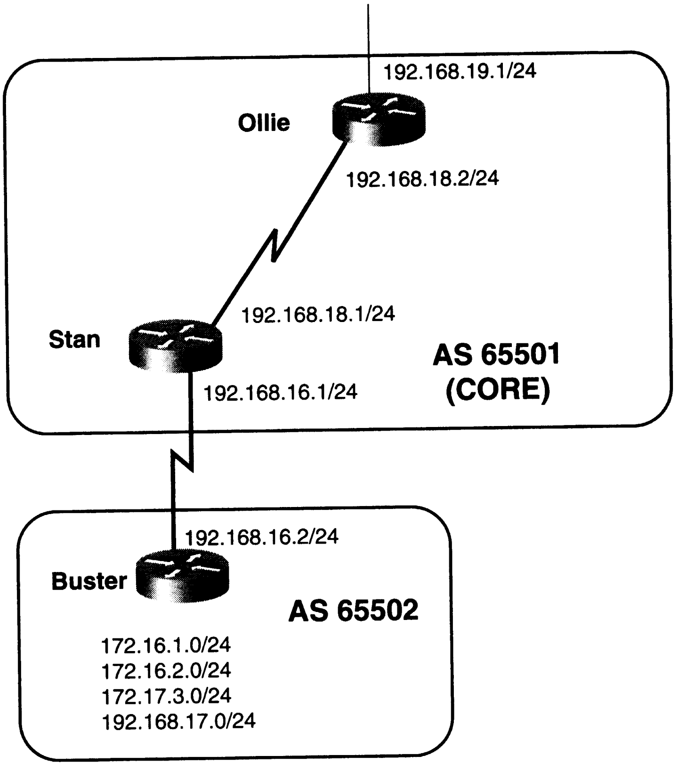
Example 1-12 *Buster Is Now Advertising the Interior Networks of AS 65502 to Stan*

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

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.18.2, 00:00:27, Serial1
C    192.168.16.0 is directly connected, Serial0
E    192.168.17.0 [140/1] via 192.168.16.2, 00:01:38, Serial0
C    192.168.18.0 is directly connected, Serial1
E    192.168.19.0 [140/1] via 192.168.18.2, 00:00:27, Serial1
E    192.168.20.0 [140/4] via 192.168.18.2, 00:00:27, Serial1
E    192.168.21.0 [140/4] via 192.168.18.2, 00:00:27, Serial1
E    192.168.22.0 [140/1] via 192.168.18.2, 00:00:27, Serial1
E    172.16.0.0 [140/1] via 192.168.16.2, 00:01:39, Serial0
E    172.17.0.0 [140/1] via 192.168.16.2, 00:01:39, Serial0
Stan#
```

Case Study: An EGP Core Gateway

By definition, an EGP core gateway can peer with multiple neighbors within multiple far autonomous systems and can pass network information from one FAS to another FAS. Because of this, the configuration of a core gateway differs slightly. Figure 1-11 shows a core router, Stan, which is peered with a router in a FAS (Buster) and a router within its LAS (Ollie).

Figure 1-11 *Core Router Stan Must Peer with Both Remote Neighbor Buster and Local Neighbor Ollie*



Example 1-13 demonstrates the EGP configuration of Stan in Figure 1-11 .

Example 1-13 *Core Gateway Configuration for Network Topology in Figure 1-11*

```
autonomous-system 65501
!
router egp 0
 network 192.168.16.0
 neighbor any
```

The LAS is still specified with the **autonomous-system** command, but the FAS is not specified by the **router egp** command. Instead, an AS number of 0 is used to specify any AS. Likewise, neighbors are specified with a **neighbor any** command, to respond to any neighbor that sends Acquisition messages. The **neighbor any** command implicitly configures neighbors, whereas the **neighbor** command explicitly configures neighbors. Core gateways can have explicitly configured neighbors, but the implicit **neighbor any** makes life simpler when there are a large number of neighbors, as might be expected at a core gateway.

Of course, at least one neighbor must have an explicit neighbor configuration; two neighbors cannot discover each other if they both have a **neighbor any** command. Example 1-14 shows the configuration for the neighbor Ollie in Figure 1-11.

Example 1-14 *Neighbor Configuration for Ollie in the Network Topology of Figure 1-11*

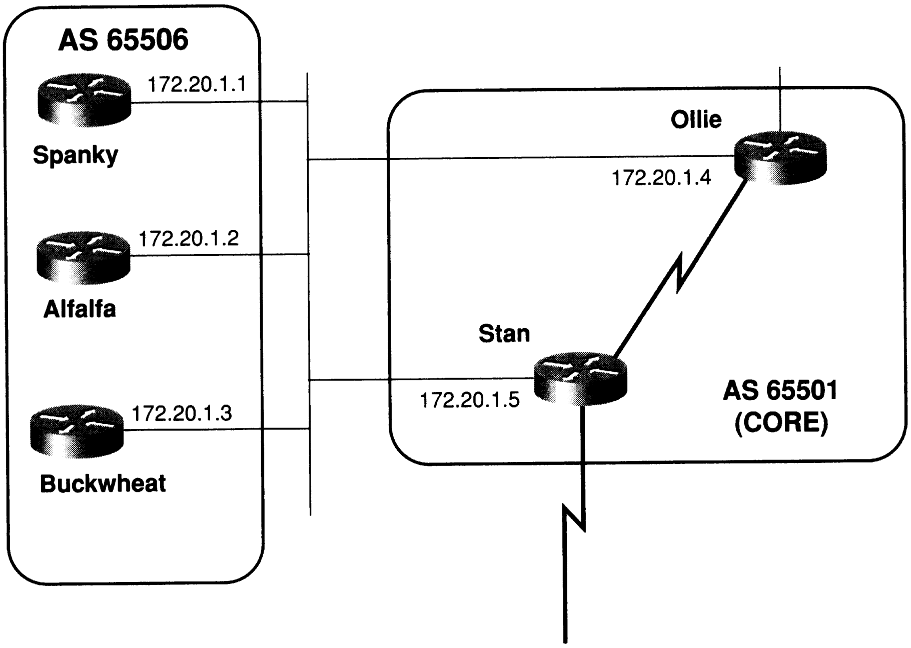
```
autonomous-system 65501
!
router egp 0
 network 192.168.19.0
 neighbor 192.168.18.1
 neighbor any
```

Although Ollie still picks up its external neighbors with the **neighbor any** command, Stan's address is explicitly configured. If it were not, Stan and Ollie would be unaware of each other's existence.

With the configuration in Example 1-14, the core gateway will pass reachability information about networks external to its own AS to every other external AS. The core gateway will not, however, pass information about the networks in its own AS. You can see in Buster's routing table of Example 1-8, for instance, that there is no entry for network 192.168.18.0. If the interior networks are to be advertised, Stan must have a **network** statement for each network to be advertised. The only **network** statement shown is for 192.168.16.0, which allows Ollie to receive information about that network. Look again at Buster's routing table. Notice that there is an entry for network 192.168.19.0. This entry is the result of the **network 192.168.19.0** statement in Ollie's configuration in Example 1-14.

What happens if a core should not peer with every EGP-speaking neighbor? In Figure 1-12, the three routers in AS 65506 are all running EGP, but Stan should peer with only Spanky and Buckwheat. Alfalfa should peer with Ollie. Of course, the core administrator could trust the administrator of AS 65506 to set up the correct peering with **neighbor** statements, but trust is seldom good enough in inter-AS routing.

Figure 1-12 *Spanky and Buckwheat Must Peer Only with Stan, Whereas Alfalfa Must Peer Only with Ollie*



In this example, all three gateways in AS 65506 have **neighbor** statements for both Stan and Ollie. To regulate the peering, an access list is used with the **neighbor any** statement, as demonstrated in the configuration for Stan in Example 1-15.

Example 1-15 *Regulating Peering with Access Lists Using the* **neighbor any** *Command*

```
autonomous-system 65501
!
router egp 0
 network 192.168.16.0
 neighbor any 10
!
access-list 10 deny 172.20.1.2
access-list 10 permit any
```

In Example 1-15, the **neighbor any** statement contains a reference to access list 10, which denies Alfalfa (172.20.1.2) and permits all other neighbors. A similar configuration at Ollie denies Spanky and Buckwheat and permits all other neighbors. Example 1-16 shows the results of this configuration.

Example 1-16 *The* **show ip egp** *Command Displays Information About EGP Neighbors*

```
Stan#show ip egp
Local autonomous system is 65501

 EGP Neighbor     FAS/LAS   State      SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.18.2    65501/65501 UP    10       3      4    60   180   4 Temp, Act
*192.168.16.2    65502/65501 UP  3:20      39     39    60   180   4 Temp, Act
```

[redacted]

```
Stan#
```

```
Ollie#show ip egp
Local autonomous system is 65501

 EGP Neighbor     FAS/LAS   State      SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.18.1    65501/65501 UP     9       4      3    60   180   4 Perm, Pass
*172.20.1.2      65506/65501 UP    13       5      5    60   180   4 Temp, Act
```

[redacted]

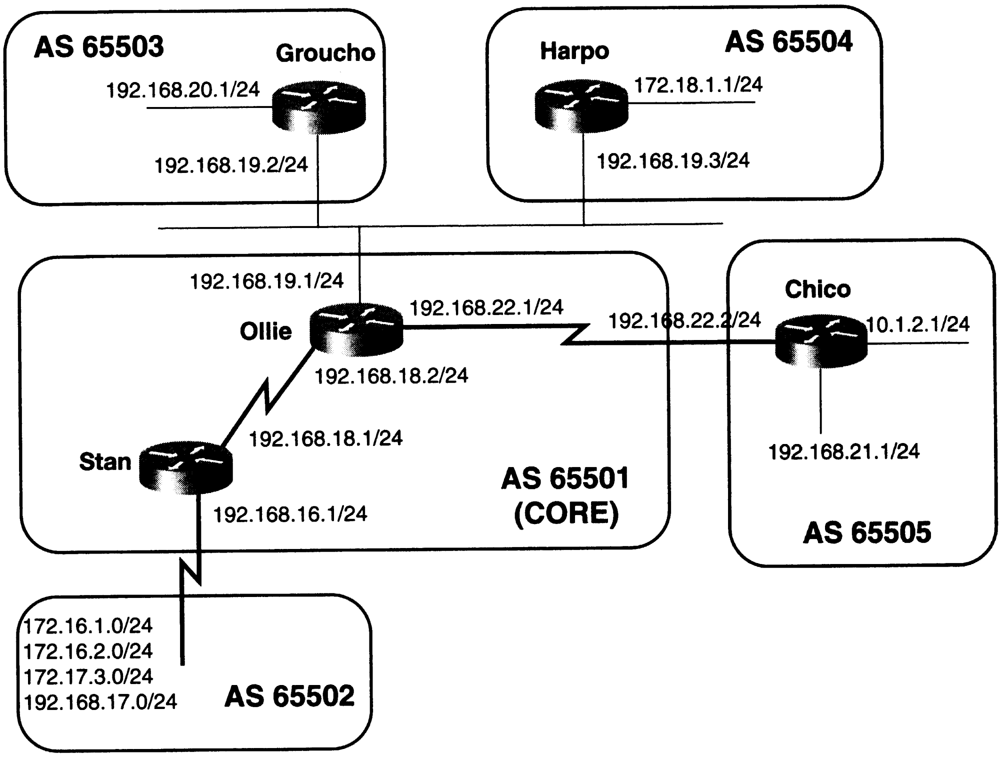
Using the **show ip egp** command with Stan and Ollie shows that Ollie is peered with Alfalfa and Stan is peered with Spanky and Buckwheat.

NOTE The details of the fields displayed by the **show ip egp** command are discussed in the section "Troubleshooting EGP." For now, the addresses of the neighbors are of interest.

Case Study: Indirect Neighbors

In Figure 1-13, three stub gateways (Groucho, Harpo, and Chico) are connected to the core gateway named Ollie. Groucho and Harpo, in separate autonomous systems, share a common Ethernet and can therefore be configured as indirect or third-party neighbors.

Figure 1-13 *EGP Indirect Neighbors*



Groucho and Harpo cannot exchange EGP information directly, but they can route packets directly to each other if Ollie advertises them as indirect neighbors. Example 1-17 shows the configuration for Ollie.

Example 1-17 *Advertising Indirect EGP Neighbors to One Another Enables the Routing of Packets Between Indirect EGP Neighbors*

```
autonomous-system 65501
!
router egp 0
 network 192.168.19.0
 network 192.168.22.0
 network 192.168.18.0
 neighbor 192.168.19.3
 neighbor 192.168.19.3 third-party 192.168.19.2
 neighbor 192.168.19.2

 neighbor 192.168.19.2 third-party 192.168.19.3
 neighbor 192.168.18.1
 neighbor any
```

In the configuration in Example 1-17, Groucho and Harpo are explicitly configured as neighbors. Following the **neighbor** statements for the two routers are **neighbor third-party** statements. These entries specify the neighbor in question and then specify that gateway's indirect neighbor on the shared Ethernet. Notice that Chico, which is not on the shared Ethernet, falls under the **neighbor any** statement. Example 1-18 shows the core gateway's indirect neighbors recorded as Third Party.

Example 1-18 *Displaying Core Gateway Indirect Neighbors*

```
Ollie#show ip egp
Local autonomous system is 65501

 EGP Neighbor     FAS/LAS     State     SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.19.3    65504/65501 UP 5TE        8    249     60   180    4 Perm, Act
*192.168.19.2    65503/65501 UP 5TE        8   3177     60   180    4 Perm, Act
*192.168.18.1    65501/65501 UP 5TE        9   3192     60   180    4 Perm, Pass
*192.168.22.2    65505/65501 UP 5TE        5   3170     60   180    4 Temp, Act
 EGP Neighbor     Third Party
*192.168.19.3     192.168.19.2
*192.168.19.2     192.168.19.3
Ollie#
```

Ollie's EGP neighbor table indicates that Groucho and Harpo (192.168.19.2 and 192.168.19.3, respectively) have been configured as indirect neighbors of each other.

Harpo's routing table (see Example 1-19) shows the results of the indirect neighbor configuration. Rather than pointing to the core gateway as the next hop to network 192.168.20.0 in AS 65503, the next hop points directly to Groucho (192.168.19.2).

Example 1-19 *Routing Table Displays Next-Hop Routes to Indirect Neighbors*

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

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.19.1, 00:02:21, Ethernet0
E    192.168.16.0 [140/4] via 192.168.19.1, 00:02:21, Ethernet0
E    192.168.17.0 [140/4] via 192.168.19.1, 00:02:21, Ethernet0
E    192.168.18.0 [140/1] via 192.168.19.1, 00:02:21, Ethernet0
C    192.168.19.0 is directly connected, Ethernet0
[redacted]
E    192.168.21.0 [140/4] via 192.168.19.1, 00:02:22, Ethernet0
E    192.168.22.0 [140/1] via 192.168.19.1, 00:02:22, Ethernet0
E    172.16.0.0 [140/4] via 192.168.19.1, 00:02:22, Ethernet0
E    172.17.0.0 [140/4] via 192.168.19.1, 00:02:22, Ethernet0
     172.18.0.0 255.255.255.0 is subnetted, 1 subnets
C       172.18.1.0 is directly connected, Loopback0
Harpo#
```

Harpo's routing table in Example 1-19 shows that network 192.168.20.0 is directly reachable via next hop 192.168.19.2. Without the indirect neighbor configuration, Harpo would have to use 192.168.19.1 as the next hop.

Case Study: Default Routes

EGP can be configured to advertise a default route in addition to more specific routes. If an AS has only a single exterior gateway, a default route is usually more efficient than a full list of exterior routes. Memory and processing cycles are conserved on the router, and bandwidth is saved on the link.

To advertise a default route into AS 65502, as illustrated previously in Figure 1-13, you configure Stan as demonstrated in Example 1-20.

Example 1-20 *Advertising a Default Route*

```
router egp 0
 network 192.168.16.0
 neighbor any
 default-information originate
 distribute-list 20 out Serial0
!
access-list 20 permit 0.0.0.0
```

The **default-information originate** command is used to generate the default route. Unlike in other protocols, when the command is used with EGP, there are no optional statements. Notice, too, that a route filter has been added, which permits only the default route to be advertised out of Stan's S0 interface to AS 65502. Without this filter, the default and all more-specific networks would be advertised. Example 1-21 shows the results of the configuration.

Example 1-21 *192.168.20.1 Is Reachable as a Result of the Default Route*

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

Gateway of last resort is 192.168.16.1 to network 0.0.0.0

C    192.168.16.0 is directly connected, Serial3
R    192.168.17.0 [120/1] via 172.16.1.2, 00:00:20, Ethernet0
     172.16.0.0 255.255.255.0 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Ethernet0
R       172.16.2.0 [120/1] via 172.16.1.2, 00:00:21, Ethernet0
R    172.17.0.0 [120/1] via 172.16.1.2, 00:00:21, Ethernet0
E*   0.0.0.0 0.0.0.0 [140/4] via 192.168.16.1, 00:00:46, Serial3
```

Example 1-21 *192.168.20.1 Is Reachable as a Result of the Default Route (Continued)*

```
Buster#ping 192.168.20.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.20.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/66/76 ms
Buster#
```

The routing table of AS 65502's exterior gateway shows that the core gateway is advertising only a default route, by which all the exterior networks in Figure 1-13 are reached.

Troubleshooting EGP

The earlier section "Shortcomings of EGP" discussed several reasons why EGP cannot be used in complex inter-AS topologies. An unexpected benefit is that by forcing a simple topology, EGP is easy to troubleshoot.

As with any routing protocol, the first step in troubleshooting EGP is examining the routing tables. If a required route is missing or an unwanted route is present, the routing tables should lead you to the source of the problem. Because the EGP metrics have very little meaning, using the routing tables for troubleshooting is greatly simplified in comparison with other routing protocols.

When examining EGP configurations, remember that the gateway must have some sort of **neighbor** statement—either explicit or **neighbor any**—for every neighbor. Understanding the use of the **network** statement, and how it differs from the **network** statement used with IGPs, is also important.

The **debug ip egp transactions** command, used several times in the "Operation of EGP" section, is a very useful troubleshooting tool. The output of this command reveals all the important information in all the EGP messages being exchanged between neighbors.

Interpreting the Neighbor Table

An examination of the EGP neighbor table using **show ip egp** will tell you about the state and configuration of a gateway's neighbors. Example 1-18 displayed the output of this command; Example 1-22 shows some additional output from the **show ip egp** command that examines Stan's neighbor table.

Example 1-22 **show ip egp** *Command Output Displays Information Useful for Troubleshooting EGP Peers*

```
Stan#show ip egp
Local autonomous system is 65501

 EGP Neighbor      FAS/LAS     State    SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.18.2    65501/65501 UP  2:08   3227     43    60   180   4 Temp, Act
*192.168.16.2    65502/65501 UP  6d17   3233   3233    60   180   4 Temp, Act
Stan#
```

You can see in Stan's neighbor table that neighbor 192.168.18.2 is an interior neighbor, because the FAS and LAS are the same (65501). The state of the neighbor is shown, as is its uptime. Whereas 192.168.18.2 has been up for just over 2 hours, 192.168.16.2 has been up for 6 days and 17 hours. The present sequence number being used by the gateway for each neighbor is shown, as is the present sequence number being used by the neighbor.

After the Hello and Poll intervals, the number of neighbor reachability messages that have been received in the past four Hello intervals is recorded. This number is used to determine whether a neighbor should be declared Up or Down, based on two values known as the *j* and *k* thresholds. The *j* threshold specifies the number of neighbor reachability messages that must be received during four Hello intervals before a Down neighbor is declared Up. The *k* threshold specifies the minimum number of neighbor reachability messages that must be received within four Hello intervals to prevent an Up neighbor from being declared Down. The thresholds, shown in Table 1-9, differ for active and passive neighbors.

Table 1-9 *EGP* j *and* k *Thresholds*

Threshold	Active	Passive	Description
j	3	1	Neighbor Up threshold
k	1	4	Neighbor Down threshold

The next field (Flags) in Example 1-22 specifies whether the neighbor is permanent or temporary. Permanent neighbors are neighbors that have been explicitly configured with a **neighbor** statement, whereas temporary neighbors have been implicitly peered under the **neighbor any** statement. In Example 1-22, you can see that both of Stan's neighbors are temporary; this fits with the configuration of Stan discussed earlier, in which there is a single **neighbor any** statement. Comparing Example 1-22 with Example 1-18, you might find it interesting that although Stan sees Ollie (192.168.18.2) as a temporary neighbor, Ollie sees Stan (192.168.18.1) as a permanent neighbor. An examination of Ollie's configuration in Example 1-23 shows why.

Example 1-23 *Neighbor Configuration of Router Ollie*

```
autonomous-system 65501
!
router egp 0
 network 192.168.19.0
 network 192.168.22.0
 network 192.168.18.0
 neighbor 192.168.19.3
 neighbor 192.168.19.3 third-party 192.168.19.2
 neighbor 192.168.19.2
 neighbor 192.168.19.2 third-party 192.168.19.3
 neighbor 192.168.18.1
 neighbor any
```

The explicit **neighbor 192.168.18.1** causes Ollie to classify Stan as a permanent neighbor.

The last field indicates whether the local router is the active or the passive neighbor. Example 1-22 shows that Stan is the active neighbor for both of its peer relationships, so you would expect Ollie to show that it is the passive neighbor. Example 1-18 bears out this assumption and also indicates that Ollie is the active neighbor for all of its other peer relationships. This is also to be expected, because AS 65501 is lower than the other AS numbers.

Case Study: Converging at the Speed of Syrup

A distinct characteristic of EGP is that nothing happens quickly. The neighbor acquisition process is slow, and the advertisement of network changes is almost glacial. As a result, you might sometimes mistakenly assume that there is a problem where none exists (except for the problematic nature of EGP itself). For example, suppose users in AS 65503 of Figure 1-13 complain that they cannot reach network 172.17.0.0 in AS 65502. When you examine Groucho's routing table, there is a route to 172.17.0.0 (see Example 1-24), but a ping to a known address on that network fails. You might be led to believe that traffic to the network is being misrouted, or *black holed.*

A clue to the problem is shown in Ollie's routing table (see Example 1-25). Notice that a new update for network 172.17.0.0 has not been received in more than 16 minutes, but the route entry for the network is still valid and is still being advertised to Ollie's neighbors.

Example 1-24 *Groucho in Figure 1-13 Has a Route to 172.17.0.0, but the Network Is Unreachable*

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

Gateway of last resort is 192.168.19.1 to network 0.0.0.0

E    10.0.0.0 [140/4] via 192.168.19.1, 00:01:23, Ethernet0
E    192.168.16.0 [140/4] via 192.168.19.1, 00:01:23, Ethernet0
E    192.168.17.0 [140/4] via 192.168.19.1, 00:01:23, Ethernet0
C    192.168.19.0 is directly connected, Ethernet0
C    192.168.20.0 is directly connected, Loopback0
E    192.168.21.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0
E    192.168.22.0 [140/1] via 192.168.19.1, 00:01:24, Ethernet0
E    172.16.0.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0
[redacted]
E    172.18.0.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0
E*   0.0.0.0 0.0.0.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0

Groucho#ping 172.17.3.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.3.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Groucho#
```

Example 1-25 *New Network Updates Are Not Being Advertised*

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

Gateway of last resort is not set

E    10.0.0.0/8 [140/1] via 192.168.22.2, 00:01:20, Serial1
E    192.168.16.0/24 [140/1] via 192.168.18.1, 00:01:13, Serial0
E    192.168.17.0/24 [140/4] via 192.168.18.1, 00:16:14, Serial0
C    192.168.18.0/24 is directly connected, Serial0
C    192.168.19.0/24 is directly connected, Ethernet0
E    192.168.20.0/24 [140/1] via 192.168.19.2, 00:02:06, Ethernet0
E    192.168.21.0/24 [140/1] via 192.168.22.2, 00:01:21, Serial1
C    192.168.22.0/24 is directly connected, Serial1
E    172.16.0.0/16 [140/4] via 192.168.18.1, 00:01:13, Serial0
[redacted]
E    172.18.0.0/16 [140/1] via 192.168.19.3, 00:01:59, Ethernet0
Ollie#
```

Stan has not included network 172.17.0.0 in the past five update messages to Ollie. There is no black hole problem here; network 172.17.0.0 has just become unreachable due to a disconnected Ethernet interface on a router in AS 65502. EGP will not declare a route down until it has failed to receive six consecutive updates for the route. Couple this with an update interval of 180 seconds, and you will see that EGP will take 18 minutes to declare a route down. Only then will it stop including the network in its own updates. In the internetwork of Figure 1-13, 54 minutes will pass between the time the exterior gateway of AS 65502 declares network 172.17.0.0 down and the time Groucho declares the network down!

End Notes

[1] Eric Rosen, "RFC 827: EXTERIOR GATEWAY PROTOCOL (EGP)" (Work in Progress)

[2] Linda J. Seamonson and Eric C. Rosen, "RFC 888: 'STUB' EXTERIOR GATEWAY PROTOCOL" (Work in Progress)

[3] D.L. Mills, "RFC 904: Exterior Gateway Protocol Formal Specification" (Work in Progress)

[4] J. Rekhter, "RFC 1092: EGP and Policy Based Routing in the New NSFNET Backbone" (Work in Progress)

Looking Ahead

This chapter has explored both the motives for inventing an inter-AS routing protocol and the reasons why EGP has proven inadequate in that role. Chapter 2 introduces the protocol that has replaced EGP, the Border Gateway Protocol, and examines its operation.

Table 1-10 summarizes the commands used in this chapter.

Table 1-10 *Chapter 1 Command Review*

Command	What It Does
autonomous-system *local-as*	Specifies the local autonomous system in which the EGP router resides
debug ip egp transactions	Displays information about EGP message exchanges and state changes
default-information originate	Causes EGP to advertise a default route
neighbor *ip-address*	Specifies the IP address of an EGP neighbor
neighbor any [*access-list-number* | *name*]	Tells EGP to attempt to peer with any router that initiates the Neighbor Acquisition Protocol
neighbor any third-party *ip-address* [**internal** | **external**]	Configures an indirect EGP neighbor
neighbor *ip-address* **third-party** *third-party-ip-address* [**internal** | **external**]	Configures EGP to send updates regarding indirect neighbors

continues

Table 1-10 *Chapter 1 Command Review (Continued)*

Command	What It Does
network *network-number*	Specifies networks in the IGP routing table that should be advertised to EGP peers
router egp *remote-as*	Configures an EGP routing process
router egp 0	Configures an EGP core gateway process
show ip egp	Displays information about the EGP connections and neighbors
timers egp *hello polltime*	Sets the EGP Hello and Poll intervals to a value different from the default

Review Questions

You can find the answers to the Review Questions in Appendix D, "Answers to Review Questions."

1 What is the current version of EGP?

____

____

____

2 What is an EGP interior neighbor? An EGP exterior neighbor?

____

____

____

3 What is the primary difference between an EGP stub gateway and an EGP core gateway?

____

____

____

4 Why does EGP use the concept of a core, or backbone, AS?

____

____

____

5 What is the difference between an active EGP neighbor and a passive EGP neighbor?

____

____

____

6 What is the purpose of an EGP Poll message?

____

____

____

7 What is an indirect, or third-party, neighbor?

____

____

____

8 How does EGP use its metrics to calculate the best path to a destination?

____

____

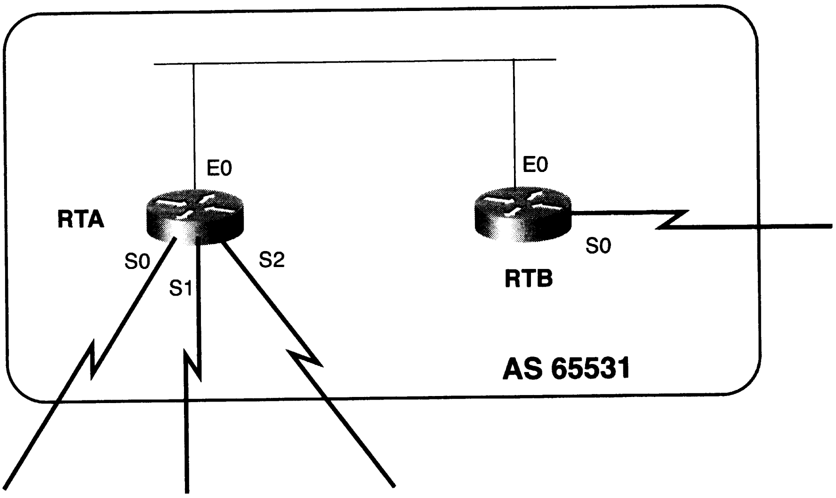
____

Configuration Exercises

You can find the answers to the Configuration Exercises in Appendix E, "Answers to Configuration Exercises."

1 Autonomous System 65531 in Figure 1-14 is a core AS.

Figure 1-14 *The Internetwork for Configuration Exercise 1*



RTA interface	Address
E0	192.168.1.1/24
S0	192.168.2.1/24
S1	192.168.3.1/24
S2	192.168.4.1/24

RTB interface	Address
E0	192.168.1.2/24
S0	192.168.5.1/24

Configure EGP on RTA and RTB, with the following constraints:

— The data link interior to the AS is not advertised to any exterior neighbor.

— RTA advertises the network attached to its S1 interface to RTB; with this exception, no other inter-AS link is advertised between RTA and RTB.

— RTA and RTB advertise a default route to their exterior neighbors, in addition to networks learned from other autonomous systems. Neither gateway advertises a default route to its internal neighbor.

2 Example 1-26 shows the route table of RTC in Figure 1-15.

Example 1-26 *The Route Table of RTC in Figure 1-15*

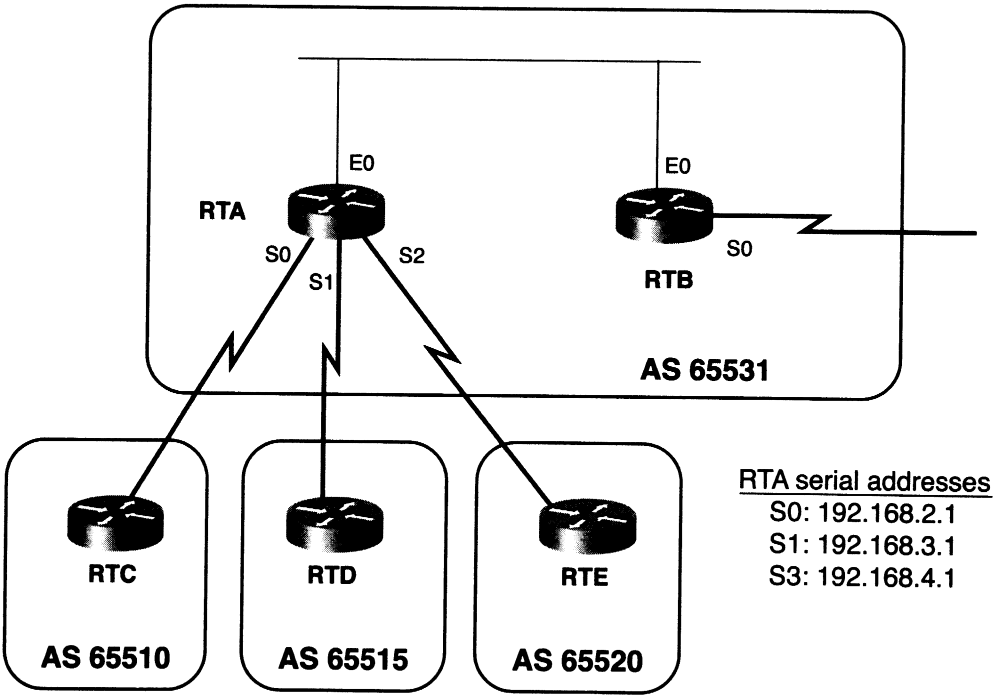
```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate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

continues

Example 1-26 *The Route Table of RTC in Figure 1-15 (Continued)*

```
Gateway of last resort is not set

I    192.168.105.0 [100/8976] via 192.168.6.2, 00:01:00, Serial1
I    192.168.110.0 [100/8976] via 192.168.6.2, 00:01:00, Serial1
I    192.168.100.0 [100/8976] via 192.168.10.2, 00:01:00, Serial2
I    192.168.120.0 [100/8976] via 192.168.10.2, 00:01:01, Serial2
C    192.168.2.0 is directly connected, Serial0
C    192.168.6.0 is directly connected, Serial1
C    192.168.10.0 is directly connected, Serial2
RTC#
```

Figure 1-15 *The Internetwork for Configuration Exercise 2*



Using redistribution, configure RTC to advertise all EGP-learned networks into AS 65510, and all internal networks except 192.168.105.0 to the core AS. Protect against route feedback by ensuring that none of the networks internal to AS 65510 are advertised back via EGP. The process ID in this configuration is the same as the local AS number.

3 Example 1-27 shows the route table of RTD in Figure 1-15.

Example 1-27 *The Route Table of RTD in Figure 1-15*

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

Gateway of last resort is not set

C    192.168.3.0 is directly connected, Serial0
C    192.168.7.0 is directly connected, Serial1
R    192.168.230.0 [120/1] via 192.168.7.2, 00:00:14, Serial1
R    192.168.200.0 [120/2] via 192.168.7.2, 00:00:15, Serial1
R    192.168.220.0 [120/1] via 192.168.7.2, 00:00:15, Serial1
R    192.168.210.0 [120/2] via 192.168.7.2, 00:00:15, Serial1
RTD#
```

Configure RTD with the following parameters:

— Only 192.168.220.0 and 192.168.230.0 are to be advertised to AS 65531.

— No routing protocol is redistributed into EGP.

— EGP is redistributed into the IGP of AS 65515.

— 192.168.3.0 is advertised into AS 65515 with a metric of 1.

— 192.168.100.0, from RTC, is advertised into AS 65515 with a metric of 1.

— 192.168.120.0, from RTC, is advertised into AS 65515 with a metric of 3.

— All other routes are advertised into AS 65515 with a metric of 5.

4 Example 1-28 shows the route table of RTE in Figure 1-15.

Example 1-28 *The Route Table of RTE in Figure 1-15*

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


Gateway of last resort is not set

O    192.168.125.0/28 [110/74] via 192.168.130.6, 00:01:03, Serial1
C    192.168.4.0/24 is directly connected, Serial0
     192.168.225.0/28 is subnetted, 1 subnets
```

continues

Example 1-28 *The Route Table of RTE in Figure 1-15 (Continued)*

```
O E2    192.168.225.160 [110/50] via 192.168.130.18, 00:01:04, Ethernet0
     192.168.215.0/24 is variably subnetted, 3 subnets, 3 masks
O       192.168.215.161/32 [110/65] via 192.168.130.6, 00:01:04, Serial1
O E2    192.168.215.192/26 [110/50] via 192.168.130.18, 00:01:04, Ethernet0
O E1    192.168.215.96/28 [110/164] via 192.168.130.6, 00:01:04, Serial1
     192.168.130.0/24 is variably subnetted, 7 subnets, 4 masks
D       192.168.131.192/27 [90/2195456] via 192.168.130.6, 00:16:49, Serial1
D       192.168.131.96/27 [90/409600] via 192.168.130.18, 00:16:49, Ethernet0
O       192.168.131.97/32 [110/11] via 192.168.130.18, 00:01:05, Ethernet0
D       192.168.131.64/27 [90/409600] via 192.168.130.18, 00:15:01, Ethernet0
D       192.168.131.8/30 [90/2195456] via 192.168.130.6, 00:16:49, Serial1
C       192.168.131.4/30 is directly connected, Serial1
C       192.168.131.16/28 is directly connected, Ethernet0
RTE#
```

Configure RTE with the following parameters:

— No IGP is redistributed into EGP.

— EGP is not redistributed into any IGP.

— All the internal networks of AS 65520 are advertised to AS 65531.

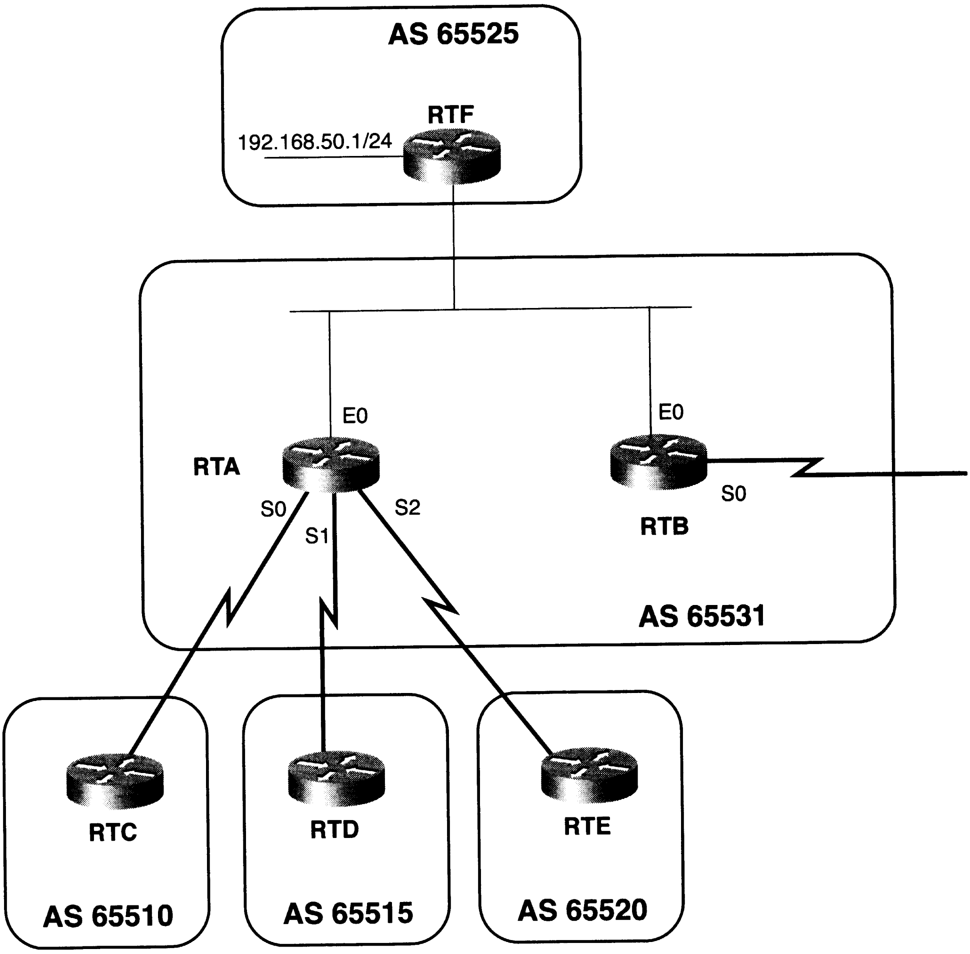
— The internal routers of AS 65520 can forward packets to any network advertised by RTA.

— All process IDs are the same as the AS number.

— All OSPF interfaces are in area 0.

5 In Figure 1-16, AS 65525 has been added to the internetwork of the previous exercises. RTF's Ethernet interface has an IP address of 192.168.1.3/24.

Figure 1-16 *The Internetwork for Configuration Exercise 5*



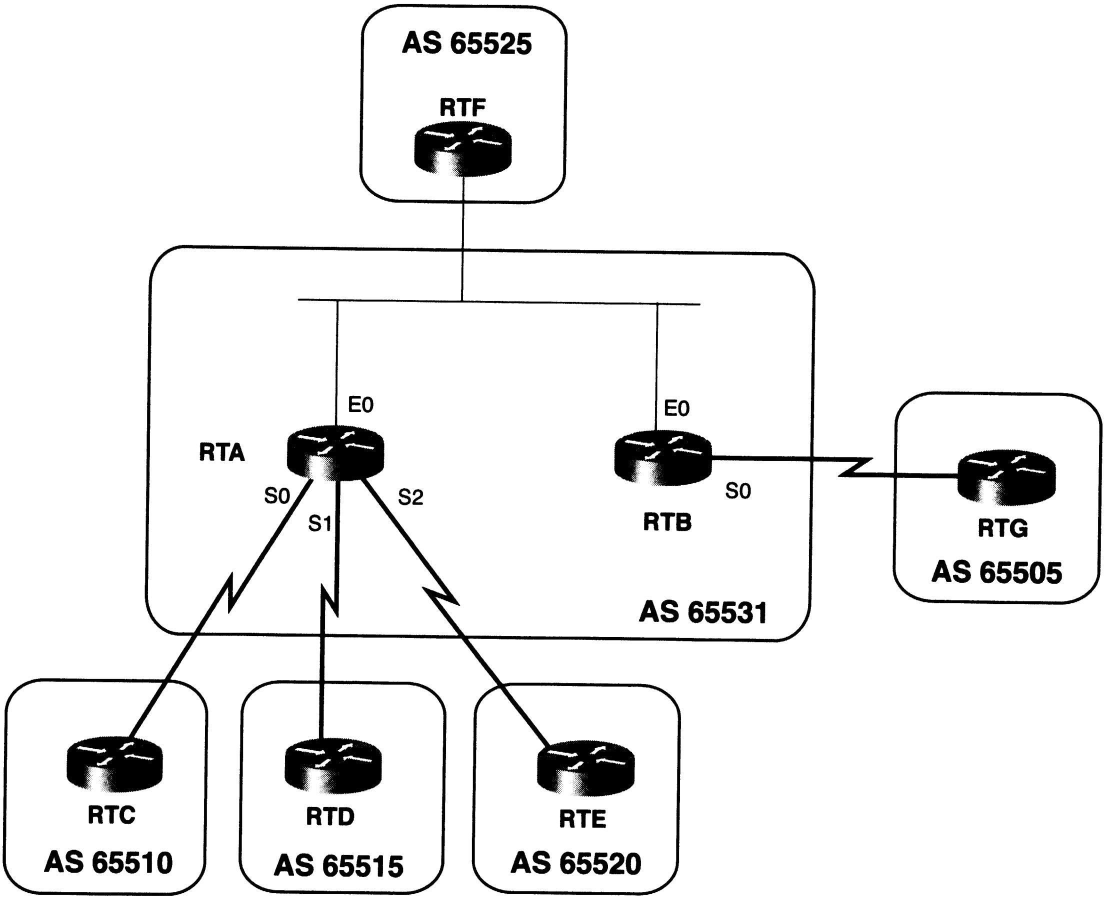
Configure this router to peer only with RTB and make any necessary configuration changes to support third-party neighbors.

Troubleshooting Exercise

You can find the answer to the Troubleshooting Exercise in Appendix F, "Answers to Troubleshooting Exercises."

1 In Figure 1-17, router RTG has been added to the internetwork.

Figure 1-17 *The Internetwork for Troubleshooting Exercise 1*

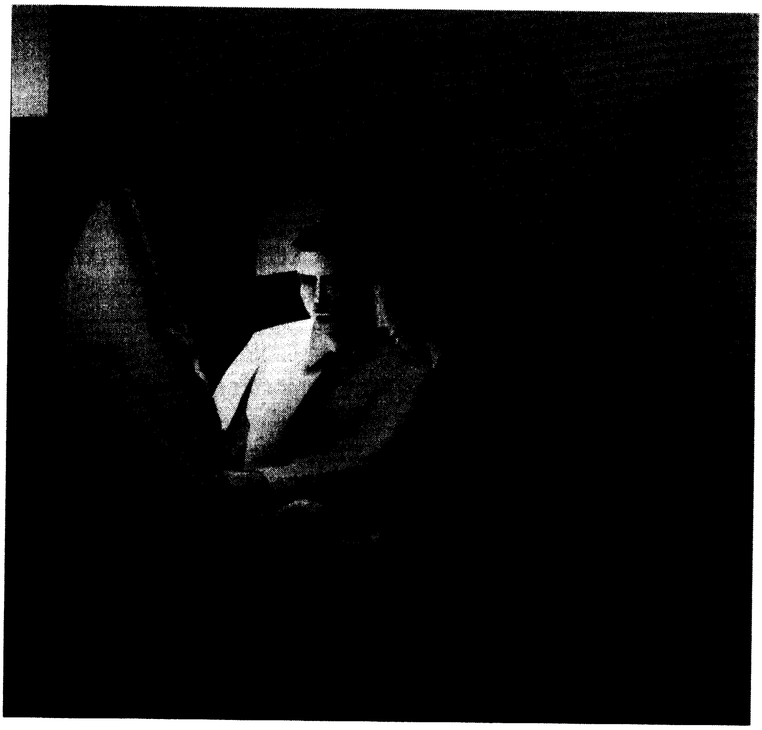


Although it is peering with RTB and exchanging reachability information, there is a configuration error. Based on the information in Example 1-29, what is the error?

Example 1-29 *The EGP Tables of RTB and RTG in Figure 1-17*

```
RTB#show ip egp
Local autonomous system is 65531

 EGP Neighbor        FAS/LAS  State    SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.1.1      65531/65531 UP     4      2      6    60   180   2 Perm, Pass
*192.168.1.3      65525/65531 UP     4      2    492    60   180   2 Perm, Pass
*192.168.5.2      65505/65531 UP     3      2     33    60   180   3 Temp, Pass
 EGP Neighbor       Third Party
*192.168.1.1        192.168.1.3(e)
*192.168.1.3        192.168.1.1
RTB#
```

```
RTG#show ip egp
Local autonomous system is 65505

 EGP Neighbor        FAS/LAS  State    SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.5.1      65505/65505 UP     9     36      3    60   180   4 Perm, Act
RTG#
```



from Cisco Voice over Frame Relay, ATM and IP

by Steve McQuerry, Kelly McGrew, and Stephen Foy

(1-58705-00645)

Cisco Press

About the Editors

Steve McQuerry, CCIE #6108, is an instructor and consultant with more than ten years of networking industry experience. He is a Certified Cisco Systems Instructor (CCSI) teaching routing and switching concepts for Global Knowledge. In addition to teaching the CVoice course, Steve regularly delivers the ICND, BCMSN, BSCN, CIT, CID, and OSPF/BGP courses. Additionally, Steve has developed and taught custom Cisco switching courses for large corporate customers. In his consulting capacity, he provides design, integration, and troubleshooting services to enterprise customers. Previous to founding his own network consulting and education firm, Steve was a Senior Consultant at the Information Connection, where he worked on various network implementation projects including responsibilities for network design, project management, security planning, and LAN/WAN troubleshooting. As a member of the Network Planning Team at the University of Kentucky Hospital, Steve designed and implemented networking solutions for the hospital's LAN and WAN. Steve holds a B.S. degree in engineering physics from the Eastern Kentucky University.

Kelly McGrew is a Certified Cisco Systems Instructor (CCSI) and vice-president of mcgrew.net inc., a network training and course development firm. He has worked as a trainer throughout the world. Kelly holds the CCNP-Voice Access Specialist and CCDA certifications. He has more than 15 years of experience in the networking industry, including experience with a variety of LAN and WAN protocols seldom seen in today's IP-centric world. Kelly has held a variety of positions for leaders in the networking industry. These include positions as a network systems engineer for CompuServe Network Services and MCI/WorldCom, an instructor/consultant for Chesapeake Computer Consultants, Inc., a program manager for Microsoft Corporation, and an instructor/consultant under a leased-employee relationship for Cisco Systems, Inc. He is a graduate of The Evergreen State College (B.A.) and obtained an M.B.A. from City University. Kelly is an associate member of the IEEE and member of the ASTD. Kelly currently focuses on teaching and course development in the Voice over Layer
2/IP arena. He currently resides in Olympia, Washington, with his wife, Tammy (also a CCSI and the president of mcgrew.net inc.), their son, Duncan, and the world's best doggie, Lady Buttons.

Stephen Foy is an internetworking instructor for Global Knowledge, Cisco's largest worldwide training partner, with more than 18 years of experience in the networking industry. Stephen is a Cisco Certified Network Professional (CCNP) and Certified Cisco Systems Instructor (CCSI) conducting classes for various vendors including Cisco Systems, Digital Equipment, and Motorola. He teaches the ICND, ACRC, BCRAN, CATM, MCCM, and CVoice classes on a regular basis.

Contents at a Glance

Bold chapters are elements included in this folio.

After reading this chapter, you should be able to perform the following tasks:

- Review, identify, and define digital telephony fundamentals and basics.
- Contrast digital and analog signaling.
- Identify and contrast the different digital frame formats, signaling formats, and coding types.
- Compare the various levels of voice quality.
- Categorize the various types of digital voice compression.
- Examine the ISDN digital architecture and identify its basic components.

CHAPTER 3

Introduction to Digital Voice Technology

As you learned in Chapter 1, "Merging Voice and Data Networks," voice can be transmitted across a traditional network in either analog format or digital format. This chapter covers the basics of voice in a digital network.

As shown in Chapter 2, "Introduction to Analog Technology," analog transmission requires one set of wires for each telephone call. Each FXS or FXO requires two wires, and variations of E&M can use up to eight wires for each call. If you are provisioning multiple voice channels between a PBX and a router using traditional analog interfaces, port density becomes a problem. The number of individual port connections on both the router and the PBX can be cost prohibitive and difficult to maintain.

The digitized alternative to this problem, commonly called T1 in the U.S. (and E1 in most other countries), assumes a fixed number of digitized voice channels that consume a fixed amount of bandwidth across a serial, time-division multiplexed set of wires. Because the interface is digital and not prone to the distortion effects that are common to most analog transmission methods, it provides greater reliability than analog transmission. Maladies such as crosstalk and radio frequency interference (RFI) are eliminated because only ones and zeros are transmitted. Repeaters may be placed in the transmission path for longer-distance T1 transmission lines to reshape the signal in much the same way that an Ethernet repeater does. The combination of the aggregated channels makes the transmission method very high speed (1.544 megabits per second [Mbps] for T1 and 2.048 Mbps for E1). Originally developed for voice transmission, T1 is a common replacement for individual analog interfaces.

In the following sections, we will examine the method by which analog signals are digitized, including some common compression schemes and other innovative ways to save bandwidth. Next, we'll cover standard ways in which the compression schemes are graded for quality. Then we'll move on to a discussion of the signal formats used on the transmission path. Finally, we'll delve into examination of ISDN and Signaling System 7 and the Q.SIG standard.

Digitizing Analog Signals

Expressing an analog signal as a digital one is a daunting task. Since an analog signal, by its very nature, has an infinite amount of values that can be expressed in terms of amplitude, frequency, and phase, converting those values to a comprehensive one and zero expression scheme is very difficult. A mathematical means by which the conversion could be accomplished needed to be developed, and the result of that research (explained in the next four sections) led to the development of a device called a *codec* (*coder–decoder*). The analog telephony signal (human voice) is applied at the input to the codec, and a coded one and zero digital bit stream is developed at the output. Conversely, the process can be inverted to convert the digital bit stream back to analog at the other end of the communications path, with the same codec working in reverse.

There are four steps involved in digitizing an analog signal:

1. Sampling
2. Quantization
3. Encoding
4. Compression (optional)

The following sections explain the details of each step, and focus on digitizing voice-grade telephony channels in preparation for transmission over T1 or E1 wires.

Sampling

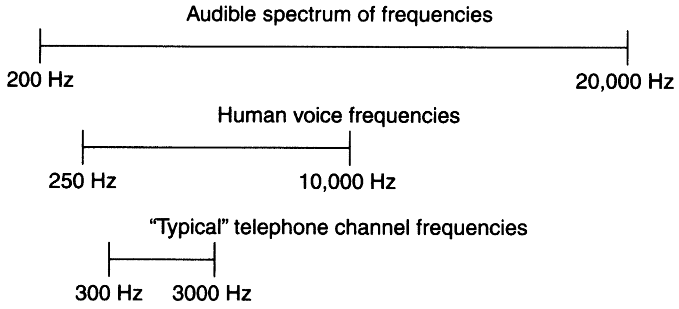
In 1924, while working for American Telephone & Telegraph, Henry Nyquist searched for a means by which analog signals could be converted to digital. His research was rooted in military applications, where a minimal amount of transmission wires could be used to multiplex a large number of analog circuits. He developed what is now known as the Nyquist Theorem. It states that in order to digitize an analog signal, the signal must be sampled at a rate equal to that of twice the highest frequency of the signal to be digitized. A voice channel's highest frequency, therefore, will be sampled only twice during one sine wave cycle in order to reproduce it as analog again. Although sampling the signal less often at the highest frequencies does not sound like very good representation, this method assumes that most of the intelligible information in an analog voice channel is nearer the middle of the bandwidth, and sampling will take place there much more frequently. Put another way, the human voice contains sounds that are more often middle-pitched than mostly high-pitched or low-pitched. The significance of the sampling will therefore be more valuable at the middle frequencies than at the high or low frequencies.

So what is the bandwidth of a voice-grade analog channel? The answer depends on the fidelity and capabilities of the analog transmission equipment. We can hear sounds from about 200 hertz (Hz) to about 20,000 Hz. Human speech is in the range from about 250 Hz to 10,000 Hz. A typical telephone channel only carries from about 300 Hz to about 3000

Hz, depending on age and other factors. Voice-grade telephony channels, then, have a limited bandwidth of frequencies compared to the natural human voice (see Figure 3-1). This limitation is imposed for the following reasons:

- The goal of a telephony channel is to transmit intelligible speech.
- The speaker must be recognizable by the listener.
- The speaker's emotions must be discernable by the listener.
- Costs and technology constraints in the telephony transmission system must be considered.

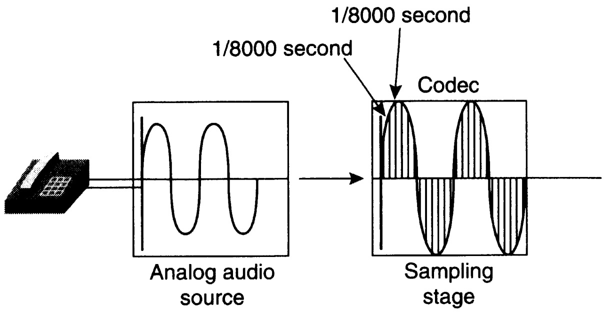
Figure 3-1 *Audible Frequencies Comparison: Human-Audible Spectrum Compared to Speech Spectrum and Telephone Channel*


Audible spectrum of frequencies
200 Hz
20,000 Hz
Human voice frequencies
250 Hz
10,000 Hz
"Typical" telephone channel frequencies
300 Hz
3000 Hz

Voice-grade telephony channels exhibit varying frequency responses because of physical differences in wires used, amplifier condition and age, and other electrical factors. In the end, all analog voice telephony channels must meet the four criteria noted above. Frequency responses of 300 Hz to 3000 Hz are common in average telephone channels, as are 270 Hz to 3300 Hz and other widely assorted variations. Because of the variations (and the lack of a standard to define the limits), Nyquist decided to drive a stake in the ground and considered all voice-grade analog channels to have a bandwidth of 0 to 4000 Hz. Our sampling rate, then, is twice 4000 Hz, or 8000 times per second.

The process of sampling an analog signal is simple. As shown in Figure 3-2, the analog waveshape is examined for its vertical amplitude at a rate equal to 8000 times per second (thus satisfying Nyquist's Theorem). Each sample thus represents a snapshot of the signal's amplitude (or vertical height) for 1/8000 second. It will take 8000 of these samples (the vertical lines in the figure) to re-create 1 second's worth of sound at the other end of the T1 line. The electronic component that accomplishes the sampling is called a codec.

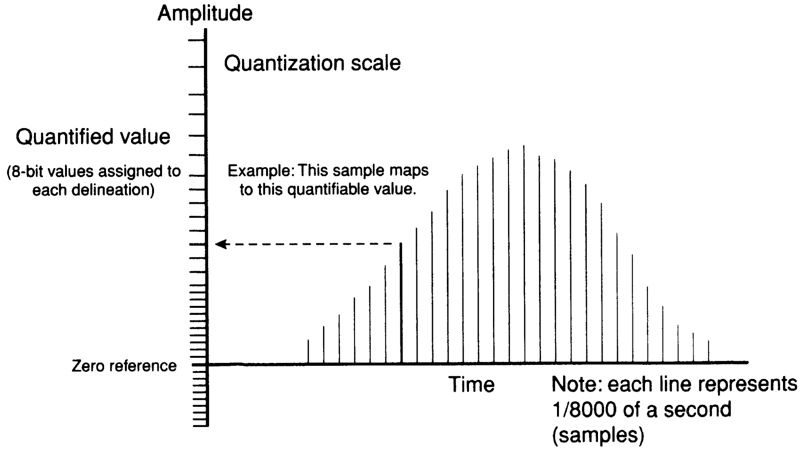
Figure 3-2 *An Analog Sine Wave Sampled at 1/8000-Second Intervals*



Quantization

The next step in the process of digitizing an analog signal is to express a mathematical value for each of the samples taken. Each sample can be quantified, or assigned a numerical value, if a scale of some sort is applied to the relative amplitude (vertical height) of the samples. Figure 3-3 shows a scale with definitive delineations to which we can assign values. In the case of digitizing analog voice, the values assigned are in the form of 8-bit binary words. Although an 8-bit binary word will produce 256 combinations, only 255 are used. There are 127 delineations above the zero reference line, 127 below the zero reference line, and 1 to mark the zero reference line itself. The bit pattern consisting of all zeros is never used, and the reason for eliminating it will be explained later. Note that the zero reference line will be represented by the all-ones bit pattern. Because of this, any of the 24 T1 channels that are idle will transmit the all-ones pattern in their respective time slots.

Figure 3-3 *Assigning Measurable Quantities to the Samples*


Amplitude
Quantization scale
Quantified value
(8-bit values assigned to each delineation)
Example: This sample maps to this quantifiable value.
Zero reference
Time
Note: each line represents 1/8000 of a second (samples)

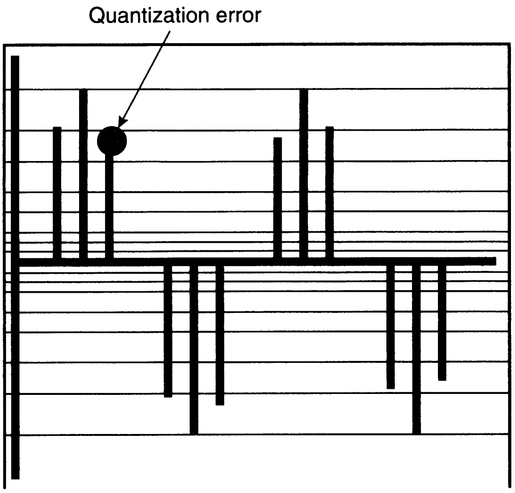
As you can see in Figure 3-3, the scale used is nonlinear in nature. This means that the delineations on the scale are set up to be spaced unevenly. Notice that the marks are close together nearer to the zero reference line, and they move further apart away from the zero reference line. This design serves two purposes. First, the human ear is designed to hear in a nonlinear fashion, and a person can discern sounds of a lower volume more clearly than loud sounds. That is, we get better granularity of the samples at lower volumes than at higher volumes as a result of the nonlinear scale. Second, the original analog signal is compressed in amplitude before it is sampled and quantified so as to reduce the amount of noise present with the signal. After sampling and quantization, less noise is transmitted to the other end as a result, so when the signal is re-created, the signal-to-noise ratio is greatly improved.

There are two standard methods today for quantifying the signal and setting up the nonlinear scale. Since the technology was first invented and deployed in the United States, the original quantifying method used by AT&T is still in use today in North America. This quantifying (or, as it is sometimes called, *companding*) method is called μ-law (pronounced "mu-law"). In the late 1960s, this digitizing technology was presented for international standardization to what was then the CCITT (Consultative Committee for International Telephone and Telegraph) and now known as the ITU-T (International Telecommunication Union Telecommunication Standardization Sector). The study group assigned to the project modified the original companding method. As a result, the international standard for companding, A-law, was released. A consequence of this ruling is that North America, which uses μ-law, is incompatible with the rest of the world, which uses a-law. (Another consequence of this is that North America uses T1 while the rest of the world uses E1.) International agreement states that while interfacing to any a-law country, the μ-law country must convert to a-law before interfacing.

NOTE In some parts of Japan, μ-law is used to better interface with North American countries.

The quantization process is not always accurate, however. Consider what would happen if a sample's amplitude fell exactly between two marks on the scale. Figure 3-4 shows the third sample to be between two valid scale delineations. How do we quantify this particular sample? For a definitive quantifiable measure, should the codec guess in the up direction or the down direction? No matter which way it guesses, the sample taken will never be exactly accurate at this 1/8000-second instance. When we re-create the sample at the far end, it will be inaccurate because we cannot represent a value between two scale delineations. This mistake in representing value is called a *quantization error*, and it results in quantization noise when the analog signal is re-created. *Quantization noise* is noise that the listener may not necessarily hear (since it's a mistake for only 1/8000 second), but if enough quantization errors occur together, they may create noise that can be heard. For this reason, the codec applies a low-pass filter, or *smoother circuit*, to the re-created analog signal to filter noise created by quantization errors.

Figure 3-4 *Quantization Error: One of the Samples Shown Is Exactly Between Two Sampling Points*



Encoding

Once each sample is taken and then quantified against the quantization scale, the codec assigns a numerical value to the quantified sample. As noted earlier, we use 8-bit binary representations for each quantified sample. The 8-bit binary word that represents each of the marks on the quantization scale (see Figure 3-3) has a predefined format. Figure 3-5 depicts the meaning of each bit of the 8-bit word. The first bit is used to represent polarity. It simply relates whether this sample is above the zero reference line or below it.

The next 3 bits represent Segment, sometimes called Chord. These bits represent the area of the scale in which this sample is found. Each Segment has an equal number of steps, which are represented by the remaining 4 bits. For example, an 8-bit value of 10110101 decodes as follows: The quantified sample is above the zero reference line (since the leftmost bit is a 1), the sample is situated in the third segment (the next 3 bits, 011, have a decimal value of 3), and the sample is on the fifth delineation within segment three (the last 4 bits, 0101, have a decimal value of 5).

Figure 3-5 *Format of a Word Sample: Sample Is 8-Bits Long, Specific Bits Have Individual Meaning*

P	Se	Se	Se	St	St	St	St

P = Polarity
Se = Segment
St = Step

Now that three of the four steps of digitization are complete, it is helpful to review what's happened:

1 Sampling: Samples are taken at a rate of 8000 times per second.

2 Quantization: Each sample is quantified in comparison to a scale that has delineations grouped in segments.

3 Encoding: Each quantified sample will produce an encoded 8-bit word that represents the sample's amplitude.

Altogether then, 8000 samples per second times 8 bits per sample yields 64000 bits per second to represent one second of sound or speech. This particular coding scheme is designated as ITU-T standard G.711, or pulse code modulation (PCM). That's a very large quantity of bandwidth to consume across expensive WAN networks for one telephone call. But it was a very innovative way to multiplex multiple channels across a digital transmission medium (T1).

Compression

Voice compression schemes were developed as an effort to save bandwidth on the WAN. As an attempt at maintaining the quality of PCM at 64000 bits per second (bps), many different approaches were taken. The results were variations on the same theme: lower bit rates, but with quite a loss in quality. Some attempted to cut the bandwidth and follow a similar approach to PCM, while others blazed new trails with bold approaches like continuously variable slope delta (CSVD). We will present some of the more common approaches in the following sections, including those chosen by standards bodies and selected by Cisco Systems. It is important to note that two of the terms used in this chapter, coding and compression, have two different meanings. Coding is a means by which the analog signal is represented. Compression means we're trying to improve on the original 64000-bps PCM method.

NOTE Note that compressing the original PCM-coded signal is completely optional, and is done specifically to save WAN bandwidth. Any compression of the original PCM will affect voice quality. You need to balance the cost of WAN bandwidth against the cost to compress with the loss of quality of the original digitized PCM signal. Rating the resulting quality is described later in the section titled "Voice Compression Techniques Compared."

The following sections examine three technologies that can be used for voice compression:

- Wave form compression: Follows the approach used for PCM encoding
- Vocoder compression: Synthesized voice with processing intelligence

- Hybrid compression: A combination of wave form and vocoder compression

An in-depth examination of each compression approach follows, with pros and cons listed for each.

