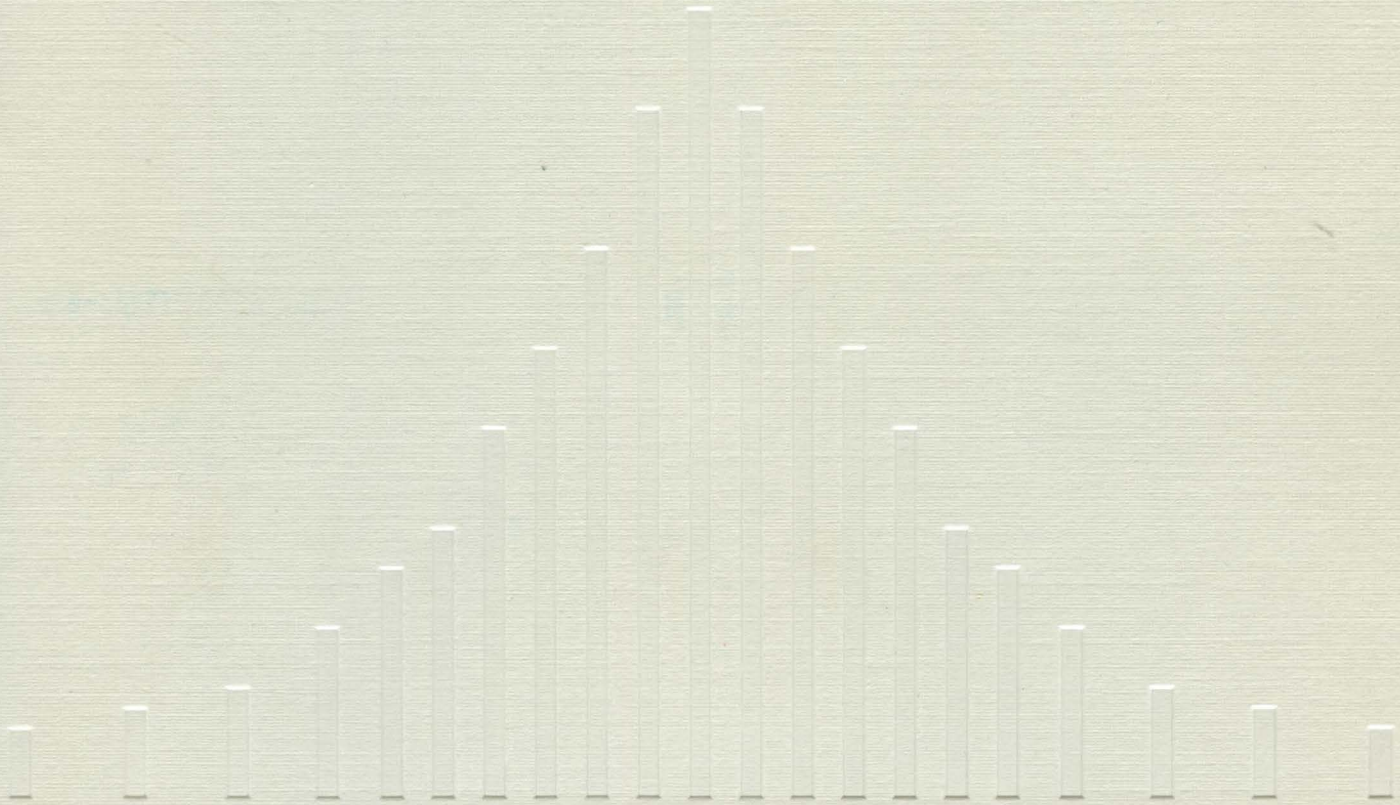
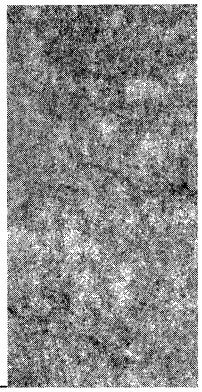
...the science of networking networks

Router Products Configuration and Reference Volume II



CISCO SYSTEMS



Router Products Configuration and Reference Volume II

Cisco Systems, Inc.
October 1991
Software Release 8.3

Corporate Headquarters:
Cisco Systems, Inc.
1525 O'Brien Drive
Menlo Park, California 94025 USA
1-415-326-1941
1-800-553-NETS
FAX 1-415-326-1989

Customer Order Number: DOC-R8.3
Cisco Document Assembly Number: 83-0017-01
Text Part Number: 78-0827-01

CISCO SYSTEMS

Router Products Configuration and Reference

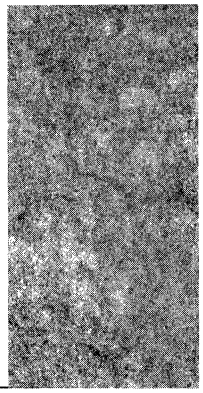
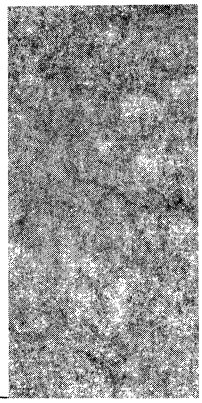


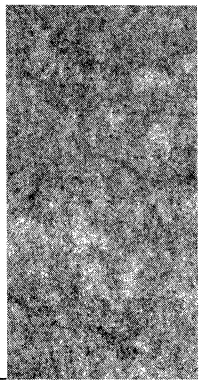
Table of Contents

Chapter 10 Routing AppleTalk 10-1

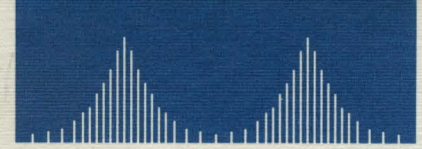
Chapter 16 Routing Novell IPX 16-1



List of Figures

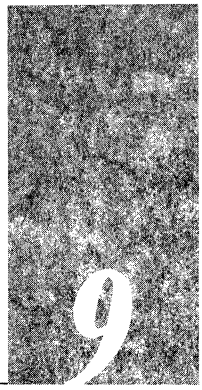


List of Tables

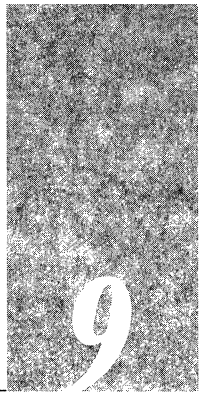


Part 4
Routing Configuration

Chapter 9
Routing Apollo Domain



Chapter 9
Routing Apollo Domain



This chapter describes how to configure Cisco Systems' implementation of the Apollo Domain routing protocol. Tasks and topics described in this chapter include:

- An overview of Apollo Domain
- How to enable Apollo Domain routing
- How to configure static routes, set update timers, and define access lists

Cisco's Implementation of Apollo Domain

The Apollo Domain routing protocol is the native-mode networking protocol for Apollo workstations. The Cisco routing software implementation supports packet forwarding and routing for the Apollo Domain network protocols on Ethernet, FDDI and serial interfaces using HDLC or X.25 encapsulation. Direct attachment to the 12-megabit Domain Token Ring is not supported.

The following restrictions apply to the Cisco implementation:

- If bridging is enabled on an Ethernet for which Apollo Domain routing is also enabled, special care must be taken. An Ethernet type code access list must be specified that filters out datagrams with the Apollo Domain type code (hexadecimal 8019). This restriction applies to MCI cards running microcode versions 1.5 or earlier.
- An IP address must be set on all media that use the ARP protocol (Ethernet and FDDI, for example). This is because Domain ARP uses the same Ethernet type value as IP ARP.
- The Cisco implementation of the Apollo Domain routing support assumes that it can use ARP to locate workstations on the local cable. Following are the versions of the Apollo operating system that support Domain ARP (D-ARP):
  - DN3000 and DN3010 nodes need version 9.7.4.1. This version of the operating system is available on a patch (ask for patch 186) from local Apollo field offices.
  - DN3500, 4000, and 4500 nodes need version 9.7.5.1 which is available on patch tape 185.
  - Version 9.7 (which provides ARP) for DN5xx-T nodes need version 9.7.4.b101. No patch is available for these machines; it is provided only on a DECnet tape.

Note: Version 10.0 does not provide ARP. You must migrate to version 10.1 and versions of the Apollo Domain operating system before successfully operating with Cisco routers. The **rtchk** and **lcnode** commands and D-ARP are not supported in Apollo's 802.5 implementation by Cisco routers.

Apollo Domain Addresses

Apollo Domain addresses are 20-bit quantities, represented by five-digit hexadecimal numbers. Each host has a single address which is used for all of its network connections.

An Apollo Domain host may have interfaces on more than one physical network (Ethernet, Domain Token Ring, serial line, and so on). Physical networks are identified by 32-bit numbers written in hexadecimal. These network numbers must be unique throughout an Apollo Domain internet. Since both the network number and the host address are needed to deliver traffic to a host, addresses are usually given as network numbers followed by host addresses separated with dots. An example would be:

```
5fe.1293c
```

Here the number `5fe` identifies the physical network, and the number `1293c` identifies the host as shown in Figure 9-1.

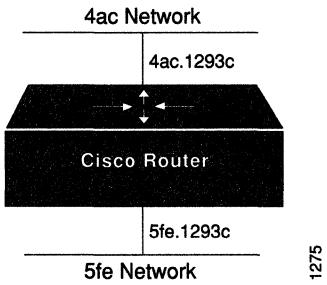


Figure 9-1 Apollo Domain Addresses

Configuring Apollo Domain Routing

There are only two commands required to enable Apollo Domain routing:

Step 1: Use the global configuration command **apollo routing** to enable routing.

Step 2: Use the interface subcommand **apollo network** to assign apollo routing to a specific interface.

All other configuration commands provide additional functionality or refinements. Each task is described in the following section. These descriptions are followed by applicable EXEC commands for monitoring and debugging Apollo Domain networks. Summaries of global configuration commands and interface subcommands described here appear at the end of this chapter.

Enabling Apollo Domain Routing

To enable or disable Domain routing and specify which system-wide host address to use, use the **apollo routing** global configuration command. The full syntax of this command follows.

apollo routing *address*

no apollo routing

The argument *address* is a unique, five-digit hexadecimal host address that you define, as described in the section "Apollo Domain Addresses." The **no apollo routing** command disables Apollo routing. See the next section for an example of how to enable Apollo Domain routing.

Assigning the Apollo Domain Network Numbers

Apollo Domain network numbers must be assigned to the appropriate interfaces. This is done using the **apollo network** interface subcommand. The full syntax of this command follows.

apollo network *number*

no apollo network

The argument *number* is an eight-digit hexadecimal number. The **no apollo network** command removes a network number from an interface.

Example:

The following is a very simple example of setting up Apollo Domain routing on a router with two Ethernet interfaces. The first step is to enable the RIP routing protocol and assign a Domain network address using the **apollo routing** command. The next step is to assign network numbers to the two interfaces.

```
apollo routing 23d5a
interface ethernet 0
apollo network 5f
interface ethernet 1
apollo network 4e
```

Configuring Static Routes

Specify static routes for an Apollo Domain network with the **apollo route** global configuration command. Full syntax of this command follows:

apollo route *network network.address*

no apollo route *network network.address*

Use of this command causes packets received for the specified network to be forwarded to the specified router (whose address is *network.address*), whether or not that router is sending out dynamic routing. Use the **no apollo route** command to remove the routes.

Example:

If the router that handled traffic for network 33 had the address *45.91ac6*, you would enter the following command.

```
apollo route 33 45.91ac6
```

Configuring Maximum Paths

To set the maximum number of multiple paths that the router will remember and use, use the **apollo maximum-paths** global configuration command. The command increases throughput by using multiple paths. It remembers higher bandwidth routes in preference to lower bandwidth routes. Full syntax of this command follows:

apollo maximum-paths *paths*

no apollo maximum-paths

The argument *paths* is the number of paths to be assigned. For a given destination, multiple paths of equal cost will be remembered. Output will be determined in round-robin fashion over these multiple paths at the packet level. The default value for *paths* is one; the **no apollo maximum-paths** command restores this default.

Example:

The following command sets three maximum paths.

```
apollo maximum-paths 3
```

The EXEC command **show apollo route** displays these additional routes and the maximum path value.

Setting Apollo Update Timers

To allow the Apollo Domain routing update timers to be set on a per-interface basis, use the **apollo update-timer** interface subcommand:

apollo update-time *seconds*

Internal Apollo Domain routing timers are affected by the value set for the *seconds* argument, as follows:

- Apollo Domain routes are marked invalid if no routing updates are heard within six times the value of the update timer.
- Apollo Domain routes are removed from the routing table if no routing updates are heard within eight times the value of the update timer.
- The default value for the *seconds* argument is 30.
- The minimum value of the *seconds* argument is 10 seconds.
- The granularity of the update timer is determined by the lowest value defined.

Example:

In the example below, the granularity of the update timer is 20 because that is the lowest value specified.

```
interface serial 0
apollo update-time 40
interface ethernet 0
apollo update-time 20
interface ethernet 1
apollo update-time 25
```

Note: Only use this command in an all-Cisco environment, and ensure that all timers are the same for all routers attached to the same network segment.

The EXEC command **show apollo interface** displays the value of these timers.

Configuring Apollo Domain Access Lists

Apollo access lists are a collection of permit and deny conditions that apply to defined Apollo network and host numbers. The router sequentially tests the network and host numbers against conditions set in the access lists.

The first match determines whether the router accepts or rejects the network and host number. Because the router stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the software rejects the network and host number.

Specifying Apollo Domain Access Lists

Use the **apollo access-list** global configuration command to specify an access condition. The full syntax of this command follows:

apollo access-list *name* {**permit** | **deny**} [*firstnet-*]*lastnet.host* [*wildcard-mask*]

no apollo access-list *name*

The argument *name* is a name defined by the network administrator for the access list. The **no apollo access-list** command removes an access list; use only the name, not all the possible parameters, when you remove the list.

Choose the permit or deny condition for this list using the **permit** or **deny** keyword.

You may define Apollo access lists for one or a selected range of Apollo networks, which are defined by network number and host number separated by a dot. The optional argument *firstnet* and the argument *lastnet.host* specify a selected network range. Use the argument *lastnet.host* to specify just one network.

The optional *wildcard-mask* argument is a wildcard mask that uses the one bits to ignore the host part of the network address. Host bits corresponding to wildcard mask bits set to zero are used in comparisons.

An access list can contain an indefinite number of actual and wildcard addresses. A wildcard address has a nonzero mask and thus potentially matches more than one actual address. The software examines the actual addresses, then the wildcard addresses. The order of the wildcard addresses is important because the software stops examining access list entries once it finds a match.

Protocol types and/or socket numbers are not useful in Apollo access lists. Also, note that Apollo access lists are named, not numbered as they are with other protocols.

Use the **no apollo access-list** command to delete the entire access list.

Example:

In the example below, the first line denies access to networks 3a through 3f, the second line denies access to a specific host and the third line permits everyone else.

```
apollo access-list eng deny 3a-3f.0 fffff
apollo access-list eng deny 5fe.1293c
apollo access-list eng permit 1-fffffff.0 ffff
```

Defining Access Groups

Use the **apollo access-group** interface subcommand to specify the interface on which the access list is defined. Full syntax of this command follows:

apollo access-group *name*

no apollo access-group *name*

Enter the user-defined *name* for the access list defined by the **apollo access-list** global configuration command for the argument *name.*

Upon receiving and routing a packet to a controlled interface, the software checks the source network and host number of the packet against that set in the access list. If the access list permits the address, the software transmits the packet.