Wave Form Compression

Wave form compression is a subset of compression schemes, which include PCM and its related derivations. This family of compression schemes is known as *wave form coding* because quantization and encoding (that is, sampling for quantity and assigning a binary value to the quantity) tracks and follows the actual analog wave form as it develops in real time. Adaptive differential pulse code modulation (ADPCM) includes a variety of wave form coding methods, including 40000-bps, 32000-bps, 24000-bps, and 16000-bps varieties. Collectively, these compression schemes are designated as ITU-T standards G.726 and G.727. The 32000-bps version was developed first, and was planned to allow the capacity of T1 to double, from 24 voice channels to 48.

As an example of how wave form compression schemes work, the 32000-bps version uses a 4-bit word, instead of an 8-bit word as PCM uses (thus resulting in 32000 bps instead of 64000 bps). An individual quantization represents the comparative difference between the quantization itself and the last sample (that is, the differential). The 4-bit coded word uses 1 bit to represent the amplitude change since the last sample (that is, an increase or a decrease in amplitude from the last sample) and the remaining 3 bits to represent how much to increase or decrease the amplitude. The relative value of the 3 bits can change based on the last few samples seen (that is, it is adaptive).

The 24000-bps version uses a 3-bit word, and the 16000-bps version uses a 2-bit word. As you may have guessed, the quality of ADPCM suffers somewhat in comparison to PCM and degrades significantly with each version. The 16000-bps version is almost unintelligible. The 40000-bps version uses a 5-bit word, and was actually developed to improve on the poor quality of the 32000-bps version.

Pros of wave form compression are:

- Reduced bandwidth consumption
- Simple and inexpensive to process

Cons of wave form compression are:

- Poorer audio quality at lower bandwidths

Vocoder Compression

A second method of speech coding is called *vocoder* (for voice coder). This method can produce low bit rates with very intelligible speech, but it sounds machine-like. It is based on the premise that the human voice is made up of a base frequency or sound, produced by the vocal cords, which is varied using a sound chamber (the mouth and throat). The variations, like the tongue against the teeth, lips formed in a circle, and the rolling of *r* as in the Spanish language, are called *fricatives*. The vocoder method assumes that the base frequency can be played continuously for each word spoken at the far end of the channel, with the transmitter sending only the fricatives.

During the vocoder processing, the coder must have the ability to recognize and encode each different fricative produced by human speech. This ability requires some measure of intelligent processing power and some preprogrammed knowledge of human speech patterns. Special processing electronics are required to accomplish this task. An example of a vocoder is the speech created by the computer used by the famous physicist Stephen Hawking. Although this form of synthesized voice is very practical for this type of use, vocoders are not practical for voice telephony since we must be able to identify the speaker and sense his or her emotional state.

Pros of vocoder compression are:

- Reduced bandwidth consumption

Cons of vocoder compression are:

- Expensive to process
- Requires specialized electronics
- Sounds synthetic
- Speaker is not recognizable

Hybrid Compression

The third category of voice coding is *hybrid coding*. It combines the best of the wave form and vocoder compression techniques to create high-quality voice at low bit rates. Used extensively in the digital cellular telephone industry, hybrid coding schemes can save significant bandwidth across a WAN channel. One of the more recent developments in this category is called *code excited linear predictive (CELP)* coding. This method maintains a "codebook" of waveshapes that are representative of sounds that the human voice can produce. Each waveshape is assigned a binary code. When the speaker talks, entire waveshapes are sampled and compared to the codebook, and the closest waveshape in the codebook to the original is sent across the channel in the form of its assigned binary pattern. Although of high quality, CELP requires powerful processing circuitry and lots of memory in order to perform its task. It also causes a significant amount of delay due to the coding process.

A second variation of CELP, called *low-delay CELP (LD-CELP)*, builds the codebook directly from the speaker's voice rather than from fixed waveshapes. This results in a shorter processing delay and possibly a more accurate voice representation. LD-CELP has been designated as ITU-T standard G.728 and operates at 16000 bps.

A third variation of CELP has been made possible by specialized microprocessor chips called *digital signal processors (DSPs).* Conjugate structure algebraic CELP (CS-ACELP) can encode very high-quality speech at a rate of 8000 bps. The original LD-CELP algorithm was modified somewhat to make it more efficient and sample more exactly to produce this complex scheme. The codebook is even more adaptive with CS-ACELP, utilizing far more complex mathematics to evaluate and encode the signal. Because the codebook is more adaptive, it can react to different languages more readily. The original CELP codebook was designed using sounds produced with American English and was therefore limited. CS-ACELP can adapt its codebook's waveshapes to many variations of human speech and can therefore adapt itself to the language being spoken. It has been designated ITU-T standard G.729 and has some basic variations.

The ITU-T Standard G.729a is also an 8000-bps CS-ACELP coding method, but its algorithm has been simplified to make it more efficient. Although it produces very high-quality voice reproduction, its fidelity is slightly less than that of the original G.729. Two other variations of G.729 are G.729B and G.729AB. These two variants are also 8 k but include a built-in VAD (Voice Activity Detection) algorithm, which serves to save even more bandwidth. VAD is described in a later section titled "Digital Speech Interpolation."

Pros of hybrid compression are:

- Excellent audio quality
- Very low bit rates
- Adapts to speaker

Cons of hybrid compression are:

- Requires specialized processing chips (DSPs)
- Requires memory
- Induces processing delay

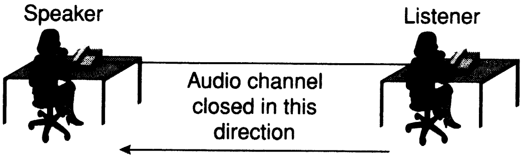
Voice Compression Techniques Compared

As the demand for voice-over-X technologies continues to increase, the IT industry will continue to develop new coding and compression techniques to conserve bandwidth and align with the particulars of the lower layer services available. When bandwidth is affordable and high quality is a must, or if connecting equipment is of traditional de facto standard, PCM coding at 64000 bps stands at the ready. Where corners can be cut and variations of ADPCM fit the need (such as in hotels and call centers), less bandwidth for less cost and quality can be satisfactory. More recent users, whose needs may not have to align with older equipment, will opt for the CELP compressions. Your choice will depend on many factors. The newer CELP standards offer very low bit rate transmission with excellent quality, and carriers and private customers will migrate toward these standards without hesitation.

Digital Speech Interpolation

In addition to compression, there are other innovative ways to save bandwidth. If we were to examine the mechanics of a human conversation, we would find that during portions of the conversation, the speaker goes silent to listen. As the listener is silent, the transmission of voice from the listener to the speaker is almost non-existent, with the exception of background noise. An opportunity exists here to close the communications path from the listener to the speaker and save bandwidth only in that direction. In addition, as the conversation develops, the speaker intentionally and unintentionally makes pauses in the conversation. These pauses—to make a point, to end a sentence, to allow the listener to digest what's been said—provide more opportunities to save expensive WAN bandwidth. The Cisco Voice Port option used to accomplish the savings is called VAD, for voice activity detection. Figure 3-6 shows how this saving can be applied.

Figure 3-6 *Voice Activity Detection: Saving Bandwidth Due to Listener's Silence*



Digital speech interpolation is a DSP function that examines voice to determine power (that is, volume) and change of power, and to determine frequency and change of frequency. When periods of silence are present, the sending entity will send an "enter silent period" notification, at which time the receiver will play a period of white noise (or *comfort noise*) to the listener. During the period of silence, the sender need not send empty packets or frames to the receiver, which means there is more room for data frames or packets to get through. When the speaker again begins to talk, the DSP again opens the audio channel, allowing speech to be transported.

Although it is valuable during a conversation, the digital speech interpolation approach can devastate some voice mail systems that end recording when speech is not present. Care should be taken when enabling this feature on Cisco voice-capable routers. Later releases of Cisco IOS allow the system administrator to control the timing of the VAD action when silence appears so as to not clip the speech between words and phrases. Also, keep in mind that the savings in bandwidth may be attractive, but because of the artificial comfort noise, it may not be aesthetically pleasing to the listener.

Telephone Voice Quality

As shown in Figure 3-7, AT&T has defined three levels of voice quality for comparing different compression schemes (see Table 3-1).

Figure 3-7 *Comparison of Voice Quality Compression Technologies*

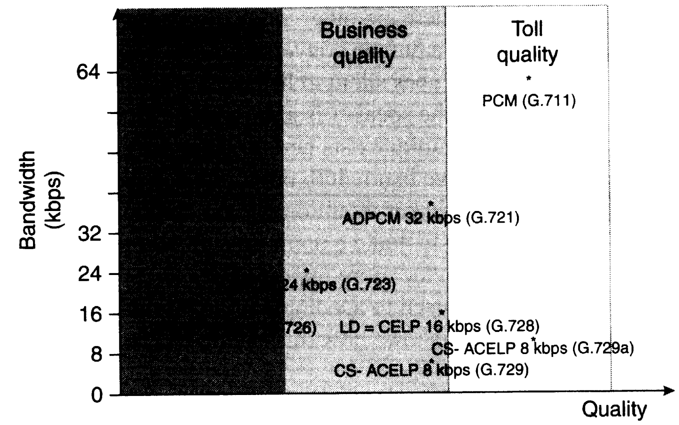


Table 3-1 *AT&T Voice Quality Levels*

Quality Level	Description	Examples
Toll	Indistinguishable from a straight "all analog" copper connection. Produces the best sound quality and has no distinguishable disturbances or distortions.	Calling within the continental United States usually results in toll quality.
Business	Noticeable distortions and poorer audio quality, but the speaker is still recognizable and an intelligent conversation can still take place.	Calling between two continents or using a poor transmission system as in third-world countries.
Unacceptable	Annoying and disturbing distortions that interfere with speech. Words and phrases are misinterpreted or not understood due to the severity of the distortions, and a conversation is difficult to maintain.	Calls to remote portable systems, as in military battlefield applications or remote oil drilling applications.

The ITU-T standard G.711 is best in the toll quality category—that is, it cannot be distinguished from straight copper wire end-to-end, and it is the digital coding standard by which all others are compared. The industry's second attempt at coding, ADPCM 32000, falls into the business quality category. ADPCM has many variations, as noted in previous chapters; as variations use less bandwidth, quality suffers. The 16000 variety of ADPCM falls into the lower category of quality called unacceptable, because the distortions and distractions are too numerous and frequent to allow an intelligible conversation.

As shown in Figure 3-7, the CS-ACELP coding methods have a quality that surpasses all of the ADPCM schemes and comes very close to PCM. These coding methods may be expensive in terms of processing power required, but their payoff in WAN bandwidth saved is undeniable.

The ITU-T, like AT&T, has defined parameters for measuring voice quality. A fair comparison of voice quality is critical to designing a comprehensive voice/data network because traditionally, the higher-bandwidth schemes tended to deliver better quality than the lower-bandwidth schemes. This can translate to higher costs for corporations deploying voice across data networks, and the solution becomes less cost-effective than was first thought.

The following sections explain the categories of quality as outlined by the ITU-T, the method of voice quality measurement used, and the quality scores for various voice compression methods.

ITU-T's Voice Quality Measurement

Unlike AT&T's three-category method, the ITU-T defines five categories of voice compression methods, as shown in Table 3-2. Each of these five categories is assigned a value, on a scale from 1 to 5, with 5 indicating toll quality and 1 indicating unsatisfactory. The following section relates how the ITU-T uses this scale to compare various voice compression methods.

Table 3-2 *ITU-T Voice Quality Levels*

Quality Level	Description	Examples
Toll quality (Grade 5)	Toll quality emulates and sounds like a copper wire. Exhibits similar quality to that of an analog end-to-end call with signal-to-noise ratios and harmonic distortions within acceptable limits.	Calls within a PBX from user to user or within a central office, such as calling a neighbor in the same geographic area.
Transparent quality (Grade 4)	This is very similar to toll quality, with some tolerable distortions and almost imperceptible distractions. The distortions may be discernable to the most critical user, but are not annoying.	Calling long distance, from state to state, or between neighboring countries.

continues

Table 3-2 *ITU-T Voice Quality Levels (Continued)*

Quality Level	Description	Examples
Conversational quality (Grade 3)	Conversational quality has perceptible distortions and annoying distractions. In this category, the user begins to noticeably hear the degraded quality of the channel and may need to ask the speaker to repeat portions of the conversation.	Intercontinental calls or calls to third-world countries.
Synthetic quality (Grade 2)	Synthetic quality is tolerable, but has very annoying distractions and poor reproduction of the speaker's voice fidelity. Because the reproduction is so poor, the listener hears what almost sounds like a machine-like re-creation.	Ship-to-shore telephony communications.
Unsatisfactory (Grade 1)	When the voice reproduction is this poor, the listener is sometimes forced to ask for a new channel. A simple exchange between the speaker and listener is strained to the point of the speaker repeating what's been said again and again.	Interference caused by malfunctioning equipment or induced from outside sources, such as radio interference.

Mean Opinion Score

The ITU-T recommends measurement of voice quality using the five categories shown in Table 3-2 by a subjective methodology called the Mean Opinion Score (MOS). Test subjects are gathered into a lab environment and asked to rate voice quality through varying methods of compression. The rating uses the five grades shown in Table 3-2. The participants listen to a recorded message that is chosen based on its varying fricatives. The message used to measure MOS for English speaking listeners is "Nowadays, a chicken leg is a rare dish." As they listen to the message, they realize that there are hard sounds (as in the "K" sound in *chicken*) and soft sounds (as in the "CH" in *chicken*), long vowels (as in the "A" sound at the end of *nowadays*) and short vowels (as in the "I" in *dish*). As the listeners rate the different compression methods, the average, or mean, of all the participants is calculated after the tests. The ITU-T considers a MOS of 4.0 as toll quality.

Although MOS is widely used, a newer method for measuring voice quality is being accepted by the industry. This method is called Perceptual Speech Quality Measurement, or PSQM, and is assigned the ITU-T standard P.861. PSQM was developed to measure voice quality in transmission systems originally developed for data, such as running voice-over IP networks. The PSQM quality measurement can sometimes be a more precise measurement tool than MOS since it is not subjective and is sensitive to impairments seen on voice-over data networks, such as delay and missing packets or frames. Some manufacturers of PSQM equipment include the capability to convert the PSQM result into MOS scores. Cisco Systems uses PSQM as a more precise means by which to measure voice quality over data networks.

You must also keep in mind that the ITU-T's recommendation for using MOS as a quality tool only specifies how to conduct the tests. The ITU itself does not publish individual MOS scores for individual codecs. Results will therefore vary from vendor to vendor and test to test.

MOS Rating of Digital Voice

Clear distinctions emerge when the MOSs of different compression schemes are listed side by side. Table 3-3 lists the MOSs of selected compression schemes as published by the ITU-T. From Table 3-3, we can see that G.711 PCM scores very high, has a short framing size or sampling interval (.125 seconds equals 8000 samples per second), and demands very little of the processor (requires .34 millions of instructions per second [MIPS]). G.726 ADPCM scores a bit lower than G.711, with more processor demand and the same sampling rate. G.729 and G.729a both require more processing power, but even though their bit rates are lower, they score significantly higher on MOS tests. This is where corporations can realize a savings in combining voice and data networks. For a little more WAN bandwidth, voice can be economically carried with data and still be delivered at a quality that approaches G.711 PCM.

Table 3-3 *Examples of ITU-T MOS Ratings of Different Compression Schemes*

Codec	Compression Method	Bit Rate	MIPS	Compression Delay (ms)	Frame Size	MOS
G.711	PCM	64	0.34	0.75	0.125	4.1
G.726	ADPCM	32	13	1	0.125	3.85
G.728	LD-CELP	16	33	3–5	0.625	3.61
G.729	CS-ACELP	8	20	10	10	3.92
G.729a	CS-ACELP	8	10.5	10	10	3.9

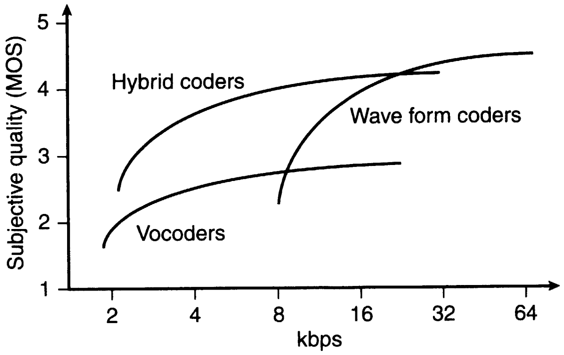
MOS Under Varying Conditions

Although it is a good starting point, the ITU's method of ratings of voice compression should be used only for comparative purposes. In the real world of deploying voice with data, we are faced with more varying conditions than simply a collection of people in a laboratory listening to quotes about chicken legs. In the real world, we must consider that we may have soft speakers, the network may exhibit bit errors (where data may be retransmitted, voice must be delivered even with errors because it's real-time), and/or the network may have framing errors. We must also consider tandem codings, where the compressed voice is decompressed at an intermediate site, then compressed again into the same or different format and forwarded to a third site. All of these conditions will cause the MOS to suffer and the voice quality to degrade quite rapidly, as in the practice of making a copy of a copy of a videotape.

MOS Scores Compared

Given the MOS ratings of different types of coding schemes, we can plot the MOS results on a comparative graphic. As shown in Figure 3-8, vocoders always tend to score very low, no matter what their bit rate. Wave form coders (such as PCM and ADPCM) score much better, but require higher bit rates. Their quality drops significantly at lower bit rates. However, the hybrid coders (such as LD-CELP and CS-ACELP) score very high even at lower bit rates. As Figure 3-8 illustrates, 8K CS-ACELP is very close in quality to 64K PCM.

Figure 3-8 *MOS Subjective Analysis*



The results shown in Figure 3-8 are simply an example from the author's perspective and do not relate scientifically to any particular test. Most typically, wave form coders tend to score highest and vocoders score lowest. Table 3-4 lists the MOS grades used to illustrate the MOS scores in Figure 3-8. As an example, hybrid coders score about 3.9 at a data rate of 8 k, which is typically representative of results for G.729.

Table 3-4 *MOS Grades*

Grade	Quality	Level of Impairments
5	Excellent	Imperceptible
4	Good	Just perceptible, not annoying
3	Fair	Perceptible and slightly annoying
2	Poor	Annoying but not objectionable
1	Bad	Very annoying and objectionable

Channel Signaling Types and Frame Formats

When Cisco voice users connect to the Public Switched Telephone Network (PSTN) or to PBX systems via a digital T1 or E1 interface, most telephony vendors agree on the standard G.711 PCM coded format. This coding format is used by default in most digital telephony systems. In order to better understand the responsibilities of both the PBX vendor (or the PSTN provider) and the installed Cisco Voice router, we must take a look at the format of the digital interface. The following sections explore the workings of both T1 and E1 as they relate to interconnecting devices for the purpose of voice transmission. We will examine the physical properties of the connections as well as the logical or framing properties. Synchronization and timing is also explained, as are methods for conveying on-hook and off-hook conditions of each telephone.

T1/DS1

Most people in the IT business unknowingly throw around acronyms and abbreviations without regard for their original meaning or their intended purpose. Such is the case when we talk about the digital service available in North America that we call T1 or DS1. In reality, the two terms have separate meanings, and each should be taken in its own context.

The term T1 dates back to its original deployment in the 1960s, when all installed circuits at this speed (1.544 Mbps) had to run on terrestrial (hence the *T* in T1) facilities. Dedicating land-line copper to the circuit was important at the time T1 was first installed because regenerative repeaters had to be installed every 4000 to 6000 feet along the path of the facility; otherwise, the signal would degrade. Therefore, when speaking of the term T1, we use it in the context of its physical characteristics. Modulation of the ones and zeros along the wire is a physical characteristic of T1. Electrical termination impedance is a physical characteristic of T1.

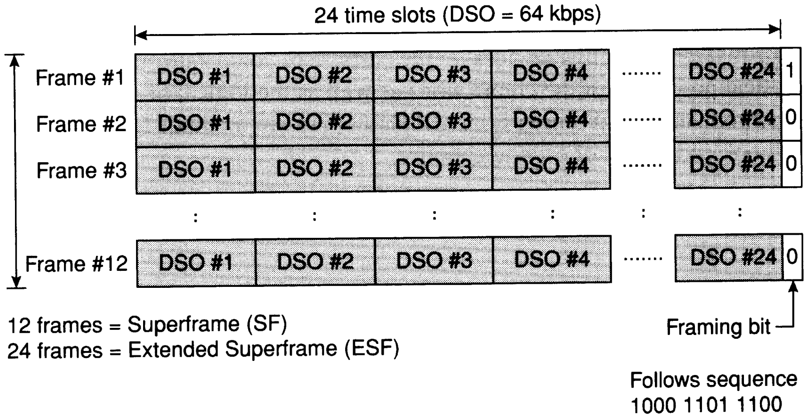
The term DS1 describes the framing characteristics of this digital transmission method (for digital signaling level 1). DS1 describes the facility as capable of carrying 24 DS0s, or a

DS1 frame, and grouping the frames into contiguously structured Superframes (that is, 12 frames) or Extended Superframes (that is, 24 frames). The details and reasoning behind the DS1 framing structure are addressed in the next section. Subsequent sections will explain options for framing structures, how the individual telephone channels signal for on-hook and off-hook, and the critical issue of end-to-end timing.

DS1 Digital Signal Superframe Format

Since the original design behind DS1 was to carry voice (not data, as most believe), we must consider that the framing format is set up so as to carry one PCM sample from each of 24 simultaneous telephone conversations along the same wire. Figure 3-9 shows the format of DS1 frames, each carrying, in sequence, an 8-bit sample from the first telephone channel, followed by an 8-bit sample from the second telephone channel, and so on up to 24 channels. The samples, when added together, result in 192 bits (24 frames × 8 bits per frame), which are sent in a serial fashion. At the end of each frame is a special bit, the 193rd bit, which serves as a synchronization function. We will address it later, in the section "Robbed Bit Signaling."

Figure 3-9 *DS1 Framing Format*



It is important to note that the transmitting device (that is, the PBX) will need to transfer 8000 frames per second to the receiving device (the Cisco Voice router) in order to re-create one second's worth of audio for all of the 24 separate telephone channels. (Recall that the sampling rate of PCM is 8000 times per second, with each sample issuing an 8-bit word at each interval.) Now for the math: If a frame consists of 193 bits, and 8000 frames are transmitted every second, then 1544000 bps (that is, 193 × 8000) is the resulting data rate.

Viewed another way, 24 telephone channels each use 64000 bps using PCM, so 24 × 64000 = 1536000 bps, plus the 8000 framing bits that must be sent (one with every frame), for a total of 1544000 bps.

Robbed Bit Signaling

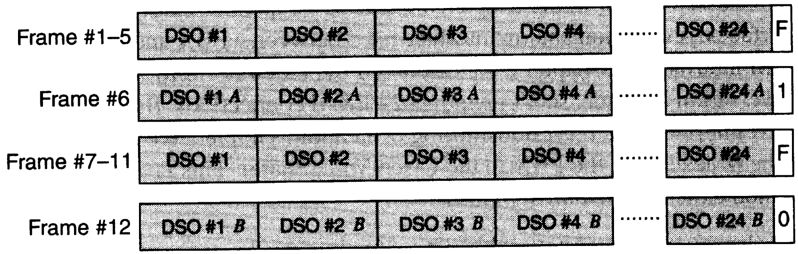
As discussed earlier in the chapter, T1 was developed to *replace* individual analog interfaces. A common analog interface used between a PBX and a Cisco Voice router is the E&M interface. In Chapter 2 we discussed the fact that E&M interfaces carry one voice conversation at a time, so T1 *emulates* 24 E&M connections. But there's more to digitizing E&M interfaces than just transmitting 8-bit audio samples. We must also convey the status of the M lead from the PBX to the Cisco Voice router. This is accomplished through a method called *robbed bit signaling*.

In order to understand how robbed bit signaling works, we must go back and take a closer look at the 193rd bit used for frame synchronization in each frame. This synchronization function is accomplished by having the 193rd bit follow a unique pattern during transmission. The pattern that the 193rd bit follows is called the *D4 framing pattern*. The sequence followed, in the Superframe format, is 1000 1101 1100. Figure 3-9 shows that the first frame in a Superframe carries a 1 bit in its 193rd position. The second frame carries a zero. The third frame also carries a zero. This continues through the 12 frame sequence at which time a new Superframe begins with the 193rd bit following the same sequence again. Using this unique bit pattern in the 193rd position, the receiving T1 device can identify frame number 1, frame number 2, frame number 3, and so on up to frame number 12 in a given Superframe.

NOTE In the 12-bit D4 framing pattern, a receiving T1 device need only receive 4 frames to declare synchronization. This is because any 4 consecutive framing bits in the sequence are unique from any other 4. If a T1 device receives 4 frames, for example, with the framing pattern 0110, it can establish that these are frame numbers 4, 5, 6, and 7 in a Superframe. If you study the framing pattern, you will see that any 4 consecutive bits are unique.

What does all this have to do with signaling? The robbed bit signaling method will identify frame number 6, and use the least significant bit in each of the 24 channels to convey the status of the M-lead from the PBX to the Cisco Voice router. (Remember that there are 24 separate E&M emulated connections, so there are 24 separate M-leads to convey.) As each of the 24 channels in frame number 6 are representing an 8-bit PCM word (coded from a 1/8000-second sample), the transmitting T1 device robs, or overwrites, the least significant bit and uses it to convey the M-lead. This happens in the 12th frame as well. The robbed bit from the 6th frame is called the A bit, and the robbed bit from the 12th frame is called the B bit. Figure 3-10 shows the format of the 6th and 12th frames. Taken together, the A and B bits can convey four different states: 00, 01, 10, and 11. This becomes important when telephones and telephone lines are capable of more than just on-hook and off-hook. A taken with B allows the PBX to convey additional signals, such as transfer or hold.

Figure 3-10 *Robbed Bit Signaling Format*



A and *B* represent robbed bits in each 8-bit sample.

Since robbed bit signaling overwrites and uses one of the bits from an 8-bit audio sample in each of the DS0s (that is, every 6th frame), the samples in the 6th and 12th frame lose some of their quantization reference. Instead of the sample being represented by 8 bits, 7 bits are effectively used for each channel every 6th frame (the 8-bit samples are left alone for all the other frames). As a result, quantization errors are introduced, thereby compromising the quality of the received and re-created analog signal. Smoother circuits, as discussed earlier in the chapter, are then even more important to get rid of the resulting quantization noise.

Extended Superframe

In the early 1980s, AT&T identified a need to include an in-band management channel for T1. This management channel would be used to command T1 devices to initiate loopbacks on certain DS0s to be tested, and to return the number of framing or other such errors to the central office for evaluation purposes. In addition, the format of T1 could be changed in such a way as to allow a CRC to be included to identify, but not correct, framing errors. It was decided that the 8000 "framing" bits would be repurposed for these uses. As this new framing method was designed, 4000 of the 8000 bits would be used for the management channel, 2000 of the bits would be used for CRC checks, and 2000 would still be used for frame synchronization.

From this idea, the Extended Superframe (ESF) format was born. While Superframe employs a format that consists of 12 contiguous frames, ESF uses 24 frames. Of the 24 frames, 12 frames carry management information in their 193rd bit position, 6 frames carry CRC information, and 6 frames carry synchronization bits. Table 3-5 illustrates the ESF format starting with the 1st frame and ending with the 24th. Each of the 24 frames in the ESF format has its 193rd bit used for a special purpose. Robbed bit signaling can now represent 16 different situations using ABCD bits, or, optionally, four situations using AB bits, or two situations using only the A bit. Notice that the first frame carries a management bit, called a data link control (DLC) bit. The second frame carries a CRC (or block check [BC]) bit. The 3rd frame carries a DLC bit again, and the 4th frame carries a synchronization, or F_e (framing extended) bit. This cycle continues until the 24th frame is transmitted, then it starts again.

Table 3-5 *Extended Superframe Format*

	Use of 193^{rd} Bit			DS0 Bits		Signaling Bits		
Frame	**Fe**	**DL**	**BC**	**Traffic**	**Signaling**	**2**	**4**	**16**
1		M		Bits 1-8				
2			C1	Bits 1-8				
3		M		Bits 1-8				
4	0			Bits 1-8				
5		M		Bits 1-8				
6			C2	Bits 1-7	Bit 8	A	A	A
7		M		Bits 1-8				
8	0			Bits 1-8				
9		M		Bits 1-8				
10			C3	Bits 1-8				
11		M		Bits 1-8				
12	1			Bits 1-7	Bit 8	A	B	B
13		M		Bits 1-8				
14			C4	Bits 1-8				
15		M		Bits 1-8				
16	0			Bits 1-8				
17		M		Bits 1-8				
18			C5	Bits 1-7	Bit 8	A	A	C
19		M		Bits 1-8				
20	1			Bits 1-8				
21		M		Bits 1-8				
22			C6	Bits 1-8				
23		M		Bits 1-8				
24	1			Bits 1-7	Bit 8	A	B	D

Two standard protocols are used on the management (M), or DLC, channel: ANSI T1.403 and AT&T's proprietary standard. The Cisco router may be configured to use either. The decision to configure for either method is determined by the protocol being used by the PBX or PSTN provider.

You can also see from Table 3-5 that the ESF format provides for robbed bit signaling in the 6th, 12th, 18th, and 24th frames. This results in an ABCD robbed bit format, allowing for 16 different signaling representations. In the event that only four representations are necessary, the 6th frame carries an A bit, the 12th frame carries a B bit, the 18th frame repeats the A bit, and the 24th frame repeats the B bit. If only two representations are needed, each robbed bit position carries the A bit.

Clocking and Line Coding

Most T1 equipment has options for clocking reference for the purpose of bit synchronization. These options are as follows:

- Internal: Synchronization is derived from an independent internal oscillator in the T1 equipment.
- Clock from Network: Synchronization is derived from the bits arriving from the received signal. The synchronization is then tied to the internal oscillator and used for transmitting all other signals.
- Loop Timed: Synchronization is derived from the received signal but is not referenced to the internal oscillator. It is, however, used to transmit signals on this interface only.

Most of the time, T1 equipment is optioned for Clock from Line, also known as Clock from Network. This clocking method ensures that the received bits will follow the clocking rate of the network, which sources its timing from a very accurate clock at the heart of the digital network. In order for the receiving T1 equipment to develop the clock, the receiving T1 device watches the pulses applied to the circuit and measures the timing between pulses to develop a timing source. The timing source develops by applying the electrical pulses on the wire to a crystal oscillator, which vibrates at a relatively constant rate. This type of clocking circuit regenerates itself by applying the crystal's output pulses back to the input of the circuit, along with the pulses received from the T1 signal for reference. An absence of pulses on the circuit causes the reference clock to drift. Transmitted ones pulse the circuit, and transmitted zeros do not. Because of the lack of pulses when transmitting zeros, carriers require that the customer follow certain rules to ensure that enough pulses are present on the circuit to develop a proper clock reference. These rules are referred to as the Ones Density rules and are stated as follows:

- No more than 15 consecutive zeros are allowed.
- At least 12.5 percent of the user traffic must be ones.

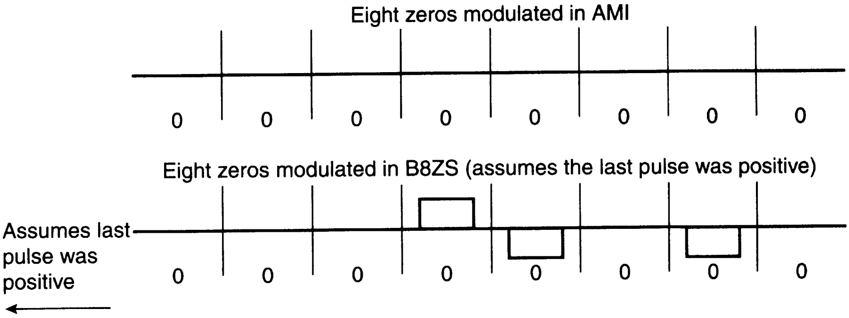
NOTE A T1 circuit is said to use a modulation scheme called Bipolar Alternate Mark Inversion Return-to-zero (or AMI). This means that when ones are to be transmitted, they will pulse the circuit with a square wave in an alternating bipolar fashion. Successive ones will pulse in opposite directions, positive and negative, and successive pulses of the same polarity are not allowed. When successive pulses do modulate in the same polarity by mistake, this error is called a bipolar violation (BPV). Zeros do not pulse the circuit at all.

When a T1 is used to carry voice traffic in the form of PCM, the Ones Density rules are always followed because there is no 8-bit sample represented by 8 zeros (see the section "Quantization" earlier in this chapter). Trouble occurs, however, when users utilize unused voice DS0s to carry data. Data has a propensity to transmit varying strings of zeros because the source of the data cannot be controlled. This may result in strings of zeros across multiple DS0s that do not pulse the clocking circuit, thereby causing the clock to drift from reference. The initial attempt at ensuring that the Ones Density rules were followed when carrying data traffic was to force one of the bits in every 8-bit DS0 to a 1. This was done only in DS0s that carry data and not in DS0s that carry voice. Effectively, then, each DS0 bearing data used 7 bits 8000 times per second for a total data rate of 56000 bits per second. This is where the 56000 data rate came from and is still in use today for circuits that use AMI line coding.

If the entire T1 is used for carrying data, the effective user traffic is reduced to 1344000 bits per second due to the ones used to ensure Ones Density rules are followed (56000 × 24 = 1344000). This made early users of T1 quite upset, since they paid for 1544000 and lost 200000 bits each second. AT&T then went back to the engineering labs to develop a modulation method to ensure Ones Density, regardless of user source (voice or data).

A new modulation scheme emerged, called Bipolar with Eight-Zero Substitution Return-to-zero (or B8ZS). In this modulation scheme, a substitution pattern containing intentional BPVs is inserted when eight consecutive zeros need to be transmitted. The substitution pattern contains BPVs in the 4th and 7th bit positions, and the receiver then reads the 8-bit sequence as 8 zeros. Figure 3-11 shows the implementation of B8ZS compared to pulses in AMI. Since the substitution pattern does contain pulses, clock reference is maintained.

Figure 3-11 *AMI and B8ZS Modulation Compared*


Eight zeros modulated in AMI
0 0 0 0 0 0 0 0
Eight zeros modulated in B8ZS (assumes the last pulse was positive)
Assumes last pulse was positive
0 0 0 0 0 0 0 0

E1 Channel Signaling

Like T1, the European digital transmission method for PCM voice, E1, has physical characteristics and framing characteristics. E1 can terminate at impedances of 75 or 120 ohms, and can use AMI or HDB3 (high density bit 3) line coding. E1 can also run balanced or unbalanced by standard, meaning that the two wires used for transmission can vary independently in voltage to represent ones and zeros, or one wire represents ground and the other wire varies in voltage.

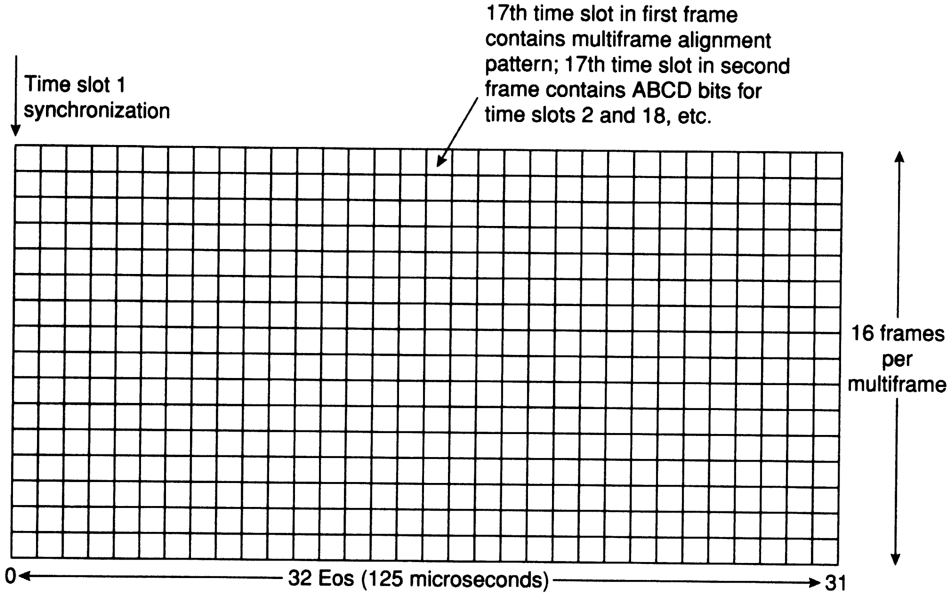
NOTE HDB3 is the E1 equivalent of B8ZS for ensuring Ones Density. HDB3 inserts a BPV in the fourth bit position of a 4-bit sequence of consecutive zeros. If the receiver sees three consecutive zeros, it checks the next bit. If it's a BPV, then the transmitter inserted it, and it is interpreted as a zero. This ensures enough pulses on the circuit for timing to be maintained.

Like T1, E1 also has structured framing characteristics and signaling methods and line coding. These methods are not always exactly like those used in T1, and vary due to the attempt to recommend a standard that can be deployed worldwide, regardless of infrastructure. These ideas are explained in the following sections.

E1 Framing and Signaling

The E1 framing method defines 32 DSOs as one frame, with 16 contiguous frames considered a Multiframe. Figure 3-12 shows the format of an E1 Multiframe. The user capacity for E1 voice channels is 30 DSOs, with the first full DSO carrying synchronization patterns (there is no separate framing bit in E1), and the center DSO used to carry ABCD signaling. The first frame in a Multiframe has its first DS0 carrying a unique 8-bit pattern establishing the beginning of a new frame. The 17th DS0 carries a different unique pattern, signifying the first frame of a Multiframe. The second frame in a Multiframe has its first DS0 carrying the same pattern as in the first frame, while the 17th DS0 is used to carry ABCD for the second DS0 and the 18th DS0. This continues until the 16th frame is completed, thus finishing one Multiframe.

Figure 3-12 *E1 Multiframe*


Time slot 1 synchronization
17th time slot in first frame contains multiframe alignment pattern; 17th time slot in second frame contains ABCD bits for time slots 2 and 18, etc.
16 frames per multiframe
0
32 Eos (125 microseconds)
31

E1 Line Coding

Unlike T1, E1 can be deployed with Return-to-zero line coding or Non-return-to-zero line coding. It also has two modulation formats as noted previously: AMI and HDB3. These variations are necessary to ensure that E1 can be deployed in poor infrastructures (such as those found in third-world countries) without needing expensive twisted pair and coaxial cabling combinations.

Digital Channel Signaling Types: CAS and CCS

The way that T1 carries signaling information within each of the DS0s every 6th frame is called *channel-associated signaling (CAS)*, which is synonymous with robbed bit signaling. A T1 circuit bearing encoded PCM voice channels with robbed bit signaling is said to be running in CAS mode. You will need to configure the Cisco Voice router for CAS mode when connecting the router to a CAS service, since CAS is not default for T1 interfaces.

On the other hand, E1 carries its signaling in DS0 17 all the time, so it is said to run in common channel signaling (CCS) mode. However, since the format of the ABCD bit patterns used for signaling matches that of T1 exactly, some E1 providers will still refer to their signaling as CAS. As an example, a more definitive description of CCS is ISDN BRI and PRI. Each of these services has a definite common channel used for signaling known as the D channel, which is used to set up and tear down B channel connections.

T1 and E1 Digital Telephony Compared

As is shown in Table 3-6, T1 and E1 are similar in their ability to carry PCM voice with signaling. While E1 has a larger carrying capacity than T1, both are arranged in multiple framing formats to allow signaling information to be carried. A variation of E1, called J1, is used in parts of Japan.

Table 3-6 *T1 and E1 Compared*

	T1 (ITU-T G.733)	E1/J1 (ITU-T G.732)
Sampling Frequency	8 kHz	8 kHz
Channel Bit Rate	DS0 – 64K	DS0 – 64K
Timeslots per Frame	24	32
Channels per Frame	24	30
Bits per Frame	24 × 8 + 1 = 193	32 × 8 = 256
Framing	Superframe (12) Extended Superframe (24)	Multiframe (16)
Framing Indicator	193rd bit of frame	First DS0
System Bit Rate	8000 × 193 = 1.544M	8000 × 256 = 2.048M
Signaling	Robbed bit CAS	CAS in timeslot 17

Synchronization of Digital Telephony

Because it is imperative that synchronization be maintained across a digital interface, many facets of maintaining synchronization are at work. Bit synchronization is maintained by using a primary clock reference source drawn from the circuit pulses themselves. Ones Density must be maintained in order to allow enough pulses from which to develop a reliable clock source. Time-slot synchronization is maintained by simply counting off 8 bits for each DS0. Since the entire circuit is time-division multiplexed, this task is relatively easy. Frame synchronization is maintained by having a unique framing pattern occur over the course of numerous frames grouped in a set. Superframe, Extended Superframe, and Multiframe are examples that have been discussed in this chapter. As an example, in Superframing, the 193rd bit from each frame follows a specific pattern over the course of transmitting 12 frames, then repeats itself. Extended Superframe does this over 24 frames, while E1 uses the first DS0 in its entirety.

Maintaining synchronization is even more critical when you consider that one T1 bit time lasts only 648 nanoseconds. Across miles and miles of transmission facilities, that fact becomes amazing to imagine. One 8-bit DS0 is transmitted in 5.18 microseconds, one frame takes 125 microseconds, and one Superframe lasts for only 3 milliseconds. It is amazing, when you think of it this way, that the telco service providers manage to keep it all going!

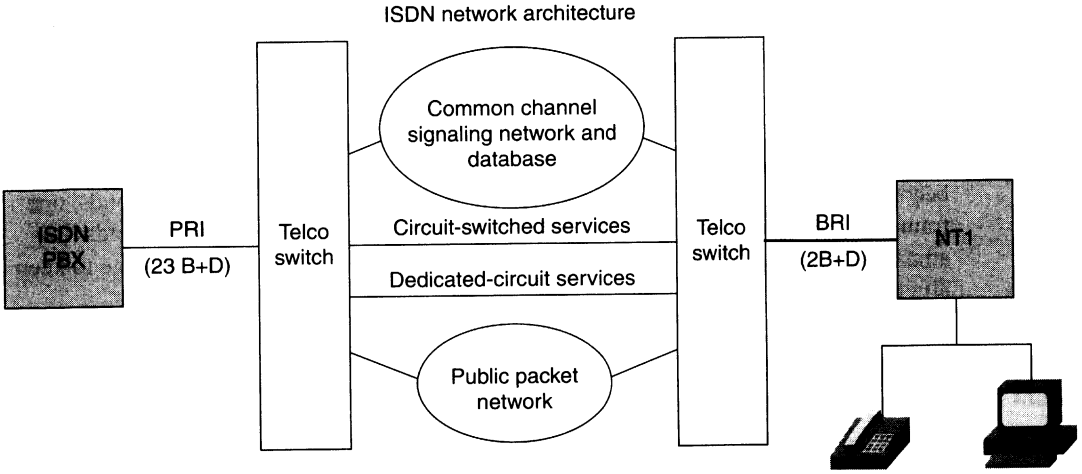
ISDN

As services to deploy voice and data over the same network become more common, ISDN will arise again, this time as a preferred connection method between PBXs and voice/data transmission equipment (such as Cisco Voice routers). Since ISDN allows access to multiple services through a common interface (2B+D, or 2 Bearer or user channels and one Delta or control channel for Basic Rate Interfaces, or BRI, and 23B+D, or 23 Bearer channels and one Delta channel for Primary Rate Interfaces, or PRI), is standards-based, and can switch channels on and off as needed, it is well suited to mixing voice and data. Data channels can be connected on the B channels when voice is not using them, and voice switches on and off on demand. In the next two sections, we will examine the architecture of ISDN networks as they may apply to voice traffic and other integrated services. We will also take a look at Signaling System 7, or SS7, an out-of-band signaling and control network originally developed by AT&T, and how ISDN lends itself to extending the services of SS7.

ISDN Network Architecture

As Figure 3-13 shows, ISDN PBXs may be installed to invoke different types of services in the telco network. Services such as circuit-switched (on-demand connections, voice or data), dedicated (such as nailed-up data connections), or packet services may be provisioned as needed. A very attractive aspect here is in considering the small office/home office (SOHO) user. Such a user has a BRI connection in the home and may need to connect to the central site for telephony services and/or data services at various times throughout the day. In consideration of these types of connections to the PBX, Cisco Voice routers have the capability to connect to PRI- and BRI-compatible PBXs. These methods are already used extensively in European countries.

Figure 3-13 *ISDN Network Architecture*


ISDN network architecture
Common channel signaling network and database
ISDN PBX
PRI
(23 B+D)
Telco switch
Circuit-switched services
Dedicated-circuit services
Telco switch
BRI
(2B+D)
NT1
Public packet network

ISDN Network Protocols

The D channel in ISDN is used to set up and tear down switched and permanent calls. At the D channel's layer 3, the ITU-T protocol Q.931 is used for this purpose. Its function is to accomplish the following:

- Establish new calls, both incoming and outgoing.
- Collect and act upon call information during the progress of the call (such as hold and transfer).
- Clear the call when requested.
- Perform miscellaneous functions, such as keeping track of the call duration, and manage and negotiate for such things as reverse charging.

As ISDN has these responsibilities between the user interface and the network interface, some of these signaling aspects need to be propagated to the telco network to accomplish requested connections and disconnections. Unfortunately, the telco network uses a different internal signaling method that is not directly compatible with Q.931.

Signaling System 7

Developed in 1984, Signaling System 7 (SS7) is an internal out-of-band signaling system within the telco network. Originally installed by the Bell System, SS7 is now funded and maintained by the major telco service providers throughout the U.S. It not only performs call setup and teardown tasks, but it also has the capability to detect a caller's source address (thus enabling caller ID) and even the caller's physical location (used by emergency response teams to detect a 911 call source).

SS7 maintains multiple databases throughout its network to keep caller source information redundant in the event of failure, and also serves to reroute call requests around failed network areas. A commercial use of SS7 might be to make a single 7-digit telephone number useful throughout all 50 of the United States, for example, to order pizza. The user need only dial only those seven digits, and SS7 would locate the caller's physical location and direct the call to the nearest of a franchise's pizza shops within the local delivery area. The physical location (that is, the caller's street address) could then be forwarded to the pizza shop on a screen to allow the shop to prepare the pizza for delivery and print directions for the delivery driver.

Q.SIG Protocol

As signaling and digital voice technologies progress, a standard means by which two PBXs can interconnect across private corporate networks is needed. The Q.SIG protocol is a standards-based protocol based on the ITU-T Q.9*XX* family of protocols, which defines a means to set up, tear down, and add additional signaling services between private PBXs. Independent of standard ISDN interconnections, Q.SIG happens to be compatible with public ISDN standards but need not be deployed with ISDN. A major advantage of Q.SIG over standard ISDN interconnections is that it adds the capability to convey nonstandard features between PBXs. This capability, known as Q.SIG GF, or Q.SIG Generic Features, provides for the conveyance of signaling and control information not provided for in standard ISDN connections. Q.SIG GF allows proprietary and nonstandard PBX signaling to be carried across the network transparently. Later versions of Cisco IOS for voice-capable routers allow the use of Q.SIQ on PBX interconnections. Q.SIG supports the following categorized services:

- **Basic service**—This category includes the signaling methods to set up, maintain, and tear down calls, similar to ISDN basic call signaling.
- **Q.SIG GF**—This category allows the conveyance of nonstandard signaling between PBXs. The transportation of this set of signaling is transparent end to end.
- **Supplementary services**—This set of signaling includes such services as call waiting, transfer, reverse charging, and call transfers.

It is important to note that Q.SIG is an alternative to standard ISDN signaling, and is used in place of Q.931 when PBXs need the additional signaling services provided for with Q.SIG. As more corporate voice networking is deployed, Q.SIG will become more prevalent for private PBX installations.

Summary

In this chapter we have discussed digital signaling. There are many different ways to encode an analog signal into digital, and these coding/compression methods yield various levels of voice quality. T1 and E1 are digital transmission methods used to convey coded analog voice channels. As they are time-division multiplexed, they carry a fixed number of channels. They also use different methods to convey signaling end to end. The formats for framing these transmission methods vary, with synchronization being the most important reason for the framing and line-coding methods. ISDN is an emerging voice PBX connection type in the U.S., with deployments already seen in Europe. Merging the capabilities of SS7 and ISDN's Q.931 to form Q.SIG will further enhance the connection options available.

Review Questions

The following questions should help you gauge your understanding of this chapter. You can find the answers in Appendix A, "Answers to Review Questions."

1 What is the rate at which analog voice samples are taken in G.711 PCM coding?

____

____

____

2 What are two examples of hybrid coding?

____

____

____

3 How does the ITU-T express voice quality measurement?

____

____

____

4 How many contiguous frames are contained in a Superframe?

____

____

____

5 What T1 line-coding method guarantees that Ones Density requirements will be met?

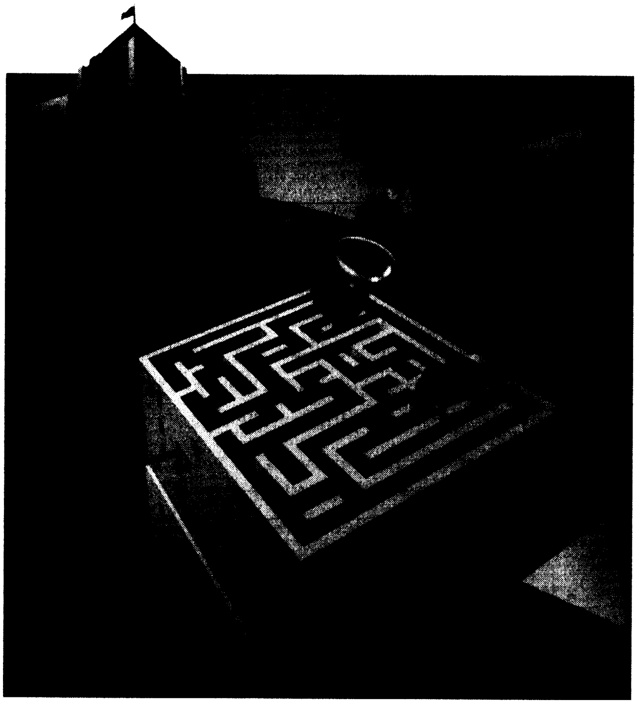
6 What is the D4 framing pattern used in T1 Superframes?

7 What is the framing format of an E1 circuit called?

8 What are the two line-coding methods used for E1?

9 How long does one bit time last in T1?

10 What is the name of the external signaling network used in the telephony network?



from Managing Cisco Network Security

by Mike Wenstrom

(1-57870-103-1)

Cisco Press

About the Author

Mike Wenstrom is an education specialist at Cisco Systems, Inc., where he designs, develops, and delivers training on Cisco's virtual private network and network security products.

Mike has chosen a career in training and instruction to help people improve their knowledge and skills in communications technologies. He especially enjoys translating complex technical subjects into an easy-to-understand form. Mike has over 18 years of experience in many facets of technical training, having been an instructional designer, course developer, technical instructor, and project manager.

While a 21-year resident of Silicon Valley, Mike worked for Cisco Systems, Aspect Communications, Siemens, IBM, ROLM, Tymnet, NCR, and the U.S. Navy. He currently develops training for and teaches Cisco's VPN and network security products in Austin, Texas, where he resides with his wife and daughter. He graduated from Western Illinois University with a BA degree. He has an AS degree in electronics technology and is a CCNA.

Contents at a Glance

Bold chapters are elements included in this folio.

Upon completing this chapter, you will be able to do the following:

- Identify the encryption protocols Cisco uses to implement IPSec support in Cisco products
- Explain the purpose and operation of each IPSec protocol supported by Cisco products
- Identify how IPSec works
- Explain how certificate authorities (CAs) work and are used
- List the general task and step procedure for configuring IPSec
- Explain how Cisco IOS Software processes IPSec

CHAPTER 15

Understanding Cisco IPSec Support

This chapter presents an overview of IP security (IPSec) and the IPSec protocols available in Cisco products that are used to create a virtual private network (VPN). Each IPSec protocol is considered here, and subsequent chapters provide details on how to configure IPSec support in Cisco products.

This chapter first considers what a VPN is, and it surveys some of the protocols used to enable a VPN. This chapter also explains what IPSec is and summarizes the protocols and security algorithms that make up IPSec. It next looks at each protocol and algorithm in more depth, considering how each works and how IPSec uses it. It includes a summary of the tasks you must perform to configure IPSec and examines the first task in detail because it is common to all IPSec methods. This chapter concludes with a quick look at how IPSec is processed and configured in Cisco IOS Software.

Using IPSec to Enable a Secure VPN

There is much interest in the networking industry concerning VPNs—how to enable them and how they fit into the enterprise network architecture. A *VPN* is an enterprise network deployed on a shared infrastructure employing the same security, management, and throughput policies applied in a private network. VPNs are an alternative wide-area network (WAN) infrastructure that can be used to replace or augment existing private networks that utilize leased lines.

VPNs fall into three categories:

- **Remote-access**—Remote-access VPNs connect telecommuters, mobile users, or even smaller remote offices with minimal traffic to the enterprise WAN and corporate computing resources.
- **Intranets**—An intranet VPN connects fixed locations, branch offices, and home offices within an enterprise WAN.
- **Extranets**—An extranet extends limited access of enterprise computing resources to business partners such as suppliers or customers, enabling access to shared information.

A variety of protocols can be used to enable each type of VPN.

VPN Protocols

Many protocols have been developed to create VPNs. Each protocol has characteristics that enable specific VPN features. For example, IPSec (the focus of this chapter) is an industry-standard network layer encryption method that enables the establishment of authentication and encryption services between endpoints over a shared IP-based network. Other protocols enable VPN features by using tunneling, the ability to enclose or encapsulate data or protocols inside other protocols. The following are the most common tunneling protocols used to enable VPNs:

- **Generic Routing Encapsulation (GRE)**—A tunneling protocol developed by Cisco that encapsulates a wide variety of protocol packet types inside IP tunnels, creating a virtual point-to-point link to Cisco routers at remote points over an IP network.
- **Layer 2 Forwarding (L2F)**—A tunneling protocol developed by Cisco that enables a virtual private dialup network (VPDN), a system that permits dial-in networks to exist remotely to home networks while giving the appearance of being directly connected to an enterprise network.
- **Point-to-Point Tunneling Protocol (PPTP)**—A network protocol developed by Microsoft that enables the secure transfer of data from a remote client to a private enterprise server by creating a VPN across IP-based networks. PPTP supports on-demand, multiprotocol, virtual private networking over public networks such as the Internet.
- **Layer 2 Tunneling Protocol (L2TP)**—A tunneling protocol developed by Cisco and Microsoft that enables a VPDN. L2TP is an extension to the Point-to-Point Protocol (PPP) used for VPNs, merging the best features of two existing tunneling protocols: PPTP and L2F.
- **Microsoft Point-to-Point Encryption (MPPE)**—A means of converting PPP packets into an encrypted form. Enables a secure VPN over a dialup or remote network. MPPE uses the RSA RC4 encryption algorithm to provide data confidentiality.

What Is IPSec?

Cisco VPN products use the industry-standard IPSec protocol suite to enable advanced VPN features. IPSec provides a mechanism for secure data transmission over IP networks, ensuring confidentiality, integrity, and authenticity of data communications over unprotected networks such as the Internet. IPSec enables the following VPN features in Cisco products:

- **Data confidentiality**—The IPSec sender can encrypt packets before transmitting them across a network.

- **Data integrity**—The IPSec receiver can authenticate IPSec peers (devices or software that originate and terminate IPSec tunnels) and packets sent by the IPSec peer to ensure that the data has not been altered during transmission.
- **Data origin authentication**—The IPSec receiver can authenticate the source of the IPSec packets sent. This service is dependent on the data integrity service.
- **Anti-replay**—The IPSec receiver can detect and reject replayed packets, helping prevent spoofing and man-in-the-middle attacks.

IPSec is a standards-based set of security protocols and algorithms. IPSec and related security protocols conform to open standards promulgated by the Internet Engineering Task Force (IETF) and documented in Requests for Comments (RFCs) and IETF-draft papers. IPSec acts at the network layer, protecting and authenticating IP packets between participating IPSec devices (peers) such as Cisco routers, PIX Firewalls, the Cisco VPN Client, Cisco VPN Concentrators, and other IPSec-compliant products. IPSec can be used to scale from small to very large networks.

Security Associations

IPSec offers a standard way of establishing authentication and encryption services between IPSec peers. IPSec uses standard encryption and authentication algorithms (that is, mathematical formulas) called *transforms* to facilitate secure communications. IPSec uses open standards for encryption key negotiation and connection management to promote interoperability between peers. IPSec provides methods to allow negotiation of services between IPSec peers. IPSec uses security associations to specify negotiated parameters.

A Security Association (SA) is a negotiated policy or an agreed-upon way of handling the data that will be exchanged between two peer devices. An example of a policy detail is the algorithm used to encrypt data. Both peers must use the same algorithm for encryption and decryption. The active SA parameters are stored in an SA Database (SAD) in the peers.

Internet Key Exchange (IKE) is a hybrid protocol that provides utility services for IPSec: authentication of the IPSec peers, negotiation of IKE and IPSec security associations, and establishment of keys for encryption algorithms used by IPSec. IKE is based on the Internet Security Association and Key Management Protocol (ISAKMP) and Oakley, which are protocols used to manage the generation and handling of encryption keys used by IPSec transforms. IKE is also the protocol used to form SAs between potential IPSec peers. In this book, and in Cisco router and PIX Firewall configuration, IKE is synonymous with ISAKMP; IKE is the term used for both.

Both IKE and IPSec use SAs to specify parameters. The components of IPSec, SAs, and IKE are covered in more detail later in this chapter.

IPSec Equipment Infrastructure

IPSec VPN solutions can be built using multiple Cisco devices—Cisco routers, the CiscoSecure PIX Firewall, the CiscoSecure VPN client software, and the Cisco VPN 3000 and 5000 series concentrators—as building blocks. Cisco routers integrate VPN features with the rich feature set provided by Cisco IOS software, reducing network complexity and total cost of ownership of the VPN solution while enabling layered security services. The PIX Firewall is a high-performance network appliance that provides high-capacity tunnel endpoints with strong firewall features. The CiscoSecure VPN client software supports the remote-access VPN requirements for e-commerce, road warrior, and telecommuting applications, offering a complete implementation of IPSec standards and interoperability with Cisco routers and the PIX Firewall.

How IPSec Works

IPSec involves many component technologies and encryption methods, but its operation can be broken into five main steps (see Figure 15-1):

Step 1 IPSec process initiation—Traffic to be encrypted as specified by the IPSec security policy configured in the IPSec peers starts the IKE process.

Step 2 IKE Phase 1—IKE authenticates IPSec peers and negotiates IKE SAs during this phase, setting up a secure channel for negotiating IPSec SAs in Phase 2.

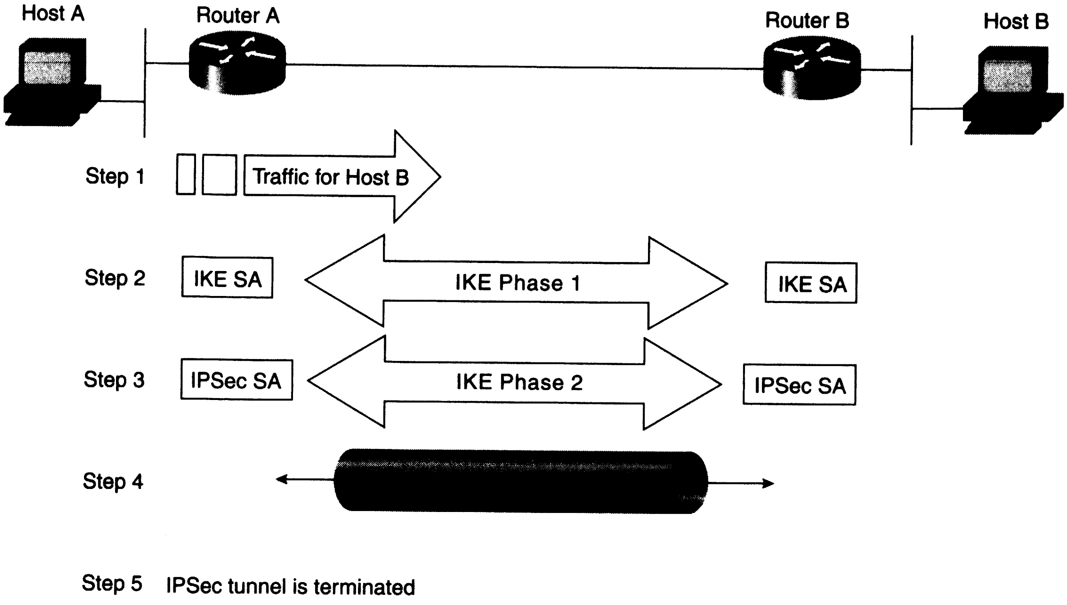
Step 3 IKE Phase 2—IKE negotiates IPSec SA parameters and sets up matching IPSec SAs in the peers.

Step 4 Data transfer—Data is transferred between IPSec peers based on the IPSec parameters and keys stored in the SA database.

Step 5 IPSec tunnel termination—IPSec SAs terminate through deletion or by timing out.

The following sections describe these steps in more detail.

Figure 15-1 *The Five Steps of IPSec*



Step 1: IPSec Process Initiation

You determine what type of traffic must be protected by IPSec as part of formulating a security policy for use with a VPN. The policy is then implemented in the configuration interface for each particular IPSec peer. For example, in Cisco routers and the PIX Firewall, access lists are used to determine which traffic to encrypt. The access lists are assigned to a crypto policy such that permit statements indicate that the selected traffic must be encrypted, and deny statements indicate that selected traffic must be sent unencrypted. With the Cisco VPN client, you use menu windows to select connections to be secured by IPSec. When traffic to be encrypted is generated or transits the IPSec client, the client initiates the next step in the process, negotiating an IKE Phase 1 exchange.

Step 2: IKE Phase 1

The basic purpose of IKE Phase 1 is to authenticate the IPSec peers and to set up a secure channel between the peers to enable IKE exchanges. IKE Phase 1 performs the following functions:

- Negotiates a matching IKE SA policy between peers to protect the IKE exchange. The IKE SA specifies negotiated IKE parameters and is bidirectional.
- Performs an authenticated Diffie-Hellman exchange with the end result of having matching shared secret keys used by IPSec encryption algorithms.
- Authenticates and protects the identities of the IPSec peers.
- Sets up a secure tunnel to negotiate IKE Phase 2 parameters.

IKE Phase 1 occurs in two modes: main mode and aggressive mode.

IKE Phase 1 Main Mode

Main mode has three two-way exchanges between the initiator and the receiver:

1. In the first exchange, the algorithms used to secure the IKE communications are agreed upon in matching IKE SAs in each peer.
2. The second exchange uses a Diffie-Hellman exchange to generate shared secret keying material used to generate shared secret keys, and to pass nonces (random numbers sent to the other party), sign them, and return them to prove their identity.
3. The third exchange verifies the other side's identity. The identity value is the IPSec peer's IP address in encrypted form.

The primary outcome of main mode is matching IKE SAs between peers to provide a protected pipe for subsequent IKE exchanges. The IKE SA specifies values for the IKE exchange: the authentication method used, the encryption and hash algorithms, the Diffie-Hellman group used (two are available), the lifetime of the IKE SA in seconds or kilobytes, and the shared secret key values for the encryption algorithms. The IKE SA in each peer is bidirectional.

IKE Phase 1 Aggressive Mode

In aggressive mode, fewer exchanges are done with fewer packets, with a resulting decrease in the time it takes to set up the IPSec session. The exchanges occur as follows:

1. On the first exchange, almost everything is squeezed into the proposed IKE SA values, the Diffie-Hellman public key, a nonce that the other party signs, and an identity packet that can be used to verify the other party's identity via a third party.
2. The receiver sends back everything that is needed to complete the exchange. The only thing left is for the initiator to confirm the exchange.

The disadvantage of using aggressive mode is that both sides exchange information before a secure channel is set up. Therefore, it is possible to sniff the wire and discover who formed the new SA. However, it is faster than main mode. Aggressive mode generally is not used by Cisco products to initiate an IKE exchange. Cisco routers and PIX Firewalls can respond to an IPSec peer originating an aggressive-mode exchange.

Step 3: IKE Phase 2

The purpose of IKE Phase 2 is to negotiate IPSec SAs to set up the IPSec tunnel. IKE Phase 2 performs the following functions:

- Negotiates IPSec SA parameters protected by an existing IKE SA
- Establishes IPSec SAs
- Periodically renegotiates IPSec SAs to ensure security
- Optionally performs an additional Diffie-Hellman exchange

IKE Phase 2 has one mode, quick mode, which occurs after IKE has established the secure tunnel in Phase 1. It negotiates a shared IPSec policy, derives shared secret keying material used for the IPSec security algorithms, and establishes IPSec SAs. Quick mode exchanges nonces that provide replay protection. The nonces are used to generate fresh shared secret key material and to prevent replay attacks from generating bogus security associations.

Quick mode is also used to renegotiate a new IPSec SA when the IPSec SA lifetime expires. Base quick mode is used to refresh the keying material used to create the shared secret key based on the keying material derived from the Diffie-Hellman exchange in Phase 1. IPSec has an option called Perfect Forward Secrecy (PFS) that increases keying material security. If PFS is specified in the IPSec policy, a new Diffie-Hellman exchange is performed with each quick mode, providing keying material that has greater entropy (key material life) and thereby greater resistance to cryptographic attacks. Each Diffie-Hellman exchange requires large exponentiations, thereby increasing CPU utilization and exacting a performance cost.

The identities of the SAs negotiated in quick mode are the IP addresses of the IKE peers.

IPSec Transforms Negotiated in Phase 2

IKE negotiates IPSec transforms (IPSec security algorithms) during Phase 2. IPSec consists of two main security protocols and a variety of supporting protocols. The IPSec transforms and associated encryption algorithms are summarized as follows:

- **Authentication Header (AH)**—A security protocol that provides authentication and optional replay-detection services. AH acts as a digital signature to ensure that data in the IP packet has not been tampered with. AH does not provide data encryption and decryption services. AH can be used either by itself or with Encapsulating Security Payload.

- **Encapsulating Security Payload (ESP)**—A security protocol that provides data confidentiality and protection with optional authentication and replay-detection services. Cisco products supporting IPSec use ESP to encrypt the data payload of IP packets. ESP can be used either by itself or in conjunction with AH.
- **Data Encryption Standard (DES)**—An encryption algorithm used to encrypt and decrypt packet data. DES is used by both IPSec and IKE. DES uses a 56-bit key, ensuring high performance yet secure encryption. DES is a symmetrical algorithm, requiring identical secret encryption keys in each IPSec peer. Diffie-Hellman is used to establish the symmetrical keys. IKE and IPSec use DES for message encryption.
- **Triple DES (3DES)**—3DES is a variant of DES that iterates three times with three separate keys, effectively tripling the strength of DES. 3DES is used by IPSec to encrypt and decrypt data traffic. 3DES uses a 168-bit key, ensuring strong encryption. IKE and IPSec use 3DES for message encryption.

IPSec transforms also use two standard hashing algorithms to authenticate data:

- **MD5 (Message Digest 5)**—MD5 is a hash algorithm used to authenticate packet data. Cisco products use the MD5 hashed message authentication code (HMAC) variant, which provides an additional level of hashing. A hash is a one-way encryption algorithm that takes an input message of arbitrary length and produces a fixed-length output message. IKE, AH, and ESP use MD5 for authentication.
- **Secure Hash Algorithm-1 (SHA-1)**—SHA is a hash algorithm used to authenticate packet data. Cisco products use the SHA-1 HMAC variant, which provides an additional level of hashing. IKE, AH, and ESP use SHA-1 for authentication.

IKE uses Diffie-Hellman to establish symmetrical keys using DES, 3DES, MD5, and SHA. Diffie-Hellman is a public-key cryptography protocol. It lets two parties establish a shared secret key over an insecure communications channel. Shared secret keys are required for DES and the HMAC algorithms. Diffie-Hellman is used within IKE to establish session keys. 768-bit and 1024-bit Diffie-Hellman groups are supported in Cisco products. The 1024-bit group is more secure.

Each IPSec SA is assigned a security parameter index (SPI), a number used to identify the IPSec SA. The SA specifies the IPSec transform used (ESP and/or AH and associated encryption and hash algorithms), the lifetime of the IPSec SA in seconds or kilobytes, whether PFS is specified, the IP addresses of the peers, the shared secret key values for the encryption algorithms, and other parameters. Each IPSec SA is unidirectional. A single IPSec SA negotiation results in two SAs—one inbound and one outbound.

IPSec AH and ESP can operate in either tunnel or transport mode. Tunnel mode is used between IPSec gateways and causes IPSec to build an entirely new IPSec header. Transport mode is generally used between a VPN client and a server and uses the existing IP header.

Step 4: Data Transfer

After IKE Phase 2 is complete and quick mode has established IPSec SAs, information is exchanged via the IPSec tunnel between IPSec peers. Packets are encrypted and decrypted using the encryption algorithms and keys specified in the IPSec SA. The IPSec SA contains a lifetime that measures traffic kilobytes or seconds. The SA contains a counter that counts down each second or each kilobyte of traffic transmitted.

Step 5: IPSec Tunnel Termination

IPSec SAs terminate because they are deleted or their lifetime expires. When the SAs terminate, the keys are also discarded. When subsequent IPSec SAs are needed for a flow, IKE performs a new Phase 2 and, if necessary, a new Phase 1 negotiation. A successful negotiation results in new SAs and new keys. New SAs can be established before the existing SAs expire so that a given flow can continue uninterrupted. Typically, Phase 2 renegotiations happen more frequently than Phase 1 renegotiations.

Technologies Used in IPSec

Let's examine the technologies that make up IPSec in more detail. The standards IPSec uses are complex, so we will consider each of the key technologies in more detail in this section. IPSec uses the following technologies:

- Authentication Header
- Encapsulating Security Payload
- Digital Encryption Standard
- Triple Digital Encryption Standard
- Internet Key Exchange
- Diffie-Hellman key agreement
- Hashed message authentication codes
- RSA security
- Certificate authority

Authentication Header

AH provides data authentication and integrity for IP packets passed between two systems. AH does not provide data confidentiality (that is, encryption) of packets. Authentication is achieved by applying a keyed one-way hash function to the packet to create a message digest. Changes in any part of the packet that occur during transit are detected by the receiver when it performs the same one-way hash function on the packet and compares the value of the message digest that the sender has supplied. The fact that the one-way hash also involves the use of a secret shared between the two systems means that authenticity is guaranteed. AH works as shown in Figure 15-2. Here are the details:

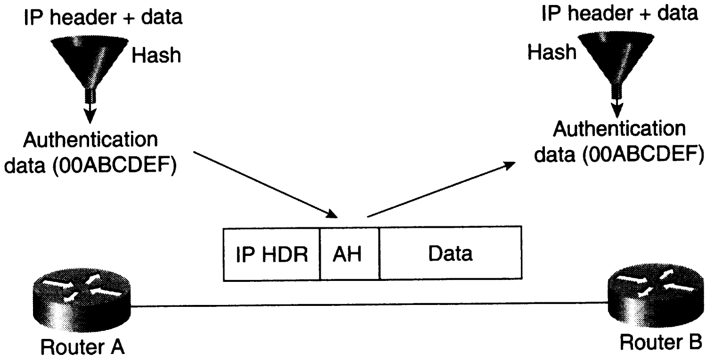
1 The IP header and data payload are hashed.

2 The hash is used to build a new AH header, which is attached to the original packet between the new AH header and the data payload.

3 The new packet is transmitted to the IPSec peer.

4 The peer hashes the IP header and data payload, extracts the transmitted hash from the AH header, and compares the two hashes. The hashes must match exactly. If even one bit is changed in the transmitted packet, the hash output on the received packet will change, and the AH header will not match.

Figure 15-2 *Authentication Header Hashing*



AH provides authentication for as much of the IP header as possible as well as for upper-level protocol data. However, some IP header fields are mutable, meaning that they change in transit. The value of the mutable fields, such as the time-to-live (TTL) field, changes as the packet transits intermediate network devices, and it might not be predictable by the sender. The values of mutable fields cannot be protected by AH. Thus, the protection provided to the IP header by AH is somewhat limited. AH may also provide optional anti-replay protection by using a sequence number in the IP packet header. RFC 2402 describes AH completely.

Encapsulating Security Payload

ESP is a security protocol used to provide confidentiality (that is, encryption), data origin authentication, integrity, optional anti-replay service, and limited traffic flow confidentiality by defeating traffic flow analysis.

ESP provides confidentiality by performing encryption at the IP packet layer. It supports a variety of symmetric encryption algorithms. The default algorithm for IPSec is 56-bit DES. This cipher must be implemented to guarantee interoperability among IPSec products. Cisco products also support the use of 3DES for strong encryption. Confidentiality may be selected independently of all other services.

Data origin authentication and connectionless integrity work together and are optional. They can also be combined with confidentiality.

The anti-replay service may be selected only if data origin authentication is selected, and its election is solely at the discretion of the receiver. Although the default calls for the sender to increment the sequence number used for anti-replay, the service is effective only if the receiver checks the sequence number. Traffic flow confidentiality requires the selection of tunnel mode. It is most effective if implemented at a security gateway, where traffic aggregation might be able to mask true source-destination patterns. Note that, although both confidentiality and authentication are optional, at least one of them must be selected.

The set of services provided by ESP depends on options that are configured during IPSec implementation and that are selected when an IPSec SA is established. However, use of confidentiality without integrity/authentication (either in ESP or separately in AH) might subject traffic to certain forms of active attacks that could undermine the confidentiality service.

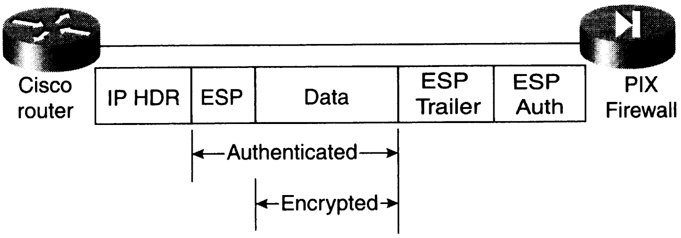
The ESP header is inserted after the IP header and before the upper-layer protocol header (transport mode) or before an encapsulated IP header (tunnel mode). RFC 2406 covers ESP completely.

ESP Encryption with a Keyed HMAC

ESP can also provide packet authentication with an optional field for authentication. Cisco IOS Software and the PIX Firewall refer to this service as *ESP HMAC*. Authentication is calculated after the encryption is done. The current IPSec standard specifies SHA1 and MD5 as the mandatory HMAC algorithms.

The main difference between the authentication provided by ESP and that provided by AH is the extent of the coverage. ESP does not protect any IP header fields unless they are encapsulated by ESP (tunnel mode). Figure 15-3 illustrates the fields protected by ESP HMAC.

Figure 15-3 *ESP HMAC Protecting the Data Payload and ESP Header*



Note that encryption covers only the data payload, and the ESP header with the ESP HMAC hash covers only the ESP header and the data payload. The IP header is not protected. ESP HMAC cannot be used alone. It must be combined with an ESP encryption protocol.

IPSec Tunnel and Transport Modes

IPSec operates in either tunnel or transport mode. Figure 15-4 illustrates tunnel mode. In tunnel mode, the entire original IP datagram is encrypted, and it becomes the payload in a new IP packet with a new IP header (HDR in Figure 15-4) and the addition of an IPSec header. Tunnel mode allows a network device, such as a PIX Firewall, to act as an IPSec gateway or proxy, performing encryption on behalf of the hosts behind the PIX. The source's router encrypts packets and forwards them along the IPSec tunnel. The destination PIX Firewall decrypts the IPSec packet, extracts the original IP datagram, and forwards it to the destination system. The major advantage of tunnel mode is that the end systems do not need to be modified to enjoy the benefits of IPSec. Tunnel mode also protects against traffic analysis. With tunnel mode, an attacker can determine only the tunnel endpoints, not the true source and destination of the tunneled packets, even if they are the same as the tunnel endpoints.

Figure 15-4 *Tunnel Mode Packets*

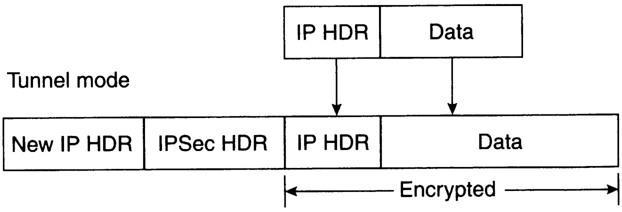
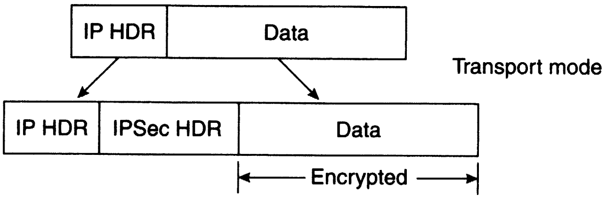


Figure 15-5 illustrates transport mode. In transport mode, only the IP payload is encrypted, and the original IP headers are left intact. An IPSec header is added. This mode has the advantage of adding only a few bytes to each packet. It also allows devices on the public network to see the packet's final source and destination. This capability allows you to enable special processing (for example, quality of service) in the intermediate network based on the information in the IP header. However, the Layer 4 header is encrypted, limiting the examination of the packet. Unfortunately, by passing the IP header in the clear, transport mode allows an attacker to perform some traffic analysis. For example, an attacker could see when many packets were sent between two IPSec peers operating in transport mode. However, the attacker would know only that IP packets were sent. He wouldn't be able to determine whether they were e-mail or another application if ESP were used.

Figure 15-5 *Transport Mode*



Using Tunnel Mode or Transport Mode

Consider some examples of when to use tunnel or transport mode. Figure 15-6 illustrates situations in which tunnel mode is used. Tunnel mode is most commonly used to encrypt traffic between secure IPSec gateways, such as between the Cisco router and the PIX Firewall, as shown in Example A in Figure 15-6. The IPSec gateways proxy IPSec for the devices behind them, such as Alice's PC and the HR servers in the figure. In Example A, Alice connects to the HR servers securely through the IPSec tunnel set up between the gateways.

Tunnel mode is also used to connect an end station running IPSec software, such as the CiscoSecure VPN client, to an IPSec gateway, as shown in Example B.

In Example C, tunnel mode is used to set up an IPSec tunnel between the Cisco router and a server running IPSec software. Note that Cisco IOS Software and the PIX Firewall set tunnel mode as the default IPSec mode.

Transport mode is used between end stations supporting IPSec, or between an end station and a gateway, if the gateway is being treated as a host. Figure 15-7 shows Example D, in which transport mode is used to set up an encrypted IPSec tunnel from Alice's PC running the Microsoft Windows 2000 client software to terminate at the Cisco VPN 3000 Concentrator, allowing Alice to tunnel L2TP over IPSec.

Figure 15-6 *Use of Tunnel Mode*

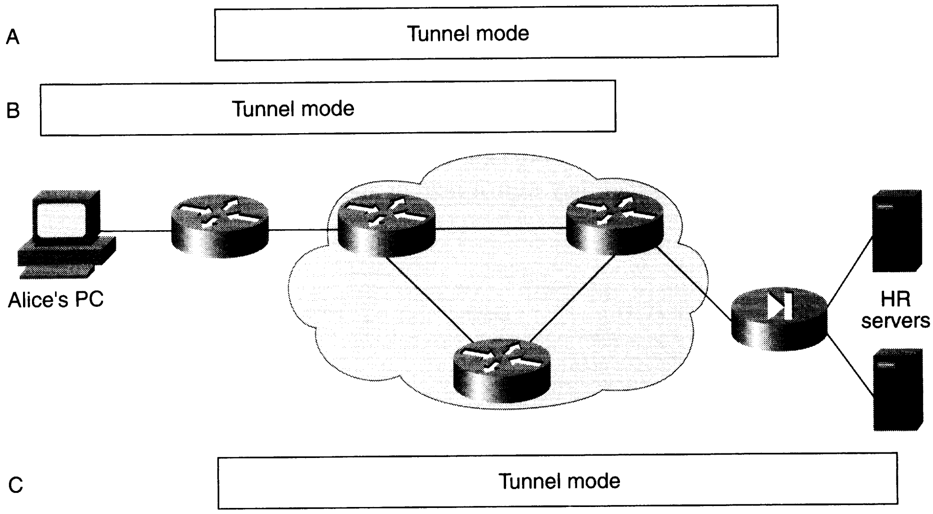
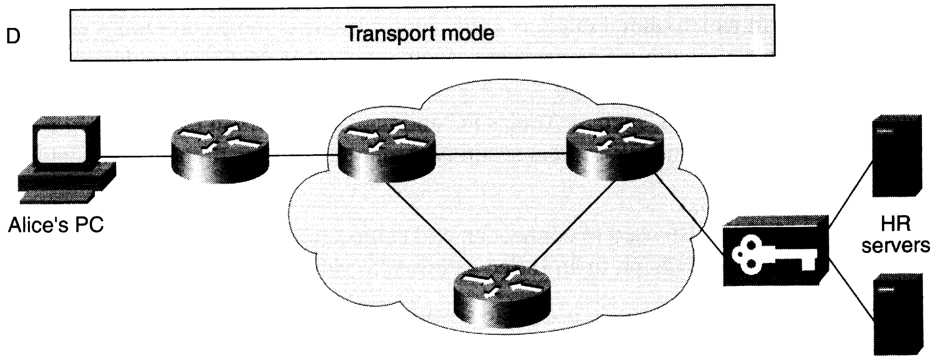


Figure 15-7 *Use of Transport Mode*



Using AH or ESP

Deciding whether to use AH or ESP in a given situation might seem complex, but it can be simplified to a few rules. When you want to make sure that data from an authenticated source gets transferred with integrity and does not need confidentiality, use the AH protocol. AH protects the upper-layer protocols and the IP header fields that do not change in transit. Protection means that the values cannot be changed without detection, so the IPSec peer will reject any altered IP datagram. AH does not protect against someone sniffing the wire and seeing the headers and data. However, because headers and data cannot be changed without the change being detected, changed packets are rejected.

If you need to keep data private (confidentiality), you use ESP. ESP encrypts the upper-layer protocols in transport mode and the entire original IP datagram in tunnel mode so that neither is readable from the wire. ESP can also provide authentication for the packets. However, when you use ESP in transport mode, the outer IP original header is not protected; in tunnel mode, the new IP header is not protected. Users will probably implement tunnel mode more than transport mode during initial IPSec usage.

Security Associations

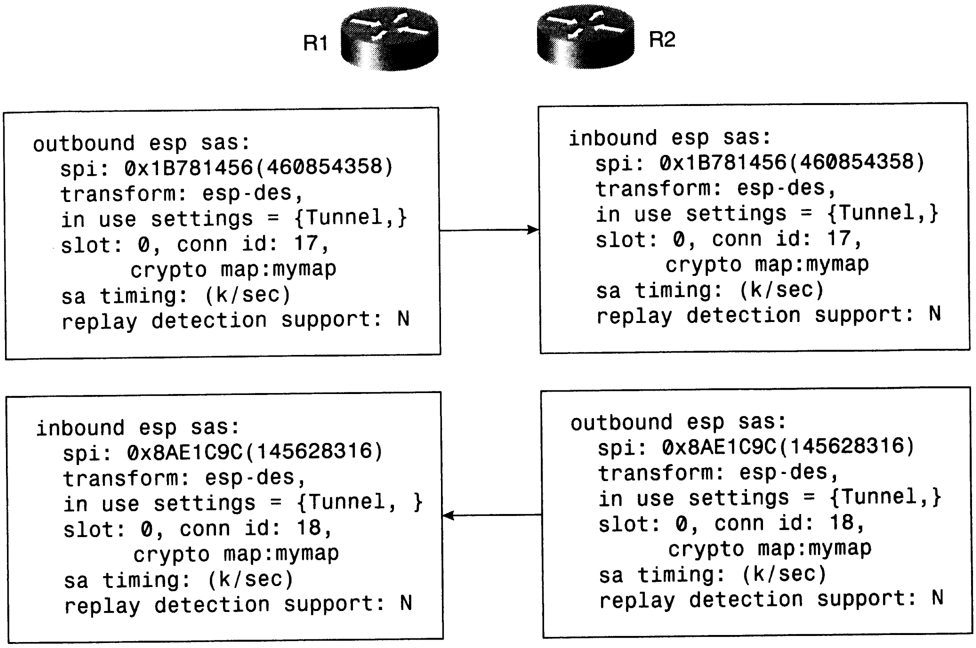
An IPSec SA is a connection between IPSec peers that determines which IPSec services are available between the peers, similar to a TCP or UDP port. Each IPSec peer maintains an SA database in memory containing SA parameters. SAs are uniquely identified by an SPI. You need to configure SA parameters and monitor SAs on Cisco products.

The IPSec SAs are set up with a quick mode exchange during IKE Phase 2. Each AH and ESP transform gets its own separate pair of IPSec SAs. Each IPSec peer agrees to set up SAs consisting of policy parameters to be used during the IPSec session. The SAs are unidirectional for IPSec, so Peer 1 will offer Peer 2 a policy. If Peer 2 accepts this policy, it sends that policy back to Peer 1. This establishes two one-way SAs between the peers. Two-way communication consists of two SAs—one for each direction.

Each SA consists of values such as a destination address, an SPI, IPSec transforms used for that session, security keys, and additional attributes such as the IPSec lifetime. The SAs in each peer have unique SPI values that are recorded in the peers' security parameter databases.

Figure 15-8 shows an example of SA parameters for two IPSec peers, Cisco Routers 1 and 2 (R1 and R2). Note that each IPSec SA is unidirectional and that the SA parameters must match on each IPSec peer. The SA parameters are configured by the system administrator, are negotiated during quick mode, and are stored in the SA database.

Figure 15-8 *Examples of IPSec SA Values*



```
outbound esp sas:
  spi: 0x1B781456(460854358)
  transform: esp-des,
  in use settings = {Tunnel,}
  slot: 0, conn id: 17,
       crypto map:mymap
  sa timing: (k/sec)
  replay detection support: N
```

```
inbound esp sas:
  spi: 0x1B781456(460854358)
  transform: esp-des,
  in use settings = {Tunnel,}
  slot: 0, conn id: 17,
       crypto map:mymap
  sa timing: (k/sec)
  replay detection support: N
```

```
inbound esp sas:
  spi: 0x8AE1C9C(145628316)
  transform: esp-des,
  in use settings = {Tunnel, }
  slot: 0, conn id: 18,
       crypto map:mymap
  sa timing: (k/sec)
  replay detection support: N
```

```
outbound esp sas:
  spi: 0x8AE1C9C(145628316)
  transform: esp-des,
  in use settings = {Tunnel,}
  slot: 0, conn id: 18,
       crypto map:mymap
  sa timing: (k/sec)
  replay detection support: N
```

Table 15-1 describes the SA parameters shown in Figure 15-8.

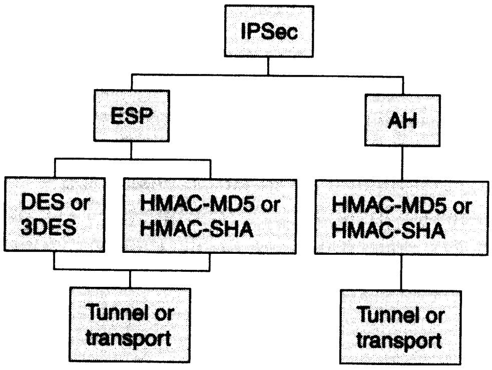
Table 15-1 *Explanation of Sample IPSec Security Association Parameters*

SA Parameter	Description
outbound esp sas: spi: 0x1B781456(460854358)	An SPI that matches inbound SPI in peer for that SA.
transform: esp-des	An IPSec transform of ESP mode to use DES.
in use settings ={Tunnel, }	The IPSec transform mode is tunnel.
slot: 0, conn id: 17, crypto map: mymap	The Cisco IOS crypto engine and crypto map information.
sa timing: (k/sec)	The SA lifetime in KB and seconds.
replay detection support: N	Replay detection that is either on or off.

IPSec Transforms

As mentioned earlier in this chapter, an IPSec transform specifies a single IPSec security protocol (either AH or ESP) with its corresponding security algorithms and mode. The AH transform is a mechanism for payload authentication. The ESP transform is a mechanism for payload encryption. Figure 15-9 illustrates possible transform combinations.

Figure 15-9 *IPSec Transforms*



Here are some examples of transforms:

- The AH protocol with the HMAC with MD5 authentication algorithm in tunnel mode is used for authentication.
- The ESP protocol with the 3DES encryption algorithm in transport mode is used for confidentiality of data.
- The ESP protocol with the 56-bit DES encryption algorithm and the HMAC with SHA authentication algorithm in tunnel mode is used for authentication and confidentiality.

Data Encryption Standard

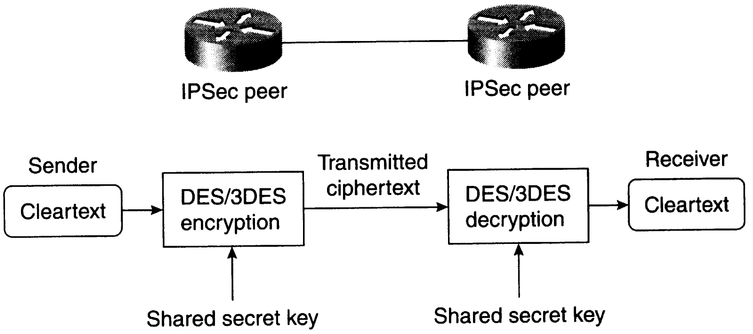
IPSec uses the 56-bit DES algorithm and 168-bit 3DES algorithm for bulk encryption in the ESP protocol and to ensure data confidentiality during IKE exchanges.

The most important feature of a cryptographic algorithm is its security against being compromised. The security of a cryptosystem, or the degree of difficulty for an attacker to determine the contents of the ciphertext, is a function of a few variables. In most protocols, the cornerstone of security lies in the secrecy of the key used to encrypt data. The DES algorithm was built so that it would be too difficult for anyone to determine the cleartext without having this key. In any cryptosystem, great lengths are taken to protect the secrecy of the encryption key.

After two IPSec peers obtain their shared secret key, they can use it to communicate with each other using the DES or 3DES encryption algorithms. Key length is a factor because it is more difficult to guess more digits than fewer. Even DES-encrypted data can be decrypted by an attacker, given enough computing power and time dedicated to finding the key. If a key were to be discovered, every packet that was encrypted with that key would easily be decrypted by the attacker. Frequently changing the shared secret keys makes it less likely that attackers can decrypt the data because there is less time to attack the key, and less data can be deciphered if a key is discovered. DES uses 56- and 168-bit key lengths.

Figure 15-10 shows how DES works and illustrates the following discussion. The components of DES encryption are the encryption and decryption algorithms, the matching shared secret keys on each peer, and the input cleartext data to be encrypted. At the core of DES is the encryption algorithm. A shared secret key is input to the algorithm. Cleartext data is fed into the algorithm in fixed-length blocks and is converted to ciphertext. The ciphertext is transmitted to the IPSec peer using ESP. The peer receives the ESP packet, extracts the ciphertext, runs it through the decryption algorithm, and outputs cleartext identical to that input on the encrypting peer.

Figure 15-10 *DES Operation*



The DES Algorithm

The DES algorithm was designed by IBM in the early 1970s. The National Security Agency (NSA) made some changes to the algorithm, approved it for general use, and published it. DES is believed to be very secure, and no one has been able to disprove this fact thus far. However, it is prudent to periodically change keys.

DES uses a 56-bit key, ensuring high-performance encryption. DES is used to encrypt and decrypt packet data. DES turns cleartext into ciphertext via an encryption algorithm. The decryption algorithm on the remote end restores cleartext from ciphertext. Shared secret keys enable the encryption and decryption. DES is a symmetrical encryption algorithm, meaning that identical 56-bit shared secret keys are required in each IPSec peer.

DES is a block-cipher algorithm, which means that it performs operations on a fixed-length block of 64 bits. Cisco's encryption algorithm incorporates cipher feedback (CFB), which further guarantees the integrity of the data received by using feedback. DES operates as follows:

1. First, DES takes a serial stream of data to be encrypted and forms it into a 64-bit block.
2. DES does a permutation of the block, after which it divides the bits into two 32-bit halves. One of the halves is run through a complex, table-specified substitution that is dependent on the key, and then the output is "exclusive ORed" with the other half of the bits. This function takes place in 16 cycles, called rounds. After each round, the two 32-bit halves are swapped.
3. Following the final round, a final permutation is applied. The resulting ciphertext is a series of bits, each of which depends on every bit of the input and every bit of the key.

The Triple DES Algorithm

3DES is also a supported encryption protocol for use in IPSec on Cisco products. The 3DES algorithm is a variant of the 56-bit DES. 3DES operates similarly to DES in that data is broken into 64-bit blocks. 3DES then processes each block three times, each time with an independent 56-bit key. 3DES effectively triples encryption strength over 56-bit DES. 3DES is a symmetrical encryption algorithm.

Internet Key Exchange

IKE is a hybrid protocol, combining the Oakley and SKEME key exchange methods inside the ISAKMP framework. IPSec uses IKE to authenticate peers, manage the generation and handling of keys used by DES and the hashing algorithms between peers, and negotiate IPSec SAs.

IPSec can be configured without IKE, but IKE enhances IPSec by providing additional features, flexibility, and ease of configuration for the IPSec standard. IKE provides the following benefits:

- It eliminates the need to manually specify all the IPSec SA parameters at both peers.
- It establishes session keys securely for use between peers.
- It allows you to specify a lifetime for the IPSec security association.
- It allows encryption keys to change during IPSec sessions.
- It allows IPSec to provide anti-replay services.
- It permits CA support for a manageable, scalable IPSec implementation.
- It allows dynamic authentication of peers.

IKE Standards

IKE uses the following methods and algorithms to accomplish its purpose:

- **ISAKMP**—A protocol framework that defines payload formats, the mechanics of implementing a key exchange protocol, and the negotiation of a security association.
- **Oakley**—A key exchange protocol that defines how to derive authenticated keying material.
- **SKEME**—A key exchange protocol that defines how to derive authenticated keying material, with rapid key refreshment.
- **DES**—An encryption algorithm that is used to encrypt packet data, ensuring confidentiality of IKE exchanges. IKE uses 56-bit DES and 3DES in Cisco products.
- **Diffie-Hellman**—A public-key cryptography protocol that lets peers establish shared secret keys over an unsecure communications channel. Diffie-Hellman is used within IKE to establish session keys.
- **MD5 and SHA (HMAC variant)**—Hash algorithms used to authenticate packet data during IKE exchanges.
- **RSA signatures and RSA encrypted nonces**—RSA signatures provide nonrepudiation, and RSA encrypted nonces (a random number used in encryption algorithms) provide repudiation. Both are used by IKE to authenticate peers.
- **X.509v3 certificates**—Digital certificates that are used with the IKE protocol when authentication requires public keys. This certificate support allows the protected network to scale by providing the equivalent of a digital ID card to each device. When two devices want to communicate, they exchange digital certificates to prove their identity (thus removing the need to manually exchange public keys with each peer or to manually specify a shared key at each peer).

IKE Phases

IKE negotiates SAs for both IKE and IPSec during two phases, with various modes, as follows:

- **Phase 1**—IKE negotiates IKE SAs during this phase.
- **Phase 2**—IKE negotiates IPSec SAs during this phase.

See the sections "Step 2: IKE Phase 1" and "Step 3: IKE Phase 2" earlier in this chapter for more information on what happens during these phases.

IKE Authentication

Potential peers in an IPSec session must authenticate themselves to each other before IKE can proceed. Peer authentication occurs during the main mode exchange during IKE Phase

1. The IKE protocol is very flexible and supports multiple authentication methods as part of the Phase 1 exchange. The two entities must agree on a common authentication protocol through a negotiation process. At this time, preshared keys, RSA-encrypted nonces, and RSA signatures are the mechanisms implemented in Cisco products:

- **Preshared keys**—The same preshared key is configured on each IPSec peer. IKE peers authenticate each other by computing and sending a keyed hash of data that includes the preshared key. If the receiving peer can independently create the same hash using its preshared key, it knows that both peers must share the same secret, thus authenticating the other peer. Preshared keys are easier to configure than manually configuring IPSec policy values on each IPSec peer, yet preshared keys do not scale well because each IPSec peer must be configured with the preshared key of every other peer it will establish a session with.
- **RSA-encrypted nonces**—Public key cryptography requires that each party generate a pseudorandom number (a nonce) and encrypt it in the other party's RSA public key. Authentication occurs when each party decrypts the other party's nonce with a local private key (and other publicly and privately available information) and then uses the decrypted nonce to compute a keyed hash. This system provides for deniable transactions. In other words, either side of the exchange can plausibly deny that he or she took part in the exchange. Cisco IOS Software is the only Cisco product that uses RSA-encrypted nonces for IKE authentication. RSA-encrypted nonces use the RSA public key algorithm.
- **RSA signatures**—With a digital signature, each device digitally signs a set of data and sends it to the other party. This method is similar to the preceding one except that it provides nonrepudiation. RSA signatures use a CA for authentication and to derive secret key values. RSA signature nonces use the RSA public key algorithm.

IKE Mode Configuration

IKE mode configuration (mode config) is an IPSec feature that allows a gateway to download an IP address (and other network-level configurations) to a client as part of an IKE negotiation. Using this exchange, the gateway gives IP addresses to the IPSec client much as a Dynamic Host Configuration Protocol (DHCP) server assigns IP addresses to a dialup client. The address supplied by mode config is known as the inside IP address and is used in the packet header (TCP or UDP) before encryption. The following steps and Figure 15-11 illustrate how mode config assigns an IP address to the IPSec client:

1 The remote user dials up his or her Internet service provider (ISP). The dialup adapter (or network interface card [NIC]) is assigned an IP address by the ISP via DHCP (address 172.16.2.121).

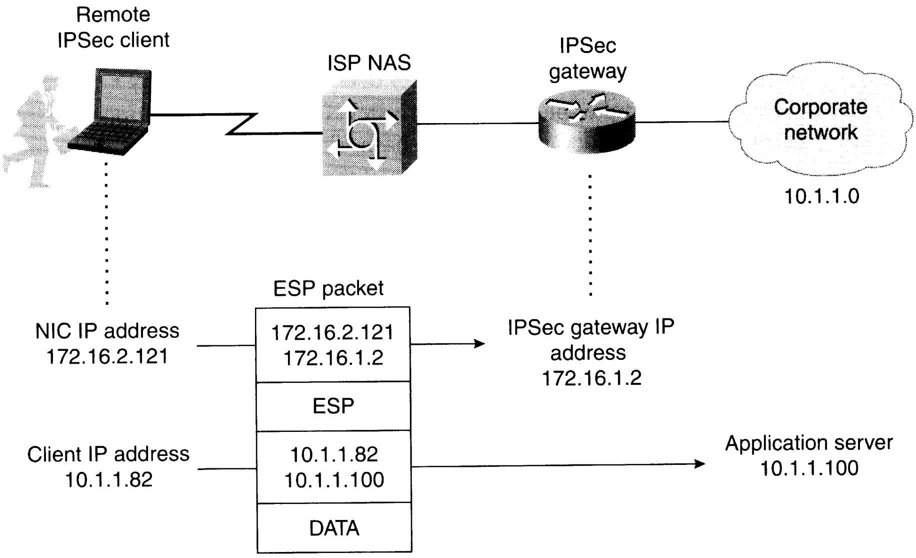
2 The IPSec client sends traffic to be encrypted and starts an IKE Phase 1 exchange to the IPSec gateway.

3 The IPSec gateway uses mode config to assign an inside IP address of 10.1.1.82 from a pool of IP addresses for clients. The inside address is used to address packets before encryption.

4 The IPSec client encrypts the packet with ESP. The ESP header uses a source address of 172.16.2.121. The destination of the ESP packet is 172.16.1.2, the IPSec gateway or tunnel endpoint.

5 The IPSec gateway receives the packet, decrypts it, and uses the inside address for the decrypted packet's IP header. The decrypted packet is sent to the corporate network destined for an application at 10.1.1.100.

Figure 15-11 *Mode Config Operation*



Mode config provides a known IP address for the client, which can be matched against IPSec policy and can be used to connect decrypted traffic to a network inside the enterprise network.

Mode config is supported in Cisco IOS Software, the CiscoSecure PIX Firewall, and the CiscoSecure VPN client. For example, using mode config, you can configure a PIX Firewall to download an IP address to a client as part of an IKE transaction. Mode config is in IETF draft status.

IKE Extended Authentication

The IKE extended authentication (XAuth) feature lets you add user authentication to IPSec for remote users. This feature provides authentication by prompting for user credentials and verifies them with the information stored in a remote security database, providing authentication, authorization, and accounting (AAA) within the VPN.

Two-factor authentication and challenge/response schemes such as SDI's SecureID and RADIUS are forms of authentication that allow a gateway, firewall, or network access server to offload the user administration and authentication to a remote security database such as a Cisco Secure ACS system or a SecureID Ace server.

IKE has no provision for user authentication. XAuth uses IKE to transfer the user's authentication information (name and password) to an IPSec gateway in a secured IKE message. The gateway uses the configured protocol (either RADIUS, SecureID, or a one-time password) to authenticate the user with a remote security database. This allows the administration of usernames and passwords to be offloaded to a remote security database within the private network that the IPSec gateway is protecting.

XAuth is negotiated between IKE Phase 1 and IKE Phase 2 at the same time as mode configuration. Authentication is performed using an existing TACACS+ or RADIUS authentication system. The XAuth feature is enabled with the **crypto map** command.

Diffie-Hellman Key Agreement

The Diffie-Hellman key agreement is a public key encryption method that provides a way for two IPSec peers to establish a shared secret key that only they know, although they are communicating over an insecure channel.

With Diffie-Hellman, each peer generates a public key/private key pair. The private key generated by each peer is kept secret and is never shared. The public key is calculated from the private key by each peer and is exchanged over the insecure channel. Each peer combines the other's public key with its own private key and computes the same shared secret number. The shared secret number is converted into a shared secret key, which is then used to encrypt data using the secret key encryption algorithms specified in the IPSec SAs, such as DES or MD5. The shared secret key is never exchanged over the insecure channel. The following steps summarize how Diffie-Hellman works:

1 The Diffie-Hellman process starts with each peer generating a large prime integer, p or q. Each peer sends the other its prime integer over the insecure channel. For example, Peer A sends p to Peer B. Each peer then uses the p and q values to generate g, a primitive root of p. Table 15-2 shows Step 1 in more detail.

Table 15-2 *Step 1 of the Diffie-Hellman Process*

Peer A Process	Peer B Process
• Generate large integer p	• Generate large integer q
• Send p to Peer B	• Send q to Peer A
• Receive q	• Receive p
• Generate g	• Generate g

2 Each peer generates a private Diffie-Hellman key (peer A: X_a; peer B: X_b) using the p and g values. Table 15-3 shows this step.

Table 15-3 *Step 2 of the Diffie-Hellman Process*

Peer A Process	Peer B Process
Generate private key X_a	Generate private key X_b

3 Each peer generates a public Diffie-Hellman key. The local private key is combined with the prime number p and the primitive root g in each peer to generate a public key, Y_a for Peer A and Y_b for Peer B. The formula for Peer A is $Y_a = g\text{^}X_a \bmod p$. The formula for Peer B is $Y_b = g\text{^}X_b \bmod p$. The exponentiation is computationally expensive. The ^ character denotes exponentiation (g to the X_a power), and mod denotes modulus or division. Table 15-4 sums up Step 3 of the process.

Table 15-4 *Step 3 of the Diffie-Hellman Process*

Peer A Process	Peer B Process
Generate the public key: $Y_a = g\text{^}X_a \bmod p$	Generate the public key: $Y_b = g\text{^}X_b \bmod p$

4 The public keys Y_a and Y_b are exchanged in public, as shown in Table 15-5.

Table 15-5 *Step 4 of the Diffie-Hellman Process*

Peer A Process	Peer B Process
Send public key Y_a to Peer A	Send public key Y_b to Peer B

5 Each peer generates a shared secret number (ZZ) by combining the public key received from the opposite peer with its own private key. The formula for Peer A is $ZZ = (Y_b\text{^}X_a) \bmod p$. The formula for Peer B is $ZZ = (Y_a\text{^}X_b) \bmod p$. The ZZ values are identical in each peer. Anyone who knows p or g, or the Diffie-Hellman public

keys, cannot guess or easily calculate the shared secret value—largely because of the difficulty in factoring large prime numbers. ZZ is also known as the value SKEYID_d in the IKE RFC 2409. Table 15-6 sums up Step 5 of the process.

Table 15-6 *Step 5 of the Diffie-Hellman Process*

Peer A Process	Peer B Process
Generate the shared secret number: ZZ = Y_b^ X_a mod p	Generate the shared secret number: ZZ = Y_a^ X_b mod p

6 Shared secret keys are derived from the shared secret number ZZ for use by DES or HMACs, as shown in Table 15-7.

Table 15-7 *Step 6 of the Diffie-Hellman process*

Peer A Process	Peer B Process
Generate shared secret key from ZZ (56-bit for DES, 168-bit for 3DES)	Generate shared secret key from ZZ (56-bit for DES, 168-bit for 3DES)

Diffie-Hellman is performed during the IKE Phase 1 main mode to initially generate keying material and to generate nonces for authentication and rekeying. It can optionally be performed during IKE Phase 2 quick mode to generate fresh keying material for IPSec SAs by combining a generated nonce with existing keying material.

The nonces are sent to the IPSec peer for authentication. The nonces are signed and returned to prove their identity (they are signed only if RSA-encrypted nonces or RSA signatures are being used for authentication), thereby providing authentication of the Diffie-Hellman exchange.

Perfect Forward Secrecy

A refresh of shared secret encryption keys occurs during an IKE Phase 2 quick mode exchange. Refreshing involves combining the current key with a random number (nonce) to create a new key using Diffie-Hellman. PFS enforces the recalculation of the shared secret key from scratch using the public key/private key generation and Diffie-Hellman techniques. The reason for the recalculation is to avoid a situation in which a hacker might have derived a particular secret key and compromised all data encrypted with that key. PFS means that a new key can be calculated that has no relationship to the preceding key.

Hashed Message Authentication Codes

IPSec uses HMACs to ensure data integrity and data origin authentication. For example, an HMAC is used to ensure data integrity and authentication during IKE Phase 1 and 2 exchanges and for IPSec AH packets. An HMAC is a mechanism for message authentication using cryptographic hash functions and a private key. A hash function or algorithm condenses a variable-length input message into a fixed-length hash of the message as output. The message hash can then be used as the message's "fingerprint." It is considered computationally infeasible to reverse the hashed value and determine the original message. Hash functions are generally fast, and the results are very secure because one-way functions are difficult if not impossible to reverse. The fundamental hash algorithms used by IPSec are the cryptographically secure MD5 and SHA-1 hash functions.

Hashing algorithms have evolved into HMACs, which combine the proven security of hashing algorithms with additional cryptographic functions. The hash produced is encrypted with the sender's private key, resulting in a keyed checksum as output. Figure 15-12 illustrates how an HMAC works. The hash function takes as input a private key and the variable-length cleartext data that needs to be authenticated. The private key length is the same as the hash's output. The HMAC algorithm is run with a resultant fixed-length checksum as output. This checksum value is sent with the message as a signature. The receiving peer runs an HMAC on the same message data that was input at the sender, using the same private key, and the resultant hash is compared with the received hash, which should match exactly.

Data integrity and data origin authentication depend on the secrecy of the secret key. If only the sender and receiver know the key and the HMAC is correct, this proves that the message must have been sent by the sender.

IPSec specifies that HMAC-MD5 and HMAC-SHA-1 are used as HMACs for IKE and IPSec.

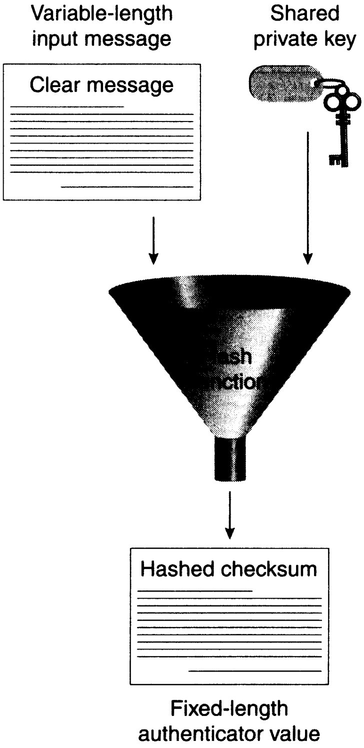
HMAC-MD5-96

IPSec uses the HMAC-MD5-96 (HMAC-MD5) encryption technique to ensure that a message has not been altered. HMAC-MD5 uses the MD5 hash developed by Ronald Rivest of the Massachusetts Institute of Technology and RSA Data Security Incorporated. It is described in RFC 1321.

HMAC-MD5 uses a 128-bit secret key. It produces a 128-bit authenticator value. This 128-bit value is truncated to the first 96 bits. After it is sent, the truncated value is stored in the authenticator field of AH or ESP-HMAC. Upon receipt, the entire 128-bit value is computed, and the first 96 bits are compared to the value stored in the authenticator field.

MD5 alone has recently been shown to be vulnerable to collision search attacks. This attack and other currently known weaknesses of MD5 do not compromise the use of MD5 within HMAC because no known attacks against HMAC-MD5 have been proven. HMAC-MD5 is recommended when the superior performance of MD5 over SHA-1 is important.

Figure 15-12 *HMAC Operation*


Variable-length input message
Shared private key
Clear message
Hash function
Hashed checksum
Fixed-length authenticator value

HMAC-SHA-1-96

IPSec uses the HMAC-SHA-1-96 (HMAC-SHA-1) encryption technique to ensure that a message has not been altered. HMAC-SHA-1 uses the SHA-1 specified in FIPS-190-1 combined with HMAC (as per RFC 2104). It is described in RFC 2404.

HMAC-SHA-1 uses a 160-bit secret key. It produces a 160-bit authenticator value. This 160-bit value is truncated to the first 96 bits. After it is sent, the truncated value is stored in the authenticator field of AH or ESP-HMAC. Upon receipt, the entire 160-bit value is computed, and the first 96 bits are compared to the value stored in the authenticator field.

SHA-1 is considered cryptographically stronger that MD5, yet it takes more CPU cycles to compute. HMAC-SHA-1 is recommended when the slightly superior security of SHA-1 over MD5 is important.

RSA Security

IPSec uses the RSA public-key cryptosystem for authentication in IKE Phase 1. RSA was developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman (hence RSA). IKE Phase 1 uses two forms of RSA: RSA signatures for digital signatures used with certificate authorities for scalable IPSec peers, and RSA encrypted nonces used with a small number of IPSec peers.

RSA generates a public key/private key pair in each IPSec peer. The public key can be transmitted over an insecure network and is used by anyone who wants to establish an IPSec session with the peer. The private key is known only to the IPSec peer and is used to decrypt data. Encryption and authentication take place without sharing the private keys: Each person uses only another's public key or his or her own private key. Anyone can send an encrypted message or verify a signed message, but only someone in possession of the correct private key can decrypt or sign a message.

RSA works as follows and is illustrated in Figure 15-13:

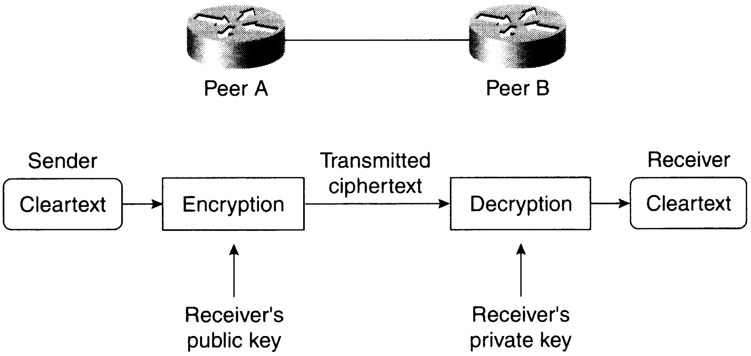
1 A cleartext message to be encrypted and the receiver's public key are input to the RSA encryption algorithm.

2 The output of the algorithm is encrypted data (ciphertext), which is transmitted to Peer B.

3 Peer B receives the ciphertext and inputs it and its own private RSA key into the RSA decryption algorithm.

4 The output is cleartext, which should match the input cleartext on Peer A.

Figure 15-13 *RSA Operation*



IKE uses the RSA algorithm in Cisco products to authenticate peers via RSA signatures and RSA encryption.

IKE Phase 1 can use preshared keys, RSA signatures, or RSA encryption to authenticate the Phase 1 exchange. Main mode generates keying material from a Diffie-Hellman exchange that must be authenticated. The first two main mode messages negotiate IKE policy, the next two exchange Diffie-Hellman public values and other data such as nonces necessary for the exchange, and the last two messages use RSA to authenticate the Diffie-Hellman exchange. The authentication method is configured in the Cisco product beforehand and is negotiated as part of the initial IKE exchange.

RSA Signatures

An RSA signature uses a nonce value and the IKE identity to exchange ancillary information, and the exchange is authenticated by signing a mutually obtainable hash. Each peer's ability to reconstruct a hash of the nonce and identity authenticates the exchange. RSA signatures work as follows:

1 RSA signature key pairs are generated with configuration commands in each peer. The RSA public keys are exchanged and authenticated, and certificates for each peer are obtained using a CA server.

2 Peer A sends its IKE identity and a signed digital certificate to peer B during IKE Phase 1. Peer B sends the same to Peer A. The signed digital certificate authenticates the IKE exchange.

RSA-Encrypted Nonces

IKE Phase 1 can use RSA-encrypted nonces to authenticate the Phase 1 exchange. Main mode generates keying material from a Diffie-Hellman exchange that must be authenticated. The first two main mode messages negotiate IKE policy, the next two exchange Diffie-Hellman public values and other data such as nonces that are necessary for the exchange, and the last two messages use RSA encryption to authenticate the Diffie-Hellman exchange if RSA encryption is the authentication method negotiated as part of the initial IKE exchange.

Simply put, RSA encryption is used to authenticate the IKE exchange by encrypting a nonce value and the IKE identity and sending it to the peer. Each peer's ability to reconstruct a hash of the nonce and identity authenticates the exchange. Consider the following steps that sum up RSA encryption operation:

1 RSA encryption key pairs are generated with configuration commands in each peer. The RSA public keys are exchanged out-of-band (via disk or e-mail) and are entered into each peer with configuration commands. Note that this manual key generation and exchange limits scalability.

2 Peer A encrypts its nonce and IKE identity with RSA encryption using Peer B's RSA public key and transmits the ciphertext to Peer B.

3 Peer B encrypts its nonce and IKE identity with RSA encryption using Peer A's RSA public key.

4 Both peers exchange the encrypted values.

5 Peer B decrypts the received ciphertext using its private RSA key and extracts Peer A's nonce and identity. It then hashes Peer A's nonce and identity and transmits the hash to Peer A.

6 Peer A decrypts the received ciphertext using its private RSA key and extracts peer B's nonce and identity. It then hashes Peer B's nonce and identity and transmits the hash to Peer B.

7 Peer A hashes its own nonce and identity and compares the hash with the one received from Peer B. The hashes should match, thus authenticating Peer B.

8 Peer B hashes its own nonce and identity and compares the hash with the one received from Peer A. The hashes should match, thus authenticating Peer A.

RSA encryption is very secure because it not only authenticates the IKE exchange but also authenticates the Diffie-Hellman exchange.

Public Key Infrastructure and CA Support

IPSec scalability, the ability to deploy large IPSec networks (those with more than 100 nodes), has been one of the greatest challenges facing implementers of network layer encryption. Digital certificate technology provides the ability for devices to easily authenticate each other in a manner that scales to very large networks.

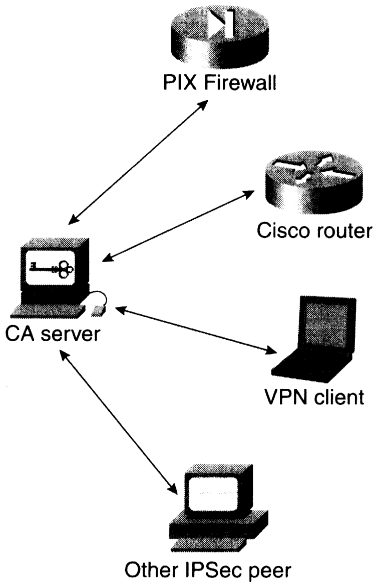
Many organizations are currently implementing a public key infrastructure (PKI) to manage digital certificates across a wide variety of applications including VPNs, secure e-mail, secure Web access, and other applications that require security. Cisco's implementation of IPSec is interoperable with the products of several leading PKI vendors. This allows you to choose the best PKI for your individual needs while knowing that it will be compatible with Cisco's network security solutions.

CAs allow the IPSec-protected network to scale by providing the equivalent of a digital identification card to each device. When two IPSec peers want to communicate, they exchange digital certificates to prove their identities. The digital certificates are obtained from a CA. CA support on Cisco products uses RSA signatures to authenticate the CA exchange.

With a CA, you do not need to configure keys between all the IPSec peers. Instead, you individually enroll each participating peer with the CA and request a certificate. When the peer has obtained a certificate from the CA, each participating peer can dynamically authenticate all the other participating peers. To add a new IPSec peer to the network, you only need to configure that new peer to request a certificate from the CA instead of making multiple key configurations with all the other existing IPSec peers.

Figure 15-14 illustrates how each IPSec peer individually enrolls with the CA server.

Figure 15-14 *CA Server Fulfilling Requests from IPSec Peers*



CA servers manage certificate requests and issue certificates to participating IPSec peers. CAs simplify the administration of IPSec peers by centralizing key management. You can use a CA with a network containing multiple IPSec-compliant devices such as PIX Firewalls, Cisco routers, the Cisco Secure VPN client, and other vendors' IPSec products.

Digital signatures, enabled by public key cryptography, provide a way to digitally authenticate devices and individual users. In public key cryptography, such as the RSA encryption system, each user has a key pair that contains both a public key and a private key. The keys are complementary—anything encrypted with one of the keys can be decrypted with the other. In simple terms, a signature is formed when data is encrypted with a sender's private key. The receiver verifies the signature by decrypting the message with the sender's public key. The fact that the message could be decrypted using the sender's public key indicates that the holder of the private key, the sender, must have created the message. This process relies on the receiver having a copy of the sender's public key and knowing with a high degree of certainty that it really does belong to the sender, not to someone pretending to be the sender.

Digital certificates provide this assurance. Digital certificates contain information to identify an IPSec peer. Digital certificates are simply a document in a specified format that contains information such as the name, serial number, company, department, or IP address of the peer and organization. It also contains a copy of the peer's public key. The certificate is itself signed by a CA, a third party that is explicitly trusted by the receiver to validate identities and to create digital certificates.

There is a problem with digital certificates. In order to validate the CA's signature, the receiver must first know the CA's public key. The CA's public key is contained in the CA's root certificate, which is installed in the IPSec peer during certificate configuration and enrollment.

IKE, a key component of IPSec, can use digital signatures to authenticate peer devices before setting up security associations. This provides scalability. When an IPSec peer enrolls with a CA, the CA provides the peer with identity (ID) certificates. The ID certificates are exchanged during IKE Phase 1 and are used to authenticate the peers, much as preshared keys are used. The ID certificate is validated with the public key of the CA itself contained in the root certificate. The certificate infrastructure is made possible by a variety of standards, which are discussed in the next section.

CA Standards Supported

Cisco routers, the Cisco Secure VPN client, and the PIX Firewall support the following open CA standards:

- **IKE**—A hybrid protocol that implements Oakley and SKEME key exchanges inside the ISAKMP framework. IKE provides authentication of the IPSec peers and negotiates IPSec keys and security associations. IKE can use digital certificates to authenticate peers.
- **Public-Key Cryptography Standard #7 (PKCS #7)**—A standard from RSA Data Security, Inc. used to encrypt and sign certificate enrollment messages.
- **Public-Key Cryptography Standard #10 (PKCS #10)**—A standard syntax from RSA Data Security, Inc. for certificate requests.
- **RSA keys**—RSA is the public key cryptographic system developed by Rivest, Shamir, and Adleman. RSA keys come in pairs: one public key and one private key.
- **X.509v3 certificates**—This standard specifies the content format of public key certificates, which are data structures that bind public key values to standard subjects. Certificate support allows the IPSec-protected network to scale by providing the equivalent of a digital ID card to each device. When two devices want to communicate, they exchange digital certificates to prove their identity (thus removing the need to manually exchange public keys with each peer or to manually specify a shared key at each peer). These certificates are obtained from a CA. X.509 is part of

the X.500 standard series created by the International Telecommunications Union (ITU). Some subjects in certificates include a unique serial number, a hashing algorithm for signing, an RSA public key, and valid dates.

- **Simplified Certification Enrollment Protocol (SCEP)**—A CA interoperability protocol that permits compliant IPSec peers and CAs to communicate so that the IPSec peer can obtain and use digital certificates from the CA. Using IPSec peers and CA servers that support SCEP provides manageability and scalability for CA support.
- **Certificate revocation lists (CRLs)**—CAs support a CRL, which is a type of certificate that lists IPSec peers that have been revoked and are no longer valid. IPSec peers can obtain the CRL from the CA. The IPSec peer should check the CRL every time an IPSec peer attempts to establish a new IKE SA to ensure that the peer is valid.
- **Registration Authority (RA)**—Some CAs have an RA as part of their implementation. An RA is essentially a server that acts as a proxy for the CA so that CA functions can continue when the CA is offline.

How CA Enrollment Works with SCEP

IPSec peers enroll with a CA using SCEP before they can use IKE configured for digital certificates. Each IPSec peer supporting SCEP contains PKI client software that enrolls with the CA in a client/server relationship using SCEP. The peer-to-CA enrollment process is summarized as follows:

1. The peer administrator configures IPSec and IKE for CA support and creates an RSA public key/private key.
2. The peer obtains the CA's public key (via the CA's own certificate and through manual authentication).
3. The peer sends its public RSA key and identity information to the CA.
4. The peer receives its own public key ID certificate and the CA's certificate from the CA.
5. The peer optionally gets the latest CRL from the CA.

After enrolling with the CA, peers exchange ID certificates via IKE, thereby exchanging their public keys. Each participant verifies the others' certificates with the CA's public key for authenticity and checks its latest copy of the CA's CRL. IPSec Security Associations (SAs) can then be set up by IKE, and a secure tunnel can be created between peers.

CA Server Support

Many vendors offer CA servers either as user-installed and user-managed software or as a managed CA service. Many CA server vendors have developed their products to interoperate with Cisco VPN products by supporting the SCEP protocol. Cisco is using the

Cisco Security Associate Program to test new CA and PKI solutions with the Cisco Secure family of products. More information on the Security Associate Program can be found at www.cisco.com/warp/customer/cc/so/neso/sqso/csap/index.shtml. The following are the vendors that interoperate with Cisco VPN products:

- **Entrust Technologies**—Entrust Technologies makes the Entrust/PKI 4.0, a CA server that supports Cisco VPN products through SCEP. Entrust/PKI 4.0 is software that is installed and administered by the user. Entrust/PKI 4.0 supports Cisco routers, the Cisco Secure VPN client, and PIX Firewalls. Entrust/PKI 4.0 runs on the Windows NT 4.0 (required for Cisco interoperability), Solaris 2.6, HP-UX 10.20, and AIX 4.3 operating systems. Refer to the Entrust Web site at www.entrust.com for more information.
- **VeriSign**—VeriSign offers OnSite 4.5, a service administered by VeriSign. Onsite 4.5 delivers a fully integrated enterprise PKI to control, issue, and manage IPSec certificates for PIX Firewalls and Cisco routers over the public Internet. Users must subscribe to the VeriSign service. Refer to the VeriSign Web site at www.verisign.com for more information.
- **Baltimore Technologies**—Baltimore has implemented support for SCEP in UniCERT (Baltimore's CA server) as well as the PKI Plus toolkit, making it easy for customers to enroll Cisco VPN products. The current release of the UniCERT CA module is available for Windows NT 4.0. Refer to the Baltimore Web site at www.baltimore.com for more information.
- **Microsoft Windows 2000 Certificate Services 5.0**—Microsoft has integrated the SCEP into its CA server through the Security Resource Kit for Windows 2000. This support lets customers utilize SCEP to obtain certificates and certificate revocation information from Microsoft Certificate Services for all Cisco's VPN security solutions. Refer to the Microsoft Web site at www.microsoft.com for more information.

IKE and IPSec Flow in Cisco IOS Software

Cisco IOS Software implements and processes IPSec in a predictable and reliable fashion. A summary of how IPSec works and the commands used to configure it are shown in Figure 15-15. The process shown in the figure assumes that you have already created your own public and private keys and that at least one access list exists. Here is the process:

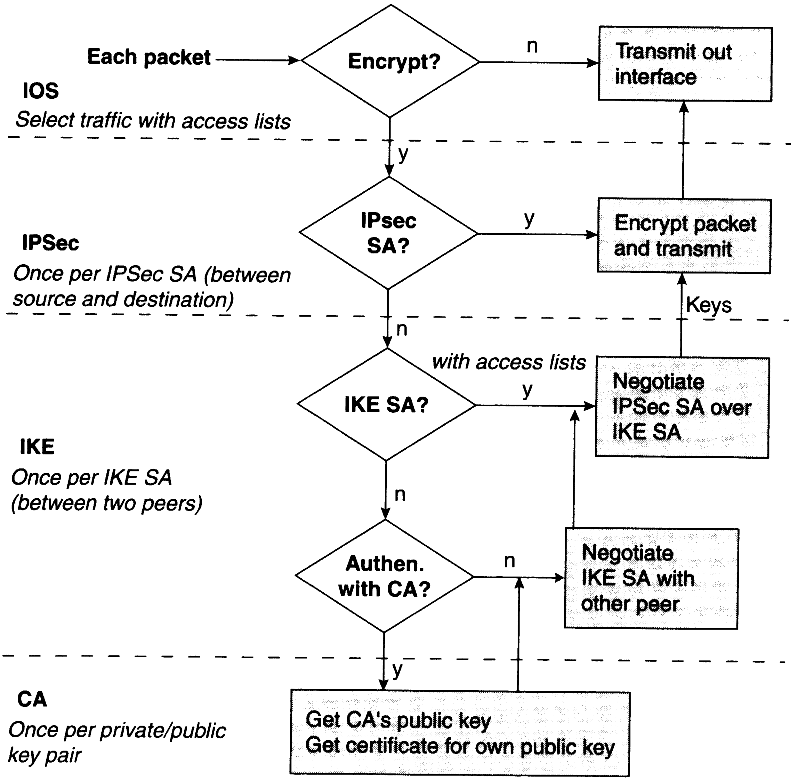
1 Cisco IOS Software uses extended IP access lists (configured with the **access-list** command) applied to an interface and crypto map to select traffic to be encrypted. Cisco IOS Software checks to see if IPSec SAs have been established. If the SA has already been established by manual configuration using the **crypto ipsec transform-**

set and **crypto map** commands, or if it was previously set up by IKE, the packet is encrypted based on the policy specified in the crypto map and is transmitted out the interface.

2 If the SA has not been established, Cisco IOS Software checks to see if an IKE SA has been configured and set up. If the IKE SA has been set up, the IKE SA governs negotiation of the IPSec SAs as specified in the IKE policy configured by the **crypto isakmp** series of commands. The packet is encrypted by IPSec and is transmitted.

3 If the IKE SA has not been set up, Cisco IOS Software checks to see if CA support has been configured to establish an IKE policy. If CA authentication is configured with the various **crypto ca** and **crypto isakmp policy** commands, the router exchanges its own digital certificate with the peer's certificates. It authenticates the peer's certificate, negotiates and establishes an IKE SA (which in turn is used to establish IPSec SAs), and encrypts and transmits the packet.