You can specify ranges of network numbers, along with host masks. While the masks may not be useful, they permit the host part to be ignored entirely.

Use the **no apollo access-group** command to remove the name.

Example:

In the example below, the access list named *eng* is assigned to the first Ethernet interface.

```
interface ethernet 0
apollo access-group eng
```

Monitoring the Apollo Domain Network

Use the EXEC commands described in this section to obtain displays of activity on the Apollo Domain network.

Displaying Apollo Interface Parameters

Use the EXEC command **show apollo interface** to display Apollo Domain parameters that have been configured on the interfaces. Enter this command at the EXEC prompt:

show apollo interface [*interface unit*]

You may specify the optional *interface* and *unit* arguments to see information for just that interface.

Following is sample output, specifying the first Ethernet interface:

```
Ethernet 0 is up, line protocol is up
  Apollo address is 123A.CAFE
  Update time is 30 seconds
  Outgoing access list is not set
```

Displaying Apollo Routes

Use the EXEC command **show apollo route** to display the Apollo Domain routing table. Enter this command at the EXEC prompt:

show apollo route

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 1 learned routes

Maximum allowed path(s) are/is 1
C Net 123A is directly connected, 0 uses, Ethernet0
C Net 123B is directly connected, 0 uses, Ethernet1
R Net 123C [1/0] via 123A.CAFB,  4 sec, 0 uses, Ethernet0
```

In the display, the leading character `R` indicates routes learned via RIP, `C` indicates connected entries, and `S` indicates statically defined entries.

Displaying Apollo Traffic Statistics

Use the EXEC command **show apollo traffic** to display information on the number and type of Apollo Domain packets transmitted and received. Enter this command at the EXEC prompt:

show apollo traffic

Following is sample output:

```
Rcvd:   8 total, 0 format errors, 0 checksum errors, 0 bad hop count,
        8 local destination, 0 multicast
Bcast:  8 received, 0 sent
Sent:   16 generated, 0 forwarded
        0 encapsulation failed, 0 no route
        0 unknown
```

In the displays:

- `format errors` are reported whenever a "bad packet" is detected (for example, corrupted header).
- `checksum errors` should not be reported, since Apollo does not use a checksum.
- `bad hop count` increments when a packets hop count exceeds 16.
- `encapsulation failed` is registered when the router is unable to encapsulate a packet.
- `unknown` counter increments when packets are encountered that the router is unable to forward (for example, misconfigured helper-address, or no route available).

Displaying the Apollo ARP Table

Use the EXEC command **show apollo arp** to display that portion of the ARP table that pertains to the Apollo Domain Address Resolution Protocol. Enter this command at the EXEC prompt:

show apollo arp

Sample output follows:

```
Protocol  Address          Age (min)     Hardware Addr
Type      Interface
Apollo    123A.CAFE              -     0000.0c00.62e6
ARPA      Ethernet0
```

Debugging the Apollo Domain Network

Use the EXEC commands described in this section to troubleshoot and monitor the Apollo Domain network transactions. Generally, these commands are entered during troubleshooting sessions with Cisco engineers. For each **debug** command, there is a corresponding **undebug** command that turns the message logging off.

debug apollo-packet

The command **debug apollo-packet** outputs information about packets received, transmitted, and forwarded.

debug apollo-routing

The command **debug apollo-routing** prints out information on Apollo Domain routing packets.

Apollo Domain Global Configuration Command Summary

The following is an alphabetical list of the Apollo Domain global configuration commands, which specify system-wide parameters for Apollo Domain support.

[no] apollo access-list *name* {**permit** | **deny**} [*firstnet-*]*lastnet.host* [*wildcard-mask*]

Specifies Apollo Domain access condition. The argument *name* is a name defined by the network administrator for the access list.

Choose the permit or deny condition for this list using the **permit** or **deny** keyword. The optional argument *firstnet* and the argument *lastnet.host* specify a selected network range. Use the argument *lastnet.host* to specify just one network. The optional *wildcard-mask* argument is a wildcard mask that uses the one bits to ignore the host part of the network address. Host bits corresponding to wildcard mask bits set to zero are used in comparisons.

[no] apollo maximum-paths *paths*

Sets the maximum number of multiple paths that the router will remember and use. The argument *paths* is the number of paths to be assigned. The default value is one, which is restored with the **no** form of the command.

[no] apollo route *network network.address*

Specifies static routes for an Apollo Domain network. Packets received for the specified network will be forwarded to the specified router, whether or not that router is sending out dynamic routing.

[no] apollo routing *address*

Enables or disables Domain routing and specifies which system-wide host address to use. The argument *address* is a unique, five-digit hexadecimal host address.

Apollo Domain Interface Subcommand Summary

The following Apollo Domain interface subcommands specify line-specific parameters for Apollo Domain support. These subcommands must be preceded by an **interface** command.

[no] apollo access-group *name*

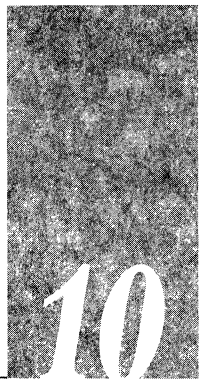
Specifies the interface on which an Apollo Domain access list is defined. Enter the user-defined *name* for the access list defined by the **apollo access-list** global configuration command for the argument *name*.

[no] apollo network *number*

Assigns Apollo Domain network numbers to the appropriate interfaces. The argument *number* is an eight-digit hexadecimal number.

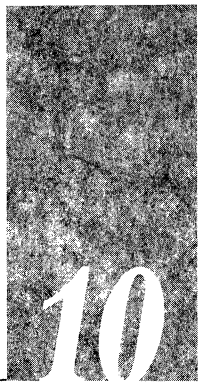
apollo update-time *seconds*

Sets the Apollo Domain routing update timers. The argument *seconds* specifies the interval between updates.



Chapter 10
Routing AppleTalk

Chapter 10
Routing AppleTalk



This chapter describes the routing process of the AppleTalk network protocol. The topics and tasks described in this chapter include:

- An overview of the AppleTalk routing protocol.
- Cisco's implementation of AppleTalk on both extended (also known as Phase II) and nonextended (Phase I) interfaces.
- Configuring AppleTalk routing.
- Configuring AppleTalk access list filters.
- Monitoring and debugging an AppleTalk network.

For more detailed information about the AppleTalk network systems, refer to Appendix F, "References and Recommended Reading."

Cisco's Implementation of AppleTalk

AppleTalk was designed as a client-server, or *distributed* network system. In other words, users share network resources, such as files and printers, with other users. Interactions with servers are essentially transparent to the user, as the computer itself determines the location of the requested material, and accesses it without requesting information from the user.

AppleTalk identifies several network entities, of which the most elemental is a *node.* A node is simply any device connected to an AppleTalk network. The most common nodes are Macintosh computers and laser printers, but many other types of computers are also capable of AppleTalk communication, including IBM PCs, DEC VAX systems and a variety of workstations. A router is considered a node on each connected network. To avoid confusion, these router nodes are referred to as *ports.* The next entity defined by AppleTalk is a *network.* An AppleTalk network is simply a single logical cable. Finally, an AppleTalk *zone* is a logical group of one or more (possibly noncontiguous) networks. These AppleTalk entities are shown in Figure 10-1.

Apple Computer has produced a variety of internetworking products with which to connect AppleTalk local area networks. Apple supports Ethernet, Token Ring, FDDITalk, and its own proprietary twisted-pair media access system (called LocalTalk). However, to allow an AppleTalk network full participation in a multiprotocol internet, a multiprotocol router is required.

All routers from Cisco Systems support the AppleTalk network protocol (both extended and nonextended) over FDDI, Ethernet, Token Ring, synchronous serial, and X.25 interfaces.

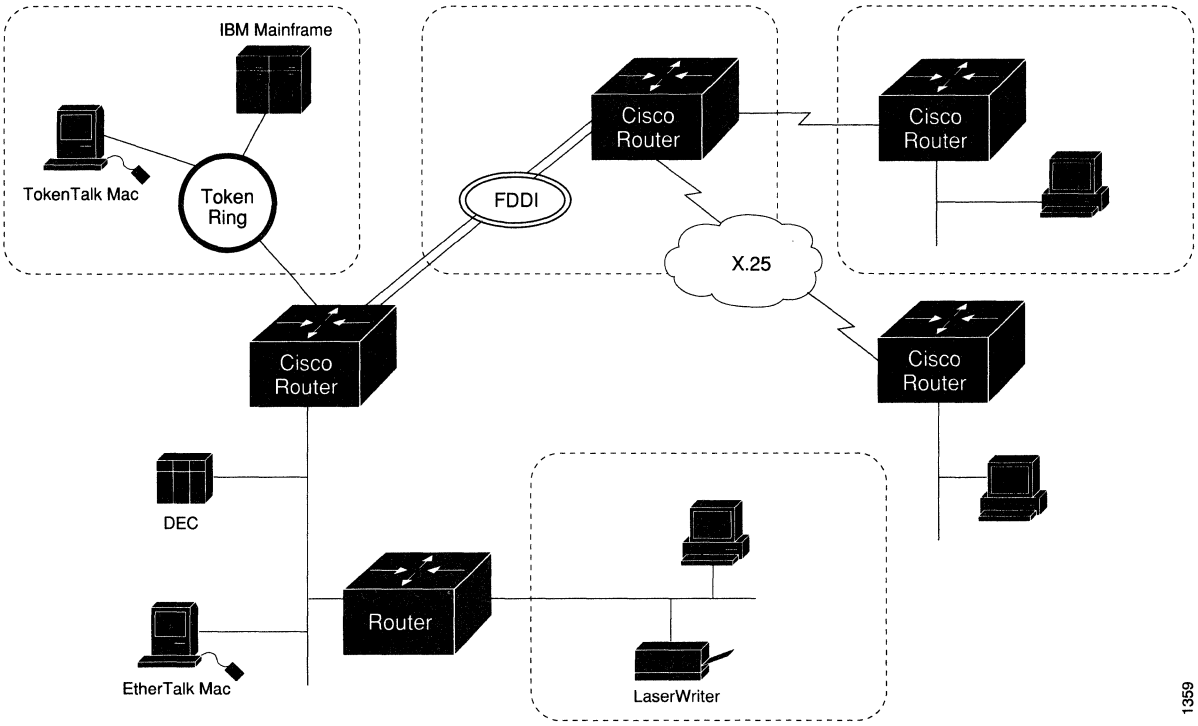


Figure 10-1 AppleTalk Entities

Figure 10-2 compares the AppleTalk protocols with the standard seven-layer OSI model, and illustrates how AppleTalk works with a variety of physical and link access mechanisms.

The Cisco AppleTalk implementation provides the following services:

- AppleTalk Address Resolution Protocol (AARP)
- Datagram Delivery Protocol (DDP)
- Routing Table Maintenance Protocol (RTMP)
- Name Binding Protocol (NBP)
- AppleTalk Echo Protocol (AEP)
- AppleTalk Transaction Protocol (ATP)
- Zone Information Protocol (ZIP)
- AppleTalk Filing Protocol (AFP)
- Print Access Protocol (PAP)

The DDP, RTMP, and AEP protocols provide end-to-end connectivity between internetworked nodes. NBP maps network names to AppleTalk internet addresses. NBP relies on ZIP to help determine which networks belong to which zones. File and print access is provided through AFP and PAP respectively, which work in concert with applications such as AppleShare and print servers.

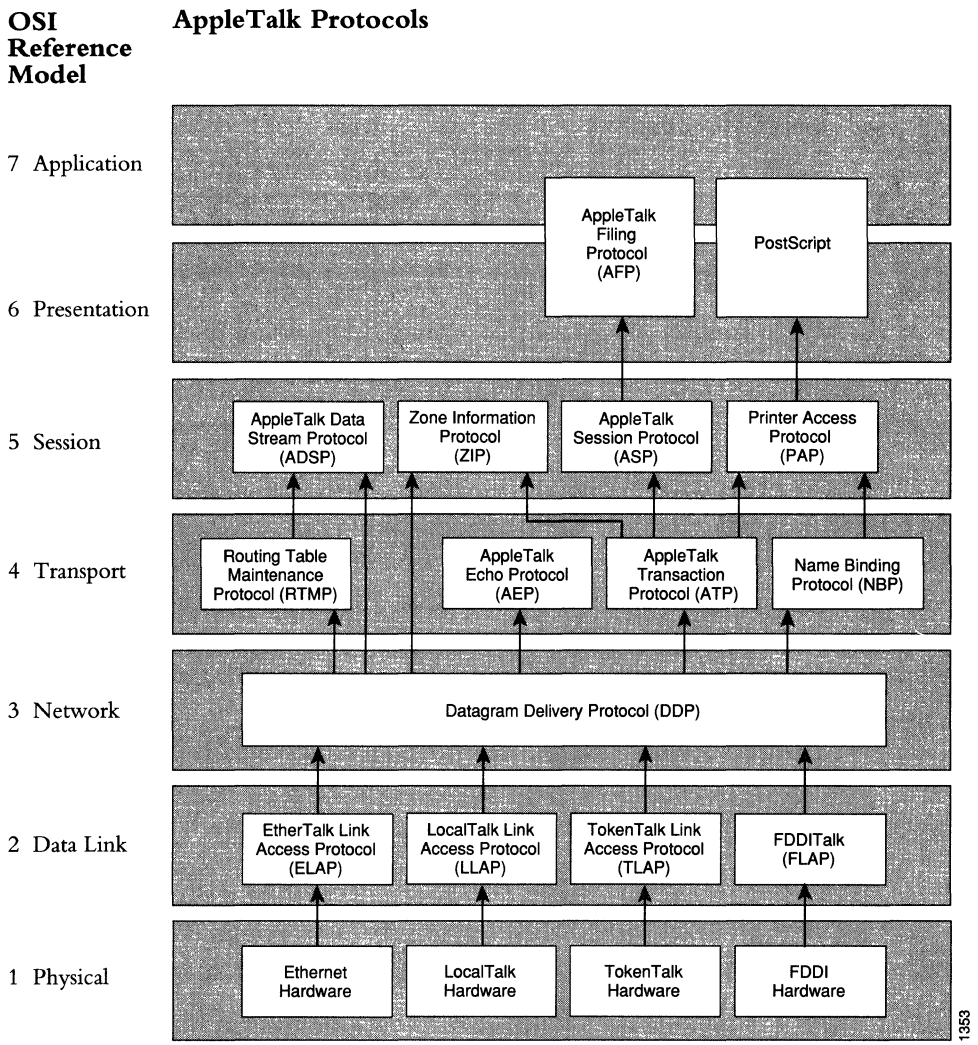


Figure 10-2 AppleTalk and the OSI Reference Model

Note: Apple Computer uses the name "AppleTalk" to refer the Apple Networking Architecture, whereas the actual transmission media used in AppleTalk Network are referred to as LocalTalk (Apple Computer's proprietary twisted pair transmission medium for AppleTalk), TokenTalk (AppleTalk over Token Ring), EtherTalk (AppleTalk over Ethernet), and FDDITalk (AppleTalk over Fiber Distributed Data Interface).

AppleTalk, like many network protocols, makes no provisions for network security. The design of the AppleTalk protocol architecture requires that security measures be executed at higher application levels. Cisco Systems supports AppleTalk distribution lists, allowing control of routing updates on a per interface basis. It is a security feature similar to those provided for other protocols.

Extended (Phase II) Versus Nonextended (Phase I) AppleTalk

AppleTalk was designed for local work groups. With the installation of over 1.5 million Macintoshes in the first five years of the product's life, Apple found that some large corporations were exceeding the design limits of AppleTalk, so they created extended AppleTalk. The extended AppleTalk architecture increases the number of nodes per AppleTalk Internet to over 16 million and an unlimited number of zones per cable. Apple also enhanced AppleTalk's routing capabilities and reduced the amount of network traffic generated by AppleTalk routers.

The introduction of the extended AppleTalk architecture also introduces the concept of *nonextended* and *extended* networks. Nonextended AppleTalk networks are sometimes called "Phase I" and extended networks are called "Phase II." Nonextended networks refer to the nonextended AppleTalk Ethernet 1.0 networks, (explicitly removed by Apple but still supported by Cisco), and to the nonextended serial line-based networks, including those configured using X.25 and LocalTalk.

Extended networks refer to the extended AppleTalk compliant networks configured on Ethernet (EtherTalk 2.0), FDDI, and Token Ring media. Samples of the AppleTalk nonextended and extended network configurations can be found in the section "AppleTalk Configuration Examples."

The AppleTalk extended-network architecture provides extensions compatible with nonextended AppleTalk internets. The AppleTalk extended architecture was designed to remove the previous limits of 254 concurrently active AppleTalk nodes per cable, as well as the previous limit of one AppleTalk zone name per cable. Extended AppleTalk contains better algorithms for choosing the best routers for traffic and is designed to minimize the amount of broadcast traffic generated for routing updates.

Another important feature in extended AppleTalk is the ability of a single AppleTalk cable to be assigned more than one network number. The size of the range of network numbers assigned to a cable determines the maximum number of concurrently active AppleTalk devices that can be supported on that cable, which is 254 devices per network number.

Cisco routers running software Release 8.2 or later support both extended and nonextended AppleTalk. An interface may be configured for either extended or nonextended AppleTalk operation. This allows an easy transition between the protocols; however, see the guideline in the section "Configuration Guidelines" before mixing versions of AppleTalk in your network.

Nonextended AppleTalk Addressing

AppleTalk addresses are 24 bits long. They consist of two components: a 16-bit network number, and an 8-bit node number. The Cisco AppleTalk software parses and displays these addresses as a sequence of two *decimal* numbers, first the network number, then the node number, separated by a dot. For example, node 45 on network 3 is written as *3.45*. A node is any AppleTalk-speaking device attached to the network. Each enabled AppleTalk interface on a router is a node on its connected network.

AppleTalk Zones

When a router is used to join two or more AppleTalk networks into an internet, the component physical networks remain independent of each other. A network manager may assign to these network conceptual groupings known as *zones*.

There are two main reasons to create zones in an AppleTalk internet: to simplify the process of locating and selecting network devices, and to allow for the creation of departmental work groups which may exist on several different and possibly geographically separated networks.

For example, consider a large AppleTalk internet which may contain hundreds or thousands of shared resources and devices. Without a method of dividing this large number of resources and devices into smaller groups of devices, a user might have to scroll through hundreds or thousands of resource/device names in the Chooser to select the one resource to be used. By creating small, conceptual groups of resource and device names, a user may now choose the resource they need much more quickly and easily than if they were sorting through a very long list of names.

A zone may include many networks, which need not be physically co-located. A zone is not limited by geographical area. The partitioning afforded by zone names is conceptual, not physical.

Zones are defined by the network manager during router configuration. When a Cisco router is configured, each AppleTalk-configured interface must be associated with exactly one zone name for nonextended networks, or one or more zone names for extended networks. Until a zone name has been assigned, AppleTalk routing features are disabled for that interface. The section "Configuring AppleTalk Routing" later in this chapter, describes the subcommands to use in the zone naming process.

It is very important that routers explicitly configured with zone information be configured correctly.

Name Binding Protocol (NBP)

The Name Binding Protocol (NBP) maps network entity names with internet addresses. It allows users to specify descriptive or symbolic names, while other software processes refer to numerical addresses for the same entities. With NBP, almost all user-level programs respond to names instead of numbers. When users select an AppleTalk device, they are using the NBP protocol to bind the address to the device name. Numerical addresses assigned to zones are primarily used by the router software and by network managers in the **ping** process (see the section "The AppleTalk Ping Command" later in this chapter).

NBP provides four basic services for binding names to nodes and zones:

- Name registration
- Name deletion
- Name look-up
- Name confirmation

The nature of the AppleTalk addressing scheme is inherently volatile and node addresses change frequently. Therefore, NBP associates numerical addresses with aliases which continue to reference the correct address if the address changes.

Zone Information Protocol (ZIP)

NBP uses the Zone Information Protocol (ZIP) to determine which networks belong to which zones. A Cisco router uses ZIP to maintain the network-number-to-zone-name mapping of the AppleTalk internet.

Along with a routing table, each router maintains a data structure known as the *zone information table* (ZIT). The table provides a listing of network numbers for each network in every zone. Each entry is a *tuple* (an inseparable network-number-hop-number set) that matches a network number with a zone name as supplied by the network manager.

Dynamic Address Assignment

AppleTalk provides for *dynamic address assignment*. With dynamic address assignment, not all fields of an AppleTalk address need to be specified in the configuration of a router. If there is another AppleTalk router on the network, it may be able to supply the network number. A preconfigured router on an AppleTalk network acts as a *seed router,* sending out address information to other routers on its connected network.

Seed routers are routers that come up and verify the configuration. If the configuration is valid, they start functioning. Seed routers come up even if no other routers are on the network. On the other hand, *nonseed routers* must first communicate with a seed router before it can function. A nonseed router must verify the configuration with another functioning router. The configuration must match exactly for the router to function.

A node number can be chosen by a network manager, but, alternatively, may be negotiated between AppleTalk hosts on the network. Note that network numbers may be assigned by routers only.

Unspecified parts of the AppleTalk address are entered as zero. For example:

34.5	Represents a fully qualified address (net 34, node 5).
0.5	Is a partially qualified address (net unspecified, node 5).
122.0	Represent net 122, node unspecified.
0.0	Is completely unspecified.

Node numbers are automatically assigned by AppleTalk (in other words, configured as zero) or when the specified address is in use it randomly chooses an initial value. The node will first try the node number that was its most recent address. If that number is unavailable, the node then searches for the next available address. If it reaches 254 without finding an available number, it cycles back to 1 and continues until it finds a free address. Nonextended network address restrictions are as follows: user node numbers are from 1 to 127 while servers/printer node numbers are from 128 to 254.

For nonseed routers, an interface will participate in the routing of only local traffic until its network number has been determined. If zero has been specified for a network number, that interface will not route any packets until it receives its network number from a seed router.

Receipt of a routing table update informs the router of the network number for the interface on which the packet was received. Every routing table update includes the network number of the network the packet was sent on. Therefore, the router is able to determine the network number of the receiving interface.

As long as one fully configured (seed) router exists on an Ethernet or Token Ring cables, other interfaces and routers directly attached to that cable need not be configured; they can take their information from the initial router. However, once the configuration process has stabilized for a particular AppleTalk internet, all routers thereafter should be configured as seed routers. Note that synchronous serial and X.25 network interfaces must be explicitly configured on each router to be used as AppleTalk transports.

Note: The same Macintosh may have several different AppleTalk address over time. Routing tables, maintained by RTMP, contain network and zone information.

RTMP routing tables contain an entry for each network a datagram can reach. Each entry includes the router port which leads to the destination network, the node ID of the next router to receive the packet, and the distance in hops to the destination network. Periodic exchange of routing tables allows the routers in an internet to ensure that they supply current and consistent information.

Node information is maintained by tables appropriate to the media (usually AARP tables).

Figure 10-3 shows a sample RTMP table and the corresponding network topology.

Extended AppleTalk Addressing

AppleTalk addresses, as explained in the section "Nonextended AppleTalk Addressing," earlier in this chapter, are composed of a 16-bit network and an 8-bit node number. In nonextended AppleTalk, nodes within a single cable can communicate using only their 8-bit node numbers.

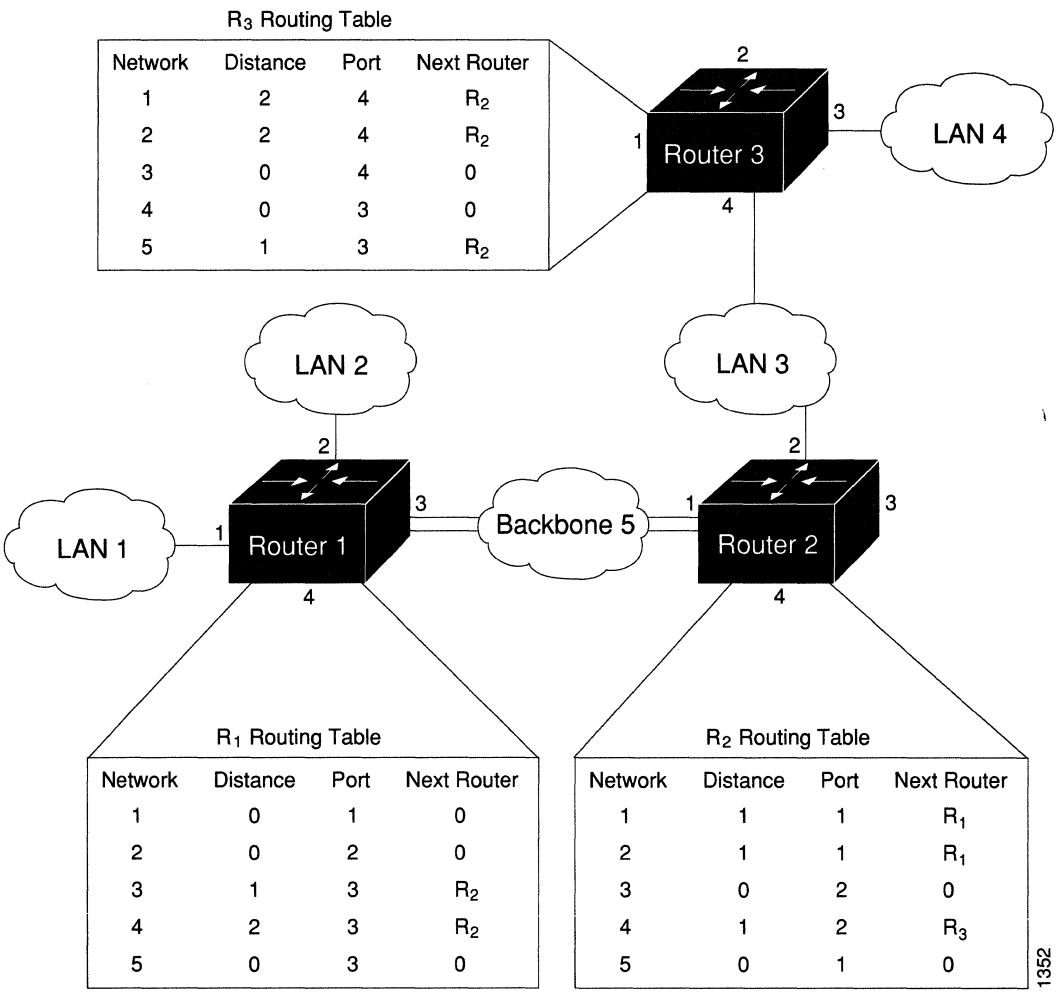


Figure 10-3 Sample AppleTalk Routing Table

A node in extended AppleTalk is *always* identified by its network *and* node number. Dynamic address resolution when a router is not present includes the assignment of a random network number within a small range, as well as a node number. When a router is present in the network, a node starts up using its newly acquired address for a short period of time. It then immediately requests a list of valid network numbers from the router or routers. The node then uses these to determine its actual AppleTalk address by selecting an unassigned address. (Be aware that this two-step process causes the nonextended AppleTalk to be incompatible with the extended AppleTalk routing.)

A new concept of cable *ranges* is introduced with the extended AppleTalk. Cables now have ranges of network numbers and multiple zones that may exist on them, so that a node can access anything that is in any of the zones that are on the same cable as the node itself. But the node can exist in only one zone and on only one network.

In an extended AppleTalk network, the mapping of a single network number to a zone name is no longer valid. End nodes are expected to know the zone to which they belong, or to choose from the list of available zones provided by a router. The router maintains a default zone which new nodes will use automatically if they have not previously chosen a zone.

Configuring AppleTalk Routing

This section provides an overview on how to configure Cisco AppleTalk routing.

Configuration Overview

The AppleTalk interface configuration is different for the two types of AppleTalk interfaces; extended and nonextended.

Configuring a nonextended AppleTalk interface involves the following steps:

Step 1: Enable AppleTalk routing with the **appletalk routing** command.

Step 2: Assign the nonextended AppleTalk addresses with the **appletalk address** interface subcommand.

Step 3: Assign the zone name with the **appletalk zone** interface subcommand.

Configuring an extended AppleTalk interface involves these steps:

Step 1: Enable AppleTalk routing with the **appletalk routing** command.

Step 2: Assign the extended AppleTalk cable range parameters with the **appletalk cable-range** command.

Step 3: Assign the zone name or names with the **appletalk zone** interface subcommand.

The software also provides commands for fine tuning the AppleTalk network, for configuring packet filtering mechanisms, monitoring, maintaining and troubleshooting network operation. Alphabetically arranged summaries of the commands described in this chapter are also provided at the end of the chapter.

Configuration Guidelines

Follow these guidelines when configuring your AppleTalk network on a Cisco router:

- A Macintosh that contains an Ethernet card must run EtherTalk version 2.0 or later to support extended AppleTalk. A Macintosh with only a LocalTalk interface does not require any changes.
- Shiva FastPath routers must run K-Star version 8.0 or later and be explicitly configured for extended AppleTalk.
- Apple's Internet Router software, version 2.0, supports a transition mode for translation between the nonextended AppleTalk and the extended AppleTalk on the same network.

- For the Internet Router version 1.0, transition mode requires the Apple upgrade utility and a special patch file from Apple.
- If your AppleTalk internet contains any routers that support only nonextended AppleTalk, the following configuration restrictions must be observed. These restrictions are not enforced, but unpredictable behavior may result if they are violated. All routers in a network must support extended AppleTalk before these restrictions may be lifted.
  - Cable ranges of only one (*666-666*, for example) are permitted.
  - Each AppleTalk network may have only one zone associated with it.

A general understanding of Cisco's representation of AppleTalk addresses is necessary before configuration of the router. Refer to the sections "Cisco's Implementation of AppleTalk," "Nonextended AppleTalk Addressing," and "Extended AppleTalk Addressing" earlier in this chapter.

Enabling AppleTalk Routing

Before you can configure AppleTalk routing, you need to enable AppleTalk protocol processing. To do that, use the **appletalk routing** global configuration command. The full command syntax follows:

appletalk routing

no appletalk routing

The **appletalk routing** configuration command enables AppleTalk protocol processing. The **no appletalk routing** disables all AppleTalk processing.

Assigning Nonextended (Phase I) AppleTalk Address

To assign AppleTalk addresses for nonextended networks, use the **appletalk address** interface subcommand. Its full syntax follows.

appletalk address *address*

no appletalk address

The argument *address* assigns AppleTalk addresses on the interfaces that will be used for the AppleTalk protocol. It assigns one AppleTalk address per interface. This step must be done before assigning zone names.

Note: Use this subcommand to configure nonextended interfaces.

The **no appletalk address** subcommand disables nonextended AppleTalk processing on the interface.

Example:

These commands begin AppleTalk routing and assign address 1.129 to interface Ethernet 0.