Figure 15-15 *IKE and IPSec Operation and Configuration Commands*


Each packet
IOS
Select traffic with access lists
Encrypt?
n
Transmit out interface
y
IPsec SA?
y
Encrypt packet and transmit
IPSec
Once per IPSec SA (between source and destination)
Keys
n
with access lists
IKE SA?
y
Negotiate IPSec SA over IKE SA
IKE
Once per IKE SA (between two peers)
n
Authen. with CA?
n
Negotiate IKE SA with other peer
y
CA
Once per private/public key pair
Get CA's public key
Get certificate for own public key

Configuring IPSec Encryption Task Overview

This section presents an overview of the four key tasks involved in configuring IPSec encryption using preshared keys on Cisco routers and the PIX Firewall. This section also discusses the first major task, Task 1: Prepare for IPSec, because it is common to all IPSec configuration methods. Subsequent chapters present the details of how to configure Cisco routers and the PIX Firewall for IPSec:

- **Task 1: Prepare for IPSec**—Preparing for IPSec involves determining the detailed encryption policy, determining how to establish keys between peers, identifying the hosts and networks you want to protect, determining details about the IPSec peers, determining which IPSec features you need, and ensuring that existing access lists used for packet filtering permit IPSec.
- **Task 2: Configure IKE**—Configuring IKE involves enabling IKE, creating the IKE policies, and validating the configuration.
- **Task 3: Configure IPSec**—IPSec configuration includes defining the transform sets, creating crypto access lists (access lists used to determine which traffic to encrypt), creating crypto maps (a template to enact an IPSec policy), and applying crypto maps to interfaces enabling IPSec.
- **Task 4: Test and verify IPSec**—Use **show**, **debug**, and related commands to verify that IPSec encryption works and to troubleshoot problems.

The following section discusses Task 1: Prepare for IPSec in more detail because this task is common to all methods of configuring IPSec. Tasks 2, 3, and 4 are covered in detail in subsequent chapters:

- Chapter 16, "Configuring Cisco IOS IPSec," covers configuring Cisco IOS IPSec using preshared keys.
- Chapter 17, "Configuring PIX Firewall IPSec Support," covers configuring PIX Firewall IPSec using preshared keys.
- Chapter 18, "Scaling Cisco IPSec Networks," covers configuring Cisco IOS and PIX Firewall IPSec using CA support. This chapter covers an additional task, configuring CA support.

Task 1: Prepare for IPSec

Successfully implementing an IPSec network requires that you plan before you begin configuring individual IPSec peers. Configuring IPSec encryption can be complicated. You should begin this task by defining the IPSec security policy based on the overall company security policy. See the sample XYZ Company policy in Appendix B, "An Example of an XYZ Company Network Security Policy." Some planning steps are as follows:

Step 1 Determine an IKE (IKE phase 1) policy between IPSec peers based on the number and location of the peers.

Step 2 Determine an IPSec (IKE Phase 2) policy to include IPSec peer details such as IP addresses and IPSec modes.

Step 3 Check the current configuration by using the **write terminal**, **show isakmp**, and **show crypto map** commands as well as other **show** commands.

Step 4 Ensure that the network works without encryption.

Step 5 Ensure that existing packet filtering access lists permit IPSec traffic in Cisco routers and the PIX Firewall.

The following sections examine these steps in detail.

Step 1: Determine an IKE Policy

You should determine the IKE policy details to enable the selected authentication method, and then configure it. Having a detailed plan reduces the chances of improper configuration. The planning you do here affects IKE Phase 1, main and aggressive modes. Some planning steps include the following:

- **Determine the key distribution method**—Select a key distribution method based on the numbers and locations of IPSec peers. For a small network, you might want to manually distribute keys. For larger networks, you might want to use a CA server to distribute digital certificates to support scalability of IPSec peers. You must then configure IKE to support the selected key distribution method.
- **Determine the authentication method**—Match the authentication method with the key distribution method. Cisco routers support either preshared keys, RSA-encrypted nonces, or RSA signatures to authenticate IPSec peers. PIX Firewalls support either preshared keys or RSA signatures.
- **Identify IPSec peers' IP addresses and host names**—Determine the details of all the IPSec peers that will use IKE and preshared keys to establish SAs. You will use this information to configure IKE.
- **Determine IKE policies for peers**—An IKE policy defines a combination, or suite, of security parameters to be used during the IKE main mode or aggressive mode negotiation. Each IKE negotiation begins with each peer agreeing on a common (shared) IKE policy. You must determine the IKE policy suites before beginning configuration, and you must configure IKE to support the policy details determined.

NOTE Remember that IKE is synonymous with ISAKMP. Cisco routers and the PIX Firewall have named their IKE commands as ISAKMP commands. For example, you can view IKE policies with the **show isakmp policy** command in the PIX Firewall. Actually, IKE is a newer protocol that uses the older ISAKMP and Oakley protocols.

Creating IKE Policies for a Purpose

IKE negotiations must be protected, so each IKE negotiation begins with each peer agreeing on an identical IKE policy used by each IPSec peer. This policy specifies which security parameters will be used to protect subsequent IKE negotiations.

You can configure multiple IKE policies on each peer participating in IPSec. After the two peers negotiate acceptable IKE policies, the policy's security parameters are identified by an IKE SA established at each peer, and these SAs apply to all subsequent IKE traffic during the negotiation. You can create multiple prioritized policies at each peer to ensure that at least one policy will match a remote peer's policy.

IKE negotiation begins in IKE Phase 1 main mode. IKE looks for a policy that is the same on both peers. The peer that initiates the negotiation sends all its policies to the remote peer, and the remote peer tries to find a match with its policies. The remote peer looks for a match by comparing its own highest-priority policy against the other peer's received policies in its IKE policy suite. The remote peer checks each of its policies in order of its priority (highest-priority first) until a match is found.

A match is made when both policies from the two peers contain the same encryption, hash, authentication, and Diffie-Hellman parameter values and when the remote peer's policy specifies a lifetime less than or equal to the lifetime in the policy being compared. (If the lifetimes are not identical, the shorter lifetime from the remote peer's policy is used.) Assign the most-secure policy the lowest-priority number so that the most-secure policy will find a match before any less-secure policies that are configured.

If no acceptable match is found, IKE refuses negotiation, and IPSec is not established. If a match is found, IKE completes the main mode negotiation, and IPSec security associations are created during IKE Phase 2 quick mode.

Defining IKE Policy Parameters

You can select specific values for each IKE parameter per the IKE standard. You choose a value based on the security level you want and the type of IPSec peer you will connect to. Table 15-8 shows the five parameters to define in each IKE policy as well as the relative strength of each.

Table 15-8 *IKE Policy Parameters*

Parameter	Strong	Stronger
Message encryption algorithm	DES	3DES
Message integrity (hash) algorithm	MD5	SHA-1
Peer authentication method	Preshare	RSA encryption, RSA signature

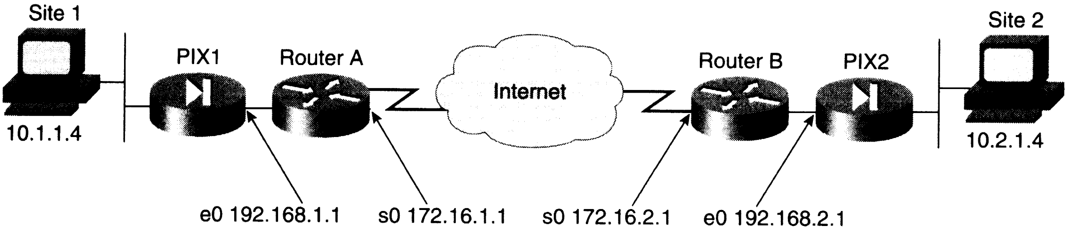
Table 15-8 *IKE Policy Parameters (Continued)*

Parameter	Strong	Stronger
Key exchange parameters (Diffie-Hellman group identifier)	Diffie-Hellman Group 1	Diffie-Hellman Group 2
IKE-established security association's lifetime	86400 seconds	Less than 86400 seconds

An IKE Policy Parameter Example for Two Peers

Figure 15-16 shows a simplified topology for the XYZ Company used in the examples in this chapter.

Figure 15-16 *XYZ Company Topology for IPSec*



Step 2: Determine IPSec (IKE Phase 2) Policy

Planning for IPSec (IKE Phase 2) is another important step you should complete before actually configuring IPSec on a Cisco router. Policy details to determine at this stage include the following:

- Select IPSec algorithms and parameters for optimal security and performance. You should determine the IPSec encryption algorithms (transforms) that will be used to secure traffic. Some IPSec algorithms require you to make trade-offs between high performance and stronger security.
- Identify IPSec peer details. You must identify the IP addresses and host names of all the IPSec peers you will connect to.
- Determine the IP addresses and applications of hosts to be protected at the local peer and remote peer.
- Select whether security associations are manually established or are established via IKE.

The goal of this planning step is to gather the precise data you will need in later steps to minimize misconfiguration.

An important part of determining the IPSec policy is to identify the IPSec peer that the Cisco router or PIX Firewall will communicate with. The peer must support IPSec as specified in the RFCs supported by Cisco products. Many different types of peers are possible, so you should identify all the potential peers. You should determine IPSec policy details for each peer before configuring IPSec. Possible peers include, but are not limited to, the following:

- Cisco routers
- The PIX Firewall
- The Cisco Secure VPN client
- Cisco VPN 3000 or 5000 series concentrators
- Other vendors' IPSec products that conform to IPSec RFCs

Table 15-9 summarizes possible IPSec policy details that you will have to determine or choose.

Table 15-9 *IPSec Policy Parameters*

IPSec Parameter	Possible Values
Transform set	AH-MD5 or AH-SHA, ESP-DES or ESP-3DES, ESP-MD5-HMAC or ESP-SHA-HMAC
IPSec mode	Tunnel or transport
Hash algorithm	MD5 or SHA-1
SA establishment	**ipsec-isakmp** or **ipsec-manual**
IPSec SA lifetime	**kilobytes** and/or **seconds**
PFS	Group 1 when using ESP-DES, Group 2 when using ESP-3DES
Peer interface	Identify peer interface (hardware or loopback)
Peer IP address or host name	Identify peer device IP address or host name
IP address of hosts to be protected	Identify hosts or networks to protect
Traffic (packet) type to be encrypted	Any traffic specified by extended IP access lists

Step 3: Check the Current Configuration

You should check the current Cisco device configuration to see if any IPSec policies already configured are useful for or might interfere with the IPSec policies you plan to configure. Previously configured IKE and IPSec policies and details can and should be used if possible to save configuration time. However, previously configured IKE and IPSec policies can also interfere with your intended policy. For example, you might add a new IKE policy that is superceded by an existing policy because the old policy has a higher priority. You will need to carefully compare existing policies with your intended new policy to ensure that the new ones fit in.

You can use the **write terminal** (**show running config** in Cisco routers) command to view the current configuration. You can also use the available **show** commands (covered in a moment) to view IKE and IPSec configuration. You can check to see if any IKE policies have previously been configured with the **show crypto isakmp policy** command in Cisco routers (**show isakmp policy** in PIX Firewalls).

Step 4: Ensure That the Network Works

Next you need to ensure that basic connectivity has been achieved between Cisco devices before configuring Cisco IOS IPSec encryption. Although a successful ICMP echo (**ping**) verifies basic connectivity between peers, you should ensure that the network works with any other protocols or ports you plan to encrypt, such as Telnet, FTP, or SQL*NET, before you begin the IPSec configuration. After IPSec is activated, basic connectivity troubleshooting can be difficult because of possible security misconfigurations and the fact that you cannot sniff the encrypted IPSec packets. Previous security settings could result in no connectivity.

Step 5: Ensure That Access Lists Permit IPSec

Perimeter routers typically implement a restrictive security policy with access lists in which only specific traffic is permitted and all other traffic is denied. Such a restrictive policy blocks IPSec traffic, so you need to add specific permit statements to the access list to allow IPSec traffic.

Ensure that access lists are configured so that protocols 50 and 51 and UDP port 500 traffic are not blocked at interfaces used by IPSec. IKE uses UDP port 500. The IPSec ESP is assigned protocol 50, and AH is assigned protocol 51. In some cases, you might need to add a statement to router access lists to explicitly permit this traffic.

A concatenated example showing access list entries permitting IPSec traffic for Router A from Router B only is shown in Example 15-1.

Example 15-1 *An Access List Permitting IPSec Traffic*

```
RouterA# show running-config
!
interface Serial0
 ip address 172.16.1.1 255.255.255.0
 ip access-group 101 in
!
access-list 101 permit ahp host 172.16.2.1 host 172.16.1.1
access-list 101 permit esp host 172.16.2.1 host 172.16.1.1
access-list 101 permit udp host 172.16.2.1 host 172.16.1.1 eq isakmp
```

Note that the protocol keyword **esp** equals the ESP protocol (number 50), the keyword **ahp** equals the AH protocol (number 51), and the **isakmp** keyword equals UDP port 500.

You might need to check your PIX Firewall to ensure that access lists on the outside interface do not block IPSec traffic. You can use the **show access-list** command to view any configured access lists. You might need to add specific access list entries to the PIX Firewall, just as you did to the perimeter router to enable IPSec traffic. A concatenated example showing access list entries permitting IPSec traffic for PIX 1 is shown in Example 15-2. Note that the source address is the peer's outside interface, and the source is PIX 1's outside interface.

Example 15-2 *Access List Entries Permitting IPSec Traffic for PIX 1*

```
access-list 102 permit ahp host 192.168.2.2 host 192.168.1.2
access-list 102 permit esp host 192.168.2.2 host 192.168.1.2
access-list 102 permit udp host 192.168.2.2 host 192.168.1.2 eq isakmp
```

Summary

This section summarizes the main points of this chapter:

- IPSec is a suite of open security protocols that work together to create a secure, scalable VPN.
- The two main protocols used by IPSec are AH, which provides data integrity and authentication with no confidentiality, and ESP, which provides all that AH provides plus data confidentiality via encryption.
- The IKE protocol is used to authenticate IPSec peers and to facilitate the creation of secret keys used by IPSec encryption algorithms.
- IKE consists of two phases. Phase 1 authenticates the IPSec peers, ensures secret key generation, and sets up a secure tunnel for Phase 2, which sets up IPSec SA.
- IPSec SAs are negotiated and established during quick mode during IKE Phase 2.
- IPSec SAs are unidirectional and specify the IPSec parameters and secret keys used for the IPSec sessions. They are stored in an SA database in dynamic memory.
- IKE uses the Diffie-Hellman key exchange agreement during Phase 1 main mode to enable the creation of secure secret keys used by IPSec encryption algorithms.
- The HMACs of HMAC-MD5 and HMAC-SHA-1 are used by IKE and IPSec to ensure data integrity and to authenticate data exchanges.
- IPSec supports RSA signatures, RSA-encrypted nonces, and preshared keys as methods to authenticate IPSec peers during IKE Phase 1.
- Digital signatures, enabled by public key cryptography, provide a means to digitally authenticate IPSec peers, enabling scalability and flexibility of an IPSec network.

Review Questions

Answer the following review questions, which delve into some of the key facts and concepts covered in this chapter:

1 What are the two main IPSec protocols, and what services does each provide?

2 What important IPSec service does AH not provide?

3 What is the difference between how tunnel mode and transport mode are used?

4 What is an IPSec security association, and how is it established?

5 Can IPSec be configured without IKE?

6 What are the benefits of using IKE?

7 What initiates the IKE process?

8 What is the purpose of IKE phase 2?

9 What is the primary purpose of a CA?

10 What are the five overall steps of the IPSec process?

References

This chapter can be considered a starting point in your journey toward understanding IPSec. The topics considered in this chapter are complex and should be studied further to more fully understand them and put them to use. Use the following references to learn more about the topics in this chapter.

RFCs related to IPSec provide detailed definitions for some of the IPSec components. They can be found at the following URL: www.ietf.org/html.charters/ipsec-charter.html.

Internet drafts are working documents of the Internet Engineering Task Force (IETF). Internet drafts define IPSec standards still in the IETF draft phase. Internet drafts can be found at the following URL: www.ietf.org/html.charters/ipsec-charter.html.

IPSec Standards

RFC 2401, S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," November 1998.

RFC 2402, S. Kent and R. Atkinson, "IP Authentication Header," November 1998.

RFC 2403, C. Madson and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," November 1998.

RFC 2404, C. Madson and R. Glenn, “The Use of HMAC-SHA-1-96 within ESP and AH,” November 1998.

RFC 2405, C. Madson and N. Doraswamy, “The ESP DES-CBC Cipher Algorithm with Explicit IV,” November 1998.

RFC 2406, S. Kent and R. Atkinson, “IP Encapsulating Security Payload (ESP),” November 1998.

RFC 2410, R. Glenn, “The NULL Encryption Algorithm and Its Use with IPSec,” November 1998.

RFC 2411, N. Doraswamy, R. Glenn, and R. Thayer, “IP Security Document Roadmap,” November 1998.

RFC 2451, R. Pereira and R. Adams, “The ESP 3DES CBC-Mode,” November 1998.

Encryption

N. Doraswamy, P. Metzger, and W. A. Simpson, “The ESP Triple DES Transform,” July 1997, draft-ietf-ipsec-ciph-des3-00.txt.

FIPS-46-2, “Data Encryption Standard,” U.S. National Bureau of Standards Federal Information Processing Standard (FIPS) Publication 46-2, December 1993, www.itl.nist.gov/div897/pubs/fip46-2.htm (supercedes FIPS-46-1).

IKE

M. Litvin, R. Shamir, and T. Zegman, “A Hybrid Authentication Mode for IKE,” December 1999, draft-ietf-ipsec-isakmp-hybrid-auth-03.txt.

R. Pereira, S. Anand, and B. Patel, “The ISAKMP Configuration Method,” August 1999, draft-ietf-ipsec-isakmp-mode-cfg-05.txt.

R. Pereira and S. Beaulieu, “Extended Authentication within ISAKMP/Oakley (XAUTH)“, December 1999, draft-ietf-ipsec-isakmp-xauth-06.txt.

RFC 2407, D. Piper, “The Internet IP Security Domain of Interpretation of ISAKMP,” November 1998.

RFC 2408, D. Maughan, M. Schertler, M. Schneider, and J. Turner, “Internet Security Association and Key Management Protocol (ISAKMP),” November 1998.

RFC 2409, D. Harkins and D. Carrel, “The Internet Key Exchange (IKE),” November 1998.

RFC 2412, H. Orman, “The OAKLEY Key Determination Protocol,” November 1998.

Hashing Algorithms

FIPS-180-1, "Secure Hash Standard," National Institute of Standards and Technology, U.S. Department of Commerce, April 1995. Also known as 59 Fed Reg 35317 (1994).

RFC 1321, R. Rivest, "The MD5 Message-Digest Algorithm," April 1992.

RFC 2085, R. Glenn and M. Oehler, "HMAC-MD5 IP Authentication with Replay Prevention," February 1997.

RFC 2104, H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," February 1997.

Public Key Cryptography

RFC 2437, B. Kaliski and J. Staddon, "PKCS #1: RSA Cryptography Specifications," October 1998.

RFC 2631, E. Rescorla, "Diffie-Hellman Key Agreement Method," June 1999.

Digital Certificates and Certificate Authorities

X. Liu, C. Madson, D. McGrew, and A. Nourse, "Cisco Systems' Simple Certificate Enrollment Protocol (SCEP)," February 2000, draft-nourse-scep-02.txt.

RFC 2314, B. Kaliski, "PKCS #10: Certification Request Syntax Version 1.5," March 1998.

RFC 2315, B. Kaliski, "PKCS #7: Cryptographic Message Syntax Version 1.5," March 1998.

RFC 2459, R. Housley, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," January 1999.

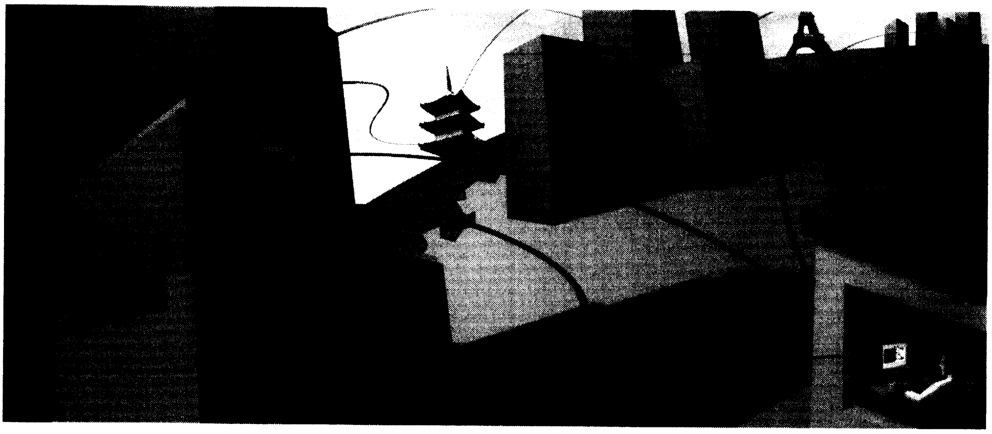
General Security

"Frequently Asked Questions About Today's Cryptography 4.0," RSA Laboratories, Redwood City, Calif., 1998. This FAQ covers the technical mathematics of cryptography as well as export law and fundamentals of information security.

IPSec, a Cisco Systems, Inc. white paper, located at www.cisco.com/warp/customer/cc/cisco/mkt/security/encryp/tech/ipsec_wp.htm. This URL requires a Cisco Connection Online username and password.

M. Kaeo, *Designing Network Security,* Cisco Press, 1999.

B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C,* Second Edition, Wiley, 1995.



from Cisco BGP-4 Command and Configuration

by William R. Parkhurst, Ph.D.

(1-58705-017-X)

Cisco Press

About the Author

William R. Parkhurst, Ph.D., CCIE #2969, is the manager of the CCIE Development group at Cisco Systems. The CCIE Development group is responsible for all new CCIE written qualification and laboratory exams. Prior to joining the CCIE team, Bill was a Consulting Systems Engineer supporting the Sprint Operation. Bill first became associated with Cisco Systems while he was a Professor of Electrical and Computer Engineering at Wichita State University. In conjunction with Cisco Systems, WSU established the first CCIE Preparation Laboratory.

Contents at a Glance

Bold chapters are elements included in this folio.

CHAPTER 9

Route Advertisement

9-1: network *ip-address*

9-2: network *ip-address* mask *network-mask*

Syntax Description:

- *ip-address*—Network to advertise to BGP peers.
- *network-mask*—Optional parameter used to advertise nonclassful network prefixes.

Defaults: None

Limitations: Up to 200 instances of the **network** command may be used in the configuration. For Cisco IOS Software Release 12.0 and later, this restriction has been removed.

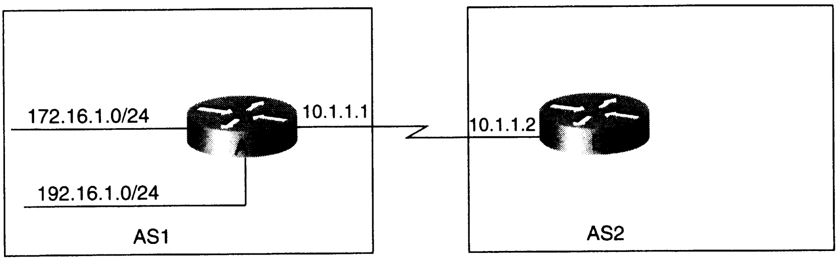
Purpose: Interior Gateway Protocols such as RIP and OSPF use the **network** command to determine on which interfaces the protocol will be active. The BGP **neighbor** command is used to determine which interfaces will run BGP. The BGP **network** command is used to determine the networks that will be advertised to BGP neighbors. In order for a network to be advertised by BGP, it must be known to the originating router. Routes learned via EBGP are automatically advertised to other EBGP neighbors. A known network is one that is directly connected, static, or learned through a dynamic routing protocol. The first form of the **network** command requires a classful IP address. A classful address is either Class A with an 8-bit subnet mask, Class B with a 16-bit subnet mask, or Class C with a 24-bit subnet mask. The second form can be used with either a classful or classless prefix.

Cisco IOS Software Release: 10.0

Configuration Example 1: Directly Connected Networks

Figure 9-1 illustrates a basic scenario for the use of the **network** command. Router A has two directly connected networks that are advertised to router B via BGP.

Figure 9-1 *Basic Use of the* **network** *Command*



```
Router A
interface loopback 0
 ip address 172.16.1.1 255.255.255.0
!
interface loopback 1
 ip address 192.16.1.1 255.255.255.0
!
router bgp 1
 neighbor 10.1.1.2 remote-as 2
 network 172.16.1.0 mask 255.255.255.0
 network 192.16.1.0
```

```
Router B
router bgp 2
 neighbor 10.1.1.1 remote-as 1
```

Notice that the *mask* option was used with network 172.16.1.0. The classful address for this network is 172.16.0.0. Because we want to advertise a subnet of 172.16.0.0, the mask option is required.

Verification

Before using the **network** command, verify that the networks are in the IP routing table using the **show ip route** command:

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

Gateway of last resort is not set
```

(Continued)

```
     172.16.0.0/24 is subnetted, 1 subnets
```

[redacted]

```
     10.0.0.0/30 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Serial0
```

[redacted]

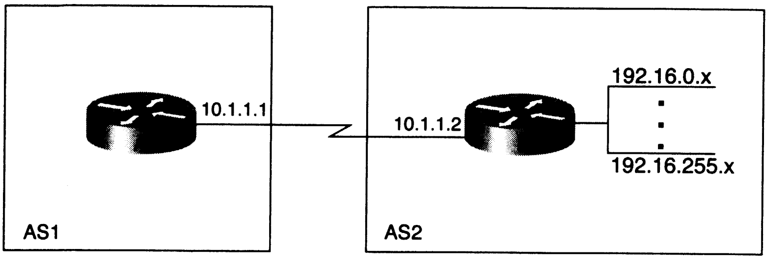
The networks to be advertised are in the IP routing table. The next step is to add the **network** commands to the BGP router configuration and to verify that the networks are in the BGP routing table using the **show ip bgp** command:

```
rtrA#show ip bgp
BGP table version is 538, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.16.1.0       0.0.0.0                  0         32768 i
*> 172.16.1.0       0.0.0.0                  0         32768 i
```

Configuration Example 2: Aggregation Using Static Routes

A static route can be used to allow BGP to advertise any network prefix. Configuring a static route installs the static network in the local IP routing table. Any route in the IP routing table can be advertised by BGP using the **network** command. Of course, the router should advertise only networks that it can actually reach. The main use of static routes with BGP is to allow the advertisement of an aggregate address. Figure 9-2 shows an ISP that owns the range of Class C addresses from 192.16.0.x through 192.16.255.x. The **network** command could be used 256 times—once for each Class C prefix—or we could use a static route to create an aggregate prefix.

Figure 9-2 *Using a Static Route and the* **network** *Command to Advertise an Aggregate Prefix*



```
Router A
router bgp 1
 neighbor 10.1.1.2 remote-as 2
```

```
Router B
router bgp 2
 neighbor 10.1.1.1 remote-as 1
 network 192.16.0.0 mask 255.255.0.0
!
ip route 192.16.0.0 255.255.0.0 null 0
```

The optional *mask* parameter is needed because 192.16.0.0/16 is a supernet of the Class C address 192.16.0.0/24. The static route has the next hop as the interface null 0. Router B has more specific routes to networks contained in the range 192.16.0.x to 192.16.255.x. Assume that network 192.16.8.0 is down. Router A thinks that 192.16.8.0 is reachable because it is receiving the advertisement to 192.16.x.x from Router B. When Router B receives a packet destined for network 192.16.8.0, the route is looked up in the IP routing table. A specific match is not found, because the network is down. Router B tries to find a shorter match. It finds 192.16.x.x and instructs Router B to send the packet to null 0 or simply discard the packet. Prefixes can also be aggregated using the **aggregate-address** command, covered in Chapter 1, "Route Aggregation."

Verification

BGP won't advertise the aggregate route unless it is in the IP routing table. As before, verify that the route is in the IP routing table and the BGP table:

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

Gateway of last resort is not set

10.0.0.0/30 is subnetted, 2 subnets
C       10.1.1.0 is directly connected, Serial0
S    192.16.0.0/16 is directly connected, Null0
```

```
rtrB#show ip bgp
BGP table version is 44, local router ID is 192.16.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.16.0.0/16    0.0.0.0                  0         32768 i
```

Troubleshooting

Step 1 Verify that the BGP neighbors are in the Established state using the **show ip bgp neighbors** command.

If the neighbor relationship is not in the Established state, see section 8-23.

Step 2 Verify that the network you are attempting to advertise is in the IP routing table. There must be an exact match between prefix and mask.

9-3: network *ip-address* backdoor

9-4: network *ip-address* mask *network-mask* backdoor

Syntax Description:

- *ip-address*—Network to advertise to BGP peers.
- *network-mask*—Optional parameter used to advertise nonclassful network prefixes.

Defaults: None

Limitations: Up to 200 instances of the **network** command may be used in the configuration. For Cisco IOS Software Release 12.0 and later, this restriction has been removed.

Purpose: When a router is running more than one IP routing protocol, the possibility exists that a particular route might be learned by two or more protocols. Because different IP routing protocols calculate the cost to a route using different metrics, the protocol cost cannot be used to select the best path. When a route is known by more than one IP routing protocol, Cisco routers use the administrative distance to select the best path, with the lowest administrative distance being preferred. EBGP routes have an administrative distance of 20, and IGPs have a higher administrative distance:

- EBGP—20
- EIGRP—90
- IGRP—100
- OSPF—110
- RIP—120
- IBGP—200

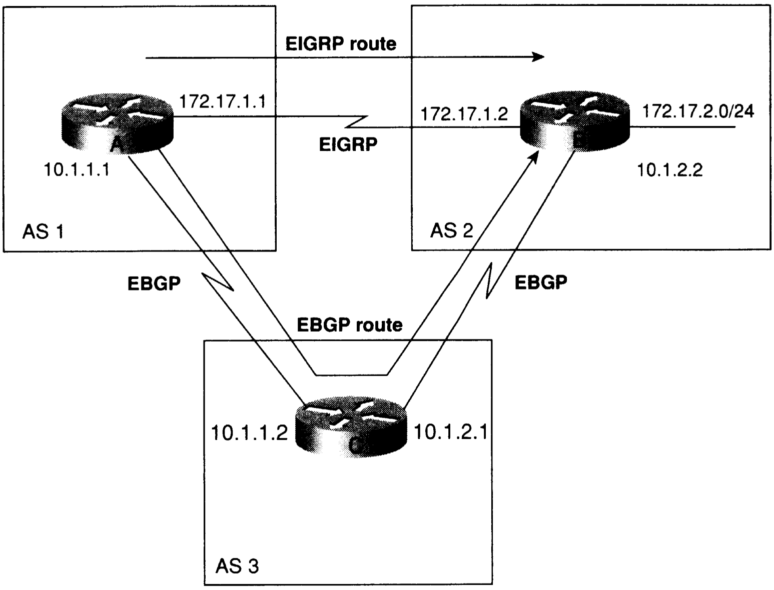
EBGP routes are preferred over IGP routes. The **backdoor** option instructs BGP to set the administrative distance for the network specified to 200, allowing the IGP route to be preferred.

Cisco IOS Software Release: 10.0

Configuration Example: Finding the Best Route Through Administrative Distance

In Figure 9-3, Router A is learning about network 172.17.2.0 via EBGP and EIGRP.

Figure 9-3 *EBGP Route to 172.17.2.0 Is Preferred Over the EIGRP Route*



```
Router A
router eigrp 1
 network 172.17.0.0
!
router bgp 1
 neighbor 10.1.1.2 remote-as 3
```

```
Router B
router eigrp 1
network 172.17.0.0
!
router bgp 2
 network 172.17.2.0 mask 255.255.255.0
 neighbor 10.1.2.1 remote-as 3
```

```
Router C
router bgp 3
 network 10.1.1.0 mask 255.255.255.252
 network 10.1.2.0 mask 255.255.255.252
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.2.2 remote-as 2
```

Because EBGP has a lower administrative distance than EIGRP, the EBGP route is installed in Router A's IP routing table.

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

Gateway of last resort is not set

     172.17.0.0/24 is subnetted, 2 subnets
C       172.17.1.0 is directly connected, Ethernet0
[redacted]
     10.0.0.0/30 is subnetted, 2 subnets
B       10.1.2.0 [20/0] via 10.1.1.2
C       10.1.1.0 is directly connected, Serial0
```

The preferred path from Router A to network 172.17.2.0 is through Router C. The shortest path to network 172.17.2.0 is through the direct connection to Router B. A number of methods can be used to modify routing table entries so that Router A prefers the direct path to network 172.17.2.0. Using the **backdoor** option is relatively easy, as shown in the following modified listing for Router A:

```
Router A
router bgp 1
[redacted]
 neighbor 10.1.1.2 remote-as 3
```

The **backdoor** option causes the network learned via EBGP to have an administrative distance of 200. The EIGRP route for network 172.17.2.0 has an administrative distance of 90, causing it to be installed in the IP routing table.

Verification

By inspecting the IP routing table on Router A, we can see that the route to 172.17.2.0 learned via EIGRP has been installed in the IP routing table, replacing the EBGP route:

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

continues

(Continued)

```
Gateway of last resort is not set

     172.17.0.0/24 is subnetted, 2 subnets
C       172.17.1.0 is directly connected, Ethernet0
D       172.17.2.0 [90/409600] via 172.17.1.2, Ethernet0
     10.0.0.0/30 is subnetted, 2 subnets
B       10.1.2.0 [20/0] via 10.1.1.2
C       10.1.1.0 is directly connected, Serial0
```

Troubleshooting

Step 1 Verify that the BGP neighbors are in the Established state using the **show ip bgp neighbors** command.

If the neighbor relationship is not in the Established state, see section 8-23.

Step 2 Before using the **backdoor** option, use the **show ip bgp** command to ensure that the route you intend to modify is in the BGP table. For example, on Router A, verify a BGP entry for network 172.17.2.0:

```
rtrA#show ip bgp
BGP table version is 43, local router ID is 192.16.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/30      10.1.1.2                 0             0 3 i
*> 10.1.2.0/30      10.1.1.2                 0             0 3 i
*> 172.17.2.0/24    10.1.1.2                               0 3 2 I
```

Step 3 If the network is in the BGP table, the **backdoor** option will work as described.

9-5: network *ip-address* route-map *route-map-name*

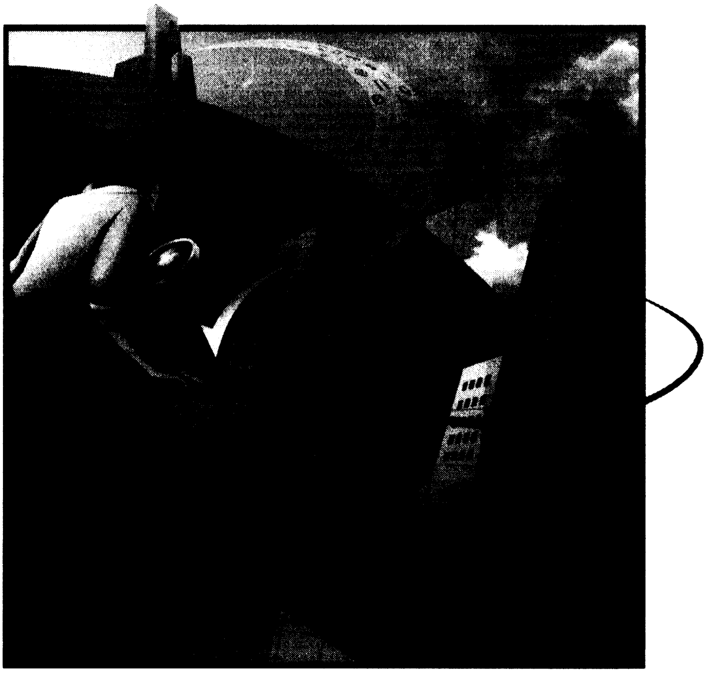
9-6: network *ip-address* mask *network-mask* route-map *route-map-name*

Purpose: In theory, these commands allow you to modify a network's BGP attributes. In our experience, they are too buggy and should not be used. See sections 8-25 and 8-26 if you need to modify the BGP attributes of received or transmitted network advertisements.

9-7: network *ip-address* weight *weight*

9-8: network *ip-address* mask *network-mask* weight *weight*

Obsolete: These commands don't work and are considered obsolete. They will be removed from future versions of Cisco IOS Software. They are included here because they exist in current Cisco IOS Software Releases, but they do nothing. See sections 8-25 or 8-35 if you need to modify the weight of received network advertisements.



from IP Quality of Service

by Srinivas Vegesna

(1-57870-116-3)

Cisco Press

About the Author

Srinivas Vegesna, CCIE #1399, is a manager in the Service Provider Advanced Consulting Services program at Cisco Systems. His focus is general IP networking, with a special focus on IP routing protocols and IP Quality of Service. In his six years at Cisco, Srinivas has worked with a number of large service provider and enterprise customers in designing, implementing, and troubleshooting large-scale IP networks. Srinivas holds an M.S. degree in Electrical Engineering from Arizona State University. He is currently working towards an M.B.A. degree at Santa Clara University.

Contents at a Glance

Bold chapters are elements included in this folio.

CHAPTER 5

Per-Hop Behavior: Resource Allocation II

The right packet scheduling mechanism for a router depends on its switching architecture. Weighted Fair Queuing (WFQ) is a scheduling algorithm for resource allocation on Cisco router platforms with a bus-based architecture. Cisco platforms using a switch fabric for packet switching tend to use a scheduling algorithm that better suits that architecture. In particular, the Cisco Catalyst family of switches and 8540 routers use the Modified Weighted Round Robin (MWRR) algorithm, and the Cisco 12000 series routers use the Modified Deficit Round Robin (MDRR) algorithm. Both MWRR and MDRR are similar in scheduling behavior to WFQ because they, too, simulate Generalized Processor Sharing (GPS).

The next two sections provide a detailed discussion of the MWRR and MDRR algorithms.

Modified Weighted Round Robin (MWRR)

Round-robin scheduling that serves a packet rather than an infinitesimal amount from each nonempty queue is the simplest way to simulate GPS. It works well in representing a GPS scheduler if all packets are the same size. Weighted Round Robin (WRR) is an extension of round-robin scheduling in which each flow is assigned a weight[1]. WRR serves a flow in proportion to its weight.

WRR scheduling is well suited when an Asynchronous Transfer Mode (ATM) switch fabric is used for switching. Internally, the switch fabric treats packets as cells, and WRR is used to schedule the cells in the queues. WRR is essentially a cell-based round robin, whereby the weight determines how many cells are scheduled in each round robin. Hence, each queue shares the interface bandwidth of the ratio of the weights independent of packet sizes.

You can schedule only packets, not cells. Therefore, all cells of a packet are served in the same pass, even when you need to borrow some weight from the future. To support variable-size packets, MWRR uses a deficit counter associated with each WRR queue. This gives MWRR some characteristics of the Deficit Round Robin (DRR) algorithm described in the next section.

Before a queue is serviced, its deficit counter is initialized to the queue's weight. A packet from a queue is scheduled only if the deficit counter is greater than zero. After serving the *n*-cell packet, the resulting counter is decremented by *n*. Packets are scheduled as long as the counter is greater than 0. Otherwise, you skip to the next queue. In each coming round, the queue's deficit counter is incremented by the queue's weight. No packet is scheduled, however, if the deficit counter is still not greater than 0. If the counter becomes greater than 0, a packet is scheduled. After serving the packet, the deficit counter is decremented by the number of cells in the packet. By using a deficit counter, MWRR works independent of the variable-length packet sizes in the long run.

The effective bandwidth for each queue is proportional to its weight:

Effective queue bandwidth = (Queue weight × Interface bandwidth) ÷ Sum of all active queue weights)

An Illustration of MWRR Operation

In this example, consider three queues with the assigned weights shown in Table 5-1. Figure 5-1 depicts the queues along with their deficit counters. Deficit counters are used to make WRR support variable packet sizes.

Table 5-1 *Weights Associated with Each Queue*

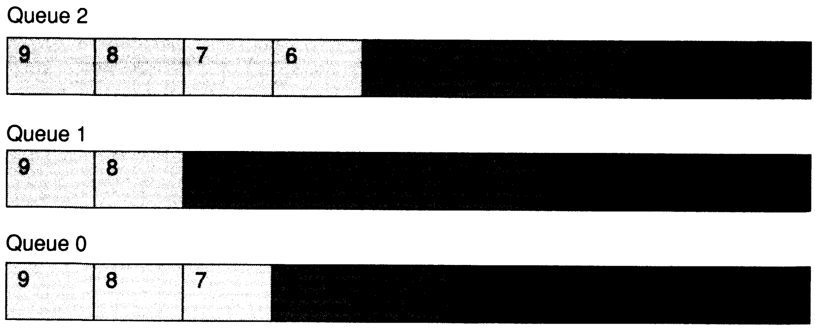
Queue	Weight
2	4
1	3
0	2

The queues show the cells queued, and the cells making up a packet are marked in the same shade of black. Queue 2, for example, has a 2-cell, 3-cell, and 4-cell packet in its queue.

Queue 0 is the first queue being served. The deficit counter is initialized to 2, the queue's weight. At the head of the queue is a 4-cell packet. Therefore, the deficit counter becomes $2 - 4 = -2$ after serving the packet. Because the deficit counter is negative, the queue cannot be served until it accumulates to a value greater than zero, as in Figure 5-2.

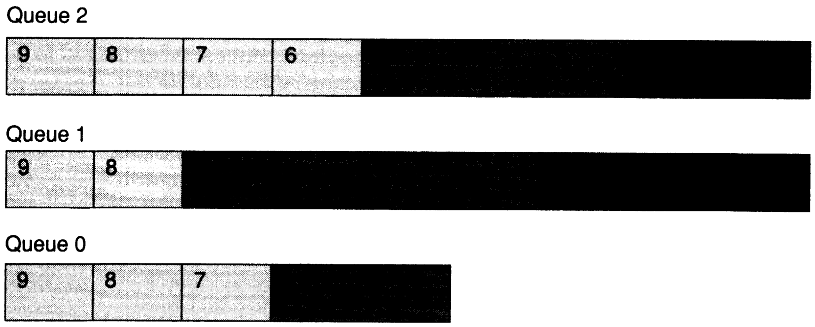
Queue 1 is the next queue to be served. Its deficit counter is initialized to 3. The 3-cell packet at the head of the queue is served, which makes the deficit counter become $3 - 3 = 0$. Because the counter is not greater than zero, you skip to the next queue, as in Figure 5-3.

Figure 5-1 *WRR Queues with Their Deficit Counters Before Start of Service*



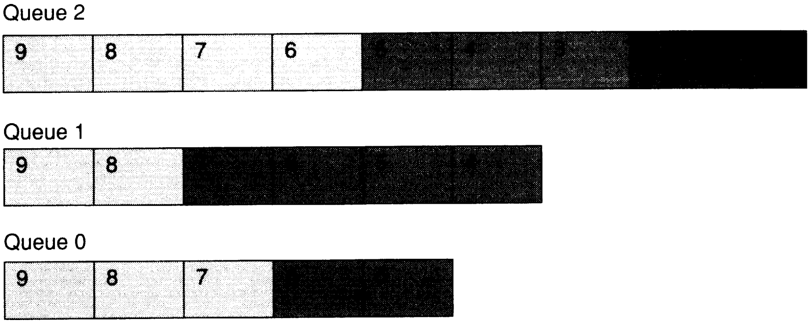
Queue	Deficit counter
0	0
1	0
2	0

Figure 5-2 *MWRR After Serving Queue 0 in the First Round*



Queue	Deficit counter
2	0
1	0
0	-2

Figure 5-3 *MWRR After Serving Queue 1 in the First Round*



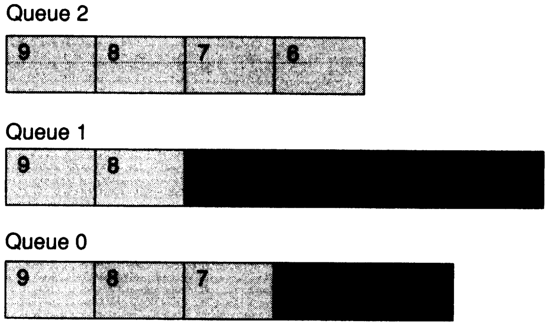
Queue	Deficit counter
2	0
1	0
0	-2

Now it is Queue 2's turn to be serviced. Its deficit counter is initialized to 4. The 2-cell packet at the head of the queue is served, which makes the deficit counter 4 – 2 = 2. The next 3-cell packet is also served, as the deficit counter is greater than zero. After the 3-cell packet is served, the deficit counter is 2 – 3 = –1, as in Figure 5-4.

Queue 0 is now served in the second round. The deficit counter from the last round was –2. Incrementing the deficit counter by the queue's weight makes the counter –2 + 2 = 0. No packet can be served because the deficit counter is still not greater than zero, so you skip to the next queue, as in Figure 5-5.

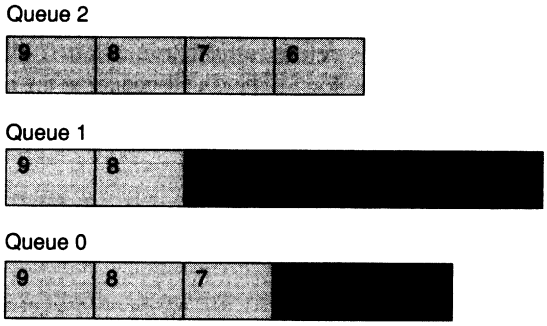
Queue 1 has a deficit counter of zero in the first round. For the second round, the deficit counter is 0 + 3 = 3. The 4-cell packet at the head of the queue is served, making the deficit counter 3 – 4 = –1, as in Figure 5-6.

Figure 5-4 *MWRR After Serving Queue 2 in the First Round*



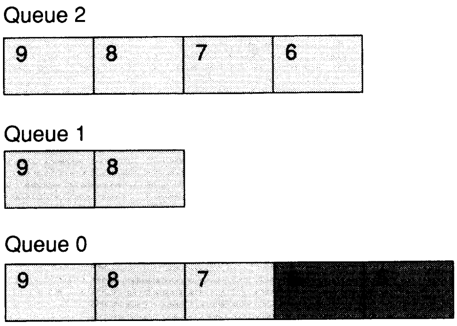
Queue	Deficit counter
2	-1
1	0
0	-2

Figure 5-5 *MWRR After Serving Queue 0 in the Second Round*



Queue	Deficit counter
2	-1
1	0
0	0

Figure 5-6 *MWRR After Serving Queue 1 in the Second Round*



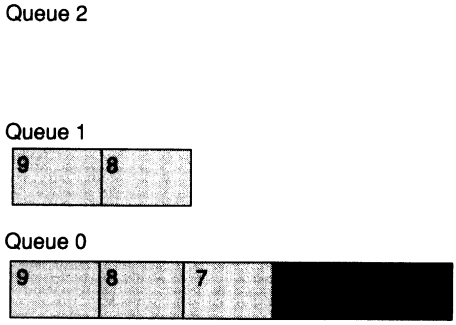
Queue	Deficit counter
2	-1
1	-1
0	0

In the second round, Queue 2's deficit counter from the first round is incremented by the queue's weight, making it –1 + 4 = 3. The 4-cell packet at the head of Queue 2 is served, making the deficit counter 3 – 4 = –1. Because Queue 2 is now empty, the deficit counter is initialized to zero, as in Figure 5-7.

Now, it is again Queue 0's turn to be served. Its deficit counter becomes 0 + 2 = 2. The 2-cell packet at the head of the queue is served, which results in a deficit counter of 2 – 2 = 0. Now skip to Queue 1, as in Figure 5-8.

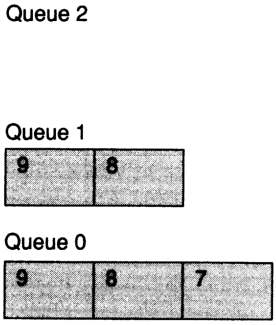
Queue 1's new deficit counter is –1 + 3 = 2. The 2-cell packet at the head of Queue 1 is served, resulting in a deficit counter of 2 – 2 = 0. The resulting Queue 1 is now empty. Because Queue 2 is already empty, skip to Queue 0, as in Figure 5-9.

Figure 5-7 *MWRR After Serving Queue 2 in the Second Round*



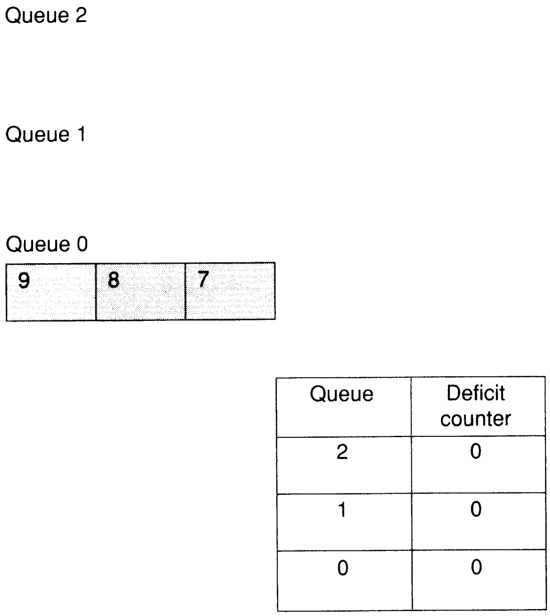
Queue	Deficit counter
2	0
1	-1
0	0

Figure 5-8 *MWRR After Serving Queue 0 in the Third Round*



Queue	Deficit counter
2	0
1	-1
0	0

Figure 5-9 *MWRR After Serving Queue 1 in the Third Round*



Queue	Deficit counter
2	0
1	0
0	0

Queue 0's deficit counter in the fourth round becomes 2. The 3-cell packet is served, which makes the deficit counter equal to –1. Because Queue 0 is now empty, reset the deficit counter to zero.

MWRR Implementation

MWRR is implemented in the Cisco Catalyst family of switches and the Cisco 8540 routers. These switches and routers differ in terms of the number of available MWRR queues and in the ways you can classify traffic into the queues.

MWRR in 8540 routers offers four queues between any interface pair based on Type of Service (ToS) group bits. Table 5-2 shows the ToS class allocation based on the IP precedence bits.

Table 5-2 *MWRR ToS Class Allocation*

IP Precedence Bits	ToS Class Bits	ToS Class Assigned
000	00	0
001	00	1
010	01	2
011	01	3

Table 5-2 *MWRR ToS Class Allocation (Continued)*

IP Precedence Bits	ToS Class Bits	ToS Class Assigned
100	10	0
101	10	1
110	11	2
111	11	3

NOTE 8500 ToS-based MWRR is similar to ToS-based Distributed WFQ (DWFQ), discussed in Chapter 4, "Per-Hop Behavior: Resource Allocation I," but differs in terms of which precedence bits are used to implement it. ToS-based DWFQ uses the two low-order precedence bits, whereas 8500 ToS-based MWRR uses the two high-order precedence bits. In both cases, the leftover bit can signify the *drop priority*. Drop priority indicates which IP precedence packets can be dropped at a higher probability between the IP precedence values making up a ToS class.

Catalyst 6000 and 6500 series switches use MWRR with two queues, Queue 1 and Queue 2, based on the Layer 2 Institute of Electrical and Electronic Engineers (IEEE) 802.1p Class of Service (CoS) field. Frames with CoS values of 0–3 go to Queue 1, and frames with CoS values of 4–7 go to Queue 2. 802.1p CoS is discussed in Chapter 8, "Layer 2 QoS: Interworking with IP QoS."

6500 series switches also implement strict priority queues as part of MWRR to support the low-latency requirements of voice and other real-time traffic.

Case Study 5-1: Class-Based MWRR Scheduling

A large Internet service provider (ISP) decides to categorize its traffic into four classes in a network backbone made up of a Cisco 8540 router based on IP precedence, as shown in Table 5-3. Each class is assigned traffic belonging to two IP precedence values—one precedence value each for normal and excess traffic. Classes 0–3 need to be allocated 15 percent, 15 percent, 30 percent, and 40 percent of the link bandwidth, respectively.

Table 5-3 *IP Precedence Allocation Based on Traffic Class*

	Critical (Class 3)	Expensive (Class 2)	Moderate (Class 1)	Cheap (Class 0)
Normal Traffic	Precedence 7	Precedence 5	Precedence 3	Precedence 1
Excess Traffic	Precedence 6	Precedence 4	Precedence 2	Precedence 0

You can enable Quality of Service (QoS)-based forwarding in an 8540 router by using the global command **qos switching**. The default weight allocation for ToS classes 0–3 is 1, 2, 4, and 8, respectively. Hence, ToS classes 0–3 get an effective bandwidth of 1/15, 2/15, 4/15, and 8/15 of the interface bandwidth.

In this case, the bandwidth allocation for classes 0–3 is 15:15:30:40 or 3:3:6:8 because the WRR scheduling weight can only be between 1–15.

Listing 5-1 shows a sample configuration to enable ToS-based MWRR globally on an 8500 router.

Listing 5-1 *Enabling ToS-Based MWRR*

```
qos switching
qos mapping precedence 0 wrr-weight 3
qos mapping precedence 1 wrr-weight 3
qos mapping precedence 2 wrr-weight 6
qos mapping precedence 3 wrr-weight 8
```

The configuration to QoS-switch according to the above criteria for traffic coming into port 1 and going out of port 0 only is given in Listing 5-2.

Listing 5-2 *Enabling ToS-Based MWRR on Specific Traffic*

```
qos switching
qos mapping <incoming interface> <outgoing interface>
    precedence 0 wrr-weight 3
qos mapping <incoming interface> <outgoing interface>
    precedence 1 wrr-weight 3
qos mapping <incoming interface> <outgoing interface>
    precedence 2 wrr-weight 6
qos mapping <incoming interface> <outgoing interface>
    precedence 3 wrr-weight 8
```

Modified Deficit Round Robin (MDRR)

This section discusses the MDRR algorithm for resource allocation available in the Cisco 12000 series routers. Within a DRR scheduler[2], each service queue has an associated *quantum value*—an average number of bytes served in each round—and a deficit counter initialized to the quantum value. Each nonempty flow queue is served in a round-robin fashion, scheduling on average packets of quantum bytes in each round. Packets in a service queue are served as long as the deficit counter is greater than zero. Each packet served decreases the deficit counter by a value equal to its length in bytes. A queue can no longer be served after the deficit counter becomes zero or negative. In each new round, each nonempty queue's deficit counter is incremented by its quantum value.

After a queue is served, the queue's deficit counter represents the amount of debit it incurred during the past round, depending on whether it was served equal to or more than its allocated quantum bytes. The amount the queue is entitled to be served in a subsequent round is reduced from the quantum bytes by a value equal to the deficit counter.

For efficiency, you should make the quantum size equal to the maximum packet size in the network. This ensures that the DRR scheduler always serves at least one packet from each nonempty flow queue.

NOTE An empty flow queue's deficit counter is reset to zero so that credits are not accumulated indefinitely, as it would eventually lead to unfairness.

The general DRR algorithm described in this section is modified to allow a *low-latency* queue. In MDRR, all queues are serviced in a round-robin fashion with the exception of the low-latency queue. You can define this queue to run in either one of two ways: in strict priority or alternate priority mode.

In *strict priority mode*, the low-latency queue is serviced whenever the queue is nonempty. This allows the lowest possible delay for this traffic. It is conceivable, however, for the other queues to starve if the high-priority, low-latency queue is full for long periods of time because it can potentially take 100 percent of the interface bandwidth.

In *alternate priority mode*, the low-latency queue is serviced alternating between the low-latency queue and the remaining CoS queues. In addition to a low-latency queue, MDRR supports up to seven other queues, making the total number of queues to eight. Assuming that 0 is the low-latency queue, the queues are served in the following order: 0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0, 6, 0, 7.

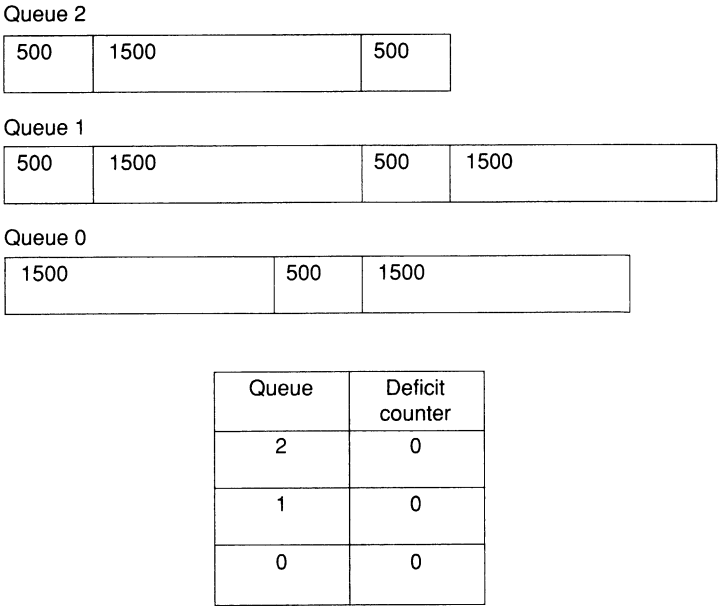
In alternate priority mode the largest delay for Queue 0 is equal to the largest single quantum for the other queues rather than the sum of all the quanta for the queues if Queue 0 were served in traditional round-robin fashion.

In addition to being DRR-draining, MDRR is not conventional round-robin scheduling. Instead, DRR is modified in such a way that it limits the latency on one user-configurable queue, thus providing better jitter characteristics.

An MDRR Example

This example, which illustrates an alternate-priority low-latency queue, defines three queues—Queue 2, Queue 1, and Queue 0, with weights of 1, 2, and 1, respectively. Queue 2 is the low-latency queue running in alternate-priority mode. All the queues, along with their deficit counters, are shown in Figure 5-10.

Figure 5-10 *Queues 0–2, Along with Their Deficit Counters*



Queue	Deficit counter
2	0
1	0
0	0

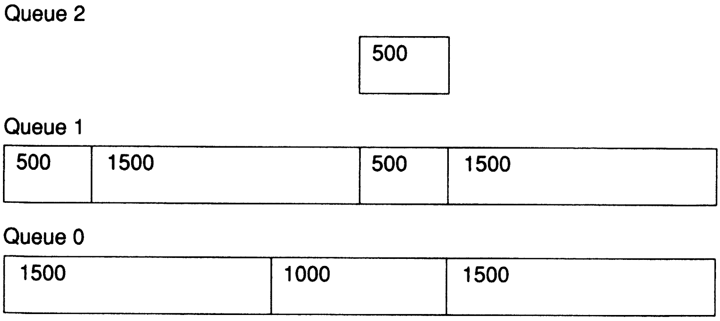
Table 5-4 provides the weight and quantum associated with each queue. When MDRR is run on the output interface queue, the interface maximum transmission unit (MTU) is used. When MDRR is run, the fabric queues.

Table 5-4 *Queues 0–2, Along with Their Associated Weights and Quantum Values*

Queue Number	Weight	Quantum = Weight × MTU (MTU = 1500 Bytes)
Queue 2	1	1500
Queue 1	2	3000
Queue 0	1	1500

On the first pass, Queue 2 is served. Queue 2's deficit counter is initialized to equal its quantum value, 1500. Queue 2 is served as long as the deficit counter is greater than 0. After serving a packet, Queue 2's size is subtracted from the deficit counter. The first 500-byte packet from the queue gets served because the deficit counter is 1500. Now, the deficit counter is updated as 1500 – 500 = 1000. Therefore, the next packet is served. After the 1500-byte packet is served, the deficit counter becomes –500 and Queue 2 can no longer be served. Figure 5-11 shows the three queues and the deficit counters after Queue 2 is served.

Figure 5-11 *MDRR After Serving Queue 2, Its First Pass*

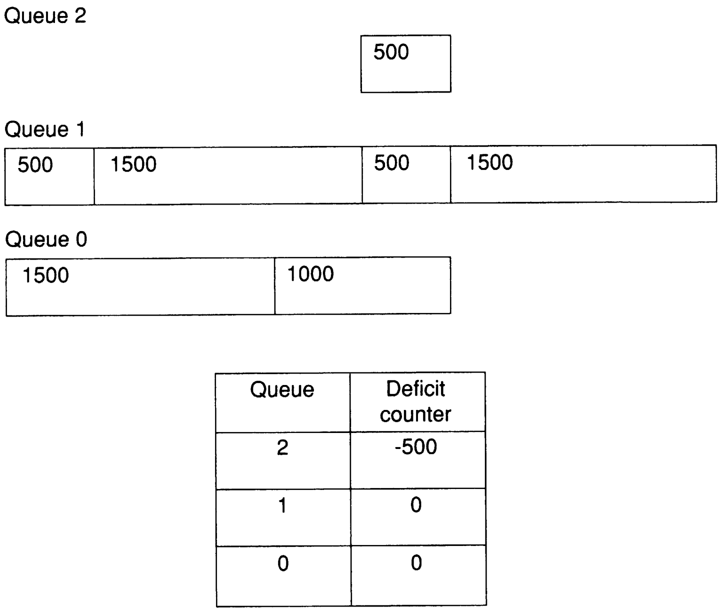


Queue	Deficit counter
2	-500
1	0
0	0

Because you are in alternate-priority mode, you alternate between serving Queue 2 and another queue. This other queue is selected in a round-robin fashion. Consider that in the round robin, it is now Queue 0's turn. The deficit counter is initialized to 1500, the quantum value for the queue. The first 1500-byte packet is served. After serving the first packet, its deficit counter is updated as 1500 – 1500 = 0. Hence, no other packet can be served in this pass. Figure 5-12 shows the three queues and their deficit counters after Queue 0 is served.

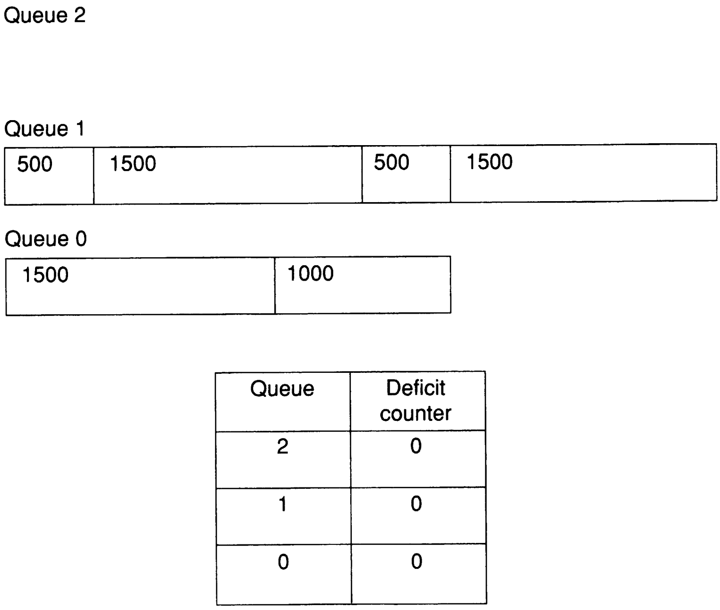
Because you alternate between the low-latency queue and the other queues served in the round robin, Queue 2 is served next. Queue 2's deficit counter is updated to –500 + 1500 = 1000. This allows the next packet in Queue 2 to be served. After sending the 500-byte packet, the deficit counter becomes 500. It could have served another packet, but Queue 2 is empty. Therefore, its deficit counter is reset to 0. An empty queue is not attended, and the deficit counter remains 0 until a packet arrives on the queue. Figure 5-13 shows the queues and the counters at this point.

Figure 5-12 *MDRR After Serving Queue 0, Its First Pass*



Queue	Deficit counter
2	-500
1	0
0	0

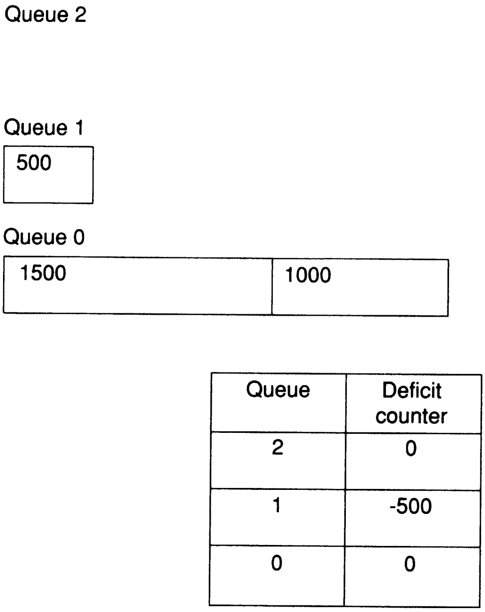
Figure 5-13 *MDRR After Serving Queue 2, Its Second Pass*



Queue	Deficit counter
2	0
1	0
0	0

Queue 1 is served next. It deficit counter is initialized to 3000. This allows three packets to be sent, leaving the deficit counter to be 3000 – 1500 – 500 – 1500 = –500. Figure 5-14 shows the queues and the deficit counters at this stage.

Figure 5-14 *MDRR After Serving Queue 1, Its First Pass*



Queue	Deficit counter
2	0
1	-500
0	0

Queue 0 is the next queue serviced and sends two packets, making the deficit counter 1500 – 1000 – 1500 = –500. Because the queue is now empty, the deficit counter is reset to 0. Figure 5-15 depicts the queues and counters at this stage.

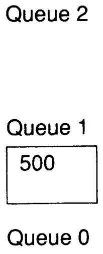
Queue 1 serves the remaining packet in a similar fashion in its next pass. Because the queue becomes empty, its deficit counter is reset to 0.

MDRR Implementation

Cisco 12000 series routers support MDRR. MDRR can run on the output interface queue (transmit [TX] side) or on the input interface queue (receive [RX] side) when feeding the fabric queues to the output interface.

Different hardware revisions of line cards termed as engine 0, 1, 2, 3, and so on, exist for Cisco 12000 series routers. The nature of MDRR support on a line card depends on the line card's hardware revision. Engine 0 supports MDRR software implementation. Line card hardware revisions, Engine 2 and above, support MDRR hardware implementation.

Figure 5-15 *MDRR After Serving Queue 0, Its Second Pass*



Queue	Deficit counter
2	0
1	-500
0	0

MDRR on the RX

MDRR is implemented in either software or hardware on a line card. In a software implementation, each line card can send traffic to 16 destination slots because the 12000 series routers use a 16x16 switching fabric. For each destination slot, the switching fabric has eight CoS queues, making the total number of CoS queues 128 (16 x 8). You can configure each CoS queue independently.

In the hardware implementation, each line card has eight CoS queues per destination interface. With 16 destination slots and 16 interfaces per slot, the maximum number of CoS queues is 16 × 16 × 8 = 2048. All the interfaces on a destination slot have the same CoS parameters.

MDRR on the TX

Each interface has eight CoS queues, which you can configure independently in both hardware- and software-based MDRR implementations.

Flexible mapping between IP precedence and the eight possible queues is offered in the MDRR implementation. MDRR allows a maximum of eight queues so that each IP precedence value can be made its own queue. The mapping is flexible, however. The number of queues needed and the precedence values mapped to those queues are user-configurable. You can map one or more precedence values into a queue.

MDRR also offers individualized drop policy and bandwidth allocation. Each queue has its own associated Random Early Detection (RED) parameters that determine its drop thresholds and DRR quantum, the latter which determines how much bandwidth it gets. The quantum (in other words, the average number of bytes taken from the queue for each service) is user-configurable.

Case Study 5-2: Bandwidth Allocation and Minimum Jitter Configuration for Voice Traffic with Congestion Avoidance Policy

Traffic is classified into different classes so that a certain minimum bandwidth can be allocated for each class depending on the need and importance of the traffic. An ISP implements five traffic classes—gold, silver, bronze, best-effort, and a voice class carrying voice traffic and requiring minimum jitter.

You need four queues, 0–3, to carry the four traffic classes (best-effort, bronze, silver, gold), and a fifth low-latency queue to carry the voice traffic.

This example shows three OC3 Point-to-Point Protocol (PPP) over Synchronous Optical Network (SONET) (PoS) interfaces, one each in slots 1–3. Listing 5-3 gives a sample configuration for this purpose.

Listing 5-3 *Defining Traffic Classes and Allocating Them to Appropriate Queues with a Minimum Bandwidth During Congestion*

```
interface POS1/0
 tx-cos cos-a

interface POS2/0
 tx-cos cos-a

interface POS3/0
 tx-cos cos-a

slot-table-cos table-a
 destination-slot 0 cos-a
 destination-slot 1 cos-a
 destination-slot 2 cos-a

rx-cos-slot 1 table-a
rx-cos-slot 2 table-a
rx-cos-slot 3 table-a

cos-queue-group cos-a
 precedence all random-detect-label 0
 precedence 0 queue 0
 precedence 1 queue 1
```

continues

Listing 5-3 *Defining Traffic Classes and Allocating Them to Appropriate Queues with a Minimum Bandwidth During Congestion (Continued)*

```
precedence 2 queue 2
precedence 3 queue 3
precedence 4 queue low-latency
precedence 5 queue 0
random-detect-label 0 50 200 2
exponential-weighting-constant 8
queue 0 10
queue 1 20
queue 2 30
queue 3 40
queue low-latency strict-priority 20
```

All interfaces for PoS1/0, PoS2/0, and PoS3/0 are configured with TX CoS based on the **cos-queue group *cos-a*** command. The ***cos-a*** command defines a CoS policy. Traffic is mapped into classes based on their IP precedence value in the packet. Each of the five classes is allocated to its individual queue, and weights are allocated based on the bandwidth allocation for each class. The bandwidth allocation for a class is proportional to its weight. The percentage of interface bandwidth allocation for Queues 0–3 and the low-latency queue is 8.33, 16.67, 25, 33.33, and 16.67, respectively. Voice is delay-sensitive but not bandwidth-intensive. Hence, it is allocated a low-latency queue with strict priority, but it doesn't need a high-bandwidth allocation.

The network supports only IP precedence values of 0–4. IP precedence 5 is allocated to Queue 0 for best-effort service. IP precedence 6 and 7 packets are control packets that a router originates. They are flagged internally and are transmitted first, regardless of the MDRR configuration.

The ***cos-a*** **cos-queue-group** command defines a Weighted Random Early Detection (WRED), a congestion avoidance policy that applies to all queues as follows:

- Minimum threshold: 50 packets
- Maximum threshold: 200 packets
- Probability of dropping packets at maximum threshold: ∫ = 50 percent
- Exponential weighting constant to calculate average queue depth: 8

WRED is discussed in Chapter 6, "Per-Hop Behavior: Congestion Avoidance and Packet Drop Policy."

The MDRR algorithm can also be applied on the input interface line card on the fabric queues delivering the packet to the destination line card. The **slot-table-cos** command defines the CoS policy for each destination line card's CoS fabric queues on the receive line card. In the example, the ***table-a*** **slot-table-cos** command defines the CoS policy for destination line cards 0–2 based on the ***cos-a*** **cos-queue-group** command. Note that the destination line card can be the same as the receive line card because the input and output interfaces for certain traffic can exist on the same line card.

The **rx-cos-slot** command applies the ***table-a*** **slot-table-cos** command to a particular slot (line card). Listing 5-4 shows the CoS configuration on the router.

Listing 5-4 *CoS Information*

```
Router#show cos
Interface      Queue Cos Group
PO1/0             cos-a
PO2/0             cos-a
PO3/0             cos-a

Rx Slot         Slot Table
1              table-a
2              table-a
3              table-a

Slot Table Name - table-0
1           cos-a
2           cos-a
3           cos-a

Cos Queue Group - cos-a
precedence all mapped label 0
red label 0
min thresh 100, max thresh 300 max prob 2
...
exponential-weighting-constant 8
queue 0  weight 10
queue 1  weight 20
queue 2  weight 40
queue 3  weight 80
queue 4  weight 10
queue 5  weight 10
queue 6  weight 10
low latency queue weight 20, priority strict
```

Note that Queues 4–6 are not mapped to any IP precedence, so they are empty queues. Only Queues 0–3 and the low-latency queue are mapped to an IP precedence, and bandwidth is allocated proportional to the queue weights during congestion.

From the line card, the **show controllers frfab/tofab cos-queue length/parameters/ variables** command shows information regarding the CoS receive and transmit queues.

Summary

In this chapter, we discuss two new scheduling algorithms, MWRR and MDRR, that are used for resource allocation. MWRR and MDRR are similar to WFQ algorithm in their scheduling behavior. MWRR and MDRR scheduling can also support voice traffic if the voice queue is made a strict priority queue. At this time, MWRR and MDRR are used in the Catalyst family of switches and Cisco 12000 series routers, respectively.

Frequently Asked Questions

Q — How does a scheduling algorithm determine resource allocation?

A — A scheduling algorithm determines which packet goes next in a queue. How often a flow's packets are served determines the bandwidth or resource allocation for the flow.

Q — Can MWRR and MDRR support voice traffic?

A — Yes. MWRR and MDRR can support voice traffic when one of their queues can be made a *strict priority queue*. A strict priority queue carries voice and other real-time traffic with low latency.

Q — How do MWRR and MDRR relate to WFQ?

A — WFQ, MWRR, and MDRR are all scheduling algorithms for resource allocation. Both MWRR and MDRR are similar in scheduling behavior to WFQ because all three algorithms simulate the GPS model. GPS is discussed in Chapter 4.

References

[1] "An Engineering Approach to Computer Networking," S. Keshav, Addison-Wesley, 1997.

[2] "Efficient Fair Queuing using Deficit Round Robin," M. Shreedhar, George Varghese, SIGCOMM 1995, pp. 231-242.

from High Availability Networking Fundamentals

by Chris Oggerino

(1-58713-017-3)

Cisco Press

About the Author

Chris Oggerino has been employed by Cisco Systems, Inc., for over five years and is currently a Serviceability Design Engineer. As a Serviceability Design Engineer at Cisco Systems, Chris spends his time improving the reliability, availability, serviceability, and usability of Cisco products. Prior to his employment with Cisco Systems, Chris spent six years doing technical support of UNIX and internetworking products, three years in microcomputer corporate sales, and four years as a programmer at a variety of companies in the Bay Area. Chris currently resides in Los Gatos, California. You can reach Chris Oggerino at chris@oggerino.org.

Contents at a Glance

Bold chapters are elements included in this folio.

CHAPTER 2

The Basic Mathematics of High Availability

Calculating availability requires mathematics. As the depth of availability research increases, the mathematical equations get more complex. This book makes it possible for people unfamiliar with calculus to perform network availability analysis. This chapter is designed to introduce you to the equations we will be using in this book.

The equations in this book are as simple as possible, while still maintaining accuracy. Some more advanced availability experts might say that I have taken a few liberties that could make the results slightly skewed. However, I believe that the equations presented in this chapter will provide perfectly adequate results for comparing network versus network availability and estimating whether a particular design is better than another design.

Determining the Availability of Network Device Components

In order to calculate the availability of a network, you have to calculate the availability of the individual network devices comprising it. And in order to calculate the availability of a network device, you have to calculate the availability of its components. The calculation of the availability of the devices' components is the starting point of the mathematics of availability.

Cisco Systems uses what is called the *Telcordia* (formerly Bell Core) *Parts Count Method.* The Parts Count Method is described in Technical Reference document TR-332, "Reliability prediction procedure for Electronic Equipment" from December 1997.

NOTE In order to avoid the complex mathematics that created the need for this simple book, this chapter summarizes the contents of TR-332. If you would like to get this document for further reading, you can purchase it from the Telcordia Web site at www.telcordia.com.

TR-332 describes the method used to determine the reliability of each component on a circuit board—capacitors, resistors, or other components. Each component is associated with a *FIT* (failures in 10^9), which represents the failures per billion hours of running time you can expect from that component. In other words, one FIT is one failure per one billion hours. TR-332 also includes factors by which you can influence the number of FITs for a particular component, based on things like temperature, environment, and quality control processes.

Cisco Systems uses a software product to do FIT calculations automatically. Cisco quality engineers feed the entire list of materials, called a *bill of materials* (BOM), that make up a circuit board into the program. Next, they tell the program any special adjustments that need to be made, such as running any particular components over their rated voltage or temperature. The program then returns the total FITs for BOM and the predicted MTBF for the circuit board.

Cisco MTBF Results

Over the past 15 years, Cisco Systems has found that the numbers produced by using Relex's (Telcordia TR-332 based) software are extremely conservative. In fact, Cisco uses a multiple of the results from the prediction software in an attempt to make the results more accurate. Even with this additional doubling of the MTBF, Cisco Systems products greatly outperform their predicted MTBFs. This is most likely a result of the software not accounting for conservatively engineered circuits and superior quality control of the components themselves.

A great example of these MTBF numbers is Cisco's 2500 line of products. Cisco has shipped more than a million 2500s to customers all over the world. When comparing the return rates to Cisco Systems repair and logistics organizations with the predicted MTBFs for the 2500s, it has been found that the product outperforms the estimates by a wide margin. In fact, 2500 routers on some networks have been running continuously, without a reboot or a glitch of any sort, for over eight years!

The MTBF numbers for Cisco products are analyzed and published internally for all Cisco employees. These numbers can be used to calculate MTBF for Cisco routers and switches at the request of customers. If you are a customer and you request an MTBF of a Cisco product, the person responding to your request will go into the MTBF database and get the MTBF for the components that make up your system. He or she will then either present you with the MTBF numbers from the database or use equations like those in this book to present the availability of the system to you.

NOTE As you will see in Chapter 4, "Factors That Affect Availability," hardware is not the only thing that affects the availability of a network: methods are available to predict software MTBF as well as hardware MTBF.

Estimating MTTR of a Network

As you learned in Chapter 1, "Introduction to High Availability Networking," MTTR is another important component of availability. You can measure MTTR in an existing network by taking the average of the time the network is down for each failure. You can predict MTTR based on a variety of methods.

The method we will use in this book to predict MTTR is based on Cisco Systems service contracts. If a customer determines that high availability is required for a network, they could purchase a four-hour, on-site service contract. This contract would be somewhat expensive but would guarantee that a repair technician and parts would be on their site no more than four hours after they called Cisco for support. With this type of service contract, I would give that company an MTTR of four hours. If the company opted for a service contract of eight hours, I would give them an MTTR of about eight hours.

For our purposes in this book, the MTTR is arbitrary. The number you use should be based on the service contract you purchase from your service vendor. In this book, we will use a variety of MTTR values so that you can be sure to understand the impact of short and long MTTRs to network availability. You will also want to note, in Chapter 4, that software MTTR can often be much shorter than hardware if the software has self correcting capabilities.

The Availability Equation and Network Device Components

Chapter 1 includes a basic description of availability. In order to calculate availability, you must use the availability equation. The availability equation (shown again in Equation 2-1) describes how you use MTBF and MTTR to find a percentage result:

The ratio of uptime to total time

In other words, the percentage availability is equal to the amount of uptime divided by the total time during some time period *t*. Time *t* will consist of both the running time and the non-running time for the box.

Equation 2-1 The Availability Equation

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

As you can see, when you have MTBF and MTTR, you can calculate availability with a simple calculator.

Availability and Uptime/Downtime

When you have an availability percentage, you can calculate uptime and downtime. Conversely, if you have the uptime or the downtime you can calculate availability.

Because availability is a percentage that represents the uptime divided by the total time, you can multiply any time period by the availability number and get the amount of uptime over that period. Of course, the difference in the total time and the uptime is the downtime.

In most cases, you are going to want the downtime per year. You will do this by subtracting the availability from 1 and then multiplying the result by the number of minutes in a year.

If you have a device that is available 99.99 percent (or 0.99990), then you can expect 52.596 minutes per year of downtime as calculated in Figure 2-1.

Figure 2-1 *Getting Downtime from Availability*

$$\begin{aligned}
\text{1 year} &= \text{525,960 minutes}^{*}\\
\text{Uptime} &= \text{Availability} * \text{Time}\\
\text{Annual Uptime} &= \text{Availability} * 525{,}960\\
\text{Annual Downtime} &= 525{,}960 - \text{Annual Uptime}
\end{aligned}$$

$$\begin{aligned}
\text{Availability} &= .9999\\
\text{Annual Uptime} &= .9999 * 525{,}960\\
\text{Annual Uptime} &= 525{,}907.4\\
\text{Annual Downtime} &= 525{,}960 - 525{,}907.4\\
&= 52.596
\end{aligned}$$

$$\begin{aligned}
\text{Downtime} &= (1 - \text{Availability}) * \text{Time}\\
\text{Annual Downtime} &= (1 - .9999) * 525{,}960\\
&= 52.596
\end{aligned}$$

*Adjusted for leap years

Determining the Availability of a Single Component

If you wanted to calculate the availability of a particular component, use the availability equation with the MTBF and MTTR for that component. If the component in question had an MTBF of 100,000 hours and an MTTR of 6 hours, then we could calculate the availability to be 0.99994. The resulting downtime would be about 31.5 minutes per year for that component as calculated in Figure 2-2.

Figure 2-2 *Finding the Downtime of a Single Component*

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

$$\text{MTBF} = 100{,}000 \text{ hours}$$

$$\text{MTTR} = 6 \text{ hours}$$

$$\text{Availability} = \frac{100{,}000}{100{,}000 + 6}$$

$$\text{Availability} = \frac{100{,}000}{100{,}006}$$

$$\text{Availability} = 0.99994$$

$$\begin{aligned}\text{Annual Downtime} &= (1 - .99994) * 525{,}960 \\ &= 31.5576 \text{ minutes}\end{aligned}$$

Determining the Availability of Multiple Components

To calculate the availability of multiple components, you must understand more equations: the serial equation and the parallel availability equation. One important thing to remember is that components can be system components (such as circuit boards) or network components (such as routers or switches).

You will use the serial availability equation whenever all the parts must work for the system (or network) to work. Equation 2-2 shows the serial availability equation.

Equation 2-2 Serial Availability Equation

$$SerialAvailability = \prod_{i=1}^{n} ComponentAvailability_{(i)}$$

i represents the component number
n represents the number of components

In a serial system, if any component fails then the entire system fails. For example, if a product consists of a power supply and a circuit board, you have a serial system. If the power supply fails, then the system fails, and if the single circuit board fails, then the system fails.

Sometimes the components in a system are in parallel, or redundant. Although there are several different parallel designs, Equation 2-3 shows the basic parallel equation. This equation applies to the situation where two devices are in parallel with each other.

Equation 2-3 Parallel Availability Equation

$$ParallelAvailability = 1 - \left[\prod_{i=1}^{n}(1 - ComponentAvailability_{(i)})\right]$$

i represents the component number
n represents the number of components

In a *parallel system,* two or more components are combined such that the system will work as long as any of the parallel components are still working. If you have two parallel components and one of them fails, the system continues (or at least should continue) to run without failure. Most systems that include parallel components also include serial components.

In order to calculate the availability of multiple components, you must understand the serial and parallel availability equations. The following sections describe these equations and how they relate to one another. First you will learn how to determine the availability of a serial system. Then you will learn how to combine parallel components to get their availability (that is, availability in parallel). Finally, we will discuss how to calculate the availability of a mixed serial/parallel system.

Serial Availability

In order to estimate the availability of a serial system, you multiply together the availability of the components. For example, in the system with a single circuit board and a power supply, you multiply the availability of the power supply by the availability of the circuit board. Figure 2-3 shows these calculations.

Figure 2-3 *Determining Serial Availability in a Two-Component System*

Power Supply = 99.999% availability
= 0.99999
Circuit Board = 99.994% availability
= 0.99994

System Availability = 0.99999 * 0.99994
= 0.99993

As you can see, two components with availability of 99.999 percent and 99.994 percent combined in a serial configuration provide 99.993 percent total system availability.

Because most systems contain more than two components, we need to use an equation that works for some number (*N*) of components. Although this equation contains a symbol that might be Greek to some of you (because it is), you can remember that all this equation does is multiply all the component availability's together, like the system in Figure 2-3. The symbol for multiplying *N* components together is a capital Greek letter Pi (Π). Figure 2-4 shows the equation and a short example.

Figure 2-4 *Determining Serial Availability in an N-Component System*

$$\text{System Availability} = \prod_{i=1}^{n} \text{availability}_{(i)}$$

Power Supply = 99.999%

Circuit Board 1 = 99.994%

Circuite Board 2 = 99.98%

$$\text{System Availability} = \prod_{i=1}^{n} \text{availability}_{(i)}$$

$$= .99999 * .99994 * .9998$$

$$= .99973$$

In English, you say, "For the components 'I' going from 1 to N (the number of components), multiply by availability$_{(i)}$." In programming this is a "for I equal 1 to N" loop, multiplying each availability by each other.

As you can see from Figure 2-4, the serial availability equation is simple to use even though is includes a Greek symbol.

Simple Parallel Availability

To find simple parallel availability, you first multiply the unavailability of each of the parallel parts. The result of the multiplication of the unavailabilities is subtracted from 1 to get the availability result. In the next section, we will talk about a more complex method of parallel calculation.

In Figure 2-5, you see a symbol with both I and N in it. As with the serial equation, this simply means to multiply each "unavailability of component (I)" by each other until *N* "unavailability of component (I)s" have been multiplied together.

Figure 2-5 *Determining Availability in a Parallel System*

$$\text{Parallel Availability} = 1 - \left[\prod_{i=1}^{n} (1 - \text{availability}_{(i)}) \right]$$

N = Number of Components in Parallel
I = The Component Number

To clarify the equation, let's look at a small example. Assume that you have a system and that inside the system are two components in parallel. For simplicity's sake, we'll say the two components are identical and both have an availability of 99.9 percent (that is, 0.999 availability). Figure 2-6 shows how we would combine these two components together to get their parallel availability.

Figure 2-6 *Determining Parallel Availability in a Two-Component System*

$$\text{Component 1} = 99.9\%\ \text{availability}$$

$$\text{Component 2} = 99.9\%\ \text{availability}$$

$$\begin{aligned}
\text{Parallel Availability} &= 1 - \left[\prod_{i=1}^{2} (1 - \text{availability}_{(i)}) \right] \\
&= 1 - \left[(1 - .999) * (1 - .999) \right] \\
&= 1 - \left[1 - .000001 \right] \\
&= .999999
\end{aligned}$$

$$\text{Percent Availability} = .999999 * 100$$

$$\text{Percent Availability} = 99.9999\%$$

It is important that we take note of a couple things about parallel availability. First, systems designed with parallel components usually have some method for moving work from the failed component to the remaining component. This feature is called a *fail-over mechanism.* Sometimes fail-over mechanisms themselves fail. We are going to exclude that probability in this book because we could continue exploring the depths of availability to no end if we went down that path. (You'll learn more about fail-over mechanisms in Chapter 4 in the network design section.) Next, we must note that simply combining a couple of items in parallel is not going to happen very often. In fact, most systems and networks include both parallel and serial components. The next section describes the method for calculating both parallel and serial scenarios.

Before we can move on to our next section on combining serial and parallel components, we need to introduce a more complex method of computing parallel availability.

N + 1 Parallel Availability

We have discussed how to calculate the availability when devices are in simple parallel arrangements. In other words, if we have any number of devices in parallel, we now know how to calculate the availability as long as any single device remains operational. This method is most often implemented in real life by having two devices when you need only one device.

In the real world, we might need more than one device for proper system or network functionality. An example is where we need at least two power supplies in our big router to supply enough power. In that situation, we might choose to have a single, backup power supply instead of two backup power supplies. This method of redundancy is called, *N + M redundancy. N* represents the number required and *M* represents the number installed.

Our coverage of N + M redundancy in this book is limited because we want to limit the mathematics required to simple equations as much as possible. However, in our examples in Chapter 7, "A Small ISP Network: An Availability Analysis," Chapter 8, "An Enterprise Network: An Availability Analysis," and Chapter 9, "A Large VoIP Network: An Availability Analysis," we will be using router and network examples which include the N + M and N + 1 redundancy methods.

Coverage of N + M is given to you by including the SHARC (System Hardware and Reliability Calculator) spreadsheet tool on the CD with this book. That tool can calculate N + M redundancy for you. The SHARC spreadsheet is introduced and used in Chapter 7. Appendix A describes the contents of the CD and the usage of the SHARC spreadsheet.

When you come across N + 1, calculate it using Equation 2-4.

Equation 2-4 The N + 1 Redundancy Equation

$$A = nA^{(n-1)} * (1 - A) + A^{n}$$

A = Total Availability
n = Number of devices

NOTE We assume an equal availability for each component.

However, N + 1 and N + M redundancies are best done by the SHARC spreadsheet. We will not use this spreadsheet until the later chapters in order to make sure we have the basic equations memorized—before we start using the easy tools!

Serial/Parallel Availability

Most systems (and networks) contain both serial and parallel components. The method we use to calculate serial/parallel availability involves two steps:

Step 1 Calculate the parallel availability for all the parallel components.

Step 2 Combine the results of Step 1, along with all the serial components, using the serial availability method.

In order to perform these calculations, you need some knowledge about the system or network. You must understand the path through which the data will travel. You must understand which components are critical or redundant. The availability of a system or a network depends on the availability of the path between Points A and B between which data must flow. Chapter 3, "Network Topology Fundamentals," covers the basics of these considerations.

Because we know the parallel equation and the serial equation, we will now build those together into the serial/parallel equations and process. The best way to illustrate serial/parallel availability is by example, as shown in Figure 2-7. In this example, we consider a system that has a single circuit board and two redundant power supplies. The circuit board is 99.994 percent available, and each of the power supplies is 99.95 percent available.

Figure 2-7 *Determining Serial/Parallel Availability in a Three-Component Parallel Example*

Circuit Board = 99.994%
Power Supplies = 99.95%

Step 1: The parallel power supply component calculations

$$\text{Parallel Availability} = 1 - \left[\prod_{i=1}^{2} (1 - \text{availability}_{(i)})\right]$$

$$= 1 - (1 - .9995) * (1 - .9995)$$

$$= .99999975$$

Step 2: The circuit board and redundant power in serial calculations

$$\text{Serial Availability} = \prod_{i=1}^{2} (\text{availability}_{(i)})$$

$$= .99994 * .99999975$$

As you can see, even though the power supplies are not all that reliable, combining two components in parallel greatly increases the availability of the overall system.

Determining Data Flow in a Network: Path Analysis

The most common situation that will occur in your studies of availability will be the combination of serial and parallel components in a large system. Furthermore, not all of these components will be required for data to flow in the scenario for which you are concerned. Many network devices connect a large number of networks together. If you are considering the availability of only the first two networks that the device joins, then you won't care if some component carrying traffic to the third network fails.

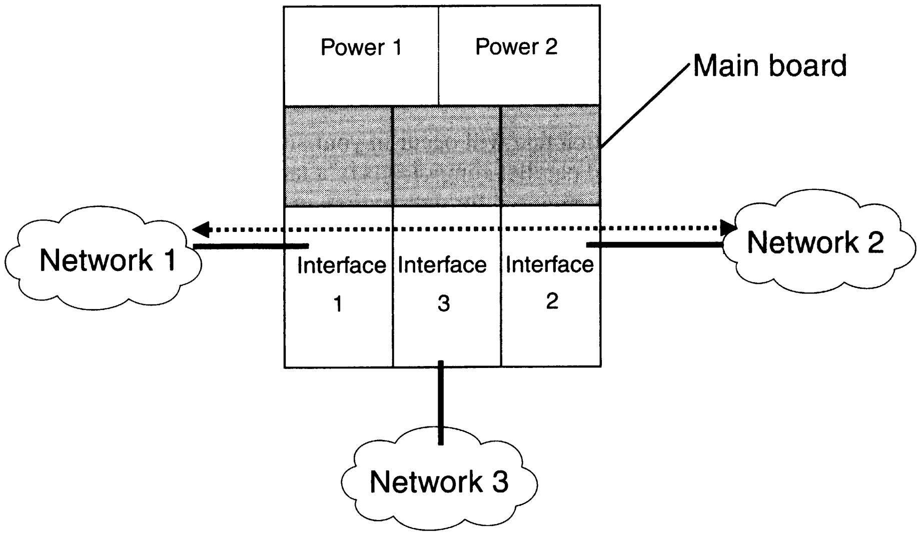
This consideration of the data flow is called *path analysis*. Path analysis makes it possible to use exactly the same equations that we have been using for calculating the availability of a system to calculate the availability of a network. In network availability calculations, we use the network devices, such as routers and switches, as components. In network computations, we might use path analysis to determine the data path through a Cisco 12000 router and use those components to analyze the availability of the Cisco 12000. Once that calculation is complete, we would use the result as a component in a network analysis.

Using Reliability Block Diagrams for Path Analysis

To perform a path analysis, creating an availability block diagram is often a good idea.

For now, we'll consider a very simple example of networking, using a large router with components that are not crucial to our calculations. We need to eliminate those from our calculations and perform the analysis of availability for something we care about. In Figure 2-8, a large network device is connected to three networks. We only want to know the availability from Network 1 to Network 2. Therefore, the hardware that supports the connection to Network 3 can be eliminated from our calculations.

Figure 2-8 *Analyzing the Path from Network 1 to Network 2*



As you can see from Figure 2-8, our device includes two redundant power supplies called Power 1 and Power 2. Several steps are involved with calculating the availability of a device (or network) which has both parallel and serial components.

In this case our steps are

Step 1 Determine the availability of each component

Step 2 Compute the availability of the dual power supplies

Step 3 Multiply the availability of all components with the result from Step 2.

Figure 2-9 shows the availability analysis from Network 1 to Network 2.

Figure 2-9 *Path Analysis: Determining the Availability from Network 1 to Network 2*

Power Supplies = 99.9%
Motherboard = 99.994%
Interface Cards = 99.95%

Step 1: Parallel calculations

Power = 1- [(1-.999) * (1-.999)]
= .999999

Step 2: Serial calculations

Power = .999999
Motherboard = .99994
Interface 1 = .9995
Interface 2 = .9995

System Availability = .999999 * .99994 * .9995 * .9995
= .998939
= 99.8939%

As you can tell from Figure 2-9, we used 99.9 percent for the power supplies, 99.994 percent for the motherboard, and 99.95 percent for each of the interface boards. After all the calculations were completed, the result of 99.8939 percent describes the availability from Network 1 to Network 2 via the device shown in the diagram. Note that interface 3 was not included in the calculations because it was not a required component. Using 525,960 minutes per year, and our standard method for calculation, we arrive at about 9.3 hours per year of downtime. Even with redundant power supplies, this system would cause a considerable amount of downtime in a network. As you proceed through this book, you will see many examples. Through these examples, you should learn a variety of ways to correct problems with too much annual downtime.

In Chapter 5, "Predicting End-to-End Network Availability: The Divide-and-Conquer Method," we are going to talk about the "divide-and-conquer" method of performing availability analysis. This will expand, greatly, on how one performs reliability block diagrams. What you should have learned here is that you need to consider which components to include in your calculations. After Chapter 5, you should know exactly how to perform the tasks.

from Cisco Secure Internet Security Solutions

by George Abe

(1-57870-177-5)

Cisco Press

About the Authors

Andrew G. Mason, CCIE #7144, CCNP Security, and CCDP, is the CEO of CCStudy.com Limited (www.ccstudy.com), a United Kingdom-based Cisco Premier Partner specializing in Cisco consulting for numerous United Kingdom-based companies. The CCStudy.com web site is a fast-growing online Cisco community for all of the Cisco Career Certifications.

Andrew has 10 years of experience in the network industry and currently is consulting for Energis-Squared, the largest ISP in the United Kingdom. He is involved daily in the design and implementation of complex secure hosted solutions, using products from the Cisco Secure product range.

Mark J. Newcomb, CCNP Security and CCDP, is a senior consulting network engineer for Aurora Consulting Group (www.auroracg.com), a Cisco Premier Partner located in Spokane, Washington, USA. Mark provides network design, security, and implementation services for clients throughout the Pacific Northwest.

Mark has more than 20 years of experience in the microcomputer industry. His current projects include designing secure communication systems for wireless devices and providing comprehensive security services to the banking industry.

Contents at a Glance

Bold chapters are elements included in this folio.

This chapter contains the following sections:

- PIX Models
- PIX Features
- PIX Configuration
- VPN with Point-to-Point Tunneling Protocol (PPTP)
- VPN with IPSec and Manual Keys
- VPN with Preshared Keys
- Obtaining Certificate Authorities (CAs)
- PIX-to-PIX Configuration
- Summary

CHAPTER 4

Cisco Secure PIX Firewall

This chapter focuses on the Cisco Secure Private Internet Exchange (PIX) Firewall. The strength of the security features within the PIX lay in the fact that it was designed solely as a firewall. Although a PIX Firewall will do a limited amount of routing, the real purposes of the PIX are to deny unrequested outside traffic from your LAN and to form secure Virtual Private Networks (VPNs) between remote locations. A router requires a great deal of configuration to act effectively as a firewall. The PIX, however, only requires six commands before it can be placed into service. The PIX is easy to configure and generally requires no routine maintenance once configured.

The larger a sphere is, the larger the surface area of that sphere. If you analogize the security concerns of an operating system to a sphere, you soon realize that the larger the operating system, the larger the "surface area" that must be defended. A router with a much larger operating system must be carefully configured to stop intruders, prevent denial of service (DoS) attacks, and secure the LAN. The PIX operating system, originally designed as a Network Address Translation (NAT) device, is not a general-purpose operating system and operates in real time, unlike both Windows NT and UNIX. Therefore, the PIX has a very small operating system that presents fewer opportunities for a security breach. The smaller the operating system, the less chance that an area has been overlooked in the development process.

The PIX does not experience any of the many security holes present within either UNIX or Windows NT. The operating system is proprietary, and its inner workings are not published for use outside of Cisco Systems. The general networking public does not have access to the source code for the PIX, and therefore, the opportunities for exploiting a possible vulnerability are limited. The inner workings of the PIX Firewall are so secret that the authors of this book were not able to gain access to them.

Several advantages to using the PIX over a router or a UNIX, Linux, or Windows NT-based firewall exist. The benefits of using a PIX include the following:

- PIX's Adaptive Security Algorithm (ASA), combined with cut-through proxy, allows the PIX to deliver outstanding performance
- Up to 500,000 connections simultaneously
- Throughput speeds up to 1000 Mbps
- Failover capabilities on most models

- An integrated appliance
- IPSec VPN support
- NAT and Port Address Translation (PAT) fully supported
- Low packet delay
- Low cost of ownership due to no OS maintenance
- Integrated Intrusion Detection System (IDS)
- High reliability, no hard disk, Mean Time Between Failure greater than 60,000 hours
- Common criteria EAL 2 certification

PIX Models

The PIX Firewall comes in four main models, with an additional model that's being phased out. Ranging in size from models designed for the home or small office through enterprise level firewalls, the PIX models allow for virtually any size of organization to be protected. The models are as follows:

- PIX 506
- PIX 515
- PIX 520/525
- PIX 535

The features of each model follow.

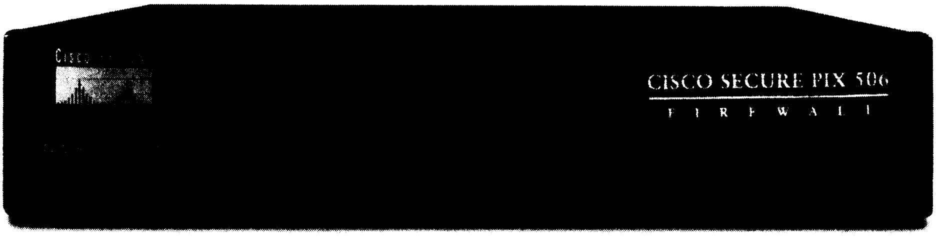
PIX 506

The PIX 506 is the smallest of the PIX Firewalls available. Currently list-priced at less than U.S. $2000, the 506 is designed for firewall protection of the home or small business office. The 506 is approximately one-half the width of the rest of the PIX models. The capabilities and hardware features of the 506 are as follows:

- 10 Mbps throughput
- 7 Mbps throughput for Triple Data Encryption Standard (3DES) connections
- Up to ten simultaneous IPSec Security Associations (SAs)
- 200 MHz Pentium MMX processor
- 32 MB SDRAM
- 8 MB Flash memory
- Two integrated 10/100 ports

A picture of the PIX 506 is shown in Figure 4-1.

Figure 4-1 *PIX 506*



PIX 515

The PIX 515 is designed for larger offices than those of the 506. There are three main advantages of the 515 over the 506. The first advantage is the ability to create demilitarized zones (DMZs) through the use of an additional network interface. The second advantage is the throughput speed and number of simultaneous connections supported. The third advantage is the ability to support a failover device that will assume the duties of the primary PIX should there be a failure. The PIX 515 comes in two models, the 515 Restricted (515-r) and the 515 Unrestricted (515-ur). The characteristics of these two models follow.

PIX 515-r:

- No failover devices supported.
- A single DMZ can be used.
- Ethernet must be the LAN protocol.
- Maximum of three interfaces may be used.
- 32 MB RAM.

PIX 515-ur:

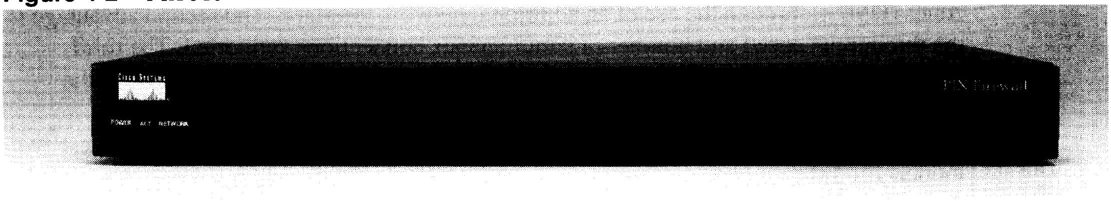
- Failover devices are supported.
- Two DMZs may be implemented.
- Ethernet must be the LAN protocol.
- Maximum of six interfaces may be used.
- 64 MB RAM.

These two models are essentially the same hardware with different memory and software. It is possible to purchase a 515-r and upgrade it to a 515-ur by adding more memory and updating the operating system. The net cost to the user is very close to the purchase price of a 515-ur. The capabilities and hardware features of the 515 follow:

- Rack mountable
- Up to 100,000 simultaneous connections
- Up to 170 Mbps throughput
- Up to four interfaces
- Up to 64 MB SDRAM
- 16 MB Flash memory
- 200 MHz Pentium MMX processor

A picture of the PIX 515 is shown in Figure 4-2.

Figure 4-2 *PIX 515*



PIX 520/525

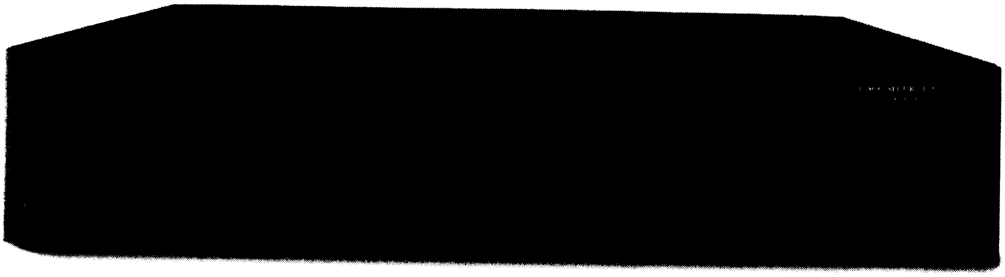
The PIX 520, sometimes called the classic PIX, is in the process of being phased out in favor of the newer design of the model 525. Both of these firewalls have the same underlying hardware.

The PIX 525 is designed for a large organization and has the following capabilities and hardware features:

- Rack mountable
- More than 256,000 simultaneous connections
- Six to eight integrated Ethernet cards
- Up to four Token Ring cards
- Up to four FDDI or four Gigabit Ethernet cards
- More than 240 Mbps throughput
- Up to 256 MB RAM

A picture of the PIX 525 is shown in Figure 4-3.

Figure 4-3 *PIX 525*



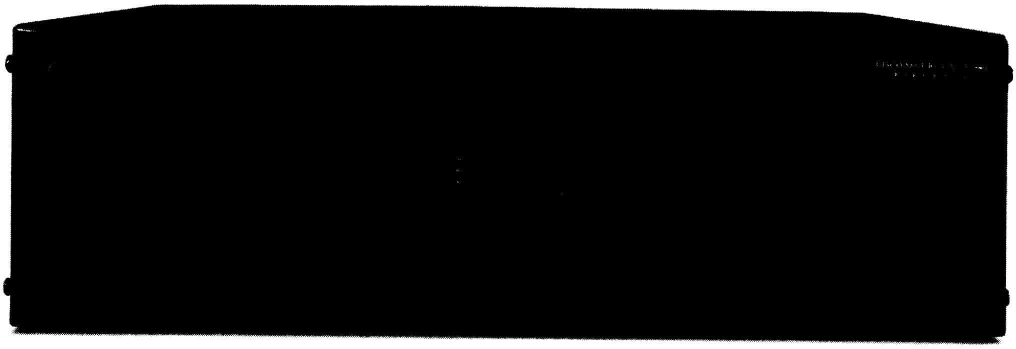
PIX 535

The PIX 535 is designed for large enterprise and Internet service provider (ISP) environments where an extreme amount of traffic must be secured. This is presently the largest PIX Firewall available and has the following capabilities and hardware features:

- Rack mountable
- More than 500,000 simultaneous connections
- Six to eight integrated Ethernet cards
- Up to four Token Ring cards
- Up to four FDDI or eight Gigabit Ethernet cards
- More than 1,000 Mbps throughput
- 512 to 1024 MB RAM

A picture of the PIX 535 is shown in Figure 4-4.

Figure 4-4 *PIX 535*



PIX Features

The PIX Firewalls, regardless of model number, all provide the same security features. The PIX is a stateful firewall that delivers full protection to the corporate network by completely concealing the nature of the internal network to those outside. The main operating features of the PIX follow:

- **Sequence random numbering**—IP spoofing generally relies on the ability to guess a sequence number. The PIX randomizes the IP sequence numbers for each session. This makes IP spoofing much more difficult to accomplish.
- **Stateful filtering**—This is a secure method of analyzing data packets that is also known as the Adaptive Security Algorithm (ASA). When data traverses from the trusted interface on the PIX to a less trusted interface, information about this packet is entered into a table. When the PIX receives a data packet with the SYN bit set, the PIX checks the table to see if, in fact, the destination host has previously sent data out to the responding host. If the table does not contain an entry showing that the local host has requested data, the packet is dropped. This technique virtually eliminates all SYN-based DoS attacks.
- **Network Address Translation (NAT)**—NAT is the process of changing the source IP address on all packets sent out by a host and changing the destination IP address of all incoming packets for that host. This prevents hosts outside of the LAN from knowing the true IP address of a local host. NAT uses a pool of IP addresses for all local hosts. The IP address a local host will receive changes as addresses are used and returned to the pool.
- **Port Address Translation (PAT)**—PAT is similar to NAT except that all local hosts receive the same IP address. Using different ports for each session differentiates local host sessions. The IP address of the local host is still changed using PAT, but the ports associated with the session are also changed. Both PAT and NAT can be used concurrently on a PIX Firewall.
- **Embedded operating system**—A UNIX, Linux, or Windows NT machine can be used as a proxy server. However, the throughput of such a machine is slower by design than that available through the PIX. A proxy server receives an Ethernet packet, strips off the header, extracts the IP packet, and then moves that packet up through the OSI model until it reaches the application layer (Layer 7), where the proxy server software changes the address. The new IP packet is rebuilt and sent down to Layer 1 of the OSI model, where it is transmitted. This uses a large number of CPU cycles and introduces delay. Because the PIX is a proprietary system, the OSI model constraints can be bypassed and made to allow cut-through proxy to operate.
- **Cut-through proxy and ASA**—The combination of cut-through proxy and ASA allows the PIX to process more than 500,000 connections simultaneously with virtually no packet delay. Cut-through proxy is the process where the first packet in a session is checked as in any proxy server, but all subsequent packets are passed through. This technique allows the PIX to transfer packets extremely efficiently.

- **DNS guard**—By default, all outgoing DNS requests are allowed. Only the first response is allowed to enter the LAN.
- **Mail guard**—Only RFC 821-specific commands are allowed to a Simple Mail Transfer Protocol (SMTP) server on an inside interface. These commands are **HELLO, MAIL, RCPT, DATA, RSET, NOOP,** and **QUIT**. The PIX responds with an OK to all other mail requests to confuse attackers. This is configured with the **fixup** command.
- **Flood defender**—This limits the total number of connections and the number of half-open connections. User Datagram Protocol (UDP) response packets that either have not been requested or arrive after a timeout period are also dropped.
- **ICMP deny**—By default, all Internet Control Message Protocol (ICMP) traffic does not get sent over the inside interface. The administrator must specifically allow ICMP traffic to enter if needed.
- **IP Frag Guard**—This limits the number of IP full-fragment packets per second per internal host to 100. This prevents DoS attacks such as LAND.c and teardrop. Additionally, this ensures that all responsive IP packets are let through only after an initial IP packet requesting the response has traversed the PIX.
- **Flood guard**—This feature is designed to prevent DoS attacks that continuously request an authentication of a user. The repetitive requests for authentication in this type of DoS attack are designed to use memory resources on a network device. The PIX relies on a subroutine that uses its own section of memory. When an excessive number of authentication requests are received, the PIX starts dropping these requests and reclaiming memory, thus defeating this form of attack.
- **Automatic Telnet denial**—By default, the PIX Firewall will not respond to any Telnet request except through the console port. When enabling Telnet, set it to allow only those connections that are actually necessary.
- **Dynamic Host Configuration Protocol (DHCP) client and server support**—The PIX can rely on a DHCP server to gain an IP address for an interface. As a DHCP server, the PIX provides IP addresses for hosts attached to one of the interfaces.
- **Secure Shell (SSH) support**—The PIX supports the SSH remote shell functionality available in SSH version 1. SSH is an application that runs on top of a connection-oriented Layer 3 protocol such as TCP. SSH provides encryption and authentication services for Telnet sessions. Support for SSH requires third-party software, which may be obtained at the following sites:

  — Windows client:

  hp.vector.co.jp/authors/VA002416/teraterm.html

  — Linux, Solaris, OpenBSD, AIX, IRIX, HP/UX, FreeBSD, and NetBSD client:

  www.openssh.com

— Macintosh client:

www.lyastor.liu.se/~jonasw/freeware/niftyssh

- **Intrusion Detection System (IDS)**— The PIX integrates the same IDS features that are available on routers through the Cisco Secure IOS. The IDS detects 53 specific types of intrusion. See Chapter 6, "Intrusion Detection Systems," for more details on IDS.
- **TCP intercept**—The PIX can act like a TCP intercept device, isolating protected hosts from direct contact through TCP connections. TCP intercept is discussed in Chapter 2, "Basic Cisco Router Security."

PIX Configuration

This section explores how to configure a PIX Firewall for a number of different scenarios. This section defines terms and gives explanations of how different scenarios require different hardware and software configurations.

The basic PIX configuration is extremely simple. By default, this configuration allows outgoing packets and responsive packets into the LAN. This configuration also denies all ICMP packets traversing the PIX from the outside to the inside, even when such a packet is in response to a ping issued from the inside.

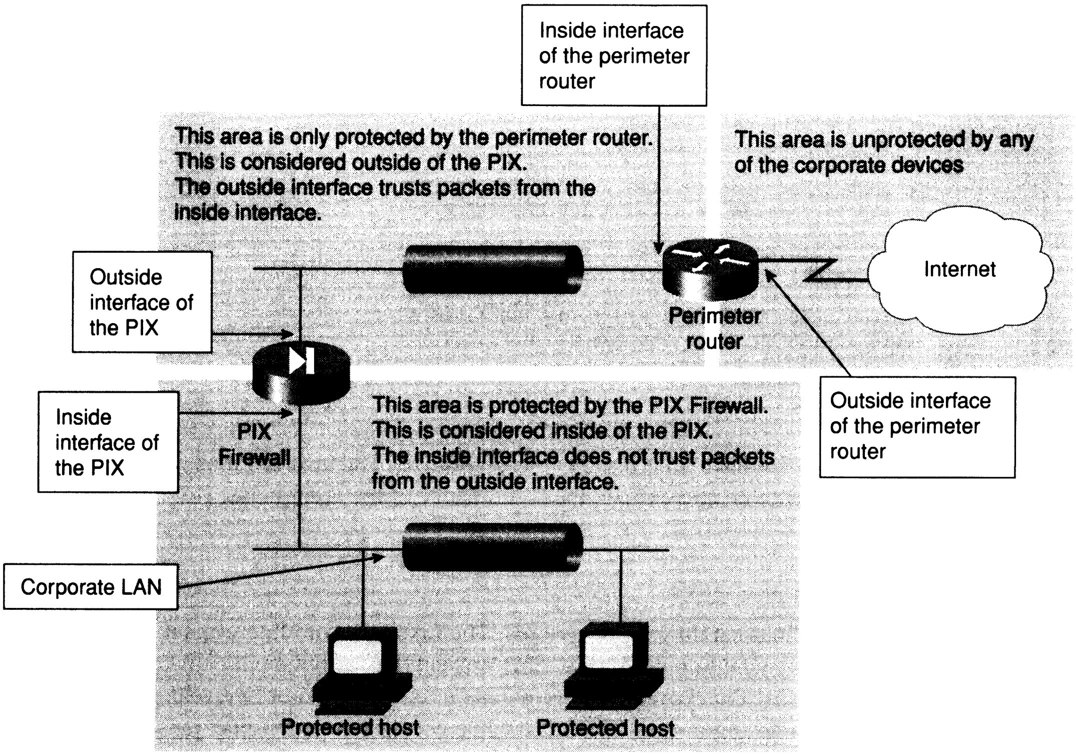
Like any other Cisco IOS, the Cisco PIX has a command-line interface (CLI). There is a user mode and an enable mode. For the moment, you will configure the PIX by connecting the console port on the PIX to a serial port on a computer using the cable you received with the PIX Firewall. Some of the commands will be familiar and some will be new. Each scenario in this section builds on the previous scenario.

If by issuing a **show config** command you see a number of items not shown on a particular configuration, do not panic. The PIX enters a number of defaults into the configuration when booting. These defaults can be changed. This chapter will deal with the most frequently used commands first. If you simply cannot wait to see what a command does, look in the index and jump ahead to the section concerning that command.

Basic Configuration

The basic configuration for the PIX is illustrated in Figure 4-5. In this scenario, the PIX is used to protect a single LAN from the Internet. Notice in Figure 4-5 that the perimeter router and the connection between the perimeter router and the outside interface of the PIX are unprotected. The perimeter router should be hardened against attacks—especially DoS attacks—because it is not protected by the PIX Firewall. Chapter 10, "Securing the Corporate Network," deals with securing a perimeter router. Any device that is outside of the PIX Firewall cannot be protected by the PIX. If possible, only the perimeter router should reside on the unprotected side of the network. Take a few minutes to study Figure 4-5, which you can use to define terms such as *inside, outside, protected,* and *unprotected.*

Figure 4-5 *Basic PIX Configuration Sample Network*


Inside interface of the perimeter router
This area is only protected by the perimeter router. This is considered outside of the PIX. The outside interface trusts packets from the inside interface.
This area is unprotected by any of the corporate devices
Outside interface of the PIX
Internet
Perimeter router
Inside interface of the PIX
PIX Firewall
This area is protected by the PIX Firewall. This is considered inside of the PIX. The inside interface does not trust packets from the outside interface.
Outside interface of the perimeter router
Corporate LAN
Protected host
Protected host

As shown in Figure 4-5, there is an inside and an outside interface on the PIX. The outside interface is less trusted than the inside interface. The inside interface has a security level of 100. The outside interface has a security level of 0. The security level is what determines whether packets originating from a particular interface are trusted by another interface. The higher the security level, the more an interface is trusted. This premise becomes more important as you build systems with multiple DMZs. When packets are trusted, they are allowed through an interface by default. When packets are not trusted, they are not allowed through by default.

For the basic configuration, you only need to add a few commands. This section takes much longer to read than it will actually take to configure the PIX. Start up the PIX Firewall and connect the inside interface into your local network. Connect the outside interface to the inside interface of your perimeter router. Do *not* connect these through the same switch or hub that runs your local network. The only path from the perimeter router to your LAN must travel through the PIX Firewall. Companies with multiple paths to the Internet should employ a PIX Firewall between each perimeter router and the LAN.

After showing you how to configure the PIX, the chapter explains what has been done. Using Telnet, enter the following commands. The lines are separated for clarity.

```
enable password enablepass encrypted
passwd password encrypted

nameif ethernet0 outside security0
nameif ethernet1 inside security100

interface ethernet0 10baset
interface ethernet1 10baset

ip address outside 192.168.1.1 255.255.255.0
ip address inside 10.1.1.254 255.255.255.0

global (outside) 1 192.168.1.100 255.255.255.0
nat (inside) 1 10.1.1.0 255.255.255.0 0 0

route outside 0.0.0.0 0.0.0.0 192.168.1.254 1
route inside 10.1.1.0 255.255.255.0 10.1.1.1 1

arp timeout 7200

write mem
```

At this point, you have your basic configuration set. The next sections walk through each line that you entered and explain the significance of the commands.

password Commands

The first two lines set up your passwords. The first password line was set with the **enable password** command to **enablepass**. This was entered with the optional keyword **encrypted**. Using **encrypted** ensures that the password will not be revealed if you print out a copy of your configuration. The second line configures your Telnet password to **password**. The same rules that apply to router passwords apply to PIX passwords. For example, the enable password controls access to the enable commands.

nameif Command

The **nameif** command is used to label your interfaces and set the security levels for each of your interfaces. The first line sets the Ethernet0 interface to be called **outside** and to have a security level of zero. The next line labels the Ethernet1 interface as **inside** with a security level of 100. In other words, Ethernet0 is from now on called **outside** instead of Ethernet0 and is completely untrusted because it has a security level of zero. Ethernet1 is now called **inside** and is completely trusted. These are both the defaults and are necessary to the configuration. Ethernet0 is always **outside** and Ethernet1 is always **inside**. **outside** always has a security level of zero, and **inside** always has a security level of 100. Except for the inside and outside interfaces, an interface may be named anything you desire and will have a security level somewhere between 0 and 100. Remember that the higher a security level, the more it is trusted.

This is important because the default behavior of the PIX Firewall is relative to the security levels associated with the interfaces in question. Every interface has a higher security level than the outside interface. Therefore, by default, packets from any interface can travel through the outside interface. Conversely, no packets from the outside interface by default can travel to any other interface.

Suppose that your PIX had two additional interfaces, Ethernet2 and Ethernet3. You enter the following two lines:

```
nameif ethernet2 joe security16
nameif ethernet3 nancy security45
```

The **joe** interface (Ethernet2) has a security level of 16 and the **nancy** interface (Ethernet3) has a security level of 45. This is feasible because you can assign any security level to an interface and can call the interface anything you choose. In this scenario, packets from nancy could travel through the joe interface without any special configurations. Packets originating at joe cannot by default travel through the nancy interface because the nancy interface has a higher security level. The advanced configurations later in this chapter expand on this concept and use more realistic names for the interfaces.

interface Command

The lines starting with **interface** accomplish two tasks. The first task is to set the speed and type of the interface. If you set interface Ethernet0 to 100BaseT, you would use the following line:

```
interface Ethernet0 100baset
```

Alternatively, if you want to set the first Token Ring interface at 16 Mbps, you would enter:

```
interface tokenring0 16
```

The second accomplishment of this line is to turn up the interface. This is the equivalent of issuing a **no shut** command on a router. The **interface** command is also the exception to the rule that you can use **inside** or **outside** instead of **Ethernet0**. The actual hardware identification must be used with the **interface** command.

Assigning IP Addresses

The next two lines assign an IP address and subnet mask to the inside and outside interfaces. The words *inside* and *outside* are used because that is what you have named with the **nameif** command. Substitute whatever name you have given to this particular interface. The IP addresses on each interface must reside on different subnets.

The full **ip address** command follows:

```
ip address interface_name ip_address subnet_mask
```

global Command

One of the strengths of the PIX Firewall is its ability to support NAT and PAT. The **global** command, in conjunction with the **nat** command, is used to assign the IP addresses that packets receive as they cross the interface. The **global** command defines a pool of global addresses. This pool provides an IP address for each outbound connection and for inbound connections resulting from these outbound connections. Whether NAT or PAT is used depends on how the **global** command is used. If you are connecting to the Internet, the global addresses should be registered. Nonroutable IP addresses are used here for illustrative purposes only. Using routable IP addresses becomes a vital consideration when using VPNs that terminate on the PIX Firewall, because without a routable IP address the VPN will never travel over the Internet. The syntax for the **global** command follows:

```
global [(interface_name)] nat_id global_ip[-global_ip]
    [netmask global_netmask]
```

The *interface_name* is the name assigned with the **nameif** command. The *nat_id* is an integer. The *nat_id* must match the number used in the **nat** command. Although almost any number can be used (as long as the number is consistent between the **global** and **nat** commands), the number 0 is reserved for special cases. The use of 0 is covered in the section "**nat** Command."

The *global-ip* can take one of two forms. The form chosen determines whether NAT or PAT is used. If PAT is to be used, enter a single IP address. All packets from all hosts will receive this address as they cross the interface. If NAT is to be used, enter an address range for the IP addresses to be seen from the outside. For example, if you wish to use the single address of 192.10.10.1, you would enter the following:

```
global (outside) 1 192.10.10.1 255.255.255.0
```

If, on the other hand, you wish to use NAT and use a whole Class C subnet, you would enter the following:

```
global (outside) 1 192.10.10.1-192.10.10.254 255.255.255.0
```

You could also use more than a Class C network by adjusting the IP addresses entered and the subnet mask. The following example uses a 23-bit subnet mask and allows you to use all IP addresses between 192.10.10.1 and 192.10.11.254. When an address range overlaps subnets, the broadcast and network addresses are not used by the **global** command.

```
global (outside) 1 192.10.10.1-192.10.11.254 255.255.254.0
```

When you want to use PAT, you use a single address instead of a range. PAT supports up to 65,535 concurrent translations. There are some limitations in the use of PAT. For example, PAT cannot be used with H.323 and multimedia applications. These types of applications expect to be able to assign certain ports within the application. PAT also does not work in conjunction with the **established** command. Because the ports are changed when using PAT, these applications fail. As in the basic configuration, the following line sets a single IP address:

```
global (outside) 1 192.168.1.100 255.255.255.0
```

The use of the **global** command requires reverse DNS PTR entries to ensure that external network addresses are accessible through the PIX Firewall. Without these PTR entries, you will see slow or intermittent Internet connectivity and File Transfer Protocol (FTP) requests consistently failing. DNS servers on a higher security level needing updates from a name server on an outside interface must use the **static** command, which will be explained in the "Realistic Configuration" section.

The subnet mask should match the subnet mask on the network segment. Use the ranges of IP addresses to limit the hosts used, not the subnet mask. In more advanced configurations later in this chapter, you will see how to use NAT and PAT together and how to use multiple global ranges.

nat Command

The **nat** command is used in conjunction with the **global** command. The **nat** command specifies from which interface connections can originate. The syntax for the **nat** command follows:

```
nat [(interface_name)] nat_id local_ip [netmask [max_connections
[em_limit]]] [norandomsequence]
```

The *nat_id* number must be the same on the **nat** and **global** command statements. Although you might have multiple **global** commands associated with an interface, only a single **nat** command can be used. Use the **no** form of the **nat** command to remove the **nat** entry, or rewrite the **nat** command with the same *nat_id* to overwrite the existing **nat** command. After issuing a **nat** command, you should enter the **clear xlate** command. This command clears all present NAT and PAT connections, which are then reestablished with the new parameters. This section will deal with using the number 0 for the *nat_id* after you have seen the other parameters within the **nat** command and the discussion on using the **nat** command with access lists.

The *local_ip* parameter can be set to a single IP address or to a whole network by adjusting the *netmask* parameter. The *local_ip* parameter specifies the internal network address to be translated. Using 0.0.0.0 allows all hosts to start outbound connections. Instead of using 0.0.0.0, you can abbreviate by using simply 0.

Use the *netmask* parameter as you would use any subnet mask. The exception is when you use 0.0.0.0 as the *netmask*. Using 0.0.0.0 means that you want to allow all hosts on the local network through. This can be abbreviated as simply 0. When allowing all hosts through, you can use 0 for both the *local_ip* and the *netmask*. Within the PIX, 0 can be substituted for where the word *any* would be used on a Cisco router. The command line might look like any of the following lines, assuming that the local inside network is 10.1.1.0 with a Class C subnet mask:

```
nat (inside) 1 0 0 0 0
nat (inside) 1 10.1.1.0 255.255.255.0 0 0
nat (inside) 1 0 255.255.255.0 0 0
nat (inside) 1 10.1.1.0 0 0 0
```

The *max connections* parameter limits the number of concurrent TCP connections through an interface. Using 0 makes the number of connections limited only by the license agreement and software installed on the PIX Firewall.

Embryonic connections are half-open TCP connections. The default of 0 does not limit the number of embryonic connections. On slower systems, entering a number for *em_limit* ensures that the system does not become overwhelmed trying to deal with embryonic connections.

The **norandomsequence** keyword is used to disable the default random sequencing of TCP packet numbers. Although usually not added to the **nat** command, this can be useful for debugging and in certain other circumstances. For example, if traffic must travel through two PIX Firewalls, the dual randomization of sequence numbers might cause the application to fail. In this case, adding the **norandomsequence** keyword to one of the PIX Firewalls should resolve the problem.

There are some special considerations for using the **nat** and **global** commands with a *nat_id* of 0. The first consideration is when using an access list to prevent NAT from occurring. For example, the following lines allow the hosts at IP addresses 10.1.1.54 and 10.1.1.113 to traverse the PIX without changing their IP addresses. All other addresses on the inside network receive translation services. The access list associated with a **nat 0** command merely prevents NAT; it does not limit accessibility to the outside.