```
!
appletalk routing
!
interface ethernet 0
appletalk address 1.129
!
```

Assigning a Cable Range for Extended AppleTalk (Phase II)

To assign the cable-range parameters, use the **appletalk cable-range** interface subcommand. The full command syntax follows.

appletalk cable-range *start-end* [*network.node*]

no appletalk cable-range

This command designates an interface to be on an extended AppleTalk network. A cable range is the network numbers assigned to an extended network.

This range is specified using the argument *start-end*, which is a pair of decimal numbers between 1 and 65,279, inclusive. The starting network number should be less than or equal to the ending network number.

Note: To use this subcommand to configure a nonextended AppleTalk interface, the starting and ending network numbers must be equal.

Specifying a cable range of 0-0 in the *start-end* argument (start = end = 0) places the interface into discovery mode, which attempts to determine cable range information from another router on that network.

The optional *network.node* argument specifies the suggested network and node number that will be used first when selecting the AppleTalk address for this interface. Note that any suggested network number must fall within the specified range of network numbers.

Use the **no appletalk cable-range** command to disable AppleTalk processing on the interface.

Example:

This command assigns a cable range of 2-2 to the interface:

```
appletalk cable-range 2-2
```

Assigning a Zone Name

Use the **appletalk zone** interface subcommand to assign a zone name to an AppleTalk interface. Full command syntax for this command follows:

appletalk zone *zonename*

no appletalk zone [*zonename*]

Interfaces which are configured for seed routing or which have discovery mode disabled must have a zone name assigned before AppleTalk processing will begin.

The argument *zonename* specifies the name of the zone for the connected AppleTalk network. The argument *zonename* may include special characters from the Apple Macintosh character set. To include a special character, insert a colon and two uppercase hexadecimal characters. The hexadecimal equivalent for special characters in the Macintosh character set may be found in character tables published by Apple Computer (see Appendix D in the text *Inside AppleTalk*, 2nd edition).

The **appletalk zone** command works for the nonextended AppleTalk interface, however, you may repeat the command to define multiple zones in an extended AppleTalk network only when a cable range has been specified.

The router selects the zone in which it will operate from the list specified. The first zone specified in the list is the *default zone*. Computers in the network will select the zone in which they will operate from the list of zone names valid on the cable to which they are connected. If an interface is using nonextended AppleTalk, repeated execution of the zone command will replace the zone name for the interface with the newly specified zone name.

The **no appletalk zone** interface subcommand deletes a zone name from a zone list or the entire zone list if none is specified. The optional zone name is ignored for nonextended AppleTalk interface configurations. The command is also ignored if the specified zone name is not in the current zone list for an interface. The list should be cleared using the **no appletalk zone** interface subcommand before configuring a new zone list.

Note: The zone list is cleared automatically when **appletalk address** or **appletalk cable-range** commands are used.

Examples:

This command assigns zone name Twilight to the interface:

```
appletalk zone Twilight
```

The following example shows use of the AppleTalk special characters sets by setting the zone name to *cisco*zone*.

```
appletalk zone cisco:A5zone
```

Setting and Resetting Discovery Mode

The discovery mode is set and reset using the **appletalk discovery** interface subcommand. The full syntax of this command follows:

appletalk discovery

no appletalk discovery

This command resets the discovery mode and allows a new cable range to be discovered. If the port information has been discovered, and the port is operational, then this command results in the port being a valid seed port.

Use the **no appletalk discovery** command to return the software to the default (off) state.

Configuring IP Encapsulation of AppleTalk Packets

Use the **appletalk iptalk** interface subcommand to encapsulate AppleTalk in IP packets in a manner compatible with the Columbia AppleTalk Package (CAP) IPtalk and the Kinetics IPtalk (KIP) implementations.

appletalk iptalk *net.node zone*

This command enables IPtalk encapsulation on an interface which already has an configured IP address. The command allows AppleTalk communication with UNIX™ hosts running older versions of CAP (Columbia AppleTalk Package) which do not support native AppleTalk EtherTalk encapsulations. Typically, Apple Macintosh users wishing to communicate with these servers would have their connections routed through a Kinetics FastPath™ router running KIP (Kinetics IP) software.

This command is provided as a migration command; newer versions of CAP provide native AppleTalk EtherTalk encapsulations and the IPtalk encapsulation is no longer required. The Cisco implementation of IPtalk assumes that AppleTalk is already being routed on the backbone, since there is currently no LocalTalk hardware interface for Cisco routers.

The Cisco implementation of IPtalk does not support manually configured AppleTalk-to-IP address mapping (**atab**). The address mapping provided is the same as the Kinetics IPtalk implementation when the **atab** facility is not enabled. This address mapping functions as follows: The IP subnet mask used on the router Ethernet interface on which IPtalk is enabled is inverted (one's complement). This result is then masked against 255 (0xFF hexadecimal). This is then masked against the low-order 8 bits of the IP address to obtain the AppleTalk node number. The following example configuration should make this more clear.

Example:

```
interface Ethernet 0
ip address 131.108.1.118 255.255.255.0
appletalk address 20.129
appletalk zone Native AppleTalk
appletalk iptalk 30.0 UDPZone
```

In this configuration, the IP subnet mask would be inverted:

```
255.255.255.0 inverted yields: 0.0.0.255
```

Masked with 255 it yields 255, and masked with the low-order 8 bits of the interface IP address it yields 118.

This means that the AppleTalk address of the Ethernet 0 interface seen in the UDPZone zone is 30.118. This caveat should be noted, however: Should the host field of an IP subnet mask for an interface be more than 8 bits wide, it will be possible to obtain conflicting AppleTalk node numbers. For instance, consider a situation where the subnet mask for the Ethernet 0 interface above is 255.255.240.0, meaning that the host field is 12 bits wide.

Configuring IP Encapsulation DDP Socket to UDP Port Mapping

Use the global configuration command **appletalk iptalk-baseport** to specify the UDP port number which is the beginning of the range of UDP ports used in mapping AppleTalk *well-known* DDP socket numbers to UDP ports. The command syntax looks like this:

appletalk iptalk-baseport *port-number*

Implementations of IPTalk prior to April, 1988 mapped well-known DDP socket numbers to privileged UDP ports start at port number 768. In April of 1988, the NIC assigned a range of UDP ports for the defined DDP well-known sockets starting at UDP port number 200 and assigned these ports the names *at-nbp*, *at-rtmp*, *at-echo* and *at-zis*. The Columbia AppleTalk Package, Release 6 and later, dynamically decides which port mapping to use. If there are no AppleTalk service entries in the */etc/services* file, CAP will use the old 768-based mapping.

This is the default UDP port mapping supported by Cisco's implementation of IPTalk. If there are service entries in the */etc/services* file for the AppleTalk services, the Cisco router configured for IPTalk encapsulation should specify the beginning of the port mapping range with the **appletalk iptalk-baseport** command. The following example configuration builds upon the example for the **appletalk iptalk** command to illustrate this concept.

Example:

```
interface Ethernet 0
ip address 131.108.1.118 255.255.255.0
appletalk address 20.129
appletalk zone Native AppleTalk
appletalk iptalk 30.0 UDPZone
appletalk iptalk-baseport 200
```

Checking Packet Routing Validity

Use the **appletalk strict-rtmp** global configuration command to enforce maximum checking of routing packets to ensure their validity. The full command syntax follows:

appletalk strict-rtmp

no appletalk strict-rtmp

The default of this command is to provides maximum checking.

Currently, strict RTMP checking consists of discarding RTMP arriving from routers not directly connected to the router performing the check. (In other words, no routed RTMP packets will be accepted.)

Use the **no appletalk strict-rtmp** command to disable the maximum checking mode.

Enabling and Disabling Routing Updates

Use the global configuration command **appletalk send-rtmp** to allow the transmission of routing updates to be disabled. The full syntax of the command is:

appletalk send-rtmp

no appletalk send-rtmp

This command allows a router to be placed on a network with AppleTalk routing enabled, but without being seen by other AppleTalk routers on the cable. The default is to send routing updates. The **no appletalk send-rtmp** command disables this default.

Changing Routing Timers

Use the global configuration command **appletalk timers** to change the time intervals used in AppleTalk routing, as follows:

appletalk timers *update-interval valid-interval invalid-interval*

The argument *update-interval* is the time, in seconds, between routing updates sent to other routers on the network. This is ten seconds by default.

The argument *valid-interval* is amount of time, in seconds, that the router will consider a route valid without having heard a routing update for that route. This is normally twice the update interval, 20 seconds by default. Once this period of time has elapsed without having heard a routing update for a route, the route becomes "suspect," indicating that a routing update has been missed.

The argument *invalid-interval* is the amount of time, in seconds, that the router will wait before marking a route invalid. Once a route has been marked invalid, the route will no longer be sent to other routers in routing updates. By default, the *invalid-interval* argument is three times the update-interval, or 30 seconds.

Note: Modification of the routing timers should not be undertaken without fully understanding the ramifications of doing so. Many other AppleTalk router vendors provide no facility for modifying their routing timers; should you adjust a Cisco router's AppleTalk timers such that routing updates do not arrive at these other routers within the normal interval, it is possible to degrade or destroy AppleTalk network connectivity.

Example:

This command increases the update interval to 20 seconds, the route valid interval to 40 seconds, and the route invalid interval to 60 seconds.

```
appletalk timers 20 40 60
```

Assigning a Proxy Network Number

When an AppleTalk internetwork contains routers which support only nonextended AppleTalk and routers which support only extended AppleTalk, then one **apple proxy-npb** global configuration command is required for each zone in which there is a router which supports only nonextended AppleTalk. The full syntax of this command follows.

appletalk proxy-npb *network-number zonename*

no appletalk proxy-npb *network-number zonename*

The argument *network-number* must be a unique network number which will be advertised via this router as if it were a real network.

The argument *zonename* is the name of the zone requiring compatibility support.

No router may have the same network number defined as a proxy network, and it cannot be associated with a physical network.

Only one proxy is needed to support a zone, but additional proxies may be defined with different network numbers if redundancy is desired. Each proxy will generate one or more packets for each forward request it receives. All other packets sent to the proxy network are discarded. Redundant proxies increase the NPB traffic linearly.

Assume your network topology looks like the one in Figure 10-4. Also assume that Router A supports only nonextended AppleTalk, that Router B supports only extended AppleTalk (not in transition mode), and that Router C supports only extended AppleTalk.

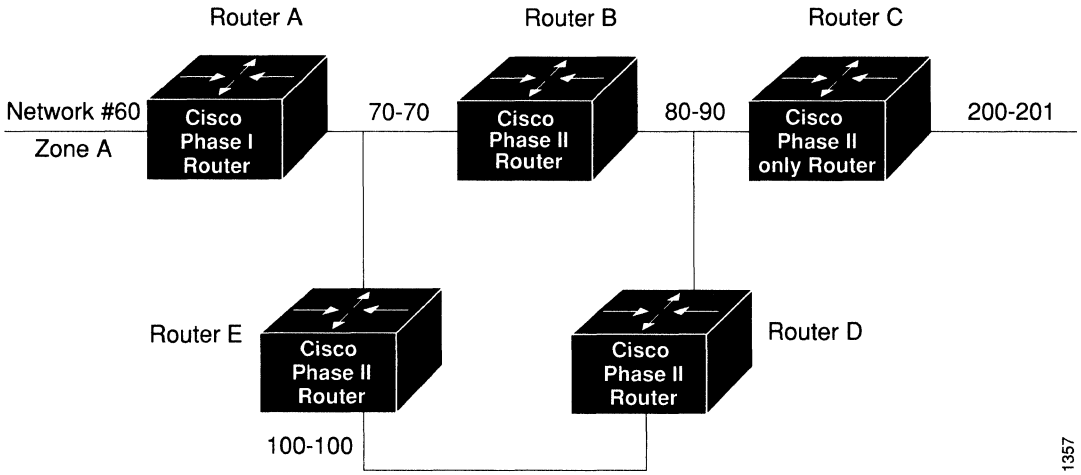


Figure 10-4 Example Network Topology

If router C generates a NBP hookup request for zone A, router B will convert this request to a forward request and send it to router A. Since router A supports only nonextended AppleTalk, it does not handle the forward request and ignores it. Hence, the NBP lookup from router C fails.

To work around this problem without putting a transition router adjacent to the nonextended only router (router A), you could configure router D with a NBP proxy.

If you configured router D with a NBP proxy as follows, any forward requests received for zone A are converted into lookup requests, and therefore, the nonextended router for Net 60 can properly respond to NBP hookup requests generated beyond router C. The following example demonstrates the command needed to describe this configuration.

Example:

```
appletalk proxy 60 A
```

Generating Checksum Verification

Use the **appletalk checksum** global configuration command to enable the generation and verification of checksums for all AppleTalk packets. The full command syntax follows:

appletalk checksum

no appletalk checksum

An incoming packet with a nonzero checksum will be verified against that checksum and discarded if in error. By default, checksum verification is enabled.

Cisco routers no longer check checksum on routed packets, thereby eliminating the need to disable checksum to allow operation of some networking applications.

Use the **no appletalk checksum** command to disable checksum verifications.

Specifying the Time Interval Between AARP Transmissions

Use the **appletalk arp interval** global configuration command to specify the time interval between retransmission of ARP packets, as follows:

appletalk arp interval *milliseconds*

The argument *milliseconds* specifies the interval. The default and minimum value is 33 milliseconds.

Lengthening the interval between packets permits responses from certain devices which respond more slowly, such as printers and overloaded file servers, to be received.

Example:

This command lengthens the AARP retry interval to 100 milliseconds.

```
appletalk arp interval 100
```

Specifying the AARP Retransmission Count

Use the **appletalk arp retransmit-count** global configuration command to specify the number of retransmissions that will be done before abandoning address negotiations and using the selected address.

appletalk arp retransmit-count *count*

The argument *count* specifies the retransmission count. The minimum value that can be specified is 1; the default is 10.

Example:

This command specifies an AARP retransmit count of 25.