```
access-list prevent_nat tcp host 10.1.1.54
access-list prevent_nat tcp host 10.1.1.113
nat (inside) 0 access-list prevent_nat
```

The access list should not attempt to prevent specific ports, because this causes the addresses to become translated. The ASA remains in effect, watching packets and preventing unauthorized access. However, the addresses within the access list are available through the outer interface without translation.

The **nat 0** command can also be used without an access list as any other *nat_id* could be used. However, using a *nat_id* of 0 without an access list causes all hosts on the network specified with the *netmask* to avoid being translated by the NAT functionality of the PIX. Previous versions of the PIX software experienced an issue when using 0 as the *nat_id*. This issue was that using 0 would cause the PIX to use proxy Address Resolution Protocol (ARP) for all inside addresses. PIX IOS versions 5.0 and above disable this behavior. If no addresses are to be translated, the **global** command is not necessary. The following example shows how all inside addresses can be prevented from being translated:

```
nat (inside) 0 0 0 0 0
```

route Command

The **route** command is used by the PIX in the same manner that static routes and default routes are used on a router. The PIX has limited routing capabilities. It is necessary for you to specify routes. As in a router, the most specific route listed takes precedence. The syntax for the route command follows:

```
route interface_name ip_address netmask gateway_ip [metric]
```

The *interface_name* is any name previously defined by the **nameif** command. The *ip_address* is the address of the internal or external network. A default route can be set with either 0.0.0.0 or 0. The *netmask* is the subnet mask of the route. A default route can use either 0.0.0.0 or 0.

The *gateway_ip* is the IP address of the next hop for the network to which you are adding a route. For example, if your inside interface supported multiple networks connected with a router whose interface is 10.1.1.20, your route statements might appear as follows:

```
route inside 10.1.2.0 255.255.255.0 10.1.1.20 2
route inside 10.1.8.0 255.255.255.0 10.1.1.20 2
route inside 10.2.13.0 255.255.255.0 10.1.1.20 2
route inside 10.11.7.0 255.255.255.0 10.1.1.20 2
```

Version 5.1 has been improved to specify automatically the IP address of a PIX Firewall interface in the **route** command. Once you enter the IP address for each interface, the PIX creates a **route** statement entry that is not deleted when you use the **clear route** command. If the **route** command uses the IP address from one of the PIX's own interfaces as the gateway IP address, the PIX uses ARP for the destination IP address in the packet instead of issuing an ARP for the gateway IP address.

The *metric* parameter is used to specify the number of hops to *gateway_ip*, not to the ultimate destination of the IP packet. A default of 1 is assumed if this parameter is not used. If duplicate routes are entered with different metrics for the same gateway, the PIX changes the metric for that route and updates the metric for the route.

arp timeout Command

The **arp timeout** command is used to specify the time that an ARP entry remains in the ARP cache before it is flushed. The number shown is the time in seconds that an ARP entry remains in the cache. The default time is 14,400 seconds, or 4 hours. In the configuration, you change the default to 2 hours with the following:

```
arp timeout 7200
```

write Command

The **write** command works in the same way that the **write** command operates in a Cisco router. For those of you relatively new to Cisco equipment, this command has largely been replaced on routers with the **copy** command. The **write** command can take any of the following formats:

```
write net [[server_ip_address]:[filename]]
write erase
write floppy
write memory
write terminal
write standby
```

The **write net** command writes across a network to a Trivial File Transfer Protocol (TFTP) server with the filename specified. If no server IP address or filename is entered, the user is prompted.

The **write erase** command clears the Flash memory configuration. The **write floppy** command writes the configuration to the floppy disk, if the PIX has a floppy. The **write memory** command stores the configuration in RAM memory. The **write terminal** command shows the current configuration on the terminal. The **write standby** command is used to write the configuration to either a failover or standby, PIX'S RAM memory.

At this point, you have completed a basic configuration. You are ready to move toward a more realistic situation, such as a network with a mail server and an FTP server.

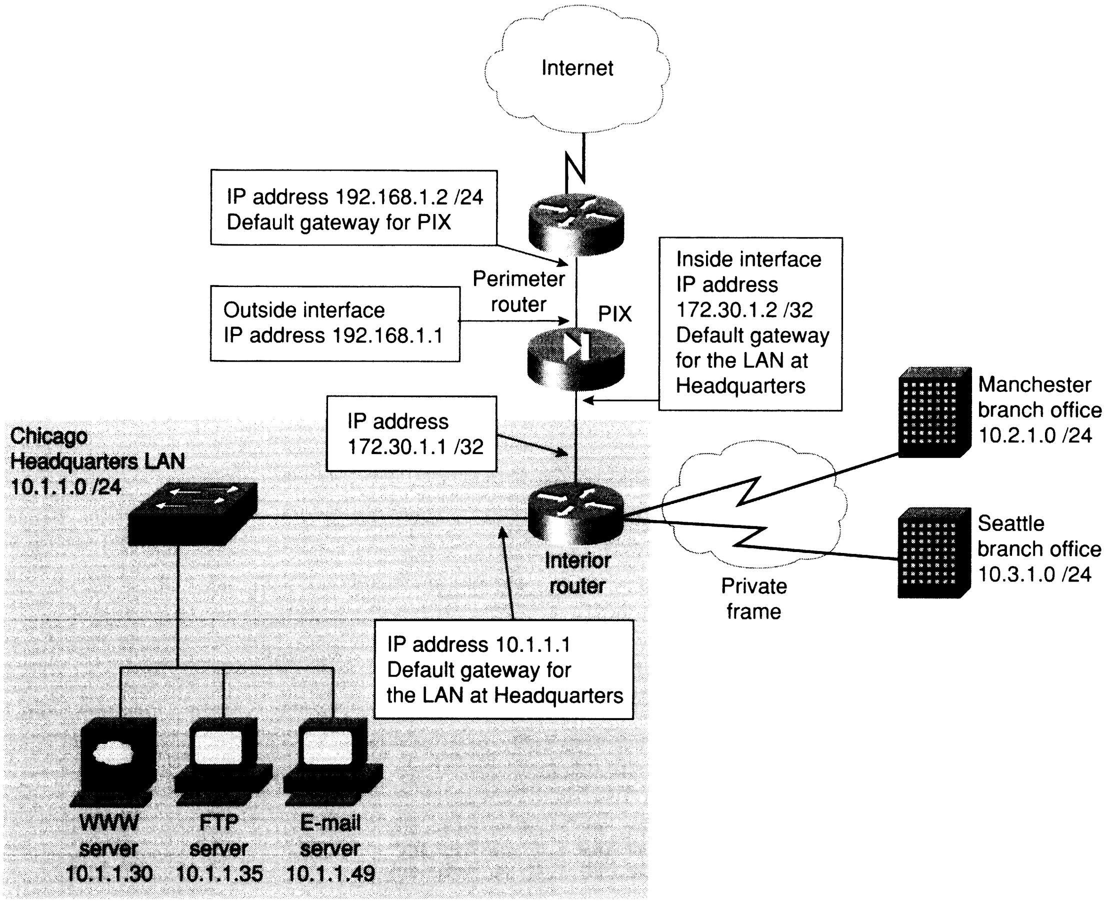
Realistic Configuration

Although the basic configuration suffices to illustrate how simple it is to configure the PIX, there are a few more items that almost all systems need. Three examples are Web services, e-mail services, and FTP services. This configuration will show how access from the outside to the inside of the PIX can be allowed.

The default configuration for the PIX Firewall is to prevent all access from an interface with a lower security level through an interface with a higher security level. The configuration in this section shows how access can be allowed without losing security protection on the whole network subnet, or even on the hosts that you allow to be seen from the outside.

Figure 4-6 shows the layout for this scenario. Note that the 192.168.1.0 /24 network has been used on the interfaces between the PIX and the perimeter router. In real life, these should be routable IP addresses, because you need people on the Internet to be able to browse your Web server, download files from your FTP server, and send and receive from your e-mail server.

Figure 4-6 *Realistic PIX Configuration*



As shown in Figure 4-6, the interior router and the inside interface of the PIX are on a separate network. This is not mandatory. However, if there is a spare Ethernet interface on the interior router and plans to use a **nat 0** command, using a spare interface on the inside router is advised, because the PIX will use ARP to a router for the address of each request. Repeated ARP requests can cause an excessive load on an overtaxed network. Connecting the PIX to a router's interface also ensures that all packets from and to the PIX are not delayed because of issues such as collisions and broadcast storms. Finally, the interior router can and should be configured with at least simple access lists to ensure that only authorized traffic is traversing the network. This might seem like too much trouble for some administrators. However, security should become a pervasive attitude throughout the network engineering staff. Having an extra layer of protection is never a waste of effort.

You now have three major design changes to make to your system. You must first allow WWW traffic to access the Web server, whose IP address is 10.1.1.30. This IP address needs to be statically translated to a routable address on the Internet. One of the easiest ways to keep track of static IP translations is to use the same last octet in both addresses. In the case of the Web server, you will use 30 as the last octet. The second change is to allow e-mail through to the mail server. The third change is to allow FTP traffic to the FTP server. All of these servers need a static translation because you cannot be guaranteed what host will be using a given outside IP address at any given time if you simply rely on the default NAT settings on the PIX and allow traffic into the LAN.

Issue a **write erase** command on the PIX. This erases the saved configuration. Turn the PIX power off and then back on to arrive at a clean state. Enter the following commands while in enable mode on the PIX. This section covers each change after the lines are entered. Again, the lines are separated for clarity.

```
enable password enablepass encrypted
passwd password encrypted

nameif ethernet0 outside security0
nameif ethernet1 inside security100

interface ethernet0 10baset
interface ethernet1 10baset

ip address outside 192.168.1.1 255.255.255.0
ip address inside 172.30.1.2 255.255.255.252

global (outside) 1 192.168.1.50-192.168.1.253 255.255.255.0
global (outside) 1 192.168.1.254 255.255.255.0
nat (inside) 1 10.1.1.0 255.255.255.0 0 0

static (inside, outside) 192.168.1.30 10.1.1.30 netmask 255.255.255.255 0 0
static (inside, outside) 192.168.1.35 10.1.1.35 netmask 255.255.255.255 0 0
static (inside, outside) 192.168.1.49 10.1.1.49 netmask 255.255.255.255 0 0

conduit permit tcp host 192.168.1.30 eq http any
conduit permit tcp host 192.168.1.35 eq ftp any
conduit permit tcp host 192.168.1.49 eq smtp any

route outside 0 0 192.168.1.2 1
route inside 10.1.1.0 255.255.255.0 172.30.1.1 1

arp timeout 7200

write mem
```

There are only a few changes from the basic configuration. You first changed the inside IP address to reflect the separate network between the PIX and the interior router. The two **global** commands shown next assign both NAT and PAT to be used by the inside hosts. Because you used a range of IP addresses, the first **global** command allows for each host on the LAN to get a dynamically assigned global address, or NAT. Once all of the available global IP addresses are in use, any hosts attempting to connect to the outside will use PAT. The second **global** line is critical because it assigns one address for use with PAT. If a single address is not reserved for use by PAT, hosts will simply not be able to get through the PIX.

The users will think that the Internet connection has been dropped, because they will receive no indication of a problem other than a lack of connection.

You might wonder why the range of IP addresses starts at 50 in the first **global** command. This allows servers to have static IP addresses. The number 50 was arbitrarily chosen. Whatever number is chosen ensures that there are sufficient reserved IP addresses for all servers on the network. You could have also reserved a set of IP addresses on the upper end of the network. The inside and outside routes were also changed to reflect the network as shown in Figure 4-6. You are now actually ready to allow users on the Internet to access your e-mail, FTP, and Web services.

Setting up to allow e-mail to traverse the PIX requires a few new commands. This replaces the **mailhost** command in previous versions of the PIX. These commands are covered later in this section. Enter the following lines into the PIX configuration.

```
static (inside, outside) 192.168.1.49 10.1.1.49 netmask 255.255.255.255 0 0
conduit permit tcp host 192.168.1.49 eq smtp any
```

That is all that is required to allow SMTP packets to traverse the PIX to the server with the 10.1.1.49 IP address. Users outside the PIX will see this server as 192.168.1.49. Packets sent to 192.168.1.49 will have NAT applied to them and will be forwarded to 10.1.1.49. Only the SMTP commands **HELLO, MAIL, RCPT, DATA, RSET, NOOP,** and **QUIT** are allowed through the PIX. The response to all other SMTP commands is an OK packet from the PIX. You added two new commands here, the **static** and the **conduit** commands. Each of them will be examined before moving on to the FTP and Web servers.

static Command

The **static** command is actually a very simple command once you are familiar with it. The purpose of the **static** command is to apply NAT to a single host with a predefined IP address. The syntax is as follows:

```
static [(internal_interface, external_interface)] global_ip local_ip [netmask
    subnet_mask] [max_connections [em_limit]] [norandomsequence]
```

The *internal_interface* and *external_interface* are names defined by the **nameif** command. The *global_ip* is the IP address seen on the outside, after NAT has been applied. The *local_ip* is the IP address used on the local host before NAT is applied. The *subnet_mask* should always be 255.255.255.0 when applied to a single host. If a network is being assigned to a single address, use the subnet mask for the network. For example, if you want the whole 10.1.4.0 network to be translated using PAT to 192.168.1.4, you use the following line:

```
static (inside, outside) 192.168.1.4 10.1.4.0 netmask 255.255.255.0 0 0
```

In this case, you also need to associate an access list with the **conduit** command. This will be covered under a more advanced configuration entitled "Dual DMZ with AAA Authentication" later in this chapter.

The *max_connections* and *em_limit* (embryonic limit) work in the same manner as with the **global** command. Using the **no** form of the command removes the **static** command. Using a **show static** command displays all of the statically translated addresses.

The **static** command is simple if you remember the order in which interface names and IP addresses appear. The order is:

```
static (high, low) low high
```

In other words, the name of the interface with the higher security level is shown first within the parenthesis, followed by the name of the lower security level interface and a closing parenthesis. This is followed by the IP address as seen on the lower security interface, then the IP address as seen on the higher security level interface. The authors remember this with the phrase "high, low, low, high." When you start looking at PIX Firewalls using one or more DMZs, the principle will hold true. Because every interface must have a unique security level, one interface must be more trusted than the other. You will still place the name of the interface with the higher security level first, followed by the less trusted interface name inside the parenthesis. Outside the parenthesis, you will show the IP address as seen on the lower security level interface, followed by the IP address as seen on the higher security level interface.

If you choose to use **nat 0** to avoid translating the IP address, you still use "high, low, low, high," but the IP addresses are the same for the global and local IP. The following is an example for when you do not use NAT on the IP address:

```
static (inside, outside) 10.1.1.49 10.1.1.49 netmask 255.255.255.255 0 0
```

conduit Command

The **conduit** command is necessary to allow packets to travel from a lower security level to a higher security level. The PIX Firewall allows packets from a higher security level to travel to a lower security level. However, only packets in response to requests initiated on the higher security level interface can travel back through from a lower security level interface. The **conduit** command changes this behavior. By issuing a **conduit** command, you are opening a hole through the PIX to the host that is specified for certain protocols from specified hosts.

The **conduit** command acts very much like adding a **permit** statement to an access list. The default behavior of the PIX is to act as if there were a deny all access list applied. Because you must allow e-mail to reach your server, you need to use the **conduit** command. The rule for access from a higher security level interface to a lower security level interface is to use the **nat** command. For access from a lower security level interface to a higher security level interface, use the **static** and **conduit** commands. As with any opening into the corporate network, this opening should be as narrow as possible. The following allows any host on the Internet to send mail to the host:

```
conduit permit tcp host 192.168.1.49 eq smtp any
```

If you wish to limit the originating IP address for e-mail, you could simply add an IP address and network mask to the end of the preceding line. You are allowed to have as many **conduit** statements as required. The following example allows SMTP traffic to enter the network from one of three networks—two with Class C subnets and the final one with a Class B subnet:

```
conduit permit tcp host 192.168.1.49 eq smtp 10.5.5.0 255.255.255.0
conduit permit tcp host 192.168.1.49 eq smtp 10.15.6.0 255.255.255.0
conduit permit tcp host 192.168.1.49 eq smtp 10.19.0.0 255.255.0.0
```

The combination of the **static** declaration and the **conduit** command can allow FTP traffic through your network. You have allowed FTP traffic to the FTP server with the following two lines:

```
static (inside, outside) 192.168.1.35 10.1.1.35 netmask 255.255.255.255 0 0
conduit permit tcp host 192.168.1.35 eq ftp any
```

It is possible to have multiple **conduit** commands associated with a single IP address. For example, the following lines allow SMTP, FTP, and HTTP services to gain access to a single server:

```
static (inside, outside) 192.168.1.35 10.1.1.35 netmask 255.255.255.255 0 0
conduit permit tcp host 192.168.1.35 eq ftp any
conduit permit tcp host 192.168.1.35 eq http any
conduit permit tcp host 192.168.1.35 eq smtp any
```

Notice that there is a single **static** statement for the host. Although some versions of the PIX IOS will allow you to enter multiple **static** commands for a single address, only the first **static** command is used. The PIX only allows the use of the host in the first **static** command. If you are using multiple **conduit** commands, you might deny some networks while allowing others. Alternatively, you might allow traffic from some networks, but not from others. In the following example, you deny FTP traffic from the 10.5.1.0 /24 network, while allowing traffic from all other networks:

```
static (inside, outside) 192.168.1.35 10.1.1.35 netmask 255.255.255.255 0 0
conduit deny tcp host 192.168.1.35 eq ftp 10.5.1.0 255.255.255.0
conduit permit tcp host 192.168.1.35 eq ftp any
```

Remote Site Configuration

At this point, you have a configuration that allows the main office to communicate through the Internet. You allowed access to the Web, FTP, and mail servers. What you do not have is access from the remote sites in Manchester and Seattle. The reason you do not have access is that the **nat** statement only applies to the Chicago LAN. You can easily add access to the Seattle and Manchester offices by adding the following lines:

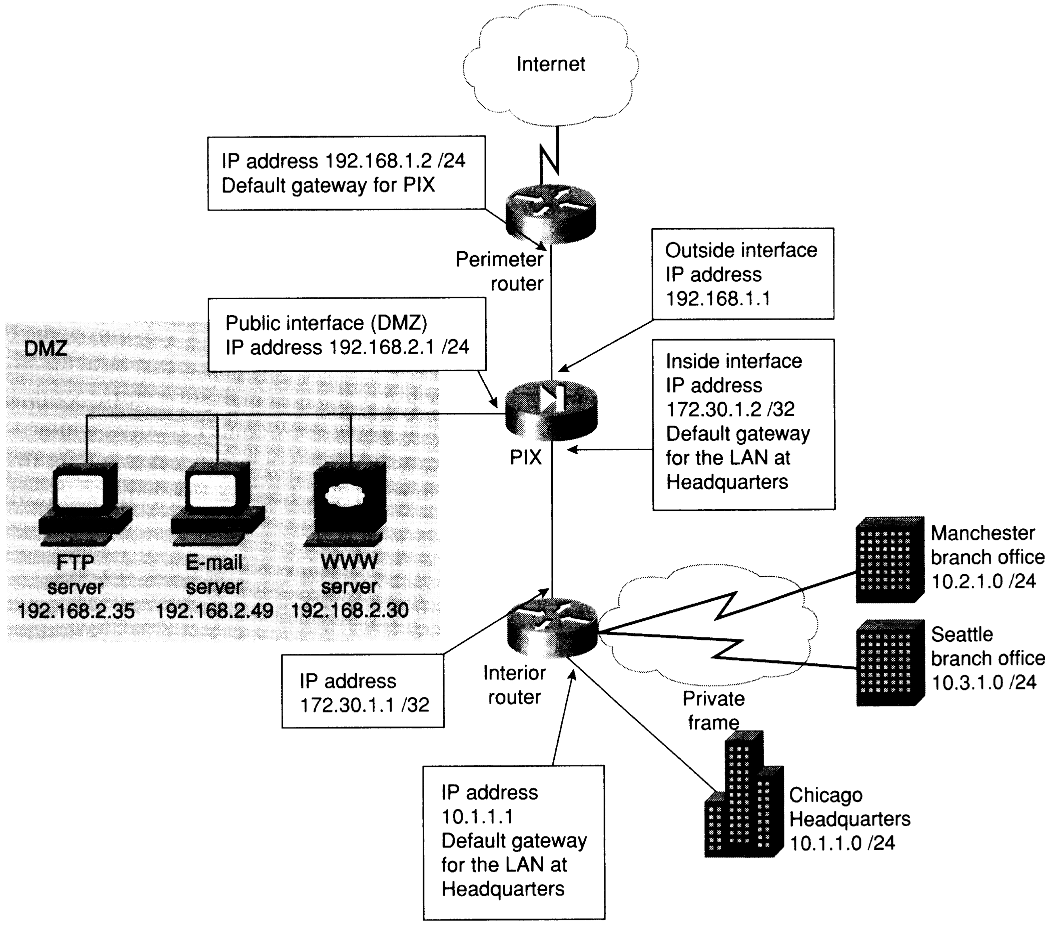
```
nat (inside) 1 10.2.1.0 255.255.255.0 0 0
nat (inside) 1 10.3.1.0 255.255.255.0 0 0
route inside 10.2.1.0 255.255.255.0 172.30.1.1 1
route inside 10.3.1.0 255.255.255.0 172.30.1.1 1
```

The next configuration adds a DMZ and allows configuration of the PIX through something other than the console. The configuration also enables SNMP, a syslog server, and filter URLs.

Single DMZ Configuration

This configuration moves the FTP, Web, and e-mail servers to a DMZ. All traffic destined for these servers will not touch the LAN. When using a DMZ, it is critical that no connection between the LAN and the DMZ be maintained except through the PIX Firewall. Connecting the LAN to the DMZ in any way except through the firewall defeats the purpose of the DMZ. Figure 4-7 shows that a third interface has been added to the PIX. This interface will be used as a DMZ.

Figure 4-7 *Single DMZ Configuration*



The configuration will need a few changes from the previous one. Look through the following configuration. This section will discuss where changes have been made and the ramifications of those changes after the configuration. As before, the blank lines are for clarity.

```
hostname pixfirewall

enable password enablepass encrypted
passwd password encrypted

nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 public security 50

interface ethernet0 auto
interface ethernet1 auto
interface ethernet2 auto

ip address outside 192.168.1.1 255.255.255.0
ip address inside 172.30.1.2 255.255.255.252
ip address public 192.168.2.1 255.255.255.0

fixup protocol http 80
fixup protocol http 10120
fixup protocol http 10121
fixup protocol http 10122
fixup protocol http 10123
fixup protocol http 10124
fixup protocol http 10125
fixup protocol ftp 21
fixup protocol ftp 10126
fixup protocol ftp 10127

snmp-server community ourbigcompany
snmp-server location Seattle
snmp-server contact Mark Newcomb Andrew Mason
snmp-server host inside 10.1.1.74
snmp-server enable traps

logging on
logging host 10.1.1.50
logging trap 7
logging facility 20
no logging console

telnet 10.1.1.14 255.255.255.255
telnet 10.1.1.19 255.255.255.255
telnet 10.1.1.212 255.255.255.255

url-server (inside) host 10.1.1.51 timeout 30
url-server (inside) host 10.1.1.52
filter url http 0 0 0 0

global (outside) 1 192.168.1.50-192.168.1.253 255.255.255.0
global (outside) 1 192.168.1.254 255.255.255.0
nat (inside) 1 10.1.1.0 255.255.255.0 0 0
nat (inside) 1 10.2.1.0 255.255.255.0 0 0
nat (inside) 1 10.3.1.0 255.255.255.0 0 0
nat (public) 1 192.168.2.1 255.255.255.0 0 0

static (public, outside) 192.168.1.30 192.168.2.30
static (public, outside) 192.168.1.35 192.168.2.35
static (public, outside) 192.168.1.49 192.168.2.49

conduit permit tcp host 192.168.1.30 eq http any
conduit permit tcp host 192.168.1.35 eq ftp any
conduit permit tcp host 192.168.1.49 eq smtp any
conduit permit tcp any eq sqlnet host 192.168.1.30
```

```
route outside 0 0 192.168.1.2 1
route inside 10.1.1.0 255.255.255.0 172.30.1.1 1
route inside 10.2.1.0 255.255.255.0 172.30.1.1 1
route inside 10.3.1.0 255.255.255.0 172.30.1.1 1
route public 192.168.2.0 255.255.255.0 192.168.2.1

arp timeout 7200

clear xlate
write mem
```

The **hostname** command has been added as the first line in this configuration. This merely identifies the host when you Telnet in for configuration.

You add a new interface, name it **public**, and assign a security level of 50 with the following line:

```
nameif ethernet2 public security 50
```

Because the security level of this interface is less than the inside and greater than the outside, some default behaviors come into play. By default, packets from the outside interface are not allowed into this network. Packets from the inside are, by default, allowed into this network.

You also changed the speeds for all of the interfaces. You are now using the keyword **auto** with the **interface** command. This allows the interface to connect in whatever form is most appropriate, based on the equipment to which it is connected. You added an IP address for the new network card and a subnet mask for the network.

fixup Command

Several **fixup** commands were entered. Some **fixup** commands appear in the configuration by default, others are added as needed. The **fixup protocol** commands allow changing, enabling, and disabling the use of a service or protocol through the PIX Firewall. The ports specified for each service are listened to by the PIX Firewall. The **fixup protocol** command causes the ASA to work on port numbers other than the defaults. The following **fixup protocol** commands are enabled by default:

```
fixup protocol ftp 21
fixup protocol http 80
fixup protocol smtp 25
fixup protocol h323 1720
fixup protocol rsh 514
fixup protocol sqlnet 1521
```

You added the following lines regarding the HTTP protocol:

```
fixup protocol http 10120
fixup protocol http 10121
fixup protocol http 10122
fixup protocol http 10123
fixup protocol http 10124
fixup protocol http 10125
```

These lines accomplish a very specific task. When HTTP traffic is seen by the PIX, it can now be on any of the previously listed ports. Before these lines were entered, the PIX would have seen what looked like HTTP traffic entering the PIX. Because the destination port was set to something other than the default of 80, that traffic would be denied. For example, if an outside user tried to connect to the Web server with the following URL, the user would be denied:

http://www.ourcompany.com:10121

The reason for the denial is that the **:10121** at the end of the URL specifies that the connection should be made on port 10121, rather than on the default port of 80. The Web developers have specific reasons for wanting to allow users to connect to these ports. The configuration allows the users to connect with these ports, and you still maintain the same safeguards regarding HTTP traffic that is true for port 80.

Similarly, the developers have specific reasons for wanting to change the defaults. The developers decided that users requiring FTP access should be able to gain access through the default port of 21 or ports 10126 and 10127. You have no idea why they want to do this, nor do you really care. What you care about is that you can open these ports to FTP traffic, and only FTP traffic, without compromising the network security. To accomplish this, you add the following lines:

```
fixup protocol ftp 21
fixup protocol ftp 10126
fixup protocol ftp 10127
```

It should be noted that the **fixup protocol** command is global in nature. For example, when you told the PIX that port 10121 was part of the HTTP protocol, this applied to all interfaces. You cannot selectively cause port 10121 to be regarded as HTTP traffic on one interface, but not on another interface.

There might be times when it is necessary to disable one of the default **fixup protocol** commands. For example, if your company develops e-mail software and the PIX is used to separate the test network from the corporate network. In this case, you might want to allow more commands than **HELLO**, **MAIL**, **RCPT**, **DATA**, **RSET**, **NOOP**, and **QUIT** to travel through the PIX. In this case, using the **no** form of the **fixup protocol** command will disable the feature. An example of removing the Mailguard feature is as follows:

```
no fixup protocol smtp 25
```

SNMP Commands

You add SNMP to the PIX because you want to be informed when errors occur. You can browse the System and Interface groups of MIB-II. All SNMP values within the PIX Firewall are read-only (RO) and do not support browsing (SNMPget or SNMPwalk) of the Cisco syslog Management Information Base (MIB). Traps are sent to the SNMP server. In other words, SNMP can be used to monitor the PIX but not for configuring the PIX. The syntax for the commands is essentially the same as when working on a Cisco router. The following lines set the community string, the location, the contact, and the interface and IP address of the SNMP server. Because you have specified **inside** on the **snmp-server host** command, the PIX knows which interface to send SNMP traps out without the need for a specific route to this host.

```
snmp-server community ourbigcompany
snmp-server location Seattle
snmp-server contact Mark Newcomb Andrew Mason
snmp-server host inside 10.1.1.74
```

logging Commands

The following **logging** commands allow you to use a syslog server for recording events. These commands are similar to those used on a Cisco router. The **logging on** command is used to specify that logging will occur. The **logging host** command is what actually starts the logging process on the host at 10.1.1.50. The **logging trap** command sets the level of logging to be recorded, which is all events with a level of 7. Finally, the **no logging console** command is used to prevent the log messages from appearing on the console. For this to work, the PIX must know how to find the host at 10.1.1.50. Ensure that a route to this host exists.

```
logging on
logging host 10.1.1.50
logging trap 7
logging facility 20
no logging console
```

telnet Command

You added three lines to allow access to the PIX Firewall through Telnet in addition to the console port access. This is a major convenience and a major security risk. There are three reasons that we consider Telnet access a risk. The first is that Telnet limits access based on the IP address. It is very easy for a user to change the IP address on a computer, especially if the user is using an operating system such as Windows 95. This allows the possibility of a user gaining access where the user should not be able to gain access. The second concern regarding security is that, as hard as you may try to prevent it, you cannot always be sure that a user walking away from a desk will lock the terminal. Password-protected screensavers help minimize the issue, but they cannot completely resolve it. Because the PIX forms the corporation's major defense from outside intrusion, it is critical that access is limited as much as possible. The third concern regarding Telnet access is a misunderstanding on how it should be configured. This third issue is covered in this section, after examining the commands entered.

```
telnet 10.1.1.14 255.255.255.255
telnet 10.1.1.19 255.255.255.255
telnet 10.1.1.212 255.255.255.255
```

In the preceding lines, you specified a subnet mask of 32 bits for each of these IP addresses. Entering **255.255.255.255** is optional, because an IP address without a subnet mask is assumed to have a 32-bit mask associated with that address. The subnet mask used on the **telnet** command is the mask for those who should have access to the PIX, not the subnet mask for the network. Approximately 50 percent of the PIX Firewalls the authors of this book have examined have been incorrectly configured with the subnet mask of the LAN. In these cases, any user on the LAN can Telnet to the PIX Firewall. If one of these users is able to guess the password, the user can control the PIX. In the configuration section "Dual DMZ with AAA Authentication" later in this chapter, you will see how to use authentication, authorization, and accounting (AAA) services to ensure that unauthorized users cannot Telnet to the PIX Firewall.

URL Filtering

You added URL filtering for monitoring, reporting, and restricting URL access. Cisco Systems and Websense, Inc. have formed a partnership for joint marketing and coordination of technical information on a product called Websense, which is used to control the sites that users are allowed to access. For example, web sites classified as employment or violent can be blocked. Instructions on ordering Websense are included in the documentation of every PIX Firewall.

The PIX Firewall configuration for enabling URL filtering is very simple. The following three lines show the configuration. The first line tells the PIX to allow or block URL access based on the information received from the Websense server on the inside interface at the 10.1.1.51 IP address. Should a response to a request not be received within the timeout parameter of 30 seconds shown on this line, the next Websense server will be queried. The default timeout is 5 seconds. The second line shows the failover Websense server, which is also the Web server on the public interface. The third line defines that all HTTP requests will be watched. Multiple filter commands can be combined to refine what is monitored. The full syntax of the **filter** command will be shown after the command lines.

```
url-server (inside) host 10.1.1.51 timeout 30
url-server (public) host 192.168.2.30
filter url http 0 0 0 0
```

The full syntax of the **filter** command is as follows:

```
filter [activex http url] |except local_ip local_mask foreign_ip
    foreign_mask [allow]
```

The definitions of the parameters can be found in Table 4-1.

Table 4-1 **filter** *Command Parameters*

Command	Description
activex	Blocks outbound ActiveX, Java applets, and other HTML object tags from outbound packets.
url	Filters URL data from moving through the PIX.
http	Filters HTTP URLs.
except	Creates an exception to a previously stated filter condition.
local_ip	The IP address before NAT (if any) is applied. Use 0 for all IP addresses.
local_mask	The subnet mask of the local IP. Use 0 if 0 is used for the IP address.
foreign_ip	The IP address of the lower security level host or network. Use 0 for all foreign IP addresses.
foreign_mask	The subnet mask of the foreign IP. Use 0 if the foreign IP is 0.
allow	When a server is unavailable, this lets outbound connections pass through the PIX without filtering.

Additional Single-DMZ Configuration Considerations

The remaining changes to this configuration involve commands that were previously examined in this chapter. You added a new **nat** statement with the interface set as **public** to allow for translation of the public DMZ to global addresses. This eliminates the chance that anyone from the outside will see any traffic on the inside network. You can use NAT on all of the public hosts and add them to the common global pool. The command used is as follows:

```
nat (public) 1 192.168.2.1 255.255.255.0 0 0
```

Next, you change the static NAT for the Web, FTP, and e-mail servers from the inside interface to the public interface. The new lines read:

```
static (public, outside) 192.168.1.30 192.168.2.30
static (public, outside) 192.168.1.35 192.168.2.35
static (public, outside) 192.168.1.49 192.168.2.49
```

If you were using the previous configuration, you would have needed to remove the old static translations using the **no** form of the **static** command. You also added a new **conduit** statement. This statement allows any Oracle database traffic from the Web server on the public interface to enter into your inside LAN. The PIX Firewall uses port 1521 for SQL*Net. This is also the default port used by Oracle for SQL*Net, despite the fact that this value does not agree with Internet Assigned Numbers Authority (IANA) port assignments.

Because the Web server has a database running in the background, you need to allow traffic from this Web server to enter into the LAN and talk to the Oracle database servers. These tasks are accomplished with the following lines:

```
conduit permit tcp host 192.168.1.30 eq http any
conduit permit tcp host 192.168.1.35 eq ftp any
conduit permit tcp host 192.168.1.49 eq smtp any
conduit permit tcp any eq sqlnet host 192.168.1.30
```

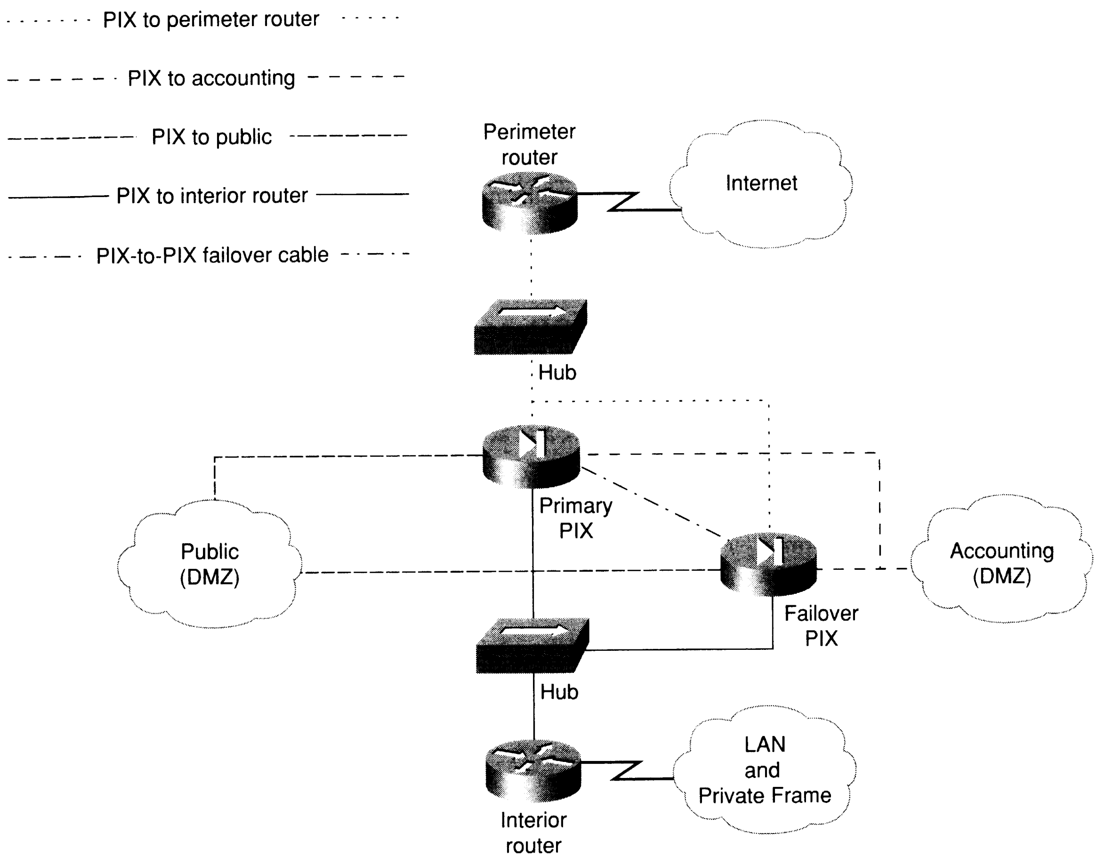
You also added a few new **route** statements. You added routes for both the Seattle and Manchester networks as well as the public network. Finally, you made sure that the NAT changes would occur by issuing a **clear xlate** command and then writing the configuration.

Dual DMZ with AAA Authentication

This section introduces AAA authorization and creates two DMZs. Chapter 10 deals extensively with AAA. This section focuses on the PIX configuration aspects of AAA. This section also introduces a failover PIX and access lists into this configuration.

Figure 4-8 shows how this network is configured. Notice that there are two PIX Firewalls, a primary and a failover. Should the primary PIX fail, the failover PIX takes over all of the duties of the primary PIX. You also have two DMZs, the public and the accounting DMZs. The accounting DMZ is used for clients on the Internet to access the accounting data for the services.

Figure 4-8 *Dual DMZ Configuration*



Although there is a failover cable that connects the serial ports on the firewalls, you also added a hub on the inside interfaces to allow connectivity between the firewalls and the interior router in order to save interfaces on the interior router. You did the same between the outside interfaces of the firewalls and the exterior router. Both PIX Firewalls must have connectivity to both DMZs for the failover PIX to operate correctly, should the primary fail.

The configuration of the primary PIX follows. This section discusses the changes made to this configuration after the listing. The blank lines were added for clarity.

```
hostname pixfirewall

enable password enablepass encrypted
passwd password encrypted

nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 public security 50
nameif ethernet3 accounting security 60
```

```
interface ethernet0 auto
interface ethernet1 auto
interface ethernet2 auto
interface ethernet3 auto

ip address outside 192.168.1.1 255.255.255.0
ip address inside 172.30.1.2 255.255.255.248
ip address public 192.168.2.1 255.255.255.0
ip address accounting 10.200.200.1 255.255.255.0

fixup protocol http 80
fixup protocol http 10120
fixup protocol http 10121
fixup protocol http 10122
fixup protocol http 10123
fixup protocol http 10124
fixup protocol http 10125
fixup protocol ftp 21
fixup protocol ftp 10126
fixup protocol ftp 10127

failover active
failover link failover

no rip inside passive
no rip outside passive
no rip public passive
no rip accounting passive
no rip inside default
no rip outside default
no rip public default
no rip accounting default

pager lines 24

aaa-server TACACS+ (inside) host 10.1.1.41 thekey timeout 20
aaa authentication include any outbound 0 0 0 0 TACACS+
aaa authorization include any outbound 0 0 0 0 TACACS+
aaa accounting include any outbound 0 0 0 0 TACACS+
aaa authentication serial console TACACS+

snmp-server community ourbigcompany
snmp-server location Seattle
snmp-server contact Mark Newcomb Andrew Mason
snmp-server host inside 10.1.1.74
snmp-server enable traps

logging on
logging host 10.1.1.50
logging trap 7
logging facility 20
no logging console

outbound limit_acctg deny 10.200.200.0 255.255.255.0
outbound limit_acctg except 10.10.1.51
outbound limit_acctg permit 10.200.200.66
outbound limit_acctg permit 10.200.200.67
apply (accounting) limit_acctg outgoing_dest

access-list acct_pub permit host 10.200.200.52
access-list acct_pub deny 10.200.200.0 255.255.255.0
access-group acct_pub in interface public
```

```
telnet 10.1.1.14 255.255.255.255
telnet 10.1.1.19 255.255.255.255
telnet 10.1.1.212 255.255.255.255

url-server (inside) host 10.1.1.51 timeout 30
url-server (inside) host 10.1.1.52
filter url http 0 0 0 0

global (outside) 1 192.168.1.50-192.168.1.253 255.255.255.0
global (outside) 1 192.168.1.254 255.255.255.0
nat (inside) 1 10.1.1.0 255.255.255.0 0 0
nat (inside) 1 10.2.1.0 255.255.255.0 0 0
nat (inside) 1 10.3.1.0 255.255.255.0 0 0
nat (public) 1 192.168.2.1 255.255.255.0 0 0
nat (accounting) 0 0 0

static (public, outside) 192.168.1.30 192.168.2.30
static (public, outside) 192.168.1.35 192.168.2.35
static (public, outside) 192.168.1.49 192.168.2.49

conduit permit tcp host 192.168.1.30 eq http any
conduit permit tcp host 192.168.1.35 eq ftp any
conduit permit tcp host 192.168.1.49 eq smtp any
conduit permit tcp any eq sqlnet host 192.168.1.30

route outside 0 0 192.168.1.2 1
route inside 10.1.1.0 255.255.255.0 172.30.1.1 1
route inside 10.2.1.0 255.255.255.0 172.30.1.1 1
route inside 10.3.1.0 255.255.255.0 172.30.1.1 1
route public 192.168.2.0 255.255.255.0 192.168.2.1
route accounting 10.200.200.0 255.255.255.0 10.200.200.1 1

arp timeout 7200

mtu inside 1500
mtu outside 1500
mtu public 1500
mtu accounting 1500

clear xlate
write mem
write standby
```

The first change made to this configuration is the added **nameif** command for the accounting DMZ, assigning a security level of 60. The next change is that you enabled this interface with the **interface** command. You then assigned an IP address to the interface. Next, you configured the failover parameters.

failover Commands

The **failover** commands are relatively simple to use. Before discussing the commands, this section takes a few moments and discusses the requirements for a failover PIX, how the primary and secondary PIX are connected, and how the failover PIX is configured.

When purchasing a PIX, consider purchasing a failover PIX at the same time. When both are purchased together, there is a significant price reduction on the failover unit. Because the PIX is generally used as the primary device protecting your network, it usually makes sense from both service and fiscal points of view to make this a redundant system.

For a PIX to failover to another PIX after failure, both firewalls must have identical hardware and identical software versions. There is a proprietary cable made specifically for connecting between PIX Firewalls. On the back of each PIX is a port labeled*failover*. The cable ends are labeled *primary* and *secondary*. Once the primary PIX is configured, turn the secondary PIX's power off. Connect the cable, and restore power to the secondary PIX. After a few seconds, the secondary PIX acquires a copy of the configuration on the primary PIX. Should the primary PIX fail, the secondary PIX starts establishing connections. However, any connections that exist when the primary PIX fails are dropped and need to be reestablished. After the secondary PIX is powered on with the failover cable connected, changes should only be made to the primary PIX. One limitation of the failover system on the PIX is the length of the failover cable. The length of the cable cannot be extended, and the cable is required to be used. Therefore, you cannot use a primary PIX in one physical location and the secondary PIX in another location.

The first command used is the **failover active** command. This command, like all commands, should only be entered on the primary PIX. This command establishes that failover is configured and that the present PIX is the primary PIX. Using the **no** form of this command forces the current PIX to become the secondary PIX.

The second command shown is the **failover link** command. You have specified that the port used for the failover is the failover port. There is one more command used regarding failover. This command, **write standby**, is shown at the bottom of the configuration. The **write standby** command should be used after each time the configuration is changed. This causes the secondary PIX to receive a copy of the current configuration.

Understanding Failover

The failover features of the PIX are similar to those used with the Hot Standby Router Protocol (HSRP) in that the standby device remains inactive until the primary device fails. The standby device, on activation, assumes the IP and Media Access Control (MAC) address of the primary unit. Likewise, the previously active device assumes the IP and MAC addresses of the formerly standby device. Because network devices do not see any change in these addresses, no new ARP entries need to be made on the hosts using the PIX Firewall.

Starting with the PIX IOS 5.0 software release, stateful failovers are supported. Prior to this release, the PIX did not maintain a copy of the connection state in the standby unit. When the primary device failed, network traffic needed to reestablish previous connections. Stateful failovers overcome this issue by passing data about the state of connections between the primary and the standby devices within *state update packets*. A single packet traversing the PIX can establish a new connection state. Because each connection state changes on a per-packet basis, every packet received by the currently active device requires a state update packet to be relayed to the inactive device. Although this process works very well, there are some latency-sensitive applications that will time out before the failover process is completed. In these cases, a new session will need to be established.

IP states are supported, as are TCP states, except those using HTTP. Almost no UDP state tables are transferred between the active and standby devices. Exceptions to this include dynamically opened ports used with multichannel protocols, such as H.323. Because DNS resolves use a single channel port, the state of DNS requests is not transferred between devices.

A dedicated LAN interface between the two PIX devices is required to achieve stateful failover. State update packets are transmitted asynchronously in the background from the active device to the standby device over the dedicated LAN interface. There are also a few configurations changes required when using stateful failover. These changes are covered in the section "Stateful Failover Configuration."

Several criteria are considered before a failover occurs. If the standby device detects that the active device is powered down, the standby device will take active control. If the failover cable is unplugged, a syslog entry is generated, but both devices maintain their present state. An exception to this is during the boot process. Should the failover cable be unplugged while the devices are booting, both devices will assume the same IP address, causing a conflict on your network. Even if you are configuring the PIX Firewalls for stateful failover using a dedicated LAN interface, the failover cable must be installed on both devices for failover to function properly.

Failover hello packets are expected on each interface every 15 seconds. When the standby device does not receive a failover hello packet within 30 seconds, the interface runs a series of tests to establish the state of the active device. If these tests do not reveal that the active device is present, the standby device assumes the active role.

A power failure on the active device is detected through the failover cable within 15 seconds. In this case, the standby device assumes the active role. A disconnected or damaged failover cable is detected within 15 seconds.

Stateful Failover Configuration

Only a few commands need to be added to a configuration to enable stateful failover. The following is a partial configuration, showing the commands necessary to enable stateful failover. After the configuration, the commands are discussed.

```
nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 failover security 60

ip address outside 192.168.1.1 255.255.255.0
ip address inside 172.30.1.1 255.255.255.0
ip address failover 10.200.200.1 255.255.255.0

failover active
failover ip address outside 192.168.1.2
failover ip address inside 172.30.1.2
failover ip address failover 10.200.200.2
failover link failover
```

Notice that the interfaces are named *failover,* and a security level is assigned to the interface with the **nameif** command. You could have named this interface anything, but for clarity, it is named failover here. This is the interface you will be using to transfer state update packets between the active and the standby devices.

After assigning IP addresses and netmasks to each of the interfaces, you are ready to start on the failover commands. Start failover with the **failover active** command. Next, use the **failover ip address** command on all of the interfaces.

When using the **failover ip address** command, you need to remember two things. First, every interface needs the **failover ip address** command entered for that interface. If an interface does not have an associated **failover ip address** command and the state of that interface is changed to down, failover will not occur. For example, if you did not add the **failover ip address** command for the outside interface and the cable connecting that interface broke, all data intended to travel through that interface will be lost. This defeats the purpose of having a failover device, because a failover device should allow all services to continue after the primary device has failed. Additionally, because both devices must have the same hardware installed, there is no reason not to enable failover to check all interfaces. The second item that you need to remember is that the **failover ip address** needs to be on the same subnet but with a different IP address than that to which the interface is set.

The final configuration required is to assign a dedicated interface to failover. Using the **failover link** command with the interface name assigned by the **nameif** command, Ethernet2 has been assigned as the failover interface in this example.

rip Commands

You added commands to disable RIP on all interfaces. Notice that each interface has two lines associated with that interface: a **no rip** *interface_name* **passive** and a **no rip** *interface_name* **default** command. Each one of these commands accomplishes a different objective. The **no rip** *interface_name* **passive** command causes the PIX to stop listening to RIP updates. The **no rip** *interface_name* **default** command causes the PIX to stop broadcasting known routes through RIP.

RIPv1 and RIPv2 are both available on the PIX through the **rip** command. Use the **no** form of the **rip** command to disable a portion of RIP. Use the **show rip** command to show the current RIP entries and the **clear rip** command to clear RIP tables. The full syntax of this command is:

```
rip interface_name default | passive [version [1 | 2]]
    [authentication [text | md5 key (key_id)]]
```

The parameters and keyword meanings are listed in Table 4-2.

Table 4-2 **rip** *Command Parameters*

Command	Description
interface_name	The interface to which this command should be applied.
default	Broadcasts a default route on the interface.
passive	Enables passive RIP (listening mode) and propagates the RIP tables based on these updates.
version	RIP version 1 or 2. Version 2 must be used if encryption is required.
authentication	Enables RIP version 2 authentication.
text	Sends RIP updates as clear text. This is not a recommended option.
md5	Sends RIP update packets using MD5 encryption. Version 2 only.
key	This is the key used to encrypt RIP updates for version 2.
key_id	The key identification value. Both sides must use the same key. Version 2 only.

pager lines Command

The **pager lines** command specifies how many lines are shown when a **show config** command is issued before a **more** prompt appears. Although this can be set to almost any value, 24 works well when using standard Telnet applications.

AAA Commands

You have enabled AAA using Terminal Access Controller Access Control System Plus (TACACS+) on your PIX for authenticating, authorizing, and accounting for users passing from the inside through the outside interface. You have also enabled TACACS+ authentication for those connecting to the PIX through the console.

The first command you need to look at is the **aaa-server** command. The example sets the server to TACACS+ on the inside interface with the IP address of 10.1.1.41. You are using **thekey** as your TACACS+ key and have set a timeout of 20 seconds. This command is also responsible for starting AAA on the PIX. The full syntax of the **aaa-server** command follows:

```
aaa-server group_tag (interface_name) host server_ip key timeout seconds
```

The parameters and keywords, along with their descriptions, are displayed in Table 4-3.

Table 4-3 **aaa-server** *Command Parameters*

Command	Description
group_tag	TACACS+ or RADIUS.
interface_name	Name of the interface where the server resides.
host	Keyword designating that a single host IP address follows.
server_ip	The IP address of the server.
key	The alphanumeric key expected at the server.
timeout	Keyword designating that the parameter following is the number of seconds.
seconds	The wait time in seconds that the PIX will wait after sending a request without receiving a response before another request is sent. The default time is 5 seconds. Four requests will be sent before timing out.

After starting AAA, you authenticated, authorized, and accounted for any outbound traffic. For a full description of these three processes, see Chapter 10. For the moment, it will suffice to say that when users attempt to send data outside, first they will be checked to ensure that they are who they claim to be, then a check will determine whether they are allowed to send the data outside, and then a record will be made that the users sent the data. You accomplish these three tasks in this example with the following three lines:

```
aaa authentication include any outbound 0 0 0 0 TACACS+
aaa authorization include any outbound 0 0 0 0 TACACS+
aaa accounting include any outbound 0 0 0 0 TACACS+
```

The key here is the word **outbound**, which means packets traversing from the inside interface through the outside interface. The **any** in these lines refers to the type of accounting service; possible values are **any**, **ftp**, **http**, **telnet**, or **protocol/port**. The four zeros refer, in order, to the local address, the local mask, the foreign IP address, and the foreign mask. The final parameter determines which service should be used, RADIUS or TACACS+. It is possible to run both TACACS+ and RADIUS at the same time. To accomplish this, merely add another **aaa-server** command with the other service.

The **aaa authentication** command has another form that allows you to authenticate connections for the serial port, the Telnet ports, and the enable mode. The full syntax of this command follows:

```
aaa authentication [serial | enable | telnet] console group_tag
```

outbound and **apply** Commands

Now that you have seen how AAA can limit outbound access through an interface, there is another way to control and limit access from a higher security level interface to a lower security level interface. This method uses PIX access lists configured with the **outbound** and **apply** commands. The first thing to remember about this type of PIX access list is that it operates in a totally different manner than a router's access list. If you are intimately familiar with router access lists, you might have a harder time accepting how PIX access lists work than those who are not so familiar with router access lists. The order of a router's access list is vitally important, because the first match will cause a rejection or acceptance. However, the PIX uses a best-fit mechanism for its access lists. This allows the administrator to deny whole ranges of IP addresses and then allow specific hosts through at a later date without having to rewrite the whole access list. The PIX access list is also neither a standard nor an extended access list, but rather a combination of the two forms.

Where a router uses two commands, **access-list** and **access-group** (or **access-class**), to define and apply an access list, the PIX uses the **outbound** and **apply** commands to define and apply an access list.

The full syntax of the **outbound** command follows:

```
outbound list_id permit | deny ip_address [netmask [java | port[-port]]]
    [protocol]
```

A description of the command parameters can be found in Table 4-4.

Table 4-4 **outbound** *Command Parameters*

Command	Description
list_id	This is an arbitrary name or number used to identify the access list. This is similar to a named access list on a router.
permit	Allows the access list to access the specified IP address and port.
deny	Denies access to the specified IP address and port.
except	Creates an exception to the previous **outbound** command. The IP address associated with an **except** statement changes depending on whether an **outgoing_src** or **outgoing_dest** parameter is used in the **apply** command. If the **apply** command uses **outgoing_src**, the IP address applies to the destination IP address. If the **apply** command uses an **outgoing_dest**, the IP address refers to the source IP address.
ip_address	The IP address associated with the **outbound permit**, **outbound deny**, or **outbound except** command.
netmask	The subnet mask associated with the IP address. Remember that this is a subnet mask, not a wildcard mask as used on routers. Where a router would have a wildcard mask of 0.0.0.255, the PIX would have a subnet mask of 255.255.255.0.
port	The port or range of ports associated with this command.
java	The keyword **java** is used to indicate port 80. When **java** is used with a **deny**, the PIX blocks Java applets from being downloaded from the IP address. By default, the PIX permits Java applets.
protocol	This limits access to one of the following protocols: UDP, TCP, or ICMP. TCP is assumed if no protocol is entered.

Now that you know how the command works, look at the effects of the commands. The first two lines of the configuration regarding access lists read:

```
outbound limit_acctg deny 10.200.200.0 255.255.255.0
outbound limit_acctg except 10.10.1.51
```

The first **outbound** command denies all packets from the Class C network at 10.1.1.0. When using the **deny** and **permit** forms of the **outbound** command, you are referring to the destination IP address. You could use the word **permit** in the example instead of **deny**, which would allow packets from these IP addresses. The effects of the second line cannot be fully determined until you look at the **apply** command. However, you can still see that an exception to the previous **deny** command exists. This exception allows packets associated with the IP address of 10.10.1.51 through the PIX. Here the word *associated* is used instead of *destination* or *source* because whether you are concerned about the source or the destination IP address is actually determined by the **apply** command. If the **apply** command specifies a source IP address, the packets from the source used with the **outbound** command are permitted or denied. If the **apply** command specifies a destination address, then packets whose destination address matches the IP address used with the **outbound** command are denied or permitted.

This is a two-step process that requires the administrator to ask two questions. First, look at the **outbound** command. Is this a permit or deny statement? Next, look at the **apply** command. Is the **apply** command concerned with the source or the destination address?

The next two lines are easy to understand. You permit access to the hosts at 10.200.200.66 and 10.200.200.67. At this point, you still do not have a definition as to whether the IP address associated with the **except** is a source or destination address. However, the **apply** command will resolve this outstanding issue. For review purposes, the two lines follow:

```
outbound limit_acctg permit 10.200.200.66
outbound limit_acctg permit 10.200.200.67
```

The **apply** statement is used to connect an access list with an interface and to define whether IP addresses specified with that access list are source or destination IP addresses. This example of the **apply** command follows:

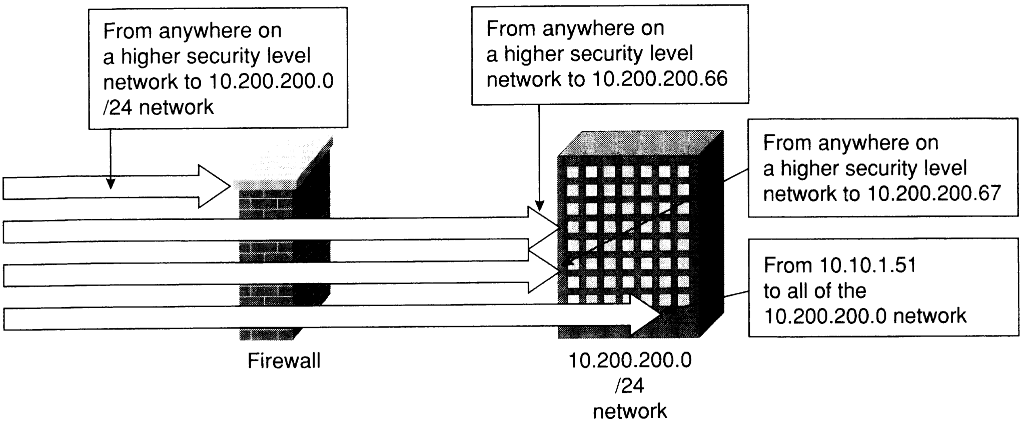
```
apply (accounting) limit_acctg outgoing_dest
```

In this example, you applied an access list to the interface previously defined as **accounting** by the **nameif** command. The access list you connected is the one called **limit_acctg**. As with a router's access lists, only one access list can be applied in a given direction on any PIX interface.

This **apply** command has applied the **except** command to source packets. The alternative would be to apply the **except** command to destination packets by using the **outgoing_src** parameter. The application of this command has a distinct effect on the access list. This effect is that the IP address specified by the **except** command is a source address.

For review purposes, look at Figure 4-9. Refer to Figure 4-9 while reviewing the following discussion about the command lines used.

Figure 4-9 *PIX* **outbound** *Command Example*


From anywhere on a higher security level network to 10.200.200.0 /24 network
From anywhere on a higher security level network to 10.200.200.66
From anywhere on a higher security level network to 10.200.200.67
From 10.10.1.51 to all of the 10.200.200.0 network
Firewall
10.200.200.0 /24 network

The following line prevents access to all of the 10.200.200.0/24 network from all hosts for all protocols. The PIX uses subnet masks, not wildcard masks.

```
outbound limit_acctg deny 10.200.200.0 255.255.255.0
```

The following line is an exception to the preceding line. Because the **apply** statement uses **outgoing_src**, the preceding denial of access to the 10.200.200.0 network does not apply to the host with the IP address of 10.10.1.51. Because the security level is higher on the network where this computer sits, this computer has access to the whole of the 10.200.200.0 network.

```
outbound limit_acctg except 10.10.1.51
```

The following line allows all hosts on all networks with a higher security level to have access to the host at 10.200.200.66.

```
outbound limit_acctg permit 10.200.200.66
```

The following line allows all hosts on all networks with a higher security level to have access to the host at 10.200.200.67.

```
outbound limit_acctg permit 10.200.200.67
```

The following line applies the access list called **limit_acctg** to the accounting interface and makes a definition for the **except** command, specifying that the IP addresses within the **except** command refer to a source address.

```
apply (accounting) limit_acctg outgoing_dest
```

It is important to remember that the order of the outbound statements is not a concern because the PIX uses a best-fit algorithm.

access-list and **access-group** Commands

There is another method of using access lists shown in this configuration. This method, heavily used in conjunction with crypto maps, will be explored further in the VPN sections later in this chapter. This section discusses the **access-list** and **access-group** commands as they relate to traffic other than encrypted traffic. Access lists on a PIX Firewall use either of these commands to limit connections between interfaces. When used to limit connections from a lower security level interface to a higher security level interface, the **access-list** command command can replace a **conduit** command. When used to limit connections from a higher security level interface to a lower security level interface, the access list can replace the **outbound** command.

Whether you are using the **access-list** command from a higher to lower or a lower to higher interface changes how you use this command. The following are some rules to keep in mind when designing PIX access lists.

For access from a higher security level interface to a lower security interface, always permit access first and then deny access afterward. Outbound connections are permitted by default. Therefore, the access list is used to limit this default behavior. Only **deny** statements need to be added, unless a **permit** is needed to override a **deny** command. Because PIX access lists are best-fit, this is a legitimate technique. In the configuration, you first allowed access for a single host at 10.200.200.52. You then denied access from all of those hosts on the 10.200.200.0 /24 network. Make sure that the netmask used on a PIX access list is really a subnet mask, and not the wildcard mask used on router access lists. This is shown in the configuration with the following two lines:

```
access-list acct_pub permit host 10.200.200.52
access-list acct_pub deny 10.200.200.0 255.255.255.0
```

When accessing from a lower to a higher security level, access is denied by default. Therefore, an access list would normally only contain **permit** statements. Again, you might have a situation where all except for a few hosts should be denied. In this case, you would use **permit** commands for the hosts to be let through the interface, along with a **deny** command for the specific hosts to be denied.

The full syntax for a PIX access list follows:

```
access-list name [deny | permit] protocol src_addr src_mask
    operator port dest_addr dest_mask operator port
```

On the PIX Firewall, access lists are applied to an interface with the **access-group** command. In the command, shown below, you apply the access list named **acct_pub** to the **public** interface.

```
access-group acct_pub in interface public
```

The **access-group** command always uses the keywords **in interface** before the interface name.

There are a few things to consider when working with PIX access lists. First, it is recommended that you do not use the **access list** command with the **conduit** and **outbound** commands. Technically, these commands will work together, however, the way these commands interact causes debugging issues. The **conduit** and **outbound** commands operate with two interfaces, while the **access-list** command applies only to a single interface. If you choose to ignore this warning, remember that the access list is checked first. The **conduit** and **outbound** commands are checked after the **access-list** command. Second, the masks used in the PIX access lists and the **outbound** command are subnet masks, not wildcard masks.

Additional Dual-DMZ Configuration Considerations

Notice that there is a **nat 0** command associated with the accounting DMZ. A **nat 0** command prevents any NAT or PAT from occurring. How could this be used to your advantage? Assuming that you do not use NAT and you assign nonroutable IP addresses to a DMZ, you can prevent anyone on the Internet from reaching this DMZ while still allowing the local LANs to reach the network. You can also provide additional protection when you are using routable IP addresses through the PIX. Whether or not you choose to use NAT on an interface does not really affect how that interface operates.

This concludes the configuration of the PIX Firewall, with the exception of VPNs. The remainder of this chapter covers VPNs, starting with Point-to-Point Tunneling Protocol (PPTP) and then moving on to IPSec VPNs.

VPN with Point-to-Point Tunneling Protocol (PPTP)

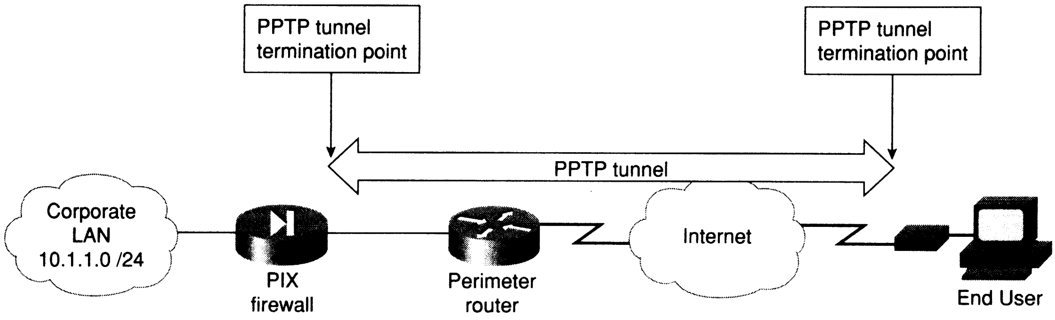
Starting with Version 5.1 of the PIX IOS, Cisco provides support for Microsoft PPTP VPN clients as an alternative to IPSec. Although PPTP is a less secure technology than IPSec, PPTP is easier to configure and maintain. PPTP also enjoys a great deal of support, especially from Microsoft clients. The PPTP is an OSI Layer 2 tunneling protocol that allows a remote client to communicate securely through the Internet. PPTP is described by RFC 2637. The PIX Firewall only supports inbound PPTP, and only a single interface can have PPTP enabled at any given time. PPTP through the PIX has been tested with Windows 95 using DUN1.3, Windows 98, Windows NT 4.0 with SP6, and Windows 2000.

The PIX Firewall supports Password Authentication Protocol (PAP), Challenge Handshake Authentication Protocol (CHAP), and Microsoft Challenge Handshake Authentication Protocol (MS-CHAP), using an external AAA server or the PIX local username and password database. Point-to-Point Protocol (PPP) with Combined Packet Protocol (CCP) negotiations with Microsoft Point-To-Point Encryption (MPPE) extensions using the RSA/RC4 algorithm and either 40- or 128-bit encryption is also supported. The compression features of MPPE are not currently supported.

To enable PPTP support, you first need to have the PIX configured to allow and deny packets in the normal fashion. The interfaces must be configured and the passwords set. After this is accomplished, you can add additional features. The sections regarding VPN in this chapter do not show all of the commands necessary to configure the PIX. Instead, this section concentrates on those commands that require configuration changes from previously shown examples or that are new commands.

Take a moment to look at Figure 4-10. Notice that the VPN tunnel is terminated on the outside interface of the PIX. Although you could terminate the VPN on the perimeter router, there are a few reasons why terminating at the PIX is preferred. The first reason is that the PIX is optimized for security operations, including VPN termination. The PIX is able to handle a much larger number of VPN terminations than most routers. The second reason is that if you terminate on the perimeter router, then only the perimeter router ensures security on the packets after the VPN tunnel has been decrypted. Because the PIX is considered the primary defense, it makes logical sense to keep packets encrypted all the way to the PIX, even if the perimeter router is running the PIX Firewall IOS.

Figure 4-10 *PIX PPTP VPN*


PPTP tunnel termination point
PPTP tunnel termination point
PPTP tunnel
Corporate LAN 10.1.1.0 /24
PIX firewall
Perimeter router
Internet
End User

The sample configuration used throughout this chapter requires changes to enable PPTP. These are shown in the following configuration. This section examines each of the new commands, after the following new configuration:

```
ip local pool thelocalpool 10.1.1.50-10.1.1.75
vpdn enable outside
vpdn group 1 accept dialin pptp
vpdn group 1 ppp authentication mschap
vpdn group 1 client configuration address local thelocalpool
vpdn group 1 client configuration dns 10.1.1.41
vpdn group 1 client configuration wins 10.1.1.9
vpdn group 1 client authentication local
vpdn username joe password joespassword
vpdn username mary password marryspassword
sysopt connection permit-pptp
```

ip local pool Command

An IP local pool is used with VPNs to reserve a range of IP addresses that will be assigned to hosts using VPNs. The addresses in this range must not be in use by any other hosts and should not be used in any other commands. Use the **show** form of the command to display all of the IP addresses within a pool. The command, reserving IP addresses of 10.1.1.50 through 10.1.1.75 and using the name **thelocalpool** follows.

```
ip local pool thelocalpool 10.1.1.50-10.1.1.75
```

vpdn Command

The **vpdn** command takes many forms. The first line, the **vpdn enable outside** command, accomplishes two tasks. First, this enables virtual private dial-up network (VPDN) support on the PIX itself. Second, VPDN is enabled on the interface labeled **outside** by the **nameif** command. Multiple interfaces accepting PPTP traffic each require a separate **vpdn enable** *interface* command. Note that the PIX Firewall only accepts incoming PPTP traffic and cannot be used to initiate a PPTP tunnel.

The basic form of the command, **vpdn group 1 accept dialin pptp**, associates the VPDN group numbered 1 within other commands. Assuming that multiple PPTP tunnels are to be terminated on this interface, you might wish to set up some users on one tunnel and other users on a different tunnel. In this case, multiple tunnels allow you to accomplish such tasks as assigning different WINS or DNS severs to individuals. The **accept dialin pptp** portion of this command tells the PIX that it should accept PPTP connections requested by outside entities.

The **vpdn group 1 ppp authentication mschap** command shown next ensures that the password authentication protocol used within VPDN group 1 is **mschap**. The other options available on this command are **pap** and **chap**.

NOTE You must also ensure that any associated Windows devices needing to use a PPTP tunnel into your network are also configured correctly. Unless you have set a Microsoft Windows client to require encrypted passwords, the client will first use a clear-text PAP password. This attempt will fail because of your PIX configuration that requires encryption. The client will then attempt to connect using the same password in an encrypted form, which will be successful. Even though the connection is ultimately successful, the password has been sent in clear text and might have been revealed to hackers. Therefore, ensure that encrypted passwords are required on all Microsoft Windows clients used with tunneled connections.

The **vpdn group 1 client configuration address local thelocalpool** command is used to assign the IP address used by the client while the client is connected through the PPTP connection. Because you created a group called **thelocalpool** and assigned the addresses of 10.1.1.50 through 10.1.1.75 to that pool, this command assigns the client to look to that pool for one of these available addresses. Limiting the total number of available IP addresses in the pool in turn limits the total number of PPTP connections that can be used simultaneously.

The **client configuration** form of the **vpdn** command is used to assign WINS and DNS servers for use by the PPTP client while the client is connected into your system. Both of these commands can take either one or two IP addresses. The order that these IP addresses are entered within the command reflects the order of their use by Windows clients.

The **vpdn group 1 client authentication local** command tells the PIX to look to the local user database to check passwords. If you are using a AAA server for client authentication, you would need to set up the PIX to recognize the AAA server and the need to authenticate PPTP users with lines similar to the following:

```
aaa-server TACACS+ (inside) host 10.1.1.41 thekey timeout 20
client authentication aaa TACACS+
```

The **vpdn username joe password joespassword** command enters Joe as a user within the local database and assigns **joespassword** to Joe. This is the password whose hash result will be sent over the connection through the MS-CHAP authentication process. You have also enabled Mary as a user with a unique password. Once the system is configured to allow one user, allowing other users involves adding a username and password to the PIX configuration.

sysopt Command

The previous commands shown in this example have set up the PPTP tunnel and users. What has not been done is to allow the users access through the firewall. The **sysopt connection permit-pptp** command allows for all authenticated PPTP clients to traverse the PIX interfaces. The **sysopt** command is used to change the default security behavior of the PIX Firewall in a number of different ways. There are many forms of this command, each acting slightly differently. Table 4-5 contains a list of the **sysopt** commands and a description of each of their functions. Each of these commands also has an associated **no** form of the command, which is used to reverse the behavior associated with the command.

Table 4-5 sysopt *Commands*

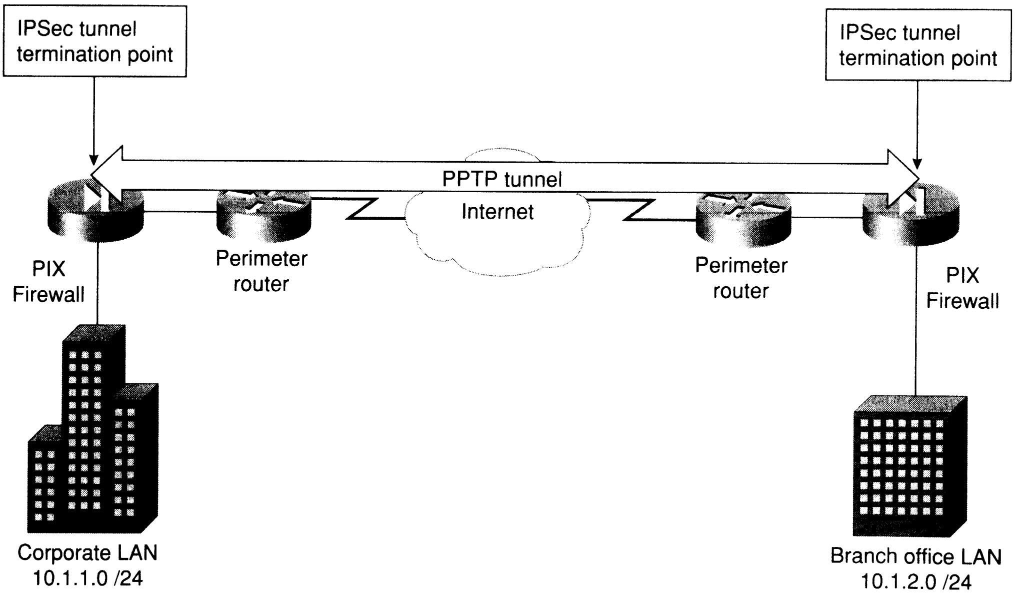
Command	Description
sysopt connection enforcesubnet	Prevents packets with a source address belonging to the destination subnet from traversing an interface. A packet arriving from the outside interface having an IP source address of an inside network is not allowed through the interface.
sysopt connection permit-ipsec	Allows traffic from an established IPSec connection to bypass the normal checking of access lists, **conduit** commands, and **access-group** commands. In other words, if an IPSec tunnel has been established, this command means that the traffic will be allowed through the interface on which the tunnel was terminated.
sysopt connection permit-pptp	Allows traffic from an established PPTP connection to bypass **conduit** and **access-group** commands and access lists.
sysopt connection tcpmss *bytes*	Forces TCP proxy connections to have a maximum segment size equal to the number specified by the parameter *bytes*. The default for *bytes* is 1380.
sysopt connection timewait	Forces TCP connections to stay in a shortened time-wait state of at least 15 seconds after the completion of a normal TCP session ends.
sysopt ipsec pl-compatible	Enables IPSec packets to bypass both NAT and the ASA features. This also allows incoming IPSec tunnels to terminate on an inside interface. For a tunnel crossing the Internet to terminate on the inside interface, the inside interface must have a routable IP address.
sysopt nodnsalias outbound	Denies outbound DNS A record replies.
sysopt noproxyarp *interface_name*	Disables proxy ARPs on the interface specified by *interface_name*.
sysopt security fragguard	Enables the IP Frag Guard feature, which is designed to prevent IP fragmentation attacks such as LAND.c and teardrop. This works by requiring responsive IP packets to be requested by an internal host before they are accepted and limits the number of IP packets to 100 per second for each internal host.

VPN with IPSec and Manual Keys

IOS versions of the PIX prior to 5.0 used a connection method involving the Private Link Encryption Card to connect between two PIX Firewalls. This method is no longer supported; IPSec is used as the alternative. If your system is still using Version 4 or earlier of the Cisco PIX IOS, it is time to upgrade.

In this configuration, you will use IPSec to connect two networks over the Internet. You will also use manual keys for this example. In this example, your main corporate office uses an internal IP address of 10.1.1.0 with a 24-bit subnet mask, while your branch office uses 10.1.2.0 with a 24-bit subnet mask. (As with any interface accessible from the Internet, the outside interface of the PIX must have a routable IP address.) Figure 4-11 shows a diagram of how these networks are connected.

Figure 4-11 *VPN with IPSec*


IPSec tunnel termination point
IPSec tunnel termination point
PPTP tunnel
Internet
PIX Firewall
Perimeter router
Perimeter router
PIX Firewall
Corporate LAN 10.1.1.0 /24
Branch office LAN 10.1.2.0 /24

You need to configure both PIX Firewalls to enable a secure tunnel between them. The configurations that follow show only the items associated with setting up the IPSec tunnels. You will see both configurations and then a discussion of the ramifications of using the commands. Keep in mind that these are examples and, therefore, do not have routable IP addresses on the outside interfaces. In real life, the outside interfaces would need routable IP addresses; inside the corporate LANs, the IP addresses do not need to be routable. The corporate PIX configuration changes are as follows:

```
ip address outside 172.30.1.1 255.255.255.252
access-list 20 permit 10.1.2.0 255.255.255.0
crypto map mymap 10 ipsec-manual
crypto map mymap 10 set transform-set myset
crypto ipsec transform-set myset ah-md5-hmac esp-des
crypto map mymap 10 match address 20
crypto map mymap 10 set peer 172.30.1.2
crypto map mymap 10 set session-key inbound ah 400
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
crypto map mymap 10 set session-key outbound ah 300
    bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
crypto map mymap 10 set session-key inbound esp 400 cipher
    cccccccccccccccccccccccccccccccc
crypto map mymap 10 set session-key outbound esp 300 cipher
    dddddddddddddddddddddddddddddddd
crypto map mymap interface outside
sysopt connection permit-ipsec
```

The branch office PIX configuration changes are as follows:

```
ip address outside 172.30.1.2 255.255.255.252
access-list 20 permit 10.1.1.0 255.255.255.0
crypto map mymap 10 ipsec-manual
crypto map mymap 10 set transform-set myset
crypto ipsec transform-set myset ah-md5-hmac esp-des
crypto map mymap 10 match address 20
crypto map mymap 10 set peer 172.30.1.1
crypto map mymap 10 set session-key inbound ah 300
    bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
crypto map mymap 10 set session-key outbound ah 400
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
crypto map mymap 10 set session-key inbound esp 300 cipher
    dddddddddddddddddddddddddddddddd
crypto map mymap 10 set session-key outbound esp 400 cipher
    cccccccccccccccccccccccccccccccc
crypto map mymap interface outside
sysopt connection permit-ipsec
```

In this example, after assigning your outside IP addresses, you added an access list. Because you decided to use manual keys, this access list might contain only a single **permit**. If you used preshared keys, the access list could contain multiple **permit** statements. The access list is used to invoke your IPSec connection. When packets are sent to this address, your PIX establishes a connection with the peer, and all data traveling between the two is carried over your tunnel.

crypto map Commands

The **crypto map** command is used extensively with IPSec. This section examines the forms of this command in Table 4-6 before examining exactly what has to be configured in the examples. The **crypto map** command's first parameter is always the *mapname*. The *mapname* parameter is an arbitrary name assigned to distinguish one map from another. Table 4-6 assumes that **crypto map** *mapname* precedes the command. As with most commands, the **no** form of a command removes the configuration.

Table 4-6 **crypto map** *mapname Parameters*

Crypto Command	Description
client authentication *aaa-server*	This is the name of a AAA server that authenticates the user during Internet Key Exchange (IKE) negotiations.
client configuration address initiate	This forces the PIX to attempt to set the IP address for each peer.
client configuration address respond	This forces the PIX to attempt to accept requests from any requesting peer.
interface *interfacename*	This specifies the interface, as defined by the **nameif** command, that the PIX will use to identify peers. When IKE is enabled and a certificate authority (CA) is used to obtain certificates, this must be the interface specified within the CA certificate.
seq-num **ipsec-isakmp | ipsec-manual [dynamic** *dynamic-map-name*]	The *seq-num* (sequence number) is the number assigned to the map entry. The *seq-num* is used in a number of forms of the **crypto map** command. **ipsec-isakmp** indicates that IKE is used to establish the security association (SA). **ipsec-manual** indicates that IKE should not be used. **dynamic** *dynamic-map-name* is an optional keyword and parameter. The keyword **dynamic** indicates that the present **crypto map** entry references a preexisting dynamic crypto map. The parameter *dynamic-map-name* is the name of the preexisting map.
seq-num **match address** *acl_name*	Traffic destined for the IP addresses with a **permit** statement within the access list defined by *acl_name* will be encrypted.
seq-num **set peer** *hostname* | *ipaddress*	This specifies the peer for this SA. A host name might be specified if the **names** command has been used. Otherwise an IP address is used.
seq-num **set pfs [groupl | group2]**	Specifies that IPSec will ask for Perfect Forward Secret (PFS). **groupl** and **group2** are optionally used to specify whether a 768-bit Diffie-Hillman prime modulus group (**groupl**) or a 1024-bit Diffie-Hillman prime modulus group (**group2**) will be used on new exchanges.

Table 4-6 **crypto map** *mapname Parameters (Continued)*

Crypto Command	Description
seq-num **set session-key inbound** | **outbound ah** *spi hex-key-string*	This sets the session keys within a **crypto map** entry. Using the keyword **inbound** specifies that the following *key-string* is for inbound traffic. Specifying the keyword **outbound** specifies that the *key-string* is for outbound traffic. One peer's outbound key string must match the other peer's inbound key string and vice versa. The *spi* parameter is used to specify the Security Parameter Index (SPI). The SPI is an arbitrarily assigned number ranging from 256 to more than 4 billion (OxFFFFFFFF). The *hex-key-string* is an arbitrary hexadecimal session key. The length of this key is determined by the transform set in use. DES uses 16 digits, MD5 uses 32, and SHA uses 40 digits.
seq-num **set session-key inbound** | **outbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]	This is very similar to the previous command, except that it is used with encapsulating security payload (ESP) instead of authentication header (AH). The keyword **esp** specifies that the ESP protocol will be used. The keyword **cipher** indicates that the following *hex-key-string* is to be used with the ESP encryption transform. The optional authenticator string is used with the ESP authentication transform.

crypto ipsec Command

You have also seen the **crypto ipsec** command used within the configurations. There are two major forms of this command, the **crypto ipsec transform-set** and the **crypto ipsec security-association lifetime** forms. Both of these can be removed with the **no** form of the command. These commands are explained in Table 4-7.

Table 4-7 **crypto ipsec** *Commands*

Crypto Command	Description
crypto ipsec set security-association lifetime seconds *seconds* | **kilobytes** *kilobytes*	If the keyword **seconds** is used, the *seconds* parameter specifies how many seconds before an SA will remain active without renegotiation. The default is 28,800 seconds, which is 8 hours. If the keyword **kilobytes** is used, the *kilobytes* parameter specifies how many kilobytes of data can pass between peers before a renegotiation must occur. The default value is 4,608,000 KB, which is approximately 4.5 GB.
crypto ipsec transform-set *transform-set-name*	This command defines the transform sets that can be used with the map entry. There can be up to a total of six *transform-set-names* used within a single line. The transform set attempts to establish an SA in the order that the sets are specified.

Now that you have seen the syntax and uses of the **crypto map** and **crypto ipsec** commands, look again at the sample configurations.

You tell the PIX that your crypto map is named **mymap** with a map number of 10 and that IKE should not be used. This is done with the following line:

```
crypto map mymap 10 ipsec-manual
```

Next, you define the name of the transform with the following:

```
crypto map mymap 10 set transform-set myset
```

The transform set is defined with the following line:

```
crypto ipsec transform-set myset ah-md5-hmac esp-des
```

You previously created an access list 20 and permitted packets originating from the remote site's network. You then set the PIX to look at access list 20. If the packets are traveling to or from an address within this access list, they will be encrypted.

```
crypto map mymap 10 match address 20
```

Set the other end of the IPSec tunnel to terminate at 172.30.1.2, which is the outside interface of the branch office's PIX:

```
crypto map mymap 10 set peer 172.30.1.2
```

Set up the inbound and outbound session keys:

```
crypto map mymap 10 set session-key inbound ah 400
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
crypto map mymap 10 set session-key outbound ah 300
    bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
crypto map mymap 10 set session-key inbound esp 400 cipher
    cccccccccccccccccccccccccccccccc
crypto map mymap 10 set session-key outbound esp 300 cipher
    dddddddddddddddddddddddddddddddd
```

Associate the crypto map with the outside interface.

```
crypto map mymap interface outside
```

Finally, permit IPSec packets into the network with the **sysopt** command.

```
sysopt connection permit-ipsec
```

The branch office PIX configuration is almost identical. The following section points out where it differs.

The branch office PIX has a different outside IP address.

```
ip address outside 172.30.1.2 255.255.255.252
```

The access list must reflect the main office's IP addresses.

```
access-list 20 permit 10.1.1.0 255.255.255.0
```

The peer is the outside IP address of the main office's PIX.

```
crypto map mymap 10 set peer 172.30.1.1
```

The session keys for the branch office are configured in the opposite order of what is configured on the main office's PIX. The inbound key on one side of a connection must equal the outbound key on the opposite side of the connection. The inbound AH session key on the Branch office is equal to the outbound AH session key on the main office's PIX. The inbound AH session key must match the main office's outbound AH session key in order for the connection to be established. The inbound ESP session key matches the main office's inbound ESP session key and the outbound ESP session key matches the main office's inbound ESP session key:

```
crypto map mymap 10 set session-key inbound ah 300
    bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
crypto map mymap 10 set session-key outbound ah 400
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
crypto map mymap 10 set session-key inbound esp 300 cipher
    dddddddddddddddddddddddddddddddd
crypto map mymap 10 set session-key outbound esp 400 cipher
    cccccccccccccccccccccccccccccccc
```

VPN with Preshared Keys

Using preshared keys is easy, once you understand the concepts presented in the previous example. The difference between this configuration and the previous one is that you are now relying on the Internet Security Association and Key Management Protocol

(ISAKMP) for exchanging keys. This section presents the configuration before exploring how it has changed. The main office's configuration is as follows:

```
hostname chicago
domain-name bigcompany.com
isakmp enable outside
isakmp policy 15 authentication pre-share
isakmp policy 15 encr 3des
crypto isakmp key isakmpkey address 172.30.1.2
crypto ipsec transform-set strong esp-sha-hmac esp-3des
access-list myaccesslist permit ip 10.1.2.0 255.255.255.0
crypto map seattletraffic 29 ipsec-isakmp
crypto map seattletraffic 29 match address myaccesslist
crypto map seattletraffic 29 set transform-set strong
crypto map seattletraffic 29 set peer 172.30.1.2
crypto map seattletraffic interface outside
sysopt connection permit-ipsec
```

The branch PIX Firewall configuration looks like this:

```
hostname seattle
domain-name bigcompany.com
isakmp enable outside
isakmp policy 21 authentication pre-share
isakmp policy 21 encryption 3des
crypto isakmp key isakmpkey address 172.30.1.1
crypto ipsec transform-set strong esp-3des esp-sha-hmac
access-list chicagolist permit ip 10.1.1.0 255.255.255.0
crypto map chicagotraffic 31 ipsec-isakmp
crypto map chicagotraffic 31 match address chicagolist
crypto map chicagotraffic 31 set transform-set strong
crypto map chicagotraffic 31 set peer 172.30.1.1
crypto map chicagotraffic interface outside
sysopt connection permit-ipsec
```

isakmp Commands

Before explaining the example, review Table 4-8 concerning the **isakmp** commands. The **isakmp** commands are very similar in syntax to the **vpdn** commands. As with most commands, using the **no** form of the command removes the configuration.

Table 4-8 **isakmp** *Commands*

Command	Description
isakmp client configuration address-pool local *localpoolname*	This command assigns a VPN client an address from within the addresses set aside by the **ip local pool** command.
isakmp enable *interfacename*	This enables ISAKMP on the interface specified by the parameter *interfacename.*
isakmp identity *address* | *hostname*	This identifies the system for IKE participation.
isakmp key *keystring* **address** *peer-address*	The *keystring* specifies the preshared key. The *peer-address* specifies the IP address of the peer.

Table 4-8 **isakmp** *Commands (Continued)*

Command	Description
isakmp peer fqdn *fqdn* **no-xauth no-config-mode**	The *fqdn* (fully qualified domain name) is the full DNS name of the peer. This is used to identify a peer that is a security gateway. The **no-xauth** option is to used if you enabled the Xauth feature and you have an IPSec peer that is a gateway. The **no-config-mode** option is used if you enabled the IKE Mode Configuration feature and you have an IPSec peer that is a security gateway.
isakmp policy *priority* **authentication pre-share | rsa-sig**	This sets the priority for the authentication and defines whether you are using pre-shared keys or RSA signatures.
isakmp policy *priority* **group1 | group2**	**group1** and **group2** are optionally used to specify whether a 768-bit Diffie-Hillman prime modulus group (**group1**) or a 1024-bit Diffie-Hillman prime modulus group (**group2**) will be used on new exchanges.
isakmp policy *priority* **hash md5 | sha**	Specifies MD5 or SHA as the hash algorithm to be used in the IKE policy.
isakmp policy *priority* **lifetime** *seconds*	Specifies how many seconds each SA should exist before expiring.

Explanation of VPN with Preshared Keys

Going back to the configuration, you can see that it is really quite simple to enable preshared keys. The following section will walk you through the configuration and explain what has been configured.

First, set the host name. The fully qualified domain name (FQDN) is set with the **domain-name** command.

```
hostname chicago
domain-name bigcompany.com
```

Then set ISAKMP to the outside interface and define that you use preshared keys and 3DES encryption.

```
isakmp enable outside
isakmp policy 15 authentication pre-share
isakmp policy 15 encr 3des
```

The ISAKMP key, whose value is **isakampkey**, is set, along with the IP address of the outside interface of the peer. Then set **transform-set** to first use **esp-sha-hmac** and then **esp-3des**.

```
crypto isakmp key isakmpkey address 172.30.1.2
crypto ipsec transform-set strong esp-sha-hmac esp-3des
```

Define an access list for use with the **crypto map** command, setting the permitted IP addresses to match the remote site's IP address.

```
access-list myaccesslist permit ip 10.1.2.0 255.255.255.0
```

Next, map the traffic to be encrypted, set the peer, and set the interface.

```
crypto map seattletraffic 29 ipsec-isakmp
crypto map seattletraffic 29 match address myaccesslist
crypto map seattletraffic 29 set transform-set strong
crypto map seattletraffic 29 set peer 172.30.1.2
crypto map seattletraffic interface outside
```

Finally, set the PIX to allow IPSec traffic through the interfaces.

```
sysopt connection permit-ipsec
```

The only real differences between the branch office and the main office configurations are that the peers are set to the other office's PIX outside interface, and the traffic to be encrypted is set to the other office's LAN.

Obtaining Certificate Authorities (CAs)

Retrieving certificate authorities (CAs) with the PIX Firewall uses almost exactly the same method as that used on routers. The following are the commands used to obtain a CA. Note that these commands might not show in a configuration. The administrator should avoid rebooting the PIX during this sequence. The steps are explained as they are shown.

First, define your identity and the IP address of the interface to be used for the CA. Also configure the timeout of retries used to gain the certificate and the number of retries.

```
ca identity bigcompany.com 172.30.1.1
ca configure bigcompany.com ca 2 100
```

Generate the RSA key used for this certificate.

```
ca generate rsa key 512
```

Then get the public key and certificate.

```
ca authenticate bigcompany.com
```

Next, request the certificate, and finally, save the configuration.

```
ca enroll bigcompany.com enrollpassword
ca save all
```

At this point, you have saved your certificates to the flash memory and are able to use them. The configuration for using an existing CA is as follows:

```
domain-name bigcompany.com
isakmp enable outside
isakmp policy 8 auth rsa-signature
ca identity example.com 172.30.1.1
ca generate rsa key 512
```

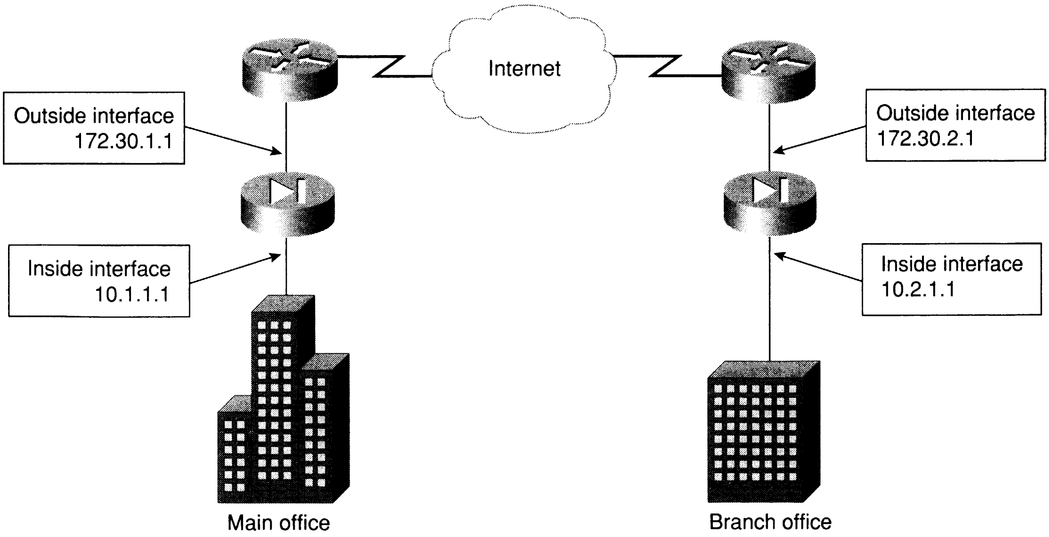
```
access-list 60 permit ip 10.1.2.0 255.255.255.0
crypto map chicagotraffic 20 ipsec-isakmp
crypto map chicagotraffic 20 match address 60
crypto map chicagotraffic 20 set transform-set strong
crypto map chicagotraffic 20 set peer 172.30.1.2
crypto map chicagotraffic interface outside
sysopt connection permit-ipsec
```

PIX-to-PIX Configuration

One advantage of using the PIX Firewall is that it has become a standard within the industry. As time passes, your business might acquire or become acquired by another company. To provide connectivity, you are faced with two choices: enabling VPNs over the Internet or using dedicated connections. Because one of the benefits of the PIX box is to allow secure VPNs, this section explores how to set up two PIX Firewalls between different locations through the Internet.

In this scenario, shown in Figure 4-10, assume that both companies trust each other totally. This means that you will not filter any traffic between the sites, and all hosts on both sites will be able to see all hosts on the other site. The peers use ISAKMP in Phase 1 to negotiate an IPSec connection in Phase 2.

Figure 4-12 *PIX-to-PIX IPSec with ISAKMP Example*



As shown in Figure 4-12, the main office uses an internal IP address of 10.1.1.1/24 with an IP address of 172.30.1.1 on the outside interface. The branch office uses an internal IP address of 10.2.1.1/24 and an IP address of 172.30.2.1 on the outside interface. The following is the configuration for the PIX Firewall at the main office. After the configuration, you will see a discussion of the commands used.

```
hostname mainofficepix
nameif ethernet0 outside security0
nameif ethernet1 inside security100
interface ethernet0 auto
interface ethernet1 auto
mtu outside 1500
mtu inside 1500
ip address outside 172.30.1.1 255.255.255.0
ip address inside 10.1.1.1 255.255.255.0
access-list 100 permit ip 10.1.1.0 255.255.255.0 10.2.1.0 255.255.255.0
nat (inside) 0 access-list 100
sysopt connection permit-ipsec
crypto ipsec transform-set maintransformset esp-des esp-md5-hmac
crypto map mymap 10 ipsec-isakmp
crypto map mymap 10 match address 100
crypto map mymap 10 set peer 172.30.2.1
crypto map mymap 10 set transform-set maintransformset
crypto map mymap interface outside
isakmp enable outside
isakmp key mysharedkey address 172.30.2.1 netmask 255.255.255.255
isakmp policy 10 authentication pre-share
isakmp policy 10 encryption des
isakmp policy 10 hash md5
isakmp policy 10 group 1
isakmp policy 10 lifetime 768
```

All of the preceding commands have been discussed previously within this chapter. There are only a few new items that you need to watch carefully to ensure that this configuration will work.

First, access list 100 must allow hosts from the branch office through the PIX Firewall. Limiting who is allowed through on the branch office network or which hosts that the branch office hosts are allowed to see is controlled through this access list. For example, assume that everyone except the branch manager in the branch office is allowed to connect only to the hosts at 10.1.1.14, 10.1.1.15, and 10.1.1.200. The branch manager, whose IP address is 10.2.1.53, is allowed to access all hosts on the main office network. In this case, your access list would be as follows:

```
access-list 100 permit ip 10.1.1.0 255.255.255.0 10.1.2.1.53 255.255.255.255
access-list 100 permit ip 10.1.1.14 255.255.255.255 10.2.1.0 255.255.255.0
access-list 100 permit ip 10.1.1.15 255.255.255.255 10.2.1.0 255.255.255.0
access-list 100 permit ip 10.1.1.200 255.255.255.255 10.2.1.0 255.255.255.0
```

Now take note of the use of the **nat 0** command to prevent NAT from occurring. In some cases, you need to enable NAT because both sites are using the same nonroutable IP addresses. This is actually a common scenario. For example, without NAT enabled and both sites using the 10.1.1.0/24 network, both PIX Firewalls will not know which network to respond to when a packet is received.

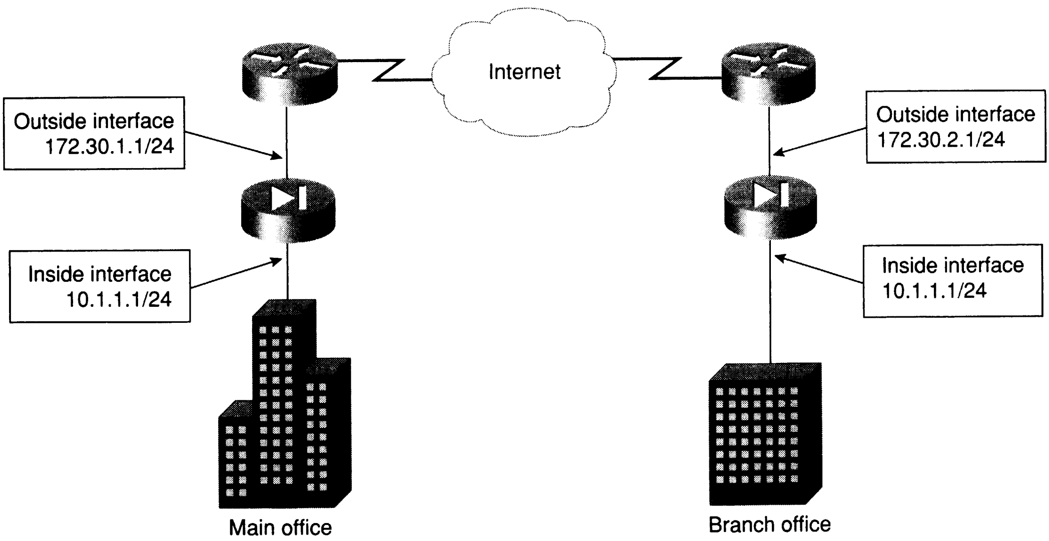
Next, you set up the Phase 2 connection. Use the **sysopt** command with the *permit-ipsec* parameter to allow packets associated with this SA through the PIX Firewall. Set up the transform set for IPSec, assign a map to the access list, and set the interface for the crypto connection. You also use the **crypto map** command to set the peer for this connection. As always, the IP address of the peer should be the outside interface of the remote PIX Firewall.

As with any ISAKMP key exchange, you need to ensure that the interface chosen is appropriate, that the key is exactly the same on both peers, and that the encryption and hash types are identical between peers.

PIX-to-PIX with Identical Internal IP Addresses

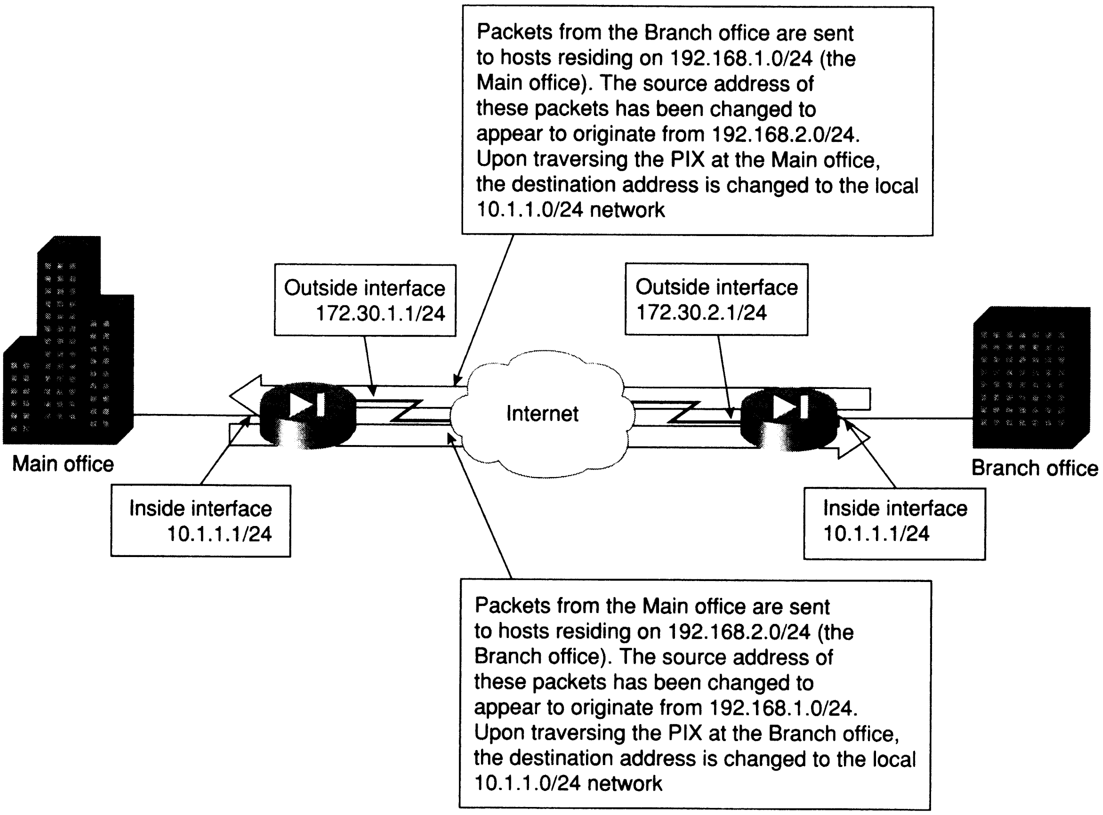
One of the issues raised by using a nonroutable IP address is the use of the IP address while another connected location is using that same address. This is a common issue when two companies connect to each other for the first time. Looking at Figure 4-13, notice that both the main and branch offices use the same internal IP address. In this situation, you will need to translate the addresses of both internal networks.

Figure 4-13 *PIX-to-PIX with Identical Internal Network Addresses*


Internet
Outside interface 172.30.1.1/24
Inside interface 10.1.1.1/24
Outside interface 172.30.2.1/24
Inside interface 10.1.1.1/24
Main office
Branch office

On the PIX at the main office, you will use NAT to translate all data destined for the branch office to the 192.168.1.0/24 network. The branch office translates all data destined for the main office to use 192.168.2.0/24 addresses. Therefore, from the point of view of the main office, the branch office appears to use 192.168.2.0/24. From the point of view of the branch office, the main office appears to use 192.168.1.0 as its internal IP addresses. Each PIX Firewall needs to be configured in a similar manner. Figure 4-14 shows how each office sees the other.

Figure 4-14 *PIX-to-PIX with Each Side Using NAT*


Packets from the Branch office are sent to hosts residing on 192.168.1.0/24 (the Main office). The source address of these packets has been changed to appear to originate from 192.168.2.0/24. Upon traversing the PIX at the Main office, the destination address is changed to the local 10.1.1.0/24 network
Outside interface 172.30.1.1/24
Outside interface 172.30.2.1/24
Internet
Main office
Branch office
Inside interface 10.1.1.1/24
Inside interface 10.1.1.1/24
Packets from the Main office are sent to hosts residing on 192.168.2.0/24 (the Branch office). The source address of these packets has been changed to appear to originate from 192.168.1.0/24. Upon traversing the PIX at the Branch office, the destination address is changed to the local 10.1.1.0/24 network

The listing of this configuration follows. This is virtually the same configuration as the previous example, with a few minor changes. First, you have to implement a global pool for use with NAT for data traveling to the branch office. Second, you have to remove the lines associated with the **nat 0** command for data traveling to the branch office. Third, you have to create a new access list called *nattobranch*, which is used by NAT to change the source address of the packets so that these packets appear to originate from the 192.168.1.0/24 network.

```
hostname mainofficepix
nameif ethernet0 outside security0
nameif ethernet1 inside security100
interface ethernet0 auto
interface ethernet1 auto
mtu outside 1500
mtu inside 1500
ip address outside 172.30.1.1 255.255.255.0
ip address inside 10.1.1.1 255.255.255.0
global (outside) 1 192.168.1.1-192.168.1.253
global (outside) 1 192.168.1.254
```