```
appletalk arp retransmit-count 25
```

AppleTalk Access and Distribution Lists

An *access list* is a list of AppleTalk network numbers kept by the Cisco router to control access to or from specific networks for a number of services.

Cisco's AppleTalk access lists provide network security by permitting or denying certain packets onto a network interface. Cisco's AppleTalk access lists are applicable to *networks only*; they may not be used for specific nodes. Network managers who are familiar with Cisco's access list support for other network protocols should note that these lists do not actually hide a network. An access-controlled network will still appear in routing information; however, the network interface will simply not allow packets to pass through it. Network managers who require a more in-depth security strategy should execute measures within the higher-level protocols, such as the AppleTalk Filing Protocol (AFP).

When defining access lists for an interface, all networks within a zone should be governed by the same access control. Although access lists are applied to network numbers, they should be created with attention to the zone in which they are located. This precaution serves two purposes. First, it prevents the partitioning of zones, which should always be a nondivisible unit of an internet. Second, it allows a router to query all the network interfaces within a zone, which prevents duplicate names among nodes.

To simplify the definition of access lists, some network managers may prefer to give each network its own zone name.

A *distribution list* is a list of AppleTalk access list numbers kept by the Cisco router which controls whether the network numbers specified by the access list are processed during the reception or transmission of routing updates. A distribution list will not prevent packets destined for a specified network number from being accepted; it will only prevent the route to the specified network from appearing in neighboring routers' AppleTalk routing tables.

Assigning an Access List to an Interface

An AppleTalk access list is assigned to an interface with the **appletalk access-group** interface subcommand. Once assigned, no packet which fails the **appletalk access-list** command will go out on that interface. The full syntax of this command follows:

appletalk access-group *list*

no appletalk access-group *list*

The argument *list* specifies the appropriate AppleTalk access list. Use the **no appletalk access-group** command to remove the list from the interface.

The EXEC command **show appletalk traffic** displays the number of packets dropped because of access control. Refer to the section on "Monitoring the AppleTalk Network" later in this chapter for more information. See the section "Filtering Networks Sent Out in Updates" for an example of the use of this command.

Controlling Interface Access

To permit or deny packets onto a specific network interface, use the **access-list** interface subcommand, as follows:

access-list *list* {**permit** | **deny**} *network*

no access-list *list*

The argument *list* is an integer between 600 and 699 and the argument *network* is an AppleTalk network. A *network* argument of –1 represents any network.

Additional permit and deny conditions may be added to the list by issuing further **access-list** commands for that list. Note that the order of specification, especially of the all networks entry, matters in how the access list treats networks.

Use the **no access-list** command to remove an entry from the list.

Example:

These commands will not permit packets destined for network number 101 to transmit the interface Ethernet 6 packets.

```
interface ethernet 6
appletalk cable-range 71-71 71.0
appletalk zone Tir'n na n'Og
access-list 602 deny 101
```

Filtering Networks Received in Updates

Use the **apple distribute-list** interface subcommand to filter input from the networks. The full syntax for this command follows.

appletalk distribute-list *access-list-number* **in**

no appletalk distribute-list *access-list-number* **in**

The argument *access-list-number* is the number of a predefined access list.

Use the keyword **in** to filter networks received in update. AppleTalk network numbers specified by the *access-list-number* argument will not be inserted into the router's AppleTalk routing table when routing updates are received. Use the **no appletalk distribute-list** command to remove this function.

This is a special command providing compatibility between nonextended-only AppleTalk routers (Cisco pre-8.2 release) and the extended AppleTalk routers (Apple's Internet Router, WellFleet, NSC, and so on).

Example:

These commands prevent the insertion of routing updates from network 10 into the routing table.

```
access-list 601 deny 10
!
interface ethernet 3
appletalk distribute-list 601 in
```

Filtering Networks Sent Out in Updates

Use the interface configuration subcommand **apple distribute-list** to filter output from networks. The full syntax of this command follows.

appletalk distribute-list *access-list-number* **out**

no appletalk distribute-list *access-list-number* **out**

The argument *access-list-number* is the number of a predefined access list.

Use the keyword **out** to suppress the AppleTalk networks specified by the *access-list-number* from being sent in updates. Use the **no appletalk distribute-list** command to remove the filter.

This is a special command providing compatibility between the nonextended-only AppleTalk routers (Cisco pre-8.2 release) and the extended AppleTalk routers (Apple's Internet Router, WellFleet, NSC, and so on).

Example:

These commands prevent network 10 on another Ethernet interface from publishing updates on interface Ethernet 0. The **appletalk access-group** command prevents packets from being sent out the interface.

```
!
access-list 601 deny 10
interface Ethernet 0
appletalk distribute-list 10 out
appletalk access-group 601
```

AppleTalk Configuration Examples

The following examples illustrate configurations of nonextended AppleTalk networks routing between two Ethernets, over HDLC, and X.25—and a configuration for an extended AppleTalk network.

Nonextended AppleTalk Routing Between Two Ethernets

This example configuration illustrates how to configure routing between two Ethernets. Ethernet 0 is on network 1, at node 128. Ethernet 1 is on network 2, at node 154. The two networks are in the "Twilight" and "No Parking" zones, respectively. See Figure 10-5 for an illustration.

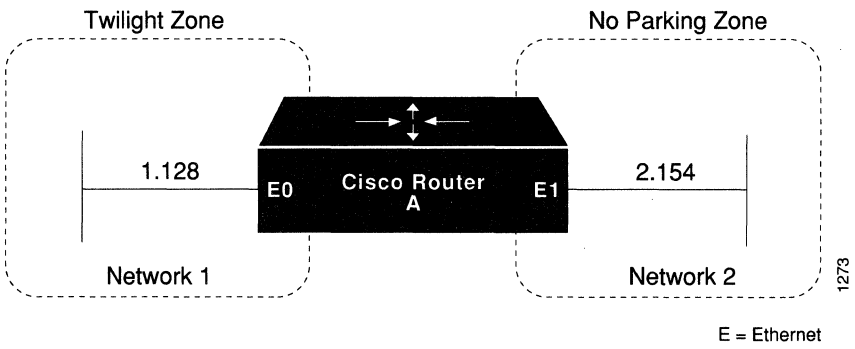


Figure 10-5 Nonextended AppleTalk Routing Between Two Ethernet Networks

Example 1:

```
!
appletalk routing
!
interface ethernet 0
appletalk address 1.128
appletalk zone Twilight
!
interface ethernet 1
appletalk address 2.154
appletalk zone No Parking
!
```

Example 2 is a variation of the above configuration. It differs in that it has other seed routers on both networks to provide the zone and network number information. In this way, the Cisco router discovers the information dynamically. Refer to Figure 10-6 for an illustration.

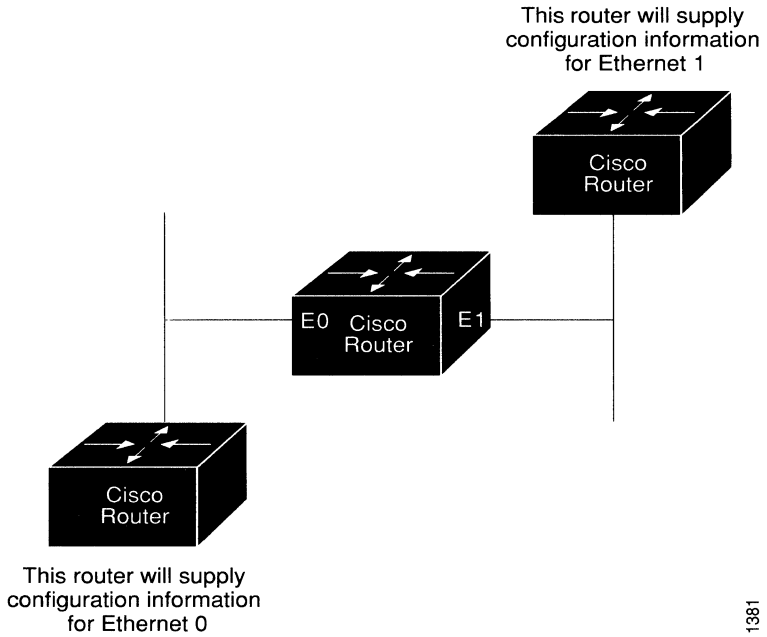


Figure 10-6 Routing Between Seed Routers

Example 2:

```
!
appletalk routing
!
interface ethernet 0
appletalk address 0.0
!
interface ethernet 1
appletalk address 0.0
!
```

Configuring Transition Mode

The Cisco router may be used to route between extended and nonextended AppleTalk networks that exist on the same cable. Many other vendors have coined the term *transition mode* for this type of routing.

To do this on the Cisco router, you must have two ports connected to the same physical cable. One port will be configured as a nonextended AppleTalk network.

Both ports must have unique network numbers because you are actually routing between two separate AppleTalk networks, an extended and a nonextended network. Figure 10-7 shows an example of the topology and configuration of such connection.

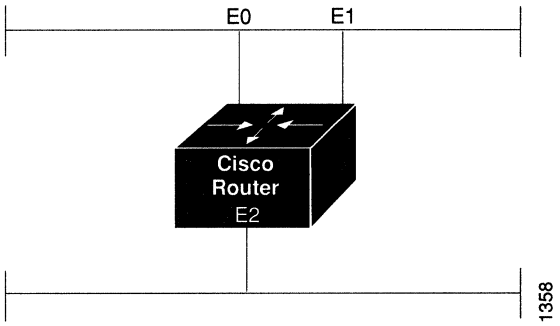


Figure 10-7 Transition Mode Topology and Configuration

Example:

```
interface ethernet 0
appletalk cable-range 2-2
appletalk zone No Parking
!
interface ethernet 1
appletalk address 3.128
appletalk zone Twilight
!
interface ethernet 2
appletalk cable-range 4-4
appletalk zone Do Not Enter
```

Note: Networks 2-2 and 4-4 in the above example have cable range of one and a single zone in their zone list. This must be true to maintain compatibility with the nonextended network, network 3.

Nonextended AppleTalk Routing over HDLC

AppleTalk's dynamic address assignment feature allows users and network managers to choose default network addresses. Configuring AppleTalk nodes over HDLC encapsulation has one major variation from normal configuration procedures: you must explicitly specify all node numbers on each end of a serial line, and at least one end of the line must also be provided with a network number. A router will not respond to the specification of zero for any node number. Moreover, if two nodes are assigned the same node numbers by user configuration, a router in an HDLC environment will not acknowledge any conflict.

A sample display of the interface configuration for both ends of the serial line follows.

Configuring Serial 1

These commands enable AppleTalk routing in interface serial 1.

Example:

```
!
interface serial 1
appletalk address 1544.1
appletalk zone Twilight
!
```

Configuring Serial 2

These commands enable AppleTalk on interface serial 2.

Example:

```
interface serial 2
appletalk address 1544.2
appletalk zone Twilight
```

You can specify that the interface is enabled using the EXEC command **show appletalk interface** as follows:

```
serial 1 is up, line protocol is up
appleTalk address is 1544.1, Valid
appleTalk zone is Twilight
Serial 2 is up, line protocol is up
AppleTalk address is 1544.2, Valid
AppleTalk zone is Twilight
```

Nonextended (Phase I) AppleTalk Routing over X.25

The configuration of X.25 networks is similar to that for HDLC encapsulation. However, you must completely and explicitly configure ***all*** network and node numbers in an X.25 environment. Note that all AppleTalk nodes within an X.25 network must be configured with the same AppleTalk network number.

X.25 configuration for AppleTalk involves mapping AppleTalk addresses to X.121 addresses, executed with the X.25 configuration subcommand **x25 map** (see the section "Configuring the Datagram Transport on Commercial X.25 Networks" in Chapter 8).

Each time a packet is sent to a particular AppleTalk address, that address is looked up in the X.25 map table in order to match it to an X.25 address. The packet is encapsulated in X.25 frames and sent to the X.25 node which is its destination.

The receiving node reassembles the X.25 frames if necessary, then strips the packet of X.25 framing information so that the original AppleTalk datagram can be processed.

In the configuration commands, the keyword **broadcast** (as used at the end of the next example) signals the following to the X.25 software: whenever a broadcast packet is sent, either each or every, but not both map entries with that flag set should receive a copy of the packet. The X.25 protocol does not provide broadcasts; therefore, they must be simulated in this manner when using X.25 as a transport protocol for another protocol that requires broadcasts, such as AppleTalk.

If the X.121 address of the router on the far end of the X.25 network is *123456789012*, and your local X.121 address is *210987654321*, and the two routers are at AppleTalk addresses *7.63* and *7.25*, you would configure these systems in the following way.

Example for First Router:

```
!
interface serial 0
appletalk address 7.25
appletalk zone Twilight
x25 map appletalk 7.63 123456789012 broadcast
!
```

Example for Second Router:

```
!
interface serial 0
appletalk address 7.63
appletalk zone Twilight
x25 map appletalk 7.25 210987654321 broadcast
!
```

Example for Third Router:

In this example, a third router has the X.121 address *333444555666* and AppleTalk address *7.100*.

```
!
interface serial 0
appletalk address 7.100
appletalk zone Twilight
x25 map appletalk 7.25 210987654321 broadcast
x25 map appletalk 7.63 123456789012 broadcast
!
```

With the addition of the third router, both the original routers need an additional **x25 map** entry:

```
x25 map appletalk 7.100 333444555666 broadcast
```

Extended AppleTalk Routing Network

The following commands illustrate how to configure an extended AppleTalk network.

Example:

```
!
appletalk routing
!
interface ethernet 0
appletalk cable-range 69-69 69.128
appletalk zone Empty Guf
appletalk zone Underworld
!
```

This configuration defines the zones *Empty Guf* and *Underworld* from which the router and the nodes may choose to reside. The equal cable range numbers allow compatibility with nonextended AppleTalk networks.

Monitoring the AppleTalk Network

Use the EXEC **show** commands described in this section to obtain displays of activity on the AppleTalk network.

Displaying the Fast-Switching Cache

Use the **show apple cache** command with the extended AppleTalk networks to display the current fast-switching cache. Enter this command at the EXEC prompt:

show apple cache

This display includes the current cache version number and all entries (valid or not). Valid entries are identified by an asterisk (*) in the first column.

Conditions that invalidate the fast-switching cache are as follows:

- Route deleted but not marked bad (and has been used)
- A route which has gone bad (and has been used)
- When you replace a route with a new metric (and it was used)
- When a neighbor transitions from suspect to bad.
- When a node address in the AARP cache changes hardware address.
- When a hardware address changes node address
- When the AARP cache gets flushed
- When an AARP entry is deleted
- When the following configuration commands are entered:
  - — After a **no appletalk routing** command
  - — After an **appletalk route-cache** command
  - — After an AppleTalk **access-list** command
- When the encapsulation for the line changes
- When a port leaves or enters Operational state

Following is a sample display of the **show apple cache** command:

```
AppleTalk routing cache version is 45
    Destination Interface MAC Header
*      4.000 Ethernet1 AA0004007BCC00000C000E8C809B81BE02
*   1544.000 Ethernet1 AA000400013400000C000E8C809B84BE02
*     33.000 Ethernet1 AA000400013400000C000E8C809B84BE02
```

Displaying AppleTalk Interface Information

The **show apple interface** command displays AppleTalk-specific interface information. Enter this command at the EXEC prompt.

show apple interface [*interface*]

The argument *interface* specifies an interface name and number to display a specific interface.

This information displayed by this command includes the extended AppleTalk cable ranges and the current interface mode, (the network verification/discovery mode, for example).

Sample displays of the **show apple interface** command follow.

Nonextended AppleTalk—Normal operation:

```
Ethernet 1 is up, line protocol is up
  AppleTalk address is 666.128, Valid
  AppleTalk zone is Underworld
```

Extended AppleTalk—Normal operation:

```
Ethernet 0 is up, line protocol is up
  AppleTalk cable range is 69-69
  AppleTalk address is 69.128, Valid
  AppleTalk zone is Empty Guf
```

Extended AppleTalk—Verification mode:

```
Ethernet 1 is up, line protocol is up
  AppleTalk routing disabled, Verifying port configuration
  AppleTalk cable range is 666-666
  AppleTalk address is 666.128, Valid
  AppleTalk zone is Underworld
```

Extended AppleTalk—Configuration error:

```
Ethernet 0 is up, line protocol is up
  AppleTalk routing disabled, Port configuration error
  AppleTalk cable range is 70-70
  AppleTalk address is 70.128, Bad
  AppleTalk zone is Empty Guf
```

When you enter the EXEC command **show apple interface** with the *interface* argument, the display looks this:

```
Ethernet 0 is up, line protocol is up
  AppleTalk cable range is 69-69
  AppleTalk address is 69.105, Valid
  AppleTalk zone is "Empty Guf"
  AppleTalk port configuration verified by 69.163
  AppleTalk discarded 3149 packets due to input errors
  AppleTalk discarded 71 packets due to output errors
  AppleTalk route cache is enabled
```

If AppleTalk routing is disabled on an interface, the display looks like this:

```
Ethernet 1 is up, line protocol is up
  AppleTalk protocol processing disabled
```

Displaying Directly Connected Routes

The **show apple neighbor** EXEC command shows the routers that are directly connected, or that are one hop away in the extended AppleTalk network. The command has this syntax:

show apple neighbor [*neighbor-address*]

The optional argument *neighbor-address* specifies address of the neighbor of the extended AppleTalk network.

For the command:

```
show apple neighbor
```

The display looks like this:

```
AppleTalk neighbors:

 31.86, Ethernet8, uptime 133:28:06, last update 1 sec ago
 81.82, Fddi0, uptime 266:11:44, last update 7 secs ago
 81.81, Fddi0, uptime 267:30:28, last update 958334 secs ago
   Neighbor is down.
 29.200, Ethernet3, uptime 263:45:50, last update 948440 secs ago
  Neighbor has restarted 2 times in 267:59:53.
   Neighbor is down.
 81.80, Fddi0, uptime 268:00:08, last update 963617 secs ago
   Neighbor is down.
 17.128, Ethernet2, uptime 133:26:43, last update 2 secs ago
  Neighbor has restarted 1 time in 268:00:21.
 69.163, Ethernet0, uptime 268:00:25, last update 1 sec ago
```

For the command:

```
show apple neighbor 69.163
```

The display looks like this:

```
Neighbor 69.163, Ethernet0, uptime 268:00:52, last update 7 secs ago
We have sent queries for 299 nets via 214 packets.
Last query was sent 4061 secs ago.

We received 152 replies and 0 extended replies.
We have received queries for 14304 nets via 4835 packets.
We sent 157 replies and 28 extended replies.
We received 0 ZIP notifies.
We received 0 obsolete ZIP commands.
We received 4 miscellaneous ZIP commands.
We received 0 unrecognized ZIP commands.
We have received 92943 routing updates.
Of the 92943 valid updates, 1320 entries were invalid.
We received 1 routing update which were very late.
Last update had 0 extended and 2 nonextended routes.
Last update detail: 2 old
```

Displaying the Network Routing Table

To show the routing table for networks, use the **show apple route** EXEC commands:

show apple route [*network*]

show apple route [*interface-name*]

This command displays either the full routing table or just the entry for the optionally specified *network* for both extended and nonextended AppleTalk networks. For the extended AppleTalk networks, the command also displays cable ranges information.

The optional *interface-name* argument specifies an interface name to report on. Displays for both nonextended and extended AppleTalk networks follow.

A sample display for a nonextended AppleTalk network:

```
Codes: R - RTMP derived, C - connected, S - static, 3 routes
C Net 258 directly connected, 1431 uses, Ethernet0, zone Twilight
R Net 6 [1/G] via 258.179, 8 sec, 0 uses, Ethernet0, zone The O
C Net 11 directly connected, 472 uses, Ethernet1, zone No Parking
R Net 2154 [1/G] via 258.179, 8 sec, 6892 uses, Ethernet0, zone LocalTalk
S Net 1111 via 258.144, 0 uses, Ethernet0, no zone set
```

In the above display, the G rating after `Net 6` indicates *good*. Alternate ratings are S for *suspect* and B for *bad*. These ratings are attained from the routing updates which occur at ten-second intervals. A separate and nonsynchronized event occurs at 20-second intervals, checking and flushing the ratings for particular routes that have not been updated. For each 20-second period that passes with no new routing information, a rating will slip from G to S to B; after one minute with no updates, that route will be flushed. Every time the router receives a useful update, the status of the route in question is reset to G. Useful updates are those advertising a route that is as good or better than the one currently in the table.

Following is a sample display for the extended AppleTalk network. Note the cable range display for *Magnolia Estates*:

```
Codes: R - RTMP derived, C - connected, 29 routes in internet

R Net 3 [1/G] via 254.163, 8 sec, Ethernet1, zone Localtalk
C Net 4 directly connected, Ethernet0, zone Twilight
C Net 6 directly connected, Ethernet3, zone Heavenly
R Net 11 [3/G] via 254.163, 8 sec, Ethernet1, zone UDP
R Net 17 [1/G] via 254.163, 8 sec, Ethernet1, zone UDP
R Net 33 [1/G] via 4.129, 1 sec, Ethernet0, zone Twilight
R Net 36 [1/G] via 254.174, 7 sec, Ethernet1, zone idontcare
R Net 55 [1/G] via 254.130, 9 sec, Ethernet1, zone Hospital
R Net 69 [1/G] via 4.129, 1 sec, Ethernet0, zone Empty Guf
R Net 70 [1/G] via 254.247, 2 sec, Ethernet1, zone Empty Guf
C Net 80 directly connected, Ethernet4, zone Light
R Net 99 [2/G] via 4.129, 1 sec, Ethernet0, zone BammBamm
C Net 254 directly connected, Ethernet1, zone Twilight
R Net 890 [2/G] via 4.129, 1 sec, Ethernet0, zone release lab
R Net 901 [2/G] via 4.129, 1 sec, Ethernet0, zone Dave's House
C Net 999-999 directly connected, Serial3, zone Magnolia Estates
R Net 2003 [4/G] via 80.129, 6 sec, Ethernet4, zone Bldg-13
R Net 2004 [2/G] via 80.129, 6 sec, Ethernet4, zone Bldg-17
R Net 2012 [2/G] via 4.130, 7 sec, Ethernet0, zone Bldg-13
R Net 2013 [3/G] via 254.163, 8 sec, Ethernet1, zone UDP
R Net 2024 [4/G] via 80.129, 3 sec, Ethernet4, zone Bldg-17
R Net 3004 [1/G] via 80.129, 3 sec, Ethernet4, zone Bldg-17
R Net 3012 [1/G] via 4.130, 5 sec, Ethernet0, zone Bldg-13
R Net 3024 [4/G] via 80.129, 3 sec, Ethernet4, zone Bldg-17
R Net 3880 [1/G] via 999.2, 0 sec, Serial3, zone Magnolia Estates
R Net 5002 [2/G] via 80.129, 3 sec, Ethernet4, zone Bldg-17
R Net 5003 [2/G] via 4.130, 5 sec, Ethernet0, zone Bldg-13
R Net 5006 [4/G] via 80.129, 3 sec, Ethernet4, zone Bldg-17
R Net 51489 [3/G] via 4.129, 8 sec, Ethernet0, zone Dave's House

[hops/state] state can be one of G:Good, S:Suspect, B:Bad
```

The next sample shows the result of the **show apple route** command with a specific network.

For the command:

```
show apple route 3880
```

The display looks like this:

```
Codes: R - RTMP derived, C - connected, 29 routes in internet

R Net 3880 [1/G] via 999.2, 7 sec, Serial3, zone Magnolia Estates
  Route installed 1:35:37
  Current gateway: 999.2, 1 hop away, updated 7 secs ago
  Zone list provided by 254.129
  Route has been updated since last RTMP was sent
  Valid zones: Magnolia Estates
```

For the command:

```
show appletalk route serial 3
```

The display looks like this:

```
Codes: R - RTMP derived, C - connected, 29 routes in internet

C Net 999 directly connected, Serial3, zone Magnolia Estates
R Net 3880 [1/G] via 999.2, 3 sec, Serial3, zone Magnolia Estates
```

Displaying AppleTalk Traffic Information

The EXEC command **show apple traffic** displays AppleTalk-specific traffic information. The command has this syntax:

show apple traffic

The statistics it displays include the total number of packets received, categorized errors, summaries of packets received for the various AppleTalk services (for example, NBP, ZIP, DDP) and for other protocols such as Echo and ARP. Several counters have also been added to monitor extended AppleTalk activity.

Following is a sample display of extended AppleTalk activity.

```
AppleTalk statistics:
  Rcvd:  719 total, 0 checksum errors, 0 bad hop count
       0 local destination, 0 access denied
       2 port disabled
  Bcast: 640 received, 164 sent
  Sent:  164 generated, 0 forwarded
         1736 encapsulation failed, 0 no route
  DDP: 719 long, 0 short, 0 wrong size
  NBP: 5 received, 0 sent, 0 forwarded, 0 lookups
  RTMP: 709 received, 240 sent, 0 requests
  ATP: 0 received
  AMP: 0 received, 0 sent
  ZIP: 3 received, 1659 sent
  Echo: 0 received, 0 illegal
  ARP: 1476 requests, 0 replies, 9 probes
  Lost: 0 no buffers
  Unknown: 0 packets
  Discarded: 1577 wrong encapsulation, 0 bad SNAP discriminator
```

Table 10-1 Show Apple Traffic Field Descriptions

Field	Description
bad hop count	Packet dropped, too many hops.
access denied	Packet dropped, access list didn't permit it.
port disabled	Packet dropped, routing disabled for port (extended AppleTalk only). Occurs because of a configuration error or a packet received while in verification/discovery mode.
encapsulation failed	Packet received for a connected network, but node not found.
wrong size	Physical packet and claimed length disagree.
no buffers	Attempted packet buffer allocation failed.
unknown	Unknown AppleTalk packet type.
wrong encapsulation	Nonextended AppleTalk packet on extended AppleTalk port, or the other way around.
bad SNAP discriminator	Extended AppleTalk packet without Apple discriminator (extended AppleTalk only). Occurs when another AppleTalk device has implemented an obsolete or incorrect packet format.

Displaying Zone Information

The **show apple zone** command displays the zone information table and has this syntax:

show apple zone [*zonename*]

Use this command to display which networks comprise each zone for both nonextended and extended AppleTalk networks.

The argument *zonename* specifies the name of the zone you are trying display information on.

In the following sample display, notice the report of cable ranges for the extended zone "Empty Guf":

```
Name                                    Network(s)
UDP                                     17 11
Heavenly                                1161 6
Hospital                                55
Bldg-17                                 82 81 14 13
CSL EtherTalk                           22
Twilight                                1544 254 36 33 4
EtherTalk                               2
Underworld                              666
Magnolia Estates                        3880 999
Light                                   80
LocalTalk                               3
Empty Guf                               69-69
Total of 12 zones
```

Displaying Information About the Sockets

The command **show apple socket** displays information about the process-level processing in all the sockets in the AppleTalk interface. Enter this command at the EXEC prompt:

show apple socket [*socket-number*]

When used with the optional *socket-number* argument, it shows information about a specific socket.

The following is the output seen when no socket number is specified:

```
Socket  Name            Owner                  Waiting*/Processed

    1   RTMP             AT RTMP                 0  99571
    2   NIS              AT NBP                  0   5425
    4   AEP              AT Maintenance          0      1
    6   ZIP              AT ZIP                  0   2704
  253   PingServ         AT Maintenance          0      1
  254   Responde         AT Maintenance          0      1
```

An asterisk (*) indicates the number of packets waiting to be processed.

When a socket is specified, only statistics for that socket are displayed, as seen in following sample output:

```
6   ZIP             AT ZIP                  0   2704
```

Maintaining the AppleTalk Network

Maintaining the AppleTalk network is a simple task. Cisco provides two EXEC commands to clear the different AppleTalk data structures.

Clearing the Neighbor Data Structures

The **clear apple neighbors** command clears the AppleTalk neighbors data structures. Enter this command at the EXEC prompt:

clear apple neighbors

Clearing the Route Data Structures

The **clear apple routes** command clears the AppleTalk route data structures. Enter this command at the EXEC prompt:

clear apple routes

The AppleTalk Ping Command

The EXEC **ping** command sends Echo Protocol datagrams to other AppleTalk nodes to verify connectivity and measure round-trip times.

When the **ping** command prompts for a protocol, specify **appletalk**. Default options are indicated with carriage returns. What follows is a sample of using **ping** with the AppleTalk protocol. To abort a ping session, type the escape sequence (by default, Ctrl-^, X).

Sample Session:

```
Protocol [ip]: appletalk
Target Appletalk address: 1024.128
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Verbose [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echos to 1024.128, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/8 ms
```

The **ping** command uses the characters in Table 10-2 to indicate the success or failure of each packet in the **ping** sequence.

Table 10-2 AppleTalk Ping Characters

Char	Meaning
!	The packet was echoed successfully from the target address.
.	The timeout period expired before an echo was received from the target address.
B	Bad, or malformed echo was received from the target address.
C	An echo was received with a bad DDP checksum.
E	Transmission of the echo packet to the target address failed.
R	The transmission of the echo packet to the target address failed for lack of a route to the target address.

Debugging the AppleTalk Network

The EXEC **debug** commands described in this section are used to troubleshoot the AppleTalk network transactions. Generally, you enter these commands during troubleshooting sessions with Cisco customer engineers.

For each **debug** command, there is a corresponding **undebug** command that turns the display off. Remember that some of these commands can be entered in groups that then display additional information.

debug appletalk

The **debug appletalk** command debugs all start-up messages and protocol routines dedicated to support start-up. This command also debugs global messages such as those regarding neighbors, ports/interfaces, and configuration. The command looks at problems with parts of Appletalk which do not have their own options in other debug commands.

debug apple-aarp

The **debug apple-aarp** command enables debugging of AppleTalk address resolution protocol. A side-effect of enabling this option is that gleaning MAC information from datagrams is disabled.

debug apple-errors

The **debug apple-errors** command reports information about errors that occur. The information displayed by this command is enhanced by enabling debugging for the specific class of errors that you are interested in. This is similar to **debug apple-packets**.

debug apple-event

The **debug apple-event** command displays debugging information about AppleTalk special events, neighbors becoming reachable/unreachable, and interfaces going up/down. Only significant events (for example, neighbor and/or route changes) are logged. This command is maintained in nonvolatile memory, if present.

apple event-logging

The **apple event-logging** configuration command causes logging of a subset of messages produced by **debug appletalk** command.

debug apple-nbp

The **debug apple-nbp** command enables debugging output from the Name Binding Protocol (NBP) routines.

debug apple-packet

The **debug apple-packet** command enables per-packet debugging output. It reports information online when a packet is received or a transmit is attempted. The command allows watching the types of packets being slow switched. It is roughly equivalent to turning on all the other AppleTalk debugging information. There will be at least one line of debugging output per AppleTalk packet processed.

The **debug apple-packet** command, when invoked in conjunction with the commands **debug apple-routing**, **debug apple-zip**, and **debug apple-nbp**, adds protocol processing information in addition to generic packet details. It reports protocol processing, and successful completion or failure information.

The **debug apple-packet** command, when invoked in conjunction with the command **debug apple-errors**, reports packet level problems such as encapsulation problems. This is the case because **debug apple-errors** is a subset of **debug apple-packets**.

debug apple-routing

The **debug apple-routing** command enables debugging output from the Routing Table Maintenance Protocol (RTMP) routines. This command can be used to monitor acquisition of routes, aging of routing table entries, and advertisement of known routes. It also reports conflicting network numbers on the same network if the network is misconfigured.

debug apple-zip

The **debug apple-zip** command enables debugging output from the Zone Information Protocol routines. This command reports significant events such as discovery of new zones and zone list queries.

AppleTalk Global Configuration Command Summary

This section lists all the global commands used with the AppleTalk interface.

no appletalk arp

Resets the **arp interval** and **arp retransmit** commands to their default values.

appletalk arp interval *milliseconds*

Specifies the time interval between retransmission of ARP packets. The argument *milliseconds* specifies the interval. The default and minimum value is 33 milliseconds.

appletalk arp retransmit-count *count*

Specifies the number of retransmissions that will be done before abandoning address negotiations and using the selected address. The argument *count* specifies the retransmission count. The minimum value that can be specified is 1 (one); the default is 20.

[no] appletalk checksum

Enables and disables the generation and verification of checksums for all AppleTalk packets. An incoming packet with a nonzero checksum will be verified against that checksum and discarded if in error. By default, checksum verification is enabled.

[no] appletalk event-logging

Causes logging of a subset of messages produced by **debug appletalk** command. The **no** form of the command turns this function off.

[no] apple proxy-npb *network-number zonename*

Required for each zone that has a nonextended-only AppleTalk router connected to a network in the zone. The argument *network-number* must be a unique network number which will be advertised via this router as if it were a real network. The argument *zonename* is the name of the zone requiring compatibility support. Only one proxy is needed to support a zone, but additional proxies can be defined with different network numbers, if redundancy is desired

[no] appletalk routing

Enables or disables the AppleTalk protocol processing.

[no] appletalk send-rtmp

Allows a router to be placed on a net with AppleTalk, enabled but without being seen. This allows disabling of routine update. The default is to allow the disabling.

[no] appletalk strict-rtmp

Enforces maximum checking of routing packets to insure their validity. The default of this command is to provides maximum checking. The **no** variation disables the maximum checking mode.

[no] appletalk timers *update-interval valid-interval invalid-interval*

Changes the time intervals (in seconds) used in AppleTalk routing. The argument *update-interval* is the time between routing updates sent to other routers on the network; the default is 10 seconds. The argument *valid-interval* is amount of time that the router will consider a route valid without having heard a routing update for that route; the default is 20 seconds, and the value is normally twice the update interval. The argument *invalid-interval* is the amount of time that the router will wait before marking a route invalid; the default is three times the update-interval, or 30 seconds.

AppleTalk Interface Subcommand Summary

This section lists, in alphabetical order, all the interface subcommands used with AppleTalk networks.

[no] access-list *list* {**permit** | **deny**} *network*

Establishes permit or deny conditions for packets on a specific network interface. The argument *list* is an integer between 600 and 699 and the argument *network* is an AppleTalk network. A *network* argument of –1 represents any network. The **no** form of the command removes an entry from the list.

[no] appletalk access-group *list*

Assigns an interface to an access list. The argument *list* specifies the appropriate AppleTalk access list. Use the **no** form of the command to remove the list from the interface.

[no] appletalk address *address*

Assigns AppleTalk addresses on the interfaces that will be used for the AppleTalk protocol. This step must be done prior to assigning zone names. Use this subcommand to configure nonextended interfaces.

[no] appletalk cable-range *start-end* [*network.node*]

Designates an interface as being on an extended AppleTalk network. This range is specified using the *start-end* parameter, which is a pair of decimal numbers between 1 and 65,279, inclusive. The starting and ending addresses can be assigned equal numbers. The optional *network.node* argument specifies the suggested network and node number that will be used first when selecting the AppleTalk address for this interface.

[no] appletalk discovery

Resets the discovery mode and allows a new cable range to be discovered. Use the **no** variation to return the software to the default (off) state.

[no] apple distribute-list *access-list-number* **in**

Filters input from networks. The argument *access-list-number* is the number of a predefined access list. The keyword **in** is used to filter networks received in update.

[no] apple distribute-list *access-list-number* **out**

Filters output from networks. The argument *access-list-number* is the number of a predefined access list. The keyword **out** is used to suppress networks from being sent in updates.

appletalk iptalk-baseport *port-number*

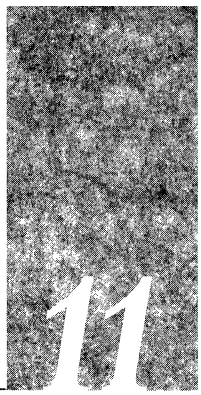
Specifies the UDP port number, which is the beginning of the range of UDP ports used in mapping AppleTalk well-known DDP socket numbers to UDP ports. The argument *port-number* is the UDP port number.

appletalk iptalk *net.node zone*

Encapsulates AppleTalk in IP packets in a manner compatible with the Columbia AppleTalk Package (CAP) IPtalk and the Kinetics IPtalk (KIP) implementations. This command enables IPtalk encapsulation on an interface which already has an configured IP address. The argument *net.node* is a network node number; the argument *zone* the AppleTalk zone.

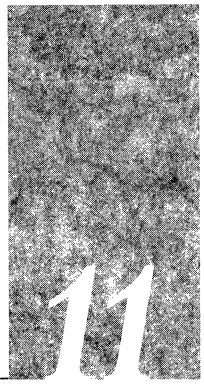
[no] appletalk zone *zonename*

Sets the zone name for the connected AppleTalk network. This command also specifies the zone name associated with the AppleTalk network for the specified interface. The argument *zonename* specifies the name of the zone for the connected AppleTalk network. The argument is ignored for nonextended AppleTalk. The command is ignored if the specified zone name is not in the zone list. The **no** form of the command deletes a zone name from a zone list or the entire zone list if none is specified.



Chapter 11
Routing CHAOSnet

Chapter 11
Routing CHAOSnet



This chapter describes Cisco Systems' implementation of the CHAOSnet routing protocol.

Cisco's Implementation of CHAOSnet

CHAOSnet is a local area network protocol developed at the Massachusetts Institute of Technology in the mid-1970s. Several Artificial Intelligence workstation manufacturers use the CHAOSnet protocol in their networking software products. The Cisco Systems router supports full CHAOSnet routing and a small subset of CHAOSnet host functions, including the status, uptime, and dump-routing-table services. The router can route CHAOSnet packets over Ethernets and synchronous serial lines.

CHAOSnet Addresses

CHAOSnet addresses are 16-bit quantities written as octal numbers. The higher-order eight bits of the address are the CHAOSnet network number, and the lower-order eight bits are the host number. Following is an example of a CHAOSnet address:

```
315.124
```

The Cisco Systems CHAOSnet implementation assumes that the CHAOSnet network corresponds one-to-one with a subnetted Internet network. For example, CHAOSnet subnet 1 must correspond to Internet subnet 1, and CHAOSnet host 360 (octal) must correspond to Internet host 240 (decimal). Because a CHAOSnet network comprised of Cisco Systems router assumes a single underlying Internet network, the router does not route CHAOSnet packets from one Internet network to another.

To form a CHAOSnet address, the router combines the lower eight or fewer bits of the Internet subnet field with the lower eight or fewer bits of the Internet host field. This approach does not assume any particular class of Internet address or subnetting scheme. However, Cisco Systems recommends that at least eight bits of subnet identifier and eight bits of host identifier be used.

Configuring CHAOSnet Routing

To start the CHAOSnet router process, use the **router chaos** global configuration command. The command syntax follows:

router chaos

no router chaos

The **router** process routes CHAOSnet packets and sends CHAOSnet routing updates. The **no router chaos** command disables CHAOSnet routing. See Chapter 14, "The IP Routing Protocols" for more information about the **router** command.

Example:

The following commands start CHAOSnet routing on network 128.88.0.0:

```
router chaos
network 128.88.0.0
```

CHAOSnet routing does not replace Internet routing; an Internet routing protocol, such as RIP, must run concurrently with CHAOSnet routing on the router. To ensure routing table consistency, the Internet routing protocol must have a greater administrative distance than the CHAOSnet routing protocol. In addition, you must configure the CHAOSnet routing process to re-advertise subnet routes derived from the Internet routing protocol.

Continuing the previous example, suppose RIP is the routing protocol running concurrently with CHAOSnet. The following router subcommand advertises RIP-derived routes to the CHAOSnet hosts on the network:

```
redistribute rip
```

See the section "Redistributing Routing Information" in Chapter 14, "The IP Routing Protocols," for more information on protocol-independent routing issues such as administrative distance and redistribution.

Monitoring CHAOSnet

Use the EXEC commands described in this section to monitor activity on the CHAOSnet.

show chaos-arp

The command **show chaos-arp** displays CHAOSnet-specific ARP entries as 16-bit octal addresses.

show ip route

The command **show ip route** displays routing entries obtained from the CHAOSnet routing protocol. In the command output, CHAOSnet entries are marked by `X` in the first column.

show ip traffic

The command **show ip traffic** displays statistics on CHAOSnet protocol operation.

Debugging CHAOSnet

Use these EXEC commands described in this section to display reports of problems and activity on the CHAOSnet.

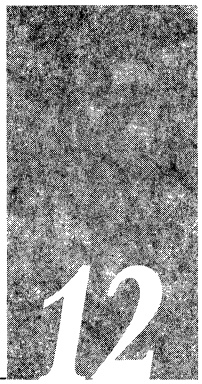
debug chaos-routing

The command **debug chaos-routing** enables logging of CHAOSnet routing activity including service requests.

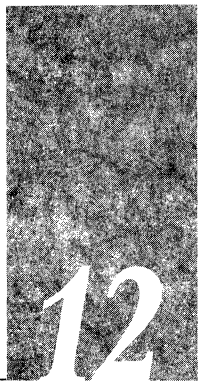
debug chaos-packet

The command **debug chaos-packet** enables logging of CHAOSnet packet transactions.

Chapter 12
Routing DECnet



Chapter 12
Routing DECnet



This chapter describes the Cisco Systems implementation of DECnet Phase IV for the Cisco network server product line. Topics and tasks described in this chapter include:

- Cisco's implementation of DECnet
- An overview of DECnet routing and addressing
- How to enable DECnet routing
- How to configure inter-area and intra-area routing costs
- How to configure access lists
- How to set default routers and priority values
- How to fine-tune DECnet performance parameters, including fast switching

DECnet Phase IV is equivalent to ISO CLNS, which is described in Chapter 15. Support for DECnet Phase IV/Phase V conversion is discussed in this chapter.

Cisco's Implementation of DECnet

Digital Equipment Corporation (DEC) designed the DECnet stack of protocols in the 1970s as part of its Digital Network Architecture (DNA). DECnet has been evolving through its lifetime to its present form known as Phase IV. DECnet support on a Cisco router includes local- and wide-area DECnet Phase IV routing over Ethernets, Token Ring, FDDI, and serial lines, with the following restrictions:

- The router uses HDLC framing rather than DEC's DDCMP framing for point-to-point lines. If you construct a network using both Cisco Systems and DEC equipment, you must ensure that each point-to-point line has the same type of equipment on both ends.
- Cisco and DECnet Phase IV routers have incompatible X.25 support. As with point-to-point lines, you must use a single vendor's equipment on the X.25 portion of your network.
- There is no standard for running DECnet over Token Ring, so you must use the same vendor's routers on a Token Ring LAN (or bridged LAN network).
- Cisco gives you additional security options through access lists.

Cisco routers can support the Address Translation Gateway (ATG), which allows the router to participate in multiple, independent DECnet networks, and to establish a user-specified address translation table for selected nodes between networks.

DEC uses some non-routable protocols that are not part of the DECnet stack. Neither Cisco nor DEC routers can route protocols like MOP (discussed later in this chapter), and LAT, the DEC terminal server protocol. These protocols must be bridged; bridging concepts are described in Chapter 20, "Configuring Transparent Bridging," in Part Five of this manual.

Cisco's DECnet Phase IV and V implementation is supported on Token Ring, provided certain implementation guidelines are met. See the sections "DECnet Phase IV to Phase V Conversion" and "Configuring DECnet on Token Ring" for more information. Since DEC has not defined a standard for running DECnet over Token Ring, all Token Ring DECnet routers on a ring must be from the same vendor.

DECnet Phase IV Addresses

DECnet Phase IV addresses are specified by area number and node number separated by a period. DECnet addresses are written as a dotted pair of area and node numbers. For example, *53.6* is node *6* in area *53*.

DECnet hosts exist as a *node* (host) in an *area*. Do not confuse the concept of *area* with an area defined by the IP, XNS, or other routing protocols. Unlike these protocols, DECnet allows for an area to span many routers, and for a single cable to have many areas attached to it. Therefore, if a host (such as a router) exists on many cables, it uses the same area/node for itself on all of them. Note how this differs from other routing protocols where each interface is given a different internetwork address. Figure 12-1 shows the DECnet approach.

The area number is six bits long (1 through 63); the node number is 10 bits long (1-1023). To determine the MAC address, the 16 bits of the DECnet address are converted into two binary-coded byte sections, then these bytes are translated into hexadecimal numbers and appended to the address *AA 00.0400* in the following order: the first byte is placed last and the second byte next to last. This example illustrates how to convert the DECnet address *12.75*:

- Area 12 Node 75 converts to binary

  00110000 01001011

- Make into two bytes, then swap order and convert to Hex:

  0100 1011 0011 0000
  4B 30

These numbers are appended to the address *AA00.0400*, so the MAC address is then *AA00.0400.4B30*.

If all this seems confusing, you can also use the EXEC **show interfaces** command to obtain the MAC address once DECnet routing is enabled.

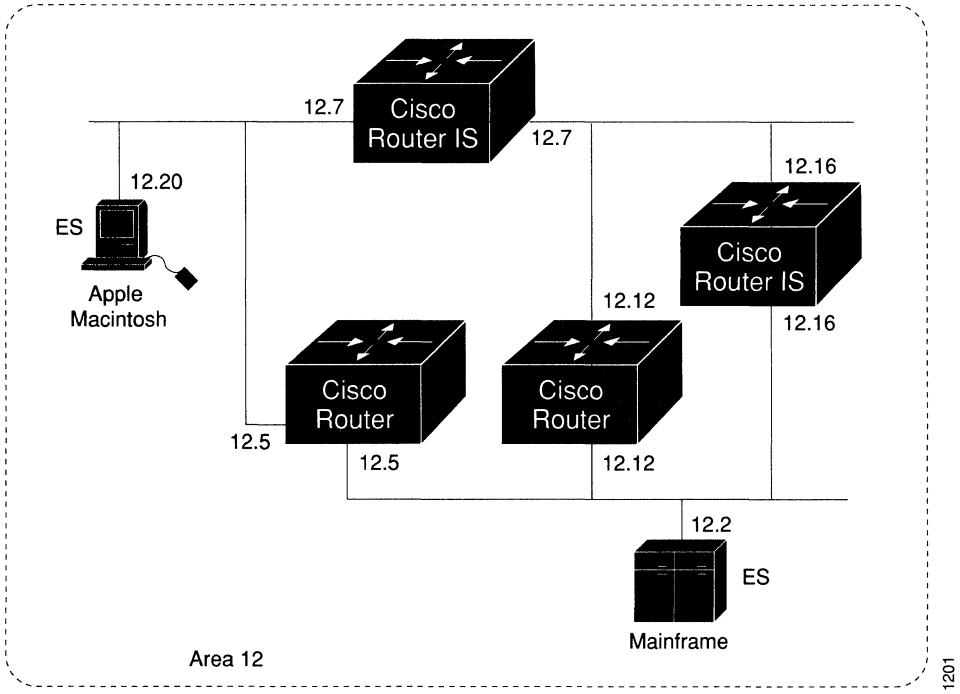


Figure 12-1 DECnet Nodes and Areas

- The DECnet Phase IV protocol associates addresses with machines, not interfaces. Therefore, a router can have only one DECnet Phase IV address for each DECnet network in which it participates. A DECnet MAC-level address is simply an encoded version of the 16-bit area/node combination. This explains why Ethernet interface addresses change on a Cisco router when the DECnet protocol is enabled.
- DECnet does not have the equivalent of the IP Address Resolution Protocol (ARP); DECnet hosts simply advertise their presence with periodic HELLO packets. Routers build local routing tables and hosts learn each other's addresses by listening to host HELLO messages. Hosts learn about nearby routers by listening to router HELLO messages.

You do not have to set each interface address manually; the **decnet routing** global configuration command automatically assigns an address to each interface for which you entered a **decnet cost** configuration command. (These commands are described later in this chapter.)

The parameters in the Cisco Systems implementation of DECnet are a subset of the parameters you can modify in DEC's Network Control Program (NCP). Cisco Systems uses the same names, the same range of allowable values, and the same defaults wherever possible. Note that you must use the configuration commands to set DECnet parameters; the Cisco Systems DECnet implementation does not set parameters by communicating with NCP.

Configuring DECnet Routing

Follow these steps to start configuring your router for DECnet routing:

Step 1: Enable DECnet routing and specify which system-wide host address to use, use with the **decnet routing** global configuration command.

Step 2: After DECnet routing has been enabled, a cost must be assigned to each interface over which DECnet should run. This enables the interface. DECnet nodes route towards a destination using the lowest path cost, so you should base your cost values on interface throughput. Use the **decnet cost** interface subcommand to set a cost value for an interface.

Step 3: Next, specify with the **decnet node-type** command the node type, either an area router—Level 1 and Level 2—or a local router routing DECnet Phase IV at Level 1 only.

Step 4: You can alter the maximum node number and maximum area number with the optional **decnet max-address** and **decnet max-area** commands.

Step 5: Finally, you must specify several commands for either intra-area or inter-area routing. These commands and their parameters must be chosen carefully, as they are dependent on each other's values in many cases.

The following sections take you through these steps in detail, as well as all the optional commands for managing performance, security, Phase IV/V conversion, and so on. The section "DECnet Configuration Examples" shows complete configuration examples for many common situations.

Enabling DECnet Routing

To enable or disable DECnet routing, use the **decnet routing** global configuration command:

decnet routing *decnet-address*

no decnet routing

The argument *decnet-address* takes as its value an address in DECnet format X.Y, where X is the area number and Y is the node number. There is no default router address; you must specify this parameter for DECnet operation.

Example:

In the example below, DECnet routing is enabled for the router in area 21 with node number 456:

```
decnet routing 21.456
```

Note: Enabling DECnet changes the MAC addresses of the router's interfaces. This is not a problem on routers equipped with nonvolatile memory. On systems that attempt to get their IP network addresses from network servers rather than from nonvolatile memory, there may be a problem with the hardware addresses changing and confusing other IP-speaking hosts. If you are attempting to use DECnet on such a configuration, be sure to set all global DECnet parameters before enabling DECnet routing on the interfaces.

Assigning the Cost

After DECnet routing has been enabled, you must assign a cost to each interface over which you want DECnet to run. (Assigning a cost in effect enables DECnet routing for an interface.) Most DECnet installations have an individualized routing strategy for using costs. Therefore, check the routing strategy used at your installation to ensure that costs you specify are consistent with those set for other hosts on the network.

The **decnet cost** interface subcommand sets a cost value for an interface:

decnet cost *cost-value*

no decnet cost

The argument *cost-value* is an integer from 1 to 63. There is no default cost for an interface, although a suggested cost for FDDI is 1, for Ethernets is 4 and for serial links is greater than 10. Use the **no decnet cost** subcommand to disable DECnet routing for an interface.

Example:

The example below establishes a DECnet routing process for the router at *21.456*, then sets a cost of four for the Ethernet 0 interface.

```
decnet routing 21.456
interface ethernet 0
decnet cost 4
```

Figure 12-2 shows four routers, three Ethernet and the various routes linking them. Each link has a different cost associated with it. The least expensive route from router 7 to router 20 is via router 12.

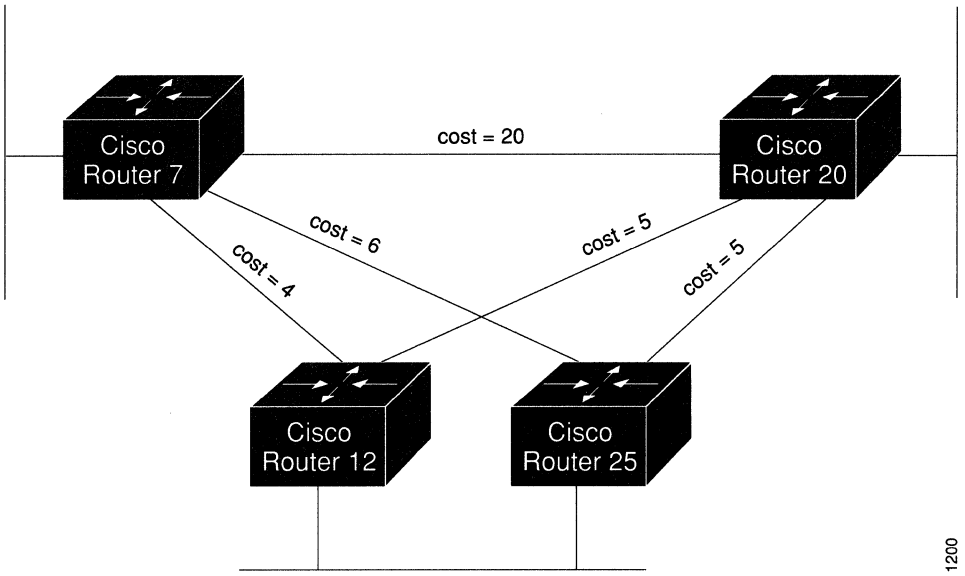


Figure 12-2 DECnet Cost Values

Specifying the Node Type

Before you use many of the global and interface configuration commands, you must specify the node type with the **decnet node-type** global configuration command. The **decnet node-type** command specifies the node type for the router.

decnet node-type {**routing-iv** | **area**}

The options are either **area** or **routing-iv**. If you specify **area,** the router participates in the DECnet routing protocol with other area routers, as described in the DEC documentation, and routes packets from and to routers in other areas. This is sometimes referred to as Level 2, or inter-area, routing. An area router does not just handle inter-area routing; it also acts as an intra-area or Level 1 router. If you specify **routing-iv** (the default), the router acts as an intra-area (standard DECnet Phase IV, Level 1 router) and ignores Level 2 routing packets. In this mode, it routes packets destined for other areas via the least-cost path to an inter-area router, exchanging packets with other end-nodes and routers in the same area.

Specifying Node Numbers and Area Sizes

DECnet routers do not have the concept of aging out a route. Therefore, all possible areas or nodes must be advertised as unreachable if they cannot be reached. Since it is best to keep routing updates small, you need to indicate the default maximum possible node and area numbers that can exist in the network. The default value for a node address to be given in an update is 1023.

You can use the **decnet max-address** global configuration command to configure the router with a different maximum node address, as follows:

decnet max-address *value*

The argument *value* is a number, less than or equal to 1023, that represents the maximum node address possible on the network. In general, all routers on the network should use the same value for this parameter.

Example:

The example below configures a small network (spanning just a department). The desire is to keep routing updates as small as possible, so the maximum address value is set to 300 instead of the default of 1023.

```
!
decnet max-address 300
!
```

Use **decnet max-area** global configuration command to set the largest number of areas that the router can handle in its routing table. The syntax is as follows:

decnet max-area *value*

The argument *value* is an area number from 1 to 63; the default is 63. Like the **decnet max-address** command value, this parameter controls the sizes of internal routing tables and of messages sent to other nodes. All routers on the network should use the same maximum address value.

Example:

In this example, the maximum number of areas that the router will save in its routing table is 45.

```
!
decnet max-area 45
!
```

Specifying the Maximum Route Cost for Inter-Area Routing

The **decnet area-max-cost** global configuration command sets the maximum cost specification value for inter-area routing. The syntax of this command follows:

decnet area-max-cost *value*

The argument *value* determines the maximum cost for a route to a distant area that the router may consider usable; the router treats as unreachable any route with a cost greater than the value you specify. A valid range for cost is from 1 to 1,022; the default is 1,022. This parameter is only valid for area routers. Make sure you've used the **decnet node-type area** command before using this command.

Example:

In this example, the node type is specified as area and the maximum cost is set to 500. Any route whose cost exceeds 500 will be considered unreachable by this router.

```
!
decnet node-type area
decnet area-max-cost 500
!
```

Use the **decnet area-max-hops** global configuration command to set the maximum hop count value for inter-area routing as follows:

decnet area-max-hops *value*

The argument *value* determines the maximum number of hops for a usable route to a distant area. The router treats as unreachable any route with a count greater than the value you specify. A valid range for the hop count is from 1 to 30; the default is 30. This parameter is only valid for area routers. Make sure you've used the **decnet node-type area** command before using this command.

Example:

This example sets the node type to area, then sets a maximum hop count of 21. This was done because it is a small network with relatively few routers for inter-area routing, so a route with a large hop count is liable to represent a problem, not an efficient route.

```
!
decnet node-type area
decnet area-max-hops 21
!
```

Specifying the Maximum Route Cost for Intra-Area Routing

The **decnet max-cost** global configuration command sets the maximum cost specification for intra-area routing. The router ignores routes within the router's local area that have a cost greater than the corresponding value of this parameter. The syntax for this command follows:

decnet max-cost *value*

The argument *value* is a cost from 1 to 1,022 (the default).

Example:

In this example, the node type is specified as DECnet Phase IV and the maximum cost is set to 335. Any route whose cost exceeds 335 will be considered unreachable by this router.

```
!
decnet node-type routing-iv
decnet max-cost 335
!
```

Use the **decnet max-hops** global configuration command to set the maximum hop count specification value for intra-area routing, as follows:

decnet max-hops *value*

The argument *value* is a hop count from 1 to 30 (the default). The router ignores routes that have a hop count greater than the corresponding value of this parameter.

Example:

This example sets the node type to DECnet Phase IV routing, then sets a maximum hop count of 2.

```
!
decnet node-type routing-iv
decnet max-hops 2
!
```

Configuring Maximum Visits

Use the **decnet max-visits** global configuration command to set the limit on the number of times a packet can pass through a router.

decnet max-visits *value*

The *value* keyword can vary from 1 to 63 (the default). If a packet exceeds *value*, the router discards the packet. DEC recommends that the value of the **max-visits** parameter be at least twice that of the **max-hops** parameter, to allow packets to still reach their destinations when routes are changing.

Example:

This example of intra-area routing configuration specifies Phase IV routing, a maximum hop count of 28, and maximum number of visits of 62 (which is more than twice 28).

```
!
decnet node-type routing-iv
decnet max-hops 28
decnet max-visits 62
!
```

Configuring Path Selection

Limiting the number of *equal cost* paths can save memory on routers with limited memory or very large configurations. Additionally, in networks with a large number of multiple paths, and end-systems with limited ability to cache out-of-sequence packets, performance may suffer when traffic is split between many paths.

Limiting the size of the routing table will not affect your router's ability to recover from network failures transparently provided that you do not make the maximum number of paths too small. If more than the specified number of equal cost paths exist and one of those paths suddenly becomes unusable, the router will discover an additional path from the paths it has been ignoring.

The first of the optional path global configuration commands, **decnet max-paths**, defines the maximum number of equal cost paths to a destination that the router will keep in its routing table, with the following syntax:

decnet max-paths *value*

The argument *value* is a decimal number equal to the maximum number of equal cost paths the router will save. The highest value accepted is 31; the default value is 1.

Example:

In the following example, some destinations have six equal cost paths, so the example specifies that the router will save no more than three equal cost paths.

```
!
decnet max-paths 3
!
```

The **decnet path-split-mode** global configuration command also helps you make decisions about equal cost paths; it specifies how the router will split the routable packets between equal cost paths. This command has two forms, as shown:

decnet path-split-mode normal
decnet path-split-mode interim

The keyword **normal** selects normal mode (the default), where equal cost paths are selected on a round-robin basis. The keyword **interim** specifies that traffic for any particular (higher-layer) session is always routed over the same path. This mode supports older implementations of DECnet (VMS versions 4.5 and earlier) that do not support out-of-order packet caching. Other sessions may take another path, thus utilizing equal cost paths that a router may have for a particular destination.

Altering DECnet Defaults

In general, you need not modify the DECnet parameters. However, under special circumstances, or when using a specific configuration, you will see better performance if you alter some of the default parameters. This section will guide you through those special circumstances.

Adjusting Timers and the Route Cache

The router broadcasts HELLO messages on all interfaces with DECnet enabled. Other hosts on the network use the HELLO messages to identify the hosts with which they can communicate directly. The router sends HELLO messages every 15 seconds by default. On

extremely slow serial lines, you may want to increase this value to reduce overhead on the line using the **decnet hello-timer** interface subcommand.

decnet hello-timer *value*
no decnet hello-timer

The keyword *value* varies from 1 to 8,191 seconds; the default is 15 seconds.

Example:

The following example increases the HELLO interval to 2 minutes (120 seconds) on serial interface 1.

```
interface serial 1
decnet hello-timer 120
```

By default, Cisco's DECnet routing software implements fast switching of DECnet datagrams by the MCI network interface board. There are times when it makes sense to disable fast switching. This is especially important when using rates slower than T1.

Fast switching uses memory space on the MCI board. In situations where a high bandwidth interface is writing large amounts of information to a low bandwidth interface, additional memory could help avoid congestion on the slow interface (also known as big-pipe/little-pipe problems). Use the **no decnet route-cache** interface subcommand to turn off fast switching.

decnet route-cache

no decnet route-cache

In a network where changes occur very seldom or do not need to be responded to immediately (it is small and uncomplicated; applications are not particularly critical; slow serial links, and so on.), increasing the time between routing updates reduces the amount of unnecessary network traffic. The **decnet routing-timer** interface subcommand specifies how often the router sends routing updates that list all the hosts that the router can reach. Other routers use this information to construct local routing tables. DEC calls this parameter the *broadcast routing timer* because they have a different timer for serial lines; the Cisco Systems DECnet implementation does not make this distinction. The syntax for the **decnet routing-timer** interface subcommand follows:

decnet routing-timer *value*

no decnet routing-timer

The argument *value* specifies a time from 1 to 65,535 seconds; the default is 40 seconds. The **no decnet routing-timer