```
access-list nattobranch permit ip 10.1.1.0 255.255.255.0 192.168.2.1 255.255.255.0
nat (inside) 1 access-list nattobranch
sysopt connection permit-ipsec
crypto ipsec transform-set maintransformset esp-des esp-md5-hmac
crypto map mymap 10 ipsec-isakmp
crypto map mymap 10 match address nattobranch
crypto map mymap 10 set peer 172.30.2.1
crypto map mymap 10 set transform-set maintransformset
crypto map mymap interface outside
isakmp enable outside
isakmp key mysharedkey address 172.30.2.1 netmask 255.255.255.255
isakmp policy 10 authentication pre-share
isakmp policy 10 encryption des
isakmp policy 10 hash md5
isakmp policy 10 group 1
isakmp policy 10 lifetime 768
```

Summary

This chapter has shown how to configure the PIX Firewall in many different ways. It started with the most basic form before moving to a more realistic configuration. This realistic configuration, allowing users through to specific services, should prove adequate for most companies that do not require the use of a DMZ.

The chapter then moved on to explore using single and multiple DMZs, along with AAA services and other examples of connections possible with the PIX Firewall. These configurations provide examples that are applicable to larger organizations.

from Integrating Voice and Data Networks

by Scott Keagy

(1-57870-196-1)

Cisco Press

About the Author

Scott Keagy (CCIE# 3985) is a cofounder of SK Networks, Inc., which specializes in voice/data network consulting. While on assignment at Cisco Systems in 1997 and 1998, he implemented the first VoFR and VoIP connectivity in its production network, for the sales offices in South and Central America. Since 1998 he has been on assignment at Pacific Bell and SBC, designing and implementing VoFR, VoATM, and VoIP solutions for their customers. He is a member of the IEEE Communications society, and he has been

in the networking and computer industries for nine years. You can contact Scott at cisco-book1@sknetworks.com.

Contents at a Glance

Bold chapters are elements included in this folio.

CHAPTER 12

Initial Network Planning and Design

If you live (or at least work) in the real world, then the success of your voice/data integration project(s) will be judged by more than whether or not the technical solution is eventually achieved. The path to that solution, in terms of time and money, is critical. If you have a cavalier approach to the initial planning phases, there is an increased likelihood that you will be surprised later in the project. And I can tell you from experience that the surprise is not usually good!

Most organizations want to save time and money. For non-technical project managers, this may translate to a reduction in discovery, analysis, and planning phases: "Let's not waste any time—we needed this done last week and we are paying by the hour!"

As a technical project manager, it is your job to resist the intuitive—albeit wrong—reaction of cutting corners on the initial phases. The extra time you spend at the beginning of a project is well rewarded by an easier implementation. The implementation phase is the wrong time to discover fundamental flaws with the design, incompatible interfaces, unsupported features, and so on. It can be very expensive and awkward to correct such problems when a project is far along the implementation path. According to an old adage, "A stitch in time saves nine." This may be updated for our purposes to, "A dollar spent on planning and design, in implementation, will save nine."

Before you place an order for new hardware, there are a number of project steps that you must complete:

- Gathering requirements and expectations for voice services
- Gathering telephony interface and signaling information
- Selecting a VoX technology
- Planning trunk and bandwidth requirements
- Selecting hardware platforms
- Reviewing proposed solutions in terms of requirements

Gathering Requirements and Expectations for Voice Services

It is important that you work with all concerned parties at the beginning of the project to determine the requirements for the project and the measures of success. Specifically, you must address the issues of calling patterns, voice-mail services, voice applications such as call centers and ACD groups, and the voice quality expectations of all concerned parties.

To understand why calling patterns are a concern, consider the following scenario:

> You call the main telephone number of Acme Corporation, and the receptionist transfers you to a person in sales. When you speak to the person in sales, you realize that you really need to speak to someone in marketing. Nobody answers the telephone when you are transferred to marketing, so you are automatically forwarded to voice mail.

This is a fairly normal business communication process, and companies expect their voice systems to accommodate these types of transactions. It is very likely that an integrated voice/data network would perform poorly when placed in this environment. Some may claim that the technology is not yet mature, but it is more commonly a problem with the design.

The inherent problem in this case is that the company may have departments in different locations. This means that every time a call is transferred between departments, it crosses a PBX/router interface and incurs another tandem encoding. Voice quality rapidly degrades as the signal undergoes coding/decoding cycles—current low bit-rate codecs tolerate a maximum of two encoding cycles in real networks. The previous call scenario may require four encodings. Now imagine that the person placing the call is using a mobile telephone; another low bit-rate codec is in the audio path. The best solution in this case is to use a G.726 or G.711 codec, which greatly increases the bandwidth consumption on the network, but preserves voice quality during tandem encodings.

Many network designers would not even consider this issue before implementing the network. The integrated network would be installed, and preliminary tests would indicate that voice quality is good. But they would forget to test call transfers between sites. After about a month, end users complain bitterly.

The point of the preceding story is to "think outside the box" when designing these networks. Do not assume you understand all of the requirements ahead of time. Talk in detail with the end users of the network to understand how they will use it. The project will be more successful if you understand the requirements, voice your concerns in a timely fashion, and address them properly. Among other issues, be sure to address voice mail from the earliest phases of the project. This is almost always a show-stopper issue if there are problems, so get it right the first time. If a centralized voice-mail system is used, you generally cannot use low bit-rate codecs and preserve good voice quality. If you are selling the project based on 8-kbps calls, you better rework your strategy.

In general, be wary of situations that require tandem encoding. Common examples include centralized call accounting, centralized voice-mail systems, and proprietary PBX features supported via transparent common channel signaling (CCS). Possible solutions include using higher bit-rate codecs, and possibly using Q.SIG instead of transparent CCS.

Here is one last thought with respect to voice/data network designs that can cause problems. Some corporations have a hierarchical data network design, based on the classic concept of a corporate headquarters, regional hubs, and satellite offices. All is well in the data-only world, with regional e-mail hubs and data centers. But this design may introduce problems for company-wide, real-time applications. Consider what happens if users in a satellite office call users in a distant satellite office. The audio path must traverse four WAN circuits:

- Originating satellite office to originating regional hub
- Originating regional hub to corporate headquarters
- Corporate headquarters to destination regional hub
- Destination regional hub to destination satellite office

The combined effect of transmission delays and variable queuing/serialization delays can severely impact voice quality. Even though the data network is well designed, it is not designed well for the addition of voice traffic. There are at least three solutions to this problem:

- Collapse the hierarchy to a hub-and-spoke topology from the corporate headquarters.
- Add extra circuits from satellite offices to corporate headquarters.
- Install circuits between offices with identified high-traffic patterns.

Each of these solutions has technical, logistical, and economic implications that are nontrivial. There are no easy fixes for these types of problems, but you are in a better position to manage them if you identify them early in the process.

You have read some of the ways your project can go awry. Hopefully, you will not repeat the mistakes presented here. Think through the consequences of your design decisions, and talk with others about issues of concern.

Gathering Telephony Trunk and Signaling Information

Early on in the project, you must evaluate your role. If you are working with a strong telecom department that has specific ideas about how to do things, then you might not want to dictate designs and requirements to them. If you alienate these folks, then your job will be more difficult. On the other hand, some organizations very badly need someone to take charge and figure out what needs to be done. These extremes are important to identify because it affects how you gather the required telephony information.

If you have a strong telecom department, you can rely on them to provide the information you need, such as what type of PBX ports will be connected to the router, and what signaling types to use. Make sure to provide feedback to them on key issues, such as how to divide number blocks for the address plan.

Be prepared to make decisions if you have a lot of unanswered questions, or you get responses such as, "We can do either," or "What do you want to do?" This is a more common scenario when you are working with telecom technicians who are not responsible for the whole network. They usually program the basic switch options using information provided to them, and leave many options in default settings. In these cases, you can choose which of the available PBX ports to use, how to provision the signaling, who provides clocking, and so on.

Refer to Chapter 2, "Enterprise Telephony Signaling," for a detailed discussion of the traditional telephony signaling types. From a project-planning perspective, you can gather the required information in phases:

- Type of equipment and physical interface connecting to the router
- Software configuration options for interface connecting to the router

It is usually easy to identify at an early stage whether a router will connect to POTS telephones, a key system (KSU), a private branch exchange (PBX), or a voice trunk provided by a carrier. It may be more onerous to determine what interfaces are available on the phone switches at each site, such as analog station cards or trunk cards. If your telecom contacts are competent, then you can work with them to determine the information you need. At least try to determine whether analog or digital interfaces will be used for routers connecting to phone switches or carrier-provided voice circuits. Make sure the telecom contacts know that proprietary digital phone sets cannot be connected to the voice ports on the router. A channel bank can interface to such phones, and connect to serial interfaces on the router for circuit emulation. Table 12-1 summarizes the types of interfaces that you may connect to Cisco multiservice routers.

Table 12-1 *Traditional Telephony Interfaces, and Corresponding Cisco Router Voice Interfaces*

If the PBX or other interface is a . . .	Then connect to this Cisco voice interface:
POTS telephone (residential phone)	FXS
Fax	FXS
Carrier provided: analog loop start/ground start (1 MB,and so on)	FXO
Carrier provided: digital T1/E1—CAS (DID, both-way, and so on)	Digital T1/E1
Carrier provided: digital T1/E1—PRI (to support E911, and so on)	Digital T1/E1—PRI

Table 12-1 *Traditional Telephony Interfaces, and Corresponding Cisco Router Voice Interfaces (Continued)*

If the PBX or other interface is a . . .	Then connect to this Cisco voice interface:
PBX/Key: analog station card (normally for fax or POTS phone)	FXO
PBX/Key: analog trunk card (normally for loop start/ground start to CO)	FXS
PBX/Key: analog E&M tie line (normally to remote PBX via CO)	E&M
PBX/Key: digital T1/E1—CAS or CCS	Digital T1/E1
PBX/Key: digital T1/E1—PRI	Digital T1/E1—PRI
Vendor-specific business set (proprietary digital phone)	none
Proprietary digital phone via telco channel bank (or other CSU)	Non-voice T1/E1 (use CES or PBX)

Often, you will have to ask questions of people who have no idea what you are talking about. When you try to find a more technical contact, you might get responses such as, "Our phone switch has been running fine for the last five years. I think the company that installed it went out of business." In these situations, your best bet is to partner with a PBX service company that has experience with the specific model of PBX or key system. In the absence of such partners, you might have to read many manuals. Worse yet, you might have to solve the enigma black box. If you are working with an ancient and questionable phone switch, it may be less expensive to purchase a new one than to spend a lot of time or money to connect it to a router. You can burn hour after hour troubleshooting, only to determine that the old phone switch is not behaving properly. "It seemed so close! It was working yesterday . . . " On the other hand, make sure you have configured everything properly (and reset interfaces) before you blame an old phone switch.

TIP If you are considering a new phone switch, it might not be a bad time to look at the Cisco Call Manager. The solution supports traditional calling features and new applications such as voice-mail/e-mail integration, using native IP telephones and simple GUI web interfaces for configuration.

Assuming you can work with the telecom contacts, and the existing phone switches seem functional, then you can proceed with the data collection. This is easier to manage if you complete a form for each site (or router). Using a form ensures that you do not forget to ask for key pieces of information, and provides written documentation of the project progress.

If problems arise during the subsequent implementation, this form may help identify what went wrong.

You are likely to have problems if you are not involved in the form-completion process. It is interesting to note that no matter how detailed, simple, or precise you make the telecom information form, you will still receive baffling results from some sources. Even though it seems like a good place to save time, your time is well spent if you work with each contact to complete the form. Your project will stay on track if you schedule time to work with each contact for this purpose. If you make each telecom contact responsible for completing the form, you will have several weeks of silence from some (or many) sources, followed by a form with blank spots and vague answers to multiple-choice questions. Do not ask how—it just happens.

Making Early Estimates

You may be pressured for cost estimates and budgetary approvals before you have enough information to develop an accurate bill of materials. If this is the case, you can determine budgetary pricing (but not an accurate bill of materials) before you know the exact PBX interfaces with which the routers must connect. You still need to know the basic router platforms, which VoX technology will be used, the number of telephony interfaces required, and whether the interfaces are analog or digital. Because gathering the exact PBX interface information is often time-consuming, you can hasten the budget-approval process by pricing the analog hardware with arbitrary analog voice interface (for example, FXS). The equipment order should not be placed until the actual bill of materials is ready—after the PBX interface information has been gathered. You must take pains to ensure that nobody places an equipment order based on the budgetary bill of materials. I have seen it happen for 50 routers.

Selecting a VoX Technology

There are at least two ways to decide whether VoFR, VoATM, or VoIP is appropriate for your network: (1) compare the relative merits and shortcomings of each technology and weigh these factors based on their importance in your network, or (2) just pick VoIP.

If you choose the more thought-intensive approach, you should consider the following criteria for the decision:

- Reliability
- Scalability
- Quality of Service
- Cost and Complexity
- Feature Support
- Existing WAN Environment

VoFR, VoATM, and VoIP are compared against each of these criteria in the following subsections.

Reliability

Because both VoFR and VoATM are link-layer technologies, they are sensitive to circuit failures. Redundant circuits increase the chance of successful call connections, but any active VoFR or VoATM calls are terminated when a circuit fails. VoIP performs better in this respect because it operates at the network layer. IP packets may be rapidly rerouted around a failed circuit, without causing active VoIP calls to be dropped.

While VoIP is resilient to circuit failures, it is sensitive to routing problems and configuration errors. Both the signaling and audio portions of VoIP rely on the existing routing protocols in the IP network. If a new device that falsely advertises routes is added to the network, then valid destinations may become unreachable. Any routing problems in the network may impact VoIP connections, even if the problems are not caused by VoIP routers. VoFR and VoATM are less likely to be affected by changes in unrelated parts of the network.

If your IP network is stable, then VoIP offers the best overall reliability.

Scalability

The main scalability issue for enterprises with VoX technologies is the management of dial peers and call routing. There are two aspects to managing dial peers that must be considered as networks grow:

- End-to-End Versus Hop-by-Hop Peering
- Number of Dial Peers per Router

End-to-End Versus Hop-by-Hop Peering

Even with default dial peers, VoFR and VoATM require more configuration and maintenance than VoIP. VoFR and VoATM dial peers must be configured at every router along a call path, whereas VoIP only requires the end routers associated with the call to have dial peers. VoIP is simpler in this respect because it builds on the services offered by IP routing protocols that are already a part of the network.

The preceding discussion assumes that PVCs are used for ATM, which is not the most scalable option. The ATM PNNI protocol can operate with E.164-based ATM addresses when creating SVCs, which allows dynamic call routing. However, this approach does not appear to be making significant advances in the market. Most companies that use ATM in the WAN have PVCs; few companies use SVCs between sites over carrier-provided Permanent Virtual Paths (PVPs). Companies can implement an E.164-based SVC solution between their own sites, but generally these networks are isolated from other companies and the PSTN as a whole. Because the technology is not ubiquitous, scalability is limited by market adoption.

Number of Dial Peers per Router

For large networks, it is not feasible to statically configure dial peers in each router. It is difficult to manage the consistency of dial-peer configurations across multiple boxes, the router configuration memory is limited, and pattern searches become inefficient as the number of dial peers increase. Hub-and-spoke networks can extensively use default call routing, but the central site routers may still be burdened. Hierarchical networks can distribute the dial peers, but still require maintenance of the dial peers in many different routers.

For VoFR and VoATM, the hierarchical approach (of which the hub-and-spoke design is a subset) enables call-routing summarization, and is the best option to scale. VoIP can take advantage of H.323 gatekeepers or SIP proxy servers, which enable dial peers to be hierarchical without forcing the VoIP signaling and audio paths across the same hierarchy. The signaling and audio paths of the call are still optimized according to IP routing, so there is no performance degradation for centralizing the dial-peer databases (other than the delay for the lookup request). The ENUM Working Group of the Internet Engineering Task Force (Transport Area) is currently defining a distributed database standard that enables mapping between E.164 telephone numbers (hence the name ENUM) and URLs, followed by a DNS resolution to an IP address. The ENUM telephone number mapping, and the Telephony Routing over IP (TRIP) protocol, will enable VoIP–call-routing scale to the entire Internet.

Quality of Service

Frame Relay offers rudimentary mechanisms to provide quality-of-service assurances. The Committed Information Rate (CIR) provides a working bandwidth guarantee, assuming the carrier does not oversubscribe its backbone too much. However, there is no guarantee for delay or delay variation. During periods of network congestion, latency across a Frame Relay PVC may increase by a factor of 20 or more (based on personal observation of the author). A consequence of this behavior is that enterprises are not empowered to provide end-to-end QoS for VoFR. You can ensure that your equipment is configured properly to enable the best performance for VoFR, but if you have an uncooperative carrier, then voice quality may suffer. Be wary of situations where VoFR traffic must cross multiple carriers' Frame Relay networks, because there is less accountability for poor performance. If you pursue a VoFR solution, you must closely monitor your frame relay provider to ensure that you are receiving the contracted CIR with a reasonable latency.

ATM offers a clear and compelling quality-of-service solution. QoS is a pillar of ATM network design, offering guarantees for bandwidth, delay, and delay variation. ATM is an excellent technology to provide real-time quality of service along with standard traffic requirements. Both VoATM and VoIP can take advantage of the QoS features of ATM.

Current IP QoS solutions meet the needs of VoIP in the enterprise. It is important to realize, however, that end-to-end QoS in an IP network relies heavily on link-layer QoS. It does not matter how high the IP precedence field is set if the packet is stuck in a congested Frame Relay network. For this reason, you must be careful when selecting a link layer for the WAN. The best options are currently ATM or leased lines. In the LAN, excess bandwidth has traditionally been the best form of QoS, but IEEE 802.1p and the RSVP Subnet Bandwidth Manager (SBM) standards now provide other options.

QoS options for the Internet are rapidly evolving. Internet telephony service providers (ITSPs) now offer VoIP-enabled backbones for their customers, but the contracts between service providers are still developing. Clearinghouse and settlement services allow ITSPs to significantly extend the reach of their QoS network, which enables subscribers to place native VoIP calls to more destinations.

Cost and Complexity

VoFR, VoATM, and VoIP all have similar requirements for interfacing with traditional telephony equipment. The differences in cost and complexity are mainly attributed to assuring QoS.

Frame Relay is an economical WAN technology for voice, provided that QoS requirements can be met. Cisco hardware options are available for small installations, and products can scale to large enterprise deployments. For companies that want to switch to VoIP, Cisco products offer a software-only upgrade path from VoFR solutions to VoIP solutions. The conceptual issues for Frame Relay QoS are somewhat challenging, but the actual router configurations are less difficult.

ATM for voice applications is more expensive than Frame Relay, but you get what you pay for. The built-in QoS guarantees across the WAN reduce the likelihood of stumbling blocks during the implementation phase. If the routers are configured correctly and the network is designed properly (that is, there are minimal tandem codecs), then VoATM quality should be good.

VoIP solutions require more router configuration commands than VoFR or VoATM solutions. In addition to configuring link-layer QoS (that is, FRF.12 fragmentation or ATM CoS parameters), VoIP requires network and transport layer QoS options such as RSVP, special queuing techniques, IP precedence, WRED, and so on. While these features require additional experience from a design and implementation perspective, they offer flexibility beyond what VoFR or VoATM can provide. A unified VoIP network can span heterogeneous link-layer technologies.

Feature Support

It is important to consider not only the traditional PBX features, but also new features and applications that are emerging as communication technologies advance.

Traditional Telephony Features

There are few inherent differences between VoFR, VoATM, and VoIP with respect to telephony feature support. Integrated voice and data networks in general support fewer telephony features than proprietary single-vendor telecom environments. Integrated voice/data networks often sacrifice some features across the WAN in favor of reducing toll costs. If your company already uses PBX equipment from a variety of vendors, many features are not available regardless of voice/data integration.

Emerging MultiService Applications

While VoFR and VoATM will remain as technologies for integrating legacy voice networks, VoIP is developing as a native end-to-end solution. Traditional telephony features are supported in hardware and software versions of IP telephones, and multiservice applications are being integrated with these products. Companies that use VoIP to integrate their legacy voice networks with their data networks will be positioned to take advantage of new multimedia applications and communications tools. There is a clear, long-term, strategic advantage to using VoIP for voice/data integration.

Existing WAN Environment

ATM, ISDN, and clear channel TDM facilities provide good performance for VoX technologies. Because these services are circuit-switched technologies, they ensure consistent delay characteristics, which is important for real-time traffic. Frame Relay can work well for real-time traffic, but it is a risky proposition. You have no assurances that performance will remain good for VoFR or VoIP across Frame Relay.

Consideration of the existing WAN environment was very important before VoIP supported FRF.12 and hardware options matured. Still, your WAN environment should heavily influence your selection of a VoX technology. The following WAN types are considered here:

- Frame Relay
- ATM
- Clear Channel TDM
- ISDN

Frame Relay

When this book was conceived, VoFR was the only reasonable option for networks with a Frame Relay WAN. Since Cisco has incorporated support for FRF.12 fragment/interleaving into VoIP platforms, this advantage has disappeared. VoIP can now provide the same level of QoS as VoFR across frame relay networks.

ATM

VoATM and VoIP are available for networks with an ATM WAN. There is no difference in quality of service between VoATM and VoIP over ATM, but there is an important bandwidth and efficiency consideration. If IP RTP header compression is not available for ATM interfaces, then VoATM is more bandwidth-efficient than VoIP across an ATM network. This is because 40 bytes of IP/UDP/RTP header information are included with the data in each VoIP packet, which means that at least two ATM cells are required to transmit each VoIP packet. VoATM encapsulates the voice coder output directly in a single ATM cell.

Another bandwidth issue arises because ATM cells carry a fixed 48-byte payload (minus 1- or 2- byte AAL headers). If the transported data does not fill the cell, then the additional payload space is padded. Considering that a typical VoIP packet (with two G.729 samples) is 60 bytes long without header compression, then it must be segmented into two ATM cells. The second cell carries about 12 to 14 bytes of data, with more than 40 padded bytes! This yields an ATM payload efficiency of 62.5 percent for a VoIP packet. VoIP does support a configurable number of codec samples per packet, and ATM payload efficiency climbs to 94 percent when VoIP is properly tuned. For example, five samples of G.729 at 10 bytes each, along with 40 bytes of IP/RTP/UDP header, yield 90 bytes of data in two ATM cells. Using the G.723.1 codec with VoIP, which would ostensibly save bandwidth, results in an ATM payload efficiency of only 73 percent. Because G.723.1 uses a 30-ms frame, placing multiple frames into a packet to improve the ATM payload efficiency is not feasible because of the additional packetizing delay incurred.

You must address the ATM payload efficiency issue for both VoIP and VoATM, or you might waste much of your WAN bandwidth, support fewer simultaneous calls, and telephony users will experience reduced Grade of Service (GoS).

Leased-Line TDM

For TDM circuits (for example, T1/E1/J1/Y1), any VoX option may be used. Private ATM across leased lines is usually not a good option, because ATM cell headers consume much of the bandwidth. You should only consider VoATM on these circuits if you also have applications such as video using ATM-CES. VoFR was a good option when VoIP hardware options were limited, but now there is no compelling reason to use it. VoIP is a good option because it has low overhead (when using RTP header compression), requires fewer dial peers, and is flexible to integrate with other WAN technologies. On low-bandwidth TDM circuits, VoIP requires Frame Relay or multilink PPP encapsulation to provide fragment/interleaving.

If other parts of your network use a given VoX technology, you should match that technology on leased-line TDM circuits. This will allow you to keep an integrated dial plan with compatible dial peers. When call passing between VoX technologies is available, this will not be necessary. By the time you read these passages, it may already be available.

ISDN

Many companies use ISDN as a backup solution to their primary frame relay, ATM, or clear channel circuits. In some countries—Japan for example—permanent ISDN connections are available. The only VoX technology that is reasonable for these environments is VoIP, which can run with PPP encapsulation. VoIP is a very attractive option when redundancy is required, because ISDN can economically provide fault tolerance. VoFR and VoATM are not compatible with ISDN.

Table 12-2 summarizes the attributes of VoFR, VoATM, and VoIP with respect to the criteria discussed in the preceding sections.

Table 12-2 *General VoX Attributes*

		Vox Technology	
Attributes	**VoFR**	**VoATM**	**VoIP**
Reliability	OK	OK	Good
Scalability	Poor	OK	Good
Quality of Service	Poor	Good	Good
Cost	Good	OK	Good
Complexity	Good	OK	Poor
Support of Telephony Features	OK	OK	Good
Emerging Applications	Poor	OK	Good

Planning Voice Trunk and Bandwidth Requirements

The object of telecom traffic planning is to determine an optimal number of voice trunks to a destination, such that a certain call success rate is achieved during peak traffic intervals. The standard models used in the telecom industry are statistical models developed by A.K. Erlang at the beginning of the 20th century:

- Erlang B
- Extended Erlang B
- Erlang C

Model Assumptions and Applicability

Each of these models assumes that time between received call attempts is random with a Poisson distribution. These models do not apply to environments that receive spikes of call traffic, such as radio call-in contests, or ticketing vendors when concerts come to town. Each of the models differs with respect to how calls are handled when the trunks in question are busy.

The *Erlang B* model is appropriate when there is an overflow path for busy trunks. For example, a PBX may reroute calls to the PSTN if the VoX trunks are all busy. This example assumes that the remote location has Direct Inward Dial (DID) trunks to facilitate direct PSTN rerouting, or else functionality is compromised (for example, a call destined for a specific person may reach an operator when routed via the PSTN). Another example is a PBX with standard tie lines as a backup to the VoX trunks.

The *Extended Erlang B* model is appropriate when there is no overflow path, and the caller hears a busy tone when the desired VoX trunks are busy. This scenario is common when a site PBX has VoX trunks for interoffice dialing, and no alternate routes through the PSTN (that is, remote sites do not have DID and operator intervention is not acceptable). This model accounts for the fact that many users will immediately redial when a call fails, which increases the amount of incoming traffic.

The *Erlang C* model is appropriate if calls are placed in a queue when the VoX trunks are busy. This model is applicable to call centers, which strive to maintain high utilization of call center agents and trunk facilities. Call centers are outside the scope of this book.

Using the Models

The Erlang models require that you provide some of the variables to solve for an unknown variable. Table 12-3 provides definitions for the variables and measurement units that are commonly associated with these models.

Table 12-3 *Definition of Common Terms for Telecom Trunk Planning*

Term	Definition
Erlang	A measure of call volume equal to 1 hour of aggregate traffic. Three calls of 20-minute duration yield 1 Erlang of call traffic.
Centi-Call Seconds (CCS)	100 seconds of calling traffic. 36 CCS = 1 Erlang. (Not to be confused with Common Channel Signaling CCS.)
Lines	The number of provisioned voice trunks that carry traffic. Each analog port is 1 line; a full T-1 CAS port is 24 lines.
Busy Hour Traffic (BHT)	Amount of call traffic (in Erlangs) that must be supported during a peak-traffic reference hour. Use high estimates for conservative trunk planning.
Blocking	Percentage of calls that cannot be accommodated because of busy trunks. A typical blocking design goal is 1 to 3 percent.
Recall Factor	When there is no overflow path for blocked calls, this is the percentage of calls that are immediately retried (for example, the end user redials the destination).

To determine the optimal number of Lines, the Erlang B model requires the Busy Hour Traffic (BHT), measured in Erlangs, and the Blocking fraction for calls attempted during the busy hour. The blocking fraction is a measure of Grade of Service (GoS). The Extended Erlang B requires the same input and the Recall Factor, which indicates how many people redial after hearing a busy tone.

The Erlang C model operates with slightly different variables. Instead of using the BHT measure, the model uses the number of calls per hour and the average call length. Instead of measuring a blocking factor, the model considers how long the callers must wait before speaking with an agent. Table 12-4 summarizes the form and application for each model type.

Table 12-4 *Applicability and Required Information for the Erlang Trunk Planning Models*

Model	Equation Form	Response When VoX Trunks Are Busy
Erlang B	Lines = f(BHT,Blocking)	Overflow to standard trunks or PSTN
Extended Erlang B	Lines = f(BHT,Blocking,Recall)	Terminate the call (user hears busy tone)
Erlang C	Lines = f(CallsPerHour,Duration,WaitTime)	Call added to a waiting queue (ACD system)

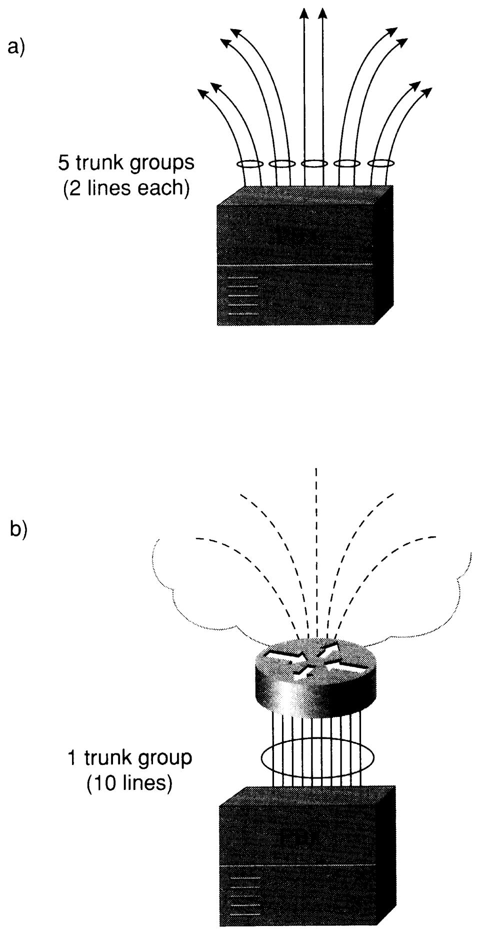
The equations to solve these models are not pretty, so network planners use reference tables that contain presolved values for the different traffic models. A few years ago, it would have been appropriate to include such a table as an appendix to this book. Now, there are numerous online tools that perform the calculation for you. Search the Web for Erlang calculator.

Adjusting the Models for VoX

The way these models apply to real networks is different for VoX networks than for traditional telecom networks. In traditional voice networks, each remote location is reachable via dedicated tie lines. There is no sharing between the lines (tandem switching is excluded for the moment to make a point). In VoX networks, the lines for all remote sites are pooled and connected to a router. Instead of having two lines to each of five different locations, ten lines are all connected to a router, which can appropriately direct the traffic. Figure 12-1 illustrates the difference between tie-line connections in traditional versus VoX networks:

The difference between independent and pooled lines affects the total number of lines required during peak loads. Consider a company with 2 lines dedicated to 50 different offices. A third call to a given office will fail, even if there are 98 idle lines! In a VoX network, all 100 lines could be provisioned for calls to any of the offices. A traditional network may require 3 lines to each office, a total of 150 lines, to accommodate the occasional peak. In the VoX environment, 100 lines may be more than sufficient to meet the total traffic requirements. The assumption here is that the peak traffic load to every destination does not occur at the same time. The ability to reduce the number of lines per site as more sites are added *(statistical multiplexing)* is a hidden benefit of integrated voice/data networks. The companies that benefit most are those with high traffic between many sites.

Figure 12-1 *Independent Tie Lines for Traditional Networks; Pooled Tie Lines for VoX Networks*



There is one concern caused by the pooled line strategy. Consider a hub-and-spoke network, where each of five remote PBXs have three lines connecting to a router, and the central site PBX has ten lines connecting to a router (taking advantage of statistical multiplexing). The first three calls between the central site and a remote site proceed normally, but all successive calls are routed in a suboptimal fashion. Figure 12-2 illustrates what happens for every call when the remote lines are busy:

1 The central PBX receives a call for the remote site and forwards it to the router, because there are free lines.

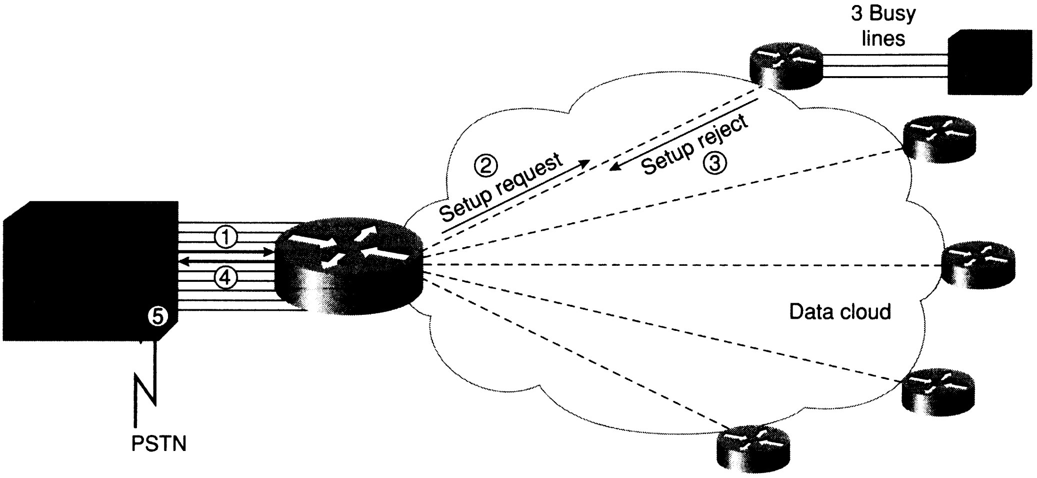
2 The central router accepts the call from the PBX, receives the digits, and attempts to establish the call with the remote router identified by the VoX dial peer.

3 Because the remote router does not have a free voice port (that is, they all are busy), it rejects the call setup request.

4 To prevent dropping the call, the central router should have an alternate dial peer pointing back to the PBX. The same digits cannot be sent back to the PBX because a routing loop would occur between the PBX and the router, which would seize all of the lines. The dial peer that points back to the PBX should add a prefix such as 91555 to the dialed digits, such that the PBX completes the call through the PSTN.

5 The PBX completes the call from the router to the PSTN. The call now occupies two lines between the router and PBX for the duration of the call.

Figure 12-2 *The Hairpin, Trombone, or Boomerang Effect When All Lines at a Remote Site Are Busy*



All calls that exceed the planned VoX capacity must exit the central PBX to the router, and immediately return to the PBX before routing to the PSTN. This situation is called a *hairpin*, *trombone*, or *boomerang*. It is undesirable because it uses two lines between the router and PBX for each call. There is no performance degradation from multiple codec cycles, because calls between local voice ports on a Cisco router are not compressed. But the wasted lines are an efficiency concern.

The solution to this problem is to have more communication between the router and the PBX, such that the path can be dynamically optimized (for example, cut the router out of the loop). This is a common feature between PBXs from the same vendor, but it has not historically been supported between vendors. QSIG is designed to fill this gap—to provide a standard telephony signaling protocol for vendor interoperability. QSIG must be supported in the PBX and in the router to take advantage of interoperable signaling.

Using the trunk or PLAR-OPX connection modes on Cisco routers is another way to solve this problem. The local router is able to refuse connection requests from the PBX when it knows the status of the remote VoX sites (via the PLAR-OPX or trunk signaling). This solution comes at the expense of dedicating voice ports on the router to specific remote destinations, which may not be the most efficient use of the voice ports.

Back to the trunk-planning issue. For VoX networks, the Erlang models should not be applied separately for traffic to each remote site. Rather, apply the models to the total traffic volume to all sites during the combined busy hour. Keep in mind that the busy hour for the combined traffic to all sites may not match the busy hour to any specific site. Applying the Erlang models to the pooled traffic (as opposed to independently for each site) yields a fewer number of required lines, in harmony with the statistical multiplexing advantage. This does not account for the hairpin effect, where two lines are required between the router and PBX for each call that is rerouted to the PSTN. To rigorously account for the hairpin effect, the Erlang traffic models should be modified.

In practice, you can observe the traffic that reroutes to the PSTN during the combined busy hour (for example, on a Cisco router, show dial peer for the peer that matches rerouted traffic, or review the call detail records), and perform an Erlang calculation on this traffic to determine the number of additional lines required for hairpinning. An alternative method is to make an educated guess about the number of extra lines required for traffic rerouted to the PSTN during the combined busy hour.

Converting Number of Trunks to Bandwidth

The Erlang calculations tell you how many voice ports you need on the router and PBX, but you must still determine the amount of bandwidth that each call consumes. This varies depending on the flavor of VoX and the codec. ATM efficiency is variable depending on the payload size. Refer to the section "Selecting a VoX Technology" in this chapter for a discussion of ATM payload efficiency.

The bandwidth calculations are not included here for every combination of codec type, samples per frame, VoX technology, and WAN transport. Instead, tools are provided to help you calculate these values yourself. You can calculate the overall bandwidth per the following equation:

$$(actual\_bandwidth) = (codec\_bandwidth) \times \frac{(payload\_length + encapsulation\_length)}{(payload\_length)}$$

This equation provides the amount of bandwidth required for each call, including the encapsulation overhead. The codec bandwidth values are provided in Table 12-5.

Table 12-5 *Bandwidth Requirements Per Call for Different Codecs*

Codec	Bandwidth (kbps)
G.711	64
G.723.1	6.3/5.3
G.726	16/24/32/40
G.728	16
G.729	8
G.729A	8

The configurable payload length must be an integer multiple of the codec sample size. This controls how many codec samples are placed in each cell, frame, or packet. Because the default values used in Cisco IOS may change, they are not provided here. You should adjust the default value if you need to:

- Increase ATM payload efficiency
- Decrease encapsulation overhead
- Reduce the effects of high cell/frame/packet loss rates

Table 12-6 summarizes the header lengths for various VoIP implementations. Add the number of bytes in the link-layer header to the number of bytes in whichever VoIP packet option you are using (for example, IP/UDP/RTP, CRTP with UDP checksums, CRTP without UDP checksums) to determine the total number of overhead bytes.

Table 12-6 *Header Length (Bytes) for VoIP over Different WAN Technologies*

WAN Technology	Link Layer	IP/UDP/RTP	CRTP	CRTP
			(UDP Checksums)	(No UDP Checksums)
HDLC	6-8	40	4	4
ML-PPP	7-9	40	4	4
Frame Relay	4	40	4	2
FRF.11 Annex C	9	40	4	2
FRF.12	8	40	4	2
ATM-AAL1	6-12	40	4	2
ATM-AAL5	13-18	40	4	2

ATM header requirements actually vary with the VoIP packet size. For all data segmented into AAL5, each cell introduces a 5-byte header, with the final cell requiring an 8-byte trailer. VoIP packets that fit into a single ATM cell payload require 5 + 8 bytes of ATM headers, and VoIP packets that fit into two ATM cell payloads require 5 + 5 + 8 bytes of ATM headers. This does not include any padding that is necessary to fill the payload field. The values in Table 12-6 assume that the IP header and codec data fit within one ATM cell when CRTP is used, and within two ATM cells when CRTP is not used.

Table 12-7 summarizes the header lengths for various VoFR implementations.

Table 12-7 *Header Length for VoFR in Different Configurations*

	Overhead (Bytes)
FRF.11 (Annex C fragmentation)	9
FRF.11 (FRF.12 fragmentation)	8
Cisco proprietary (voice-encap method)	6

VoATM is currently supported on the Cisco MC3810 router using AAL5 and AAL2. In addition to the 5-byte ATM header, there is a 4-byte VoATM header before the codec data. The ATM header lengths (for VoATM and VoIP) do not consider the payload padding, which increases the effective size of the header. You cannot determine the effective header length until you choose the payload size. Be sure to add the payload pad to the header length before calculating the required bandwidth.

Selecting Hardware to Meet Requirements

The surest way to make a book obsolete before it is published is to refer to the capabilities of specific hardware models. That being said, traditional telephony interfaces are available for the following Cisco routers as of this writing:

- 1750 Modular Access Router
- MC3810 Multi-access Concentrator
- 2600 series Modular Access Routers
- 3600 series Modular Access Routers
- AS-5300, AS-5800, and Access-Path Solutions
- 7200VXR series Core Routers
- 7500 series Core Routers

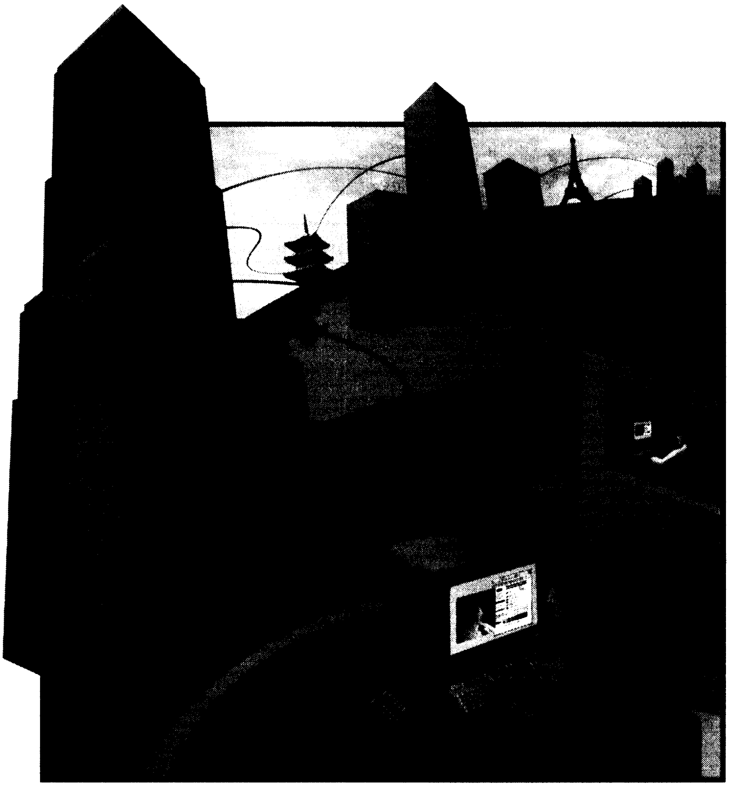
High-end ATM switches are not included here because the services they offer for voice are usually of the ATM-CES variety as opposed to VoATM. The distinction is made because the ATM-CES service provides a T1/E1 type of service, with no telephony intelligence required.

Small-scale implementations can support up to 4 analog ports (FXS/FXO/E&M) or 48 to 60 digital ports (2 T1/E1) in the 2600 series Modular Access Routers. The 3600 series offers up to three times the port density of the 2600 series, and the 7200 and 7500 routers provide higher density aggregation. The Access Server router line (for example, AS-5300, and so on) provides high-density T1/E1 telephony connections for VoIP, with value-adds like SS7 adjuncts and Interactive Voice Response (IVR).

As of this writing, there are various hardware caveats for supporting transparent pass-through of common channel signaling (CCS) and interpreted QSIG. For current hardware support and feature comparisons, you should look at Cisco Connection Online (CCO): www.cisco.com/. If you have not spent a lot of time there already, you are missing a truly incredible amount of free documentation. Cisco deserves a lot of credit for maintaining and constantly updating a wide range of product and technology information.

Reviewing Proposed Solutions in Terms of Requirements

After you have spent time developing the high-level integrated network design (that is, selecting the VoX technology, interfaces between routers and phone switches, bandwidth and trunk requirements, and hardware platforms), you must take a step back and verify whether you have met the design goals that were initially identified. Presumably, you have been mindful of the design goals at all phases of the early design, but it is a worthwhile task to review the requirements again before you commit to an equipment purchase.



from MPLS and VPN Architectures

by Ivan Pepelnjak and Jim Guichard

(1-58720-002-1)

Cisco Press

About the Authors

Jim Guichard is a senior network design consultant within Global Solutions Engineering at Cisco Systems. During the last four years at Cisco, Jim has been involved in the design, implementation, and planning of many large-scale WAN and LAN networks. His breadth of industry knowledge, hands-on experience, and understanding of complex internetworking architectures have enabled him to provide a detailed insight into the new world of MPLS and its deployment. If you would like to contact Jim, he can be reached at jguichar@cisco.com.

Ivan Pepelnjak, CCIE, is the executive director of the Technical Division with NIL Data Communications (www.NIL.si), a high-tech data communications company focusing on providing high-value services in new-world Service Provider technologies.

Ivan has more than 10 years of experience in designing, installing, troubleshooting, and operating large corporate and service provider WAN and LAN networks, several of them already deploying MPLS-based Virtual Private Networks. He is the author or lead developer of a number of highly successful advanced IP courses covering MPLS/VPN, BGP, OSPF, and IP QoS. His previous publications include *EIGRP Network Design Solutions*, by Cisco Press.

Contents at a Glance

Bold chapters are elements included in this folio.

[redacted]

CHAPTER 8

MPLS/VPN Architecture Overview

In the previous chapter, you learned about virtual private network (VPN) evolution; two major VPN models, overlay VPN and peer-to-peer VPN; and the major technologies used to implement both VPN models.

The overlay VPN model, most commonly used in a service provider network, dictates that the design and provisioning of virtual circuits across the backbone must be complete prior to any traffic flow. In the case of an IP network, this means that even though the underlying technology is connectionless, it requires a connection-oriented approach to provision the service.

From a service provider's point of view, the scaling issues of an overlay VPN model are felt most when having to manage and provision a large number of circuits/tunnels between customer devices. From a customer's point of view, the Interior Gateway Protocol design is typically extremely complex and also difficult to manage.

On the other hand, the peer-to-peer VPN model suffers from lack of isolation between the customers and the need for coordinated IP address space between them.

With the introduction of Multiprotocol Label Switching (MPLS), which combines the benefits of Layer 2 switching with Layer 3 routing and switching, it became possible to construct a technology that combines the benefits of an overlay VPN (such as security and isolation among customers) with the benefits of simplified routing that a peer-to-peer VPN implementation brings. The new technology, called MPLS/VPN, results in simpler customer routing and somewhat simpler service provider provisioning, and makes possible a number of topologies that are hard to implement in either the overlay or peer-to-peer VPN models. MPLS also adds the benefits of a connection-oriented approach to the IP routing paradigm, through the establishment of label-switched paths, which are created based on topology information rather than traffic flow.

NOTE This introduction might lead you to believe that any overlay VPN implementation can be replaced with an MPLS/VPN implementation. Unfortunately, that is not true. MPLS/VPN currently supports only IP as the Layer 3 protocol. Other protocols, such as IPX and AppleTalk, still must be tunneled across an IP backbone.

The MPLS/VPN architecture provides the capability to commission an IP network infrastructure that delivers *private* network services over a *public* infrastructure. This is the same type of service that has already been described in the previous chapter. However, the mechanisms used to provision the service are different. The MPLS/VPN technology is quite complex in itself and will be covered in a series of chapters. In this chapter, you'll see the basic MPLS/VPN concepts without going into too many details that would clutter the overall picture. In the next chapter, the detailed operation of MPLS/VPN is explained, along with the relevant configuration information to be able to provision a simple Intranet topology based on the MPLS/VPN architecture.

Case Study: Virtual Private Networks in SuperCom Service Provider Network

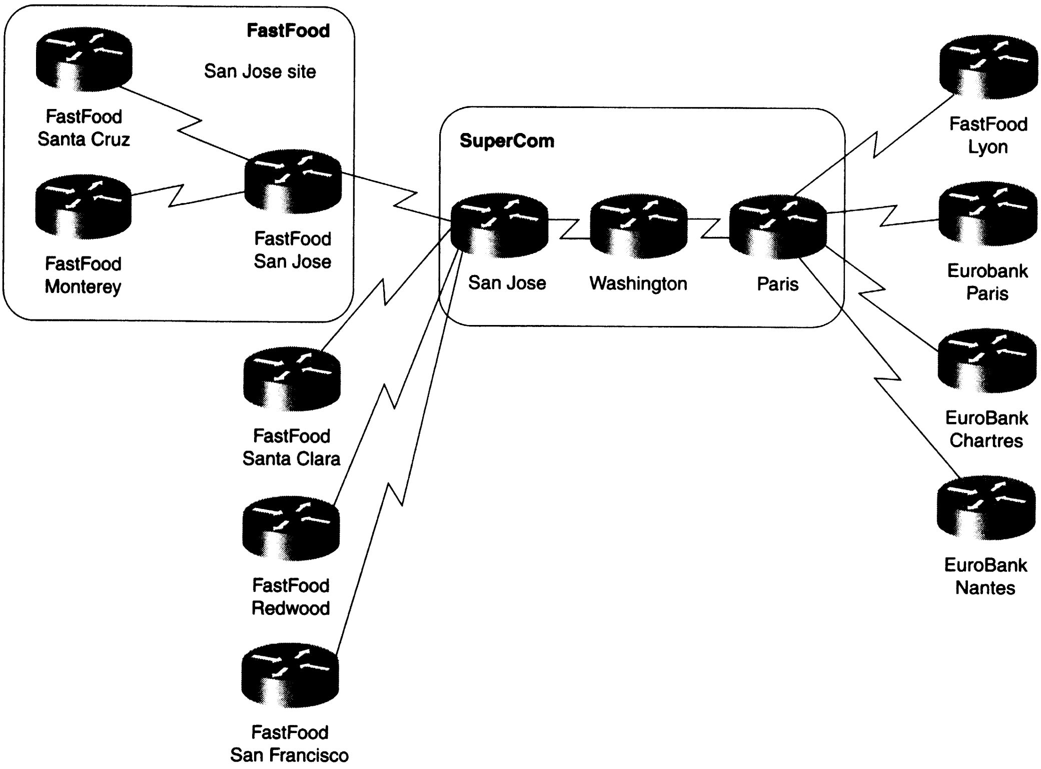
As with all complex topics, the MPLS/VPN concepts are best explained through use of a case study. Imagine a service provider (let's call it SuperCom) that is offering VPN services based on MPLS/VPN technologies. The service provider has two points of presence (POP), a U.S. POP in the San Jose area and a French POP in the Paris area. The POPs are linked through a core router located in Washington, D.C.

The service provider has two customers: FastFood, with headquarters in San Jose and branch offices in Santa Clara and Lyon; and EuroBank, with headquarters in Paris and branch offices in Chartres and San Francisco. The FastFood company has a number of other branch offices (for example, in Santa Cruz and Monterey) that are linked directly with the FastFood central site. The whole network is shown in Figure 8-1.

According to the terminology introduced in Chapter 7, "Virtual Private Network (VPN) Implementation Options," the routers in Figure 8-1 have the following roles:

- San Jose and Paris routers link the SuperCom network with its customers; they are thus provider edge (PE) routers.
- The Washington router does not have any customer connection; therefore, it's a provider (P) router.
- Customer routers connected to the SuperCom network—FastFood routers in San Jose, Santa Clara, and Lyon, as well as EuroBank routers in San Francisco, Paris, and Chartres—are customer edge (CE) routers.
- The FastFood routers in Santa Cruz and Monterey have no connection to the SuperCom network; they are customer (C) routers. All the networks connected directly to the FastFood San Jose site (Santa Cruz and Monterey networks) form a customer network (C-network) and represent a single site to the SuperCom network. The service provider does not care (and does not need to know) about the internal structure of that site.

Figure 8-1 *SuperCom Network and Its Customers*



Let's assume that both companies, FastFood and EuroBank, follow the same addressing convention—the central sites use public IP addresses, whereas all the remote sites use private IP address space (network 10.0.0.0).

NOTE The addressing scheme used by these corporations is seen more often in real customer networks, more so in cases in which the customer didn't acquire a significant portion of public IP address space several years ago.

The IP addresses used by these two companies are summarized in Table 8-1.

Table 8-1 *Address Space of FastFood and EuroBank*

Company	Site	Subnet
FastFood	San Jose	195.12.2.0/24
	Santa Clara	10.1.1.0/24
	Redwood	10.1.2.0/24
	Santa Cruz	10.1.3.0/24
	Monterey	10.1.4.0/24
	Lyon	10.2.1.0/24
EuroBank	Paris	196.7.25.0/24
	Chartres	10.2.1.0/24
	Nantes	10.2.2.0/24
	San Francisco	10.1.1.0/24

The SuperCom service provider would like to offer IP-based VPN service based on the peer-to-peer model (not a number of IP-over-IP tunnels), but it cannot do so easily because the address space of sites connected to the same router overlap.

NOTE The service provider would encounter a similar (but not so obvious) problem if the address space overlap occurred between customers connected to different POPs. The traditional peer-to-peer model requires strict uniqueness of IP address space.

SuperCom can traditionally solve the overlapping addresses issue in three ways:

- It can persuade the customers to renumber their networks. Most customers would not be willing to do that and would rather find another service provider.
- It can implement the VPN service with IP-over-IP tunnels, where the customer IP addresses are hidden from the service provider routers.
- It can implement a complex network address translation (NAT) scheme that would translate customer addresses into a different (but unique) set of addresses at the provider edge router and then translate those addresses back to the customer addresses before the packet would be sent from the egress PE-router to the CE router. Although such a solution is technically feasible, the administrative overhead is prohibitively large.

VPN Routing and Forwarding Tables

The overlapping addresses, usually resulting from usage of private IP addresses in customer networks, are one of the major obstacles to successful deployment of peer-to-peer VPN implementations. The MPLS/VPN technology provides an elegant solution to the dilemma: Each VPN has its own routing and forwarding table in the router, so any customer or site that belongs to that VPN is provided access only to the set of routes contained within that table. Any PE-router in an MPLS/VPN network thus contains a number of per-VPN routing tables and a global routing table that is used to reach other routers in the provider network, as well as external globally reachable destinations (for example, the rest of the Internet). Effectively, a number of virtual routers are created in a single physical router, as displayed in Figure 8-2 for the case of San Jose router of SuperCom network.

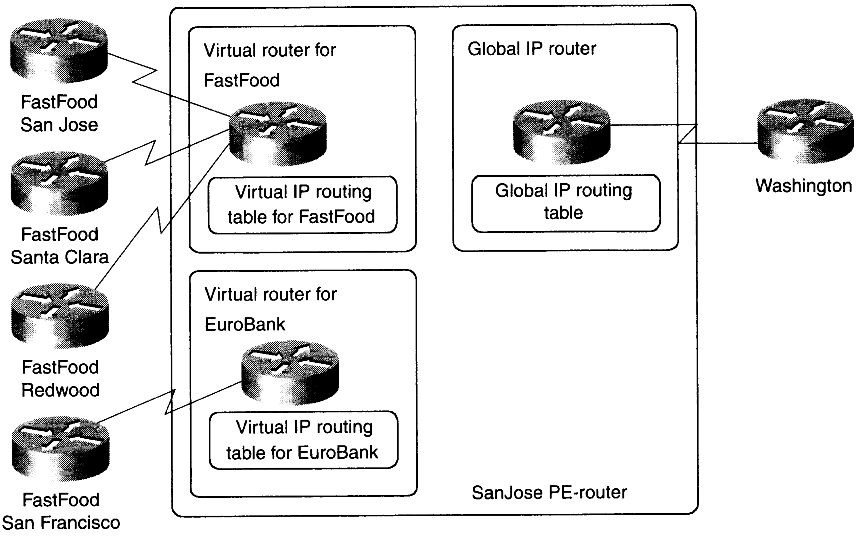
NOTE The relationship between virtual private networks and VPN routing and forwarding tables as explained in the previous paragraph is a slight simplification of the actual relationship between these two concepts. Nevertheless, it is true in cases where each site (or customer) belongs only to one VPN. The additional complexity introduced by overlapping VPNs or sites belonging to more than one VPN is explained in the section "Overlapping Virtual Private Networks," later in this chapter.

The concept of virtual routers allows the customers to use either global or private IP address space in each VPN. Each customer site belongs to a particular VPN, so the only requirement is that the address space be unique within that VPN. Uniqueness of addresses is not required among VPNs except where two VPNs that share the same private address space want to communicate.

More structures are associated with each virtual router than just the virtual IP routing table:

- A forwarding table that is derived from the routing table and is based on CEF technology.
- A set of interfaces that use the derived forwarding table.
- Rules that control the import and export of routes from and into the VPN routing table. These rules were introduced to support overlapping VPNs and are explained later in this chapter.
- A set of routing protocols/peers, which inject information into the VPN routing table. This includes static routing.
- Router variables associated with the routing protocol that is used to populate the VPN routing table.

Figure 8-2 *Virtual Routers Created in a PE-router*



The usage of these structures is explained in the rest of this chapter, and the detailed operation of each of them is explained in the next chapters.

The combination of the VPN IP routing table and associated VPN IP forwarding table is called VPN routing and forwarding instance (VRF).

NOTE

You might think that there is no difference between an IP routing table and an IP forwarding table—and usually that's true. In an MPLS environment, the only minor difference between them is the fact that the IP forwarding table also contains MPLS encapsulation information.

A major difference between the two tables arises in cases where an IP route refers to a next hop that is not directly connected. In that case, the routing table will contain the next-hop information, but not the outgoing interface or the IP address of the downstream router. The forwarding table will contain all the information needed to forward the packet toward the destination. For example, with the configuration in Example 8-1, the routing table lists the next hop for network 10.0.0.0/8 as 1.0.0.1 (as shown in Example 8-2), while the forwarding table contains the real next hop (the IP address of the downstream router), as shown in Example 8-3.

Example 8-1 *Sample Configuration with Recursive IP Routing*

```
ip route 10.0.0.0 255.0.0.0 1.0.0.1
ip route 1.0.0.1 255.255.255.255 2.0.0.2
!
interface serial 0
ip address 2.0.0.1 255.0.0.0
```

Example 8-2 *IP Routing Table for the Recursive IP Routing Example*

```
mpls router# show ip route
...
     1.0.0.0/32 is subnetted, 1 subnets
S       1.0.0.1 [1/0] via 2.0.0.2
C    2.0.0.0/8 is directly connected, Serial0
S    10.0.0.0/8 [1/0] via 1.0.0.1
...
```

Example 8-3 *CEF Forwarding Table Entry for Recursive IP Routing Example*

```
mpls router# show ip cef 10.0.0.0

10.0.0.0/8, version 87
0 packets, 0 bytes
  via 1.0.0.1, 0 dependencies, recursive
    next hop 2.0.0.2, Serial0 via 1.0.0.1/32
```

In the SuperCom case, the San Jose router contains three IP routing and forwarding tables—one table per customer and a global table used to forward non-VPN IP packets and to route VPN packets between PE-routers.

Overlapping Virtual Private Networks

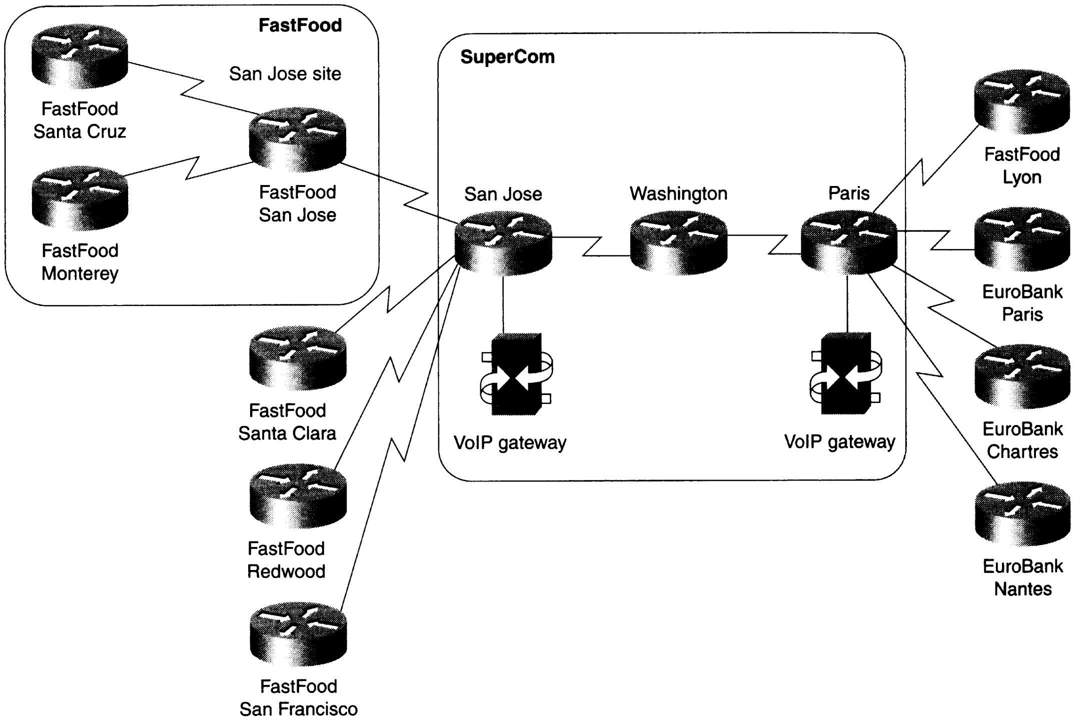
The SuperCom example might lead you to believe that a VPN is associated with a single VRF in a PE-router. Although that would be true in the case where the VPN customer needs no connectivity with other VPN customers, the situation might become more complex and require more than one VRF per VPN customer.

Imagine that SuperCom wants to extend its service offering with a Voice over IP (VoIP) service with gateways to the public voice network located in San Jose and Paris, as shown in Figure 8-3. The VoIP gateways were placed in a separate VPN to enhance the security of the newly created service. The IP addresses of these gateways are shown in Table 8-2.

Table 8-2 *IP Addresses of VoIP Gateways in SuperCom Network*

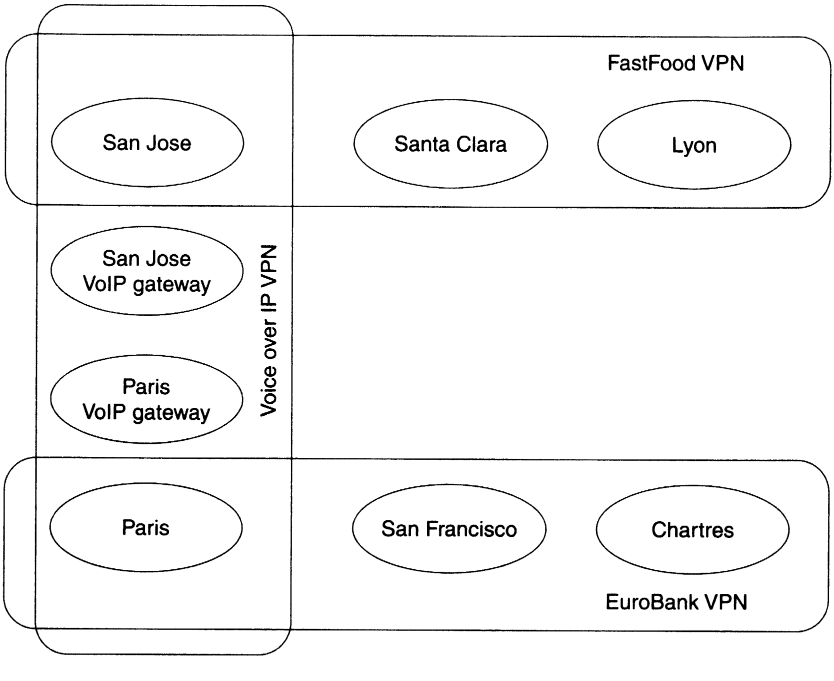
VoIP Gateway Location	VoIP Gateway IP Address
San Jose	212.15.23.12
Paris	212.15.27.35

Figure 8-3 *VoIP Gateways in SuperCom Network*



Both EuroBank and FastFood decided to use the service, but only from their central sites—the branch offices have no need for international voice connectivity. This requirement leads to an interesting problem: The central sites of both organizations need to be in two VPNs: the corporate VPN to reach their remote sites and the VoIP VPN to reach the VoIP gateways. The connectivity requirements are illustrated in Figure 8-4.

Figure 8-4 *VPN Connectivity Requirements in SuperCom Network*



NOTE The connectivity requirements in Figure 8-4 are a simplification of the requirements that you would encounter in a real service provider network. Most often, for security reasons, the customers using a common service (for example, VoIP gateways) will not see each other, but only the gateways or servers providing the service that they are using.

To support connectivity requirements similar to those in Figure 8-4, the MPLS/VPN architecture supports the concept of *sites*, where a VPN is made up of one or multiple sites. A VPN is essentially a collection of sites sharing common routing information, which means that a site may belong to more than one VPN if it holds routes from separate VPNs. This provides the capability to build intranets and extranets, as well as any other topology described in Chapter 7. A VPN in the MPLS/VPN architecture can therefore be pictured as a community of interest or a closed user group, which is dictated by the routing visibility that the site will have.

The VRF concept introduced in the previous section must be modified to support the concept of sites that can reside in more than one VPN. For example, the central site of

FastFood and EuroBank cannot use the same VRF as all other FastFood or EuroBank sites connected to the same PE-router. The central site of EuroBank, for example, needs to access the VoIP gateways, so the routes toward these gateways must be in the VRF for that site, whereas the same routes will not be in the Chartres' site VRF. Therefore, the MPLS/VPN architecture unbundles the concept of VRF from the concept of VPN. The VRF is simply a collection of routes that should be available to a particular site (or set of sites) connected to a PE-router. These routes can belong to more than one VPN.

NOTE You might be inclined at this moment to jump from a one-VPN-one-VRF model to the other extreme: one-site-one-VRF model. Although that model is theoretically correct and supports any VPN topology, it leads to more complex configurations of the PE-routers that are harder to maintain and that also use more memory. Therefore, it is recommended to keep the number of VRFs to a minimum (for example, one VRF for the customer's central site and another VRF for all remote offices connected to the same PE-router).

The relationship between the VPNs, sites, and VRFs can be summarized in the following rule, which should be used as the basis for any VRF definition in an MPLS/VPN network.

NOTE All sites that share the same routing information (usually this means that they belong to the same set of VPNs), that are allowed to communicate directly with each other, and that are connected to the same PE-router can be placed in a common VRF.

Using this rule, the minimum set of VRFs in the SuperCom network is the one outlined in Table 8-3.

Table 8-3 *VRFs in the PE-routers in the SuperCom Network*

PE-router	VRF	Sites in the VRF	VRF Belongs to VPNs
San Jose	FastFood_Central	FastFood SanJose site	FastFood, VoIP
	FastFood	FastFood Santa Clara site FastFood Redwood site	FastFood
	EuroBank	EuroBank San Francisco site	EuroBank
	VoIP	San Jose VoIP gateway	VoIP
Paris	FastFood	FastFood Lyon site	FastFood
	EuroBank_Central	EuroBank Paris site	EuroBank, VoIP

Table 8-3 *VRFs in the PE-routers in the SuperCom Network (Continued)*

PE-router	VRF	Sites in the VRF	VRF Belongs to VPNs
	EuroBank	EuroBank Chartres site EuroBank Nantes site	EuroBank
	VoIP	Paris VoIP gateway	VoIP

Route Targets

A careful reader might start asking an interesting question: If there is no one-to-one mapping between VPN and VRF, how does the router know which routes need to be inserted into which VRF? This dilemma is solved by the introduction of another concept in the MPLS/VPN architecture: the *route target*. Every VPN route is tagged with one or more route targets when it is exported from a VRF (to be offered to other VRFs). You can also associate a set of route targets with a VRF, and all routes tagged with at least one of those route targets will be inserted into the VRF.

NOTE The *route target* is the closest approximation to a *VPN identifier* in the MPLS/VPN architecture. In most VPN topologies, you can equate them, but in other topologies (usually a central services topology), a single VPN might need more than one route target for successful implementation.

NOTE The route target is a 64-bit quantity, the format of which is explained in the next chapter. For simplicity reasons, we will use names for route targets in this chapter.

The SuperCom network contains three VPNs and thus requires three route targets. The association between route targets and VRFs in the SuperCom network is outlined in Table 8-4.

Table 8-4 *Correspondence Between VRFs and Route Targets in SuperCom Network*

PE-router	VRF	Sites in the VRF	Route Target Attached to Exported Routes	Import Route Targets
San Jose	FastFood_ Central	FastFood SanJose site	FastFood, VoIP	FastFood, VoIP

continues

Table 8-4 *Correspondence Between VRFs and Route Targets in SuperCom Network (Continued)*

PE-router	VRF	Sites in the VRF	Route Target Attached to Exported Routes	Import Route Targets
	FastFood	FastFood Santa Clara site FastFood Redwood site	FastFood	FastFood
	EuroBank	EuroBank San Francisco site	EuroBank	EuroBank
	VoIP	San Jose VoIP gateway	VoIP	VoIP
Paris	FastFood	FastFood Lyon site	FastFood	FastFood
	EuroBank_ Central	EuroBank Paris site	EuroBank, VoIP	EuroBank, VoIP
	EuroBank	EuroBank Chartres site EuroBank Nantes site	EuroBank	EuroBank
	VoIP	Paris VoIP gateway	VoIP	VoIP

NOTE Based on Table 8-4, you might assume that the route targets attached to routes exported from a VRF always match the set of import route targets of a VRF. Although that's certainly true in simpler VPN topologies, there are widespread VPN topologies (for example, central services VPN) in which this assumption is not true.

Propagation of VPN Routing Information in the Provider Network

The previous sections have explained MPLS/VPN architecture from a single PE-router standpoint. Two issues have yet to be addressed:

- How will the PE-routers exchange information about VPN customers and VPN routes between themselves?
- How will the PE-routers forward packets originated in customer VPNs?

This section addresses inter-PE routing; the next section briefly describes the forwarding mechanism.

Two fundamentally different ways exist for approaching the VPN route exchange between PE-routers:

- The PE-routers could run a different routing algorithm for each VPN. For example, a copy of OSPF or EIGRP could be run for each VPN. This solution would face serious scalability problems in service provider networks with a large number of VPNs. It would also face interesting design challenges when asked to provide support for overlapping VPNs.
- The PE-routers run a single routing protocol to exchange all VPN routes. To support overlapping address spaces of VPN customers, the IP addresses used by the VPN customers must be augmented with additional information to make them unique.

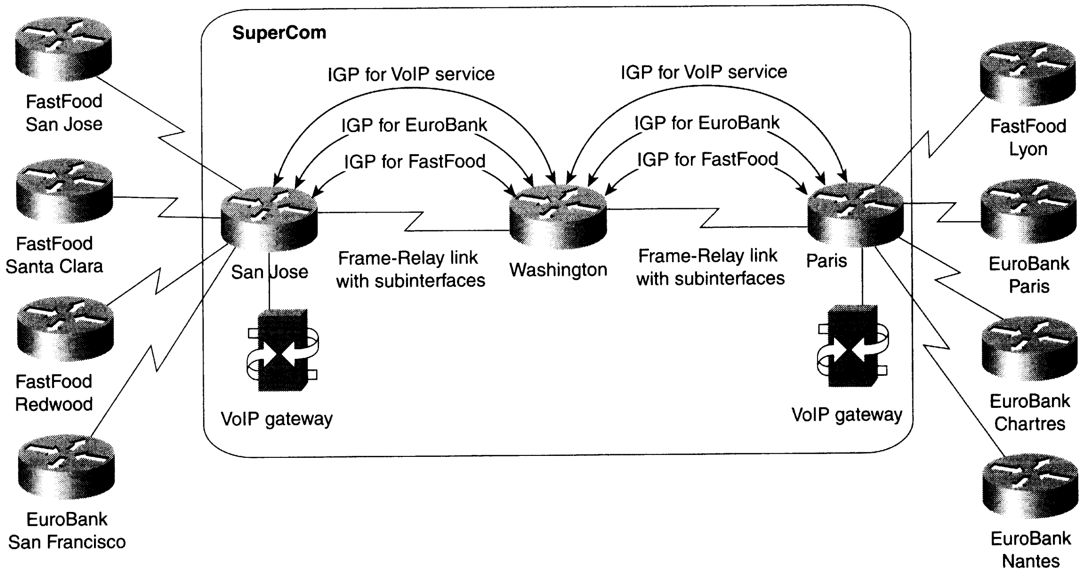
NOTE To illustrate the scalability issues that might arise from deploying one routing algorithm per VPN, consider the case where the SuperCom network would have to support more than 100 VPN customers connected to the San Jose and Paris routers with OSPF as the routing protocol. The PE-routers in the SuperCom network would run more than 100 independent copies of OSPF routing process (if that were technically possible), with each copy sending hello packets and periodic refreshments over the network. Because you cannot run more than one copy of OSPF over the same link, you would have to configure per-VPN subinterfaces (for example, using Frame Relay encapsulation) on the link between San Jose (or Paris) and Washington, resulting in an extremely complex network similar to the one shown in Figure 8-5. You would also have to run 100 different SPF algorithms and maintain 100 separate topology databases in the service provider routers.

The second approach was chosen as the building block of MPLS/VPN technology. IP subnets advertised by the CE-routers to the PE-routers are augmented with a 64-bit prefix called a *route distinguisher* to make them unique. The resulting 96-bit addresses are then exchanged between the PE-routers using a special address family of Multiprotocol BGP (hereby referred to as MP-BGP). There were several reasons for choosing BGP as the routing protocol used to transport VPN routes:

- The number of VPN routes in a network can become very large. BGP is the only routing protocol that can support a very large number of routes.
- BGP, EIGRP, and IS-IS are the only routing protocols that are multiprotocol by design (all of them can carry routing information for a number of different address families). IS-IS and EIGRP, however, do not scale to the same number of routes as BGP. BGP is also designed to exchange information between routers that are not directly connected. This BGP feature supports keeping VPN routing information out of the provider core routers (P-routers).

- GP can carry any information attached to a route as an optional BGP attribute. What's more, you can define additional attributes that will be transparently forwarded by any BGP router that does not understand them. This property of BGP makes propagation of route targets between PE-routers extremely simple.

Figure 8-5 *SuperCom Network with One IGP per VPN*

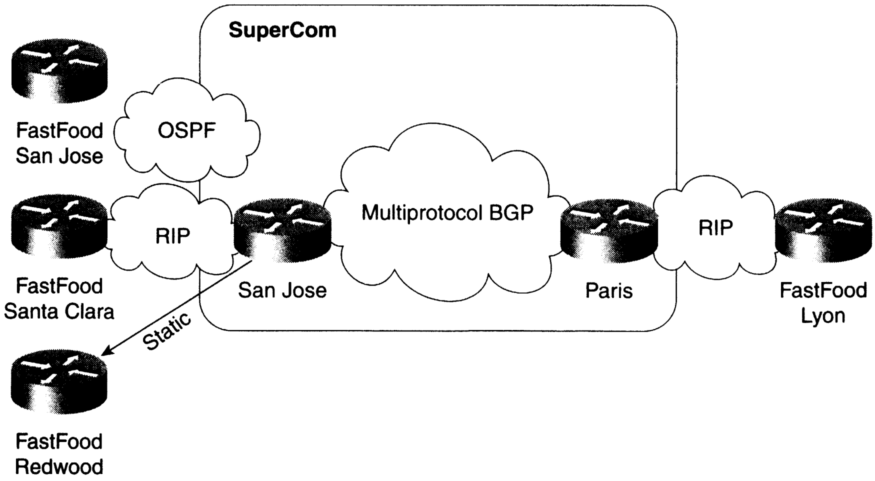


Multiprotocol BGP in the SuperCom Network

To illustrate the interaction of per-VPN routing protocols with the MP-BGP used in the service provider network core, consider the case of the FastFood customer in the SuperCom network. Let's assume that the San Jose site is using OSPF to interact with the SuperCom backbone, the Lyon and Santa Clara sites are using RIP, and the Redwood site is using no routing protocol—there is a static route configured on the San Jose PE-router and the default route configured on the Redwood router. The routing protocols used in FastFood VPN are shown in Figure 8-6.

NOTE The Washington router (the P-router in the SuperCom network) is not involved in the MP-BGP. As you'll see in the next section, the forwarding model used in MPLS/VPN does not require the P-routers to make any routing decisions based on VPN addresses; they just forward packets based on the label value attached to the packet. The P-routers, therefore, do not need to carry the VPN routes, resulting in even better scalability.

Figure 8-6 *Routing Protocols Used in FastFood VPN*



The San Jose PE-router collects routing information from the San Jose site using a per-VPN OSPF process. Similarly, the information from the Santa Clara site is collected using a per-VPN RIP process. This process is marked as Step 1 in Figure 8-7.

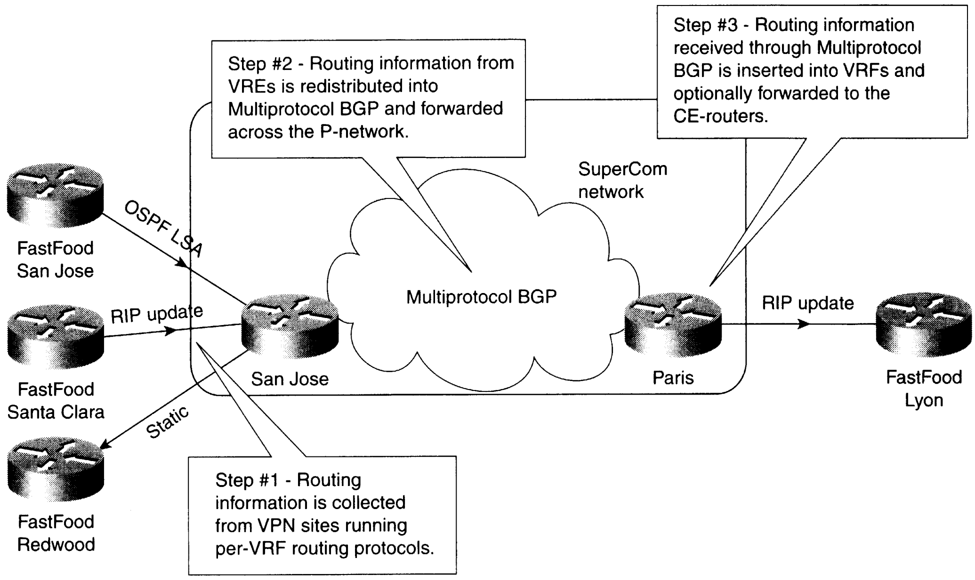
NOTE The routing protocol used within a VPN network must be limited to the VPN in question. If the same routing protocol would be used in different VPNs, the possibility of using overlapping IP addresses between VPNs would be lost, and there would be potential route leakage between VPNs.

To support overlapping VPNs, the routing protocol must be limited to a single VPN routing and forwarding (VRF) table. Each PE-router must be configured so that any routing information learnt from an interface can be associated with a particular VRF. This is done through the standard routing protocol process and is known as the *routing context.* A separate routing context is used per VRF.

Some routing protocols (for example, RIP) support several instances (or routing contexts) of the same protocol, with each instance running in a different VRF. Other protocols (for example, OSPF) require a separate copy of the routing protocol process for each VRF.

The information gathered by various routing protocols in the San Jose PE-router, as well as the static routes configured on the San Jose router, is redistributed into MP-BGP. VPN addresses are augmented with the route distinguishers at the moment of redistribution. The route export route target specified in the originating VRF is also attached to the route. The resulting 96-bit routing information is propagated by MP-BGP to the Paris router (Step 2 in Figure 8-7).

Figure 8-7 *Routing Protocol Operation in SuperCom Network*


Step #2 - Routing information from VREs is redistributed into Multiprotocol BGP and forwarded across the P-network.
Step #3 - Routing information received through Multiprotocol BGP is inserted into VRFs and optionally forwarded to the CE-routers.
SuperCom network
OSPF LSA
FastFood San Jose
RIP update
Multiprotocol BGP
RIP update
FastFood Santa Clara
San Jose
Paris
FastFood Lyon
Static
Step #1 - Routing information is collected from VPN sites running per-VRF routing protocols.
FastFood Redwood

WARNING The redistribution of the per-VPN routing information into MP-BGP is not automatic and must be manually configured on the router for each VRF (see Chapter 9, "MPLS/VPN Architecture Operation," for further details of this configuration), unless this information was learned from the customer via BGP. The omission of manual redistribution into MP-BGP is one of the most common configuration errors in MPLS/VPN deployment.

The Paris router, after receiving MP-BGP routes, inserts the received routes into various VRF tables based on the route target attribute attached to each individual route. The route distinguisher is dropped from the 96-bit route when the route is inserted into the VRF, resulting yet again in a traditional IP route. Finally, the routing information received through BGP is redistributed into the RIP process and is passed on to the Lyon site through RIP updates (Step 3 in Figure 8-7).

WARNING Similar to the redistribution of VRF routes into MP-BGP, the redistribution of routes received over the service provider backbone back into the per-VRF routing process is not automatic, unless this process is BGP; it must be manually configured if the redistribution is required by the routing design.

Contrary to the traditional BGP operation in which the internal BGP routes are not allowed to be redistributed into other routing protocols, this restriction is lifted in the MPLS/VPN environment. The VPN routes received by a PE-router through an internal MP-BGP session from another PE-router can be redistributed into other routing protocols.

VPN Packet Forwarding

In the previous section, you saw that the IP addresses used within a VPN must be prepended with a 64-bit prefix called a route distinguisher (RD) to make them unique.

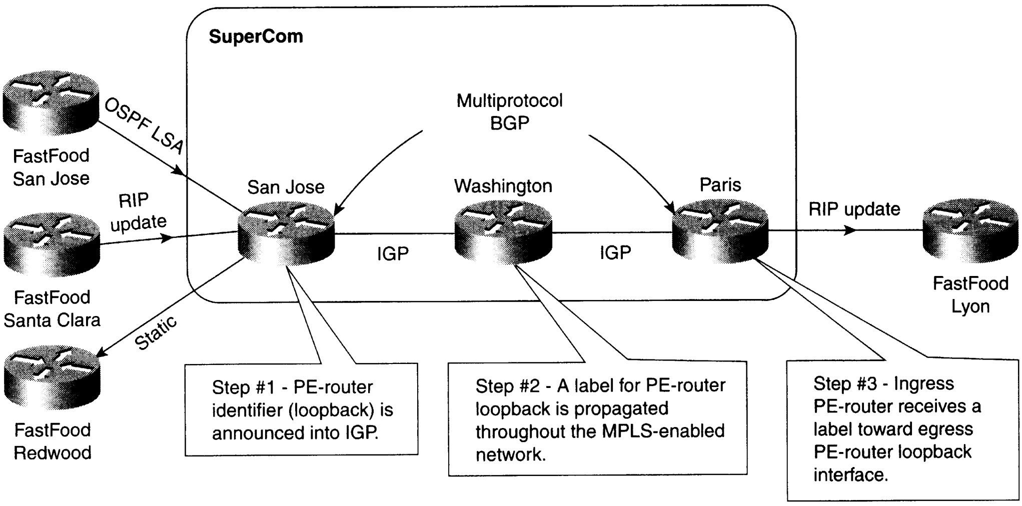
Similarly, when the VPN-originated IP packets are forwarded across the service provider backbone (the P-network), they must be augmented to make them uniquely recognizable. Yet again, several technology options are possible:

- The IP packet is rewritten to include 96-bit addresses in the packet header. This operation would be slow and complex.
- The IP packet is tunneled across the network in VPN-over-IP tunnels. This choice would make MPLS/VPN as complex as traditional IP-over-IP VPN solutions using the overlay VPN model.

With the introduction of MPLS, a third technology option was made possible: Each VPN packet is labeled by the ingress PE-router with a label uniquely identifying the egress PE-router, and is sent across the network. All the routers in the network subsequently switch labels without having to look into the packet itself. The preparatory steps for this process are illustrated in Figure 8-8.

Each PE-router needs a unique identifier (a host route—usually the loopback IP address is used), which is then propagated throughout the P-network using the usual IGP (Step 1). This IP address is also used as the BGP next-hop attribute of all VPN routes announced by the PE-router. A label is assigned in each P-router for that host route and is propagated to each of its neighbors (Step 2). Finally, all other PE-routers receive a label associated with the egress PE-router through an MPLS label distribution process (Step 3). After the label for the egress PE-router is received by the ingress PE-router, the VPN packet exchange can start.

Figure 8-8 *VPN Packet Forwarding—Preparatory Steps*


SuperCom
Multiprotocol BGP
OSPF LSA
FastFood San Jose
RIP update
FastFood Santa Clara
Static
FastFood Redwood
San Jose
Washington
Paris
IGP
IGP
RIP update
FastFood Lyon
Step #1 - PE-router identifier (loopback) is announced into IGP.
Step #2 - A label for PE-router loopback is propagated throughout the MPLS-enabled network.
Step #3 - Ingress PE-router receives a label toward egress PE-router loopback interface.

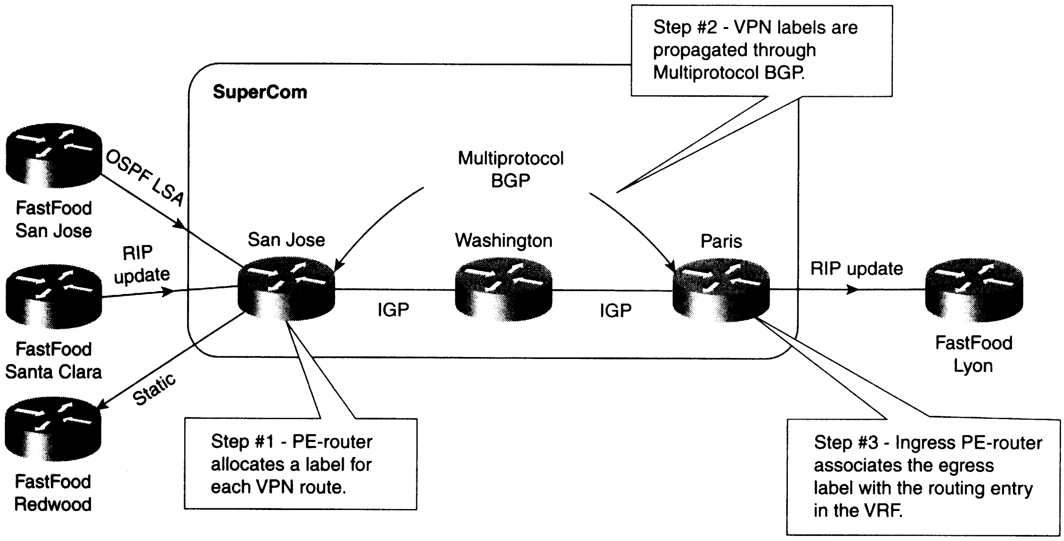
However, when the egress PE-router receives the VPN packet, it has no information to tell it which VPN the packet is destined for. To make the communication between VPN sites unique, a second set of labels is introduced, as illustrated in Figure 8-9.

Each PE-router allocates a unique label for each route in each VPN routing and forwarding (VRF) instance (Step 1). These labels are propagated together with the corresponding routes through MP-BGP to all other PE-routers (Step 2). The PE-routers receiving the MP-BGP update and installing the received routes in their VRF tables (see Figure 8-7 for additional details) also install the label assigned by the egress router in their VRF tables. The MPLS/VPN network is now ready to forward VPN packets.

When a VPN packet is received by the ingress PE-router, the corresponding VRF is examined, and the label associated with the destination address by the egress PE-router is fetched. Another label, pointing toward the egress PE-router, is obtained from the global forwarding table. Both labels are combined into an MPLS label stack, are attached in front of the VPN packet, and are sent toward the egress PE-router.

All the P-routers in the network switch the VPN packet based only on the top label in the stack, which points toward the egress PE-router. Because of the normal MPLS forwarding rules, the P-routers never look beyond the first label and are thus completely unaware of the second label or the VPN packet carried across the network.

Figure 8-9 *VPN Label Allocation*


Step #2 - VPN labels are propagated through Multiprotocol BGP.
SuperCom
Multiprotocol BGP
OSPF LSA
FastFood San Jose
RIP update
San Jose
Washington
Paris
RIP update
IGP
IGP
FastFood Santa Clara
FastFood Lyon
Static
FastFood Redwood
Step #1 - PE-router allocates a label for each VPN route.
Step #3 - Ingress PE-router associates the egress label with the routing entry in the VRF.

The egress PE-router receives the labeled packet, drops the first label, and performs a lookup on the second label, which uniquely identifies the target VRF and sometimes even the outgoing interface on the PE-router. A lookup is performed in the target VRF (if needed), and the packet is sent toward the proper CE-router.

NOTE The egress PE-router assigns labels to VPN routes in such a way that the need for additional Layer 3 lookup in the target VRF is minimized. The additional Layer 3 lookup is needed only for summary VPN routes advertised between the PE-routers.

The router just before the egress PE-router might also remove the first label in the label stack through a mechanism called *penultimate hop popping*. Refer to Chapter 2, "Frame-mode MPLS Operation," for a detailed description of this mechanism.

In the best case (no summary VPN routes and network topology that supports penultimate hop popping), the egress PE-router would perform only a single label lookup, resulting in maximum forwarding performance.

Summary

Virtual Private Networks (VPN) based on Multiprotocol Label Switching (MPLS) combine the benefits of the overlay VPN model, such as isolation and security, with the benefits of the peer-to-peer VPN model, such as simplified routing, easier provisioning, and better scalability. A number of mechanisms are needed to successfully meet all these goals:

- Each VPN needs a separate VPN routing and forwarding instance (VRF) in each PE-router to guarantee isolation and enable usage of uncoordinated private IP addresses.
- To support overlapping VPN topologies, the VRFs can be more granular than the VPNs and can participate in more than one VPN at a time. An attribute called a route target is needed to identify the set of VPNs in which a particular VRF participates. For maximum flexibility, a set of route targets can be associated with a VRF or attached to a VPN route.
- VPN IP addresses are prepended with 64-bit route distinguishers to make VPN addresses globally unique. These 96-bit addresses are exchanged between the PE-routers through MP-BGP, which also carries additional route attributes (for example, the route target) by means of optional BGP route attributes, called extended communities.
- Each PE-router needs a unique router ID (host route—usually the loopback address) that is used to allocate a label and enable VPN packet forwarding across the backbone.
- Each PE-router allocates a unique label to each route in each VRF (even if they have the same next hop) and propagates these labels together with 96-bit VPN addresses through MP-BGP.
- Ingress PE-routers use a two-level MPLS label stack to label the VPN packets with a VPN label assigned by the egress PE-router and an IGP label identifying the PE-router assigned through the regular MPLS label distribution mechanisms. The label stack is prepended to the VPN packet, and the resulting MPLS packet is forwarded across the P-network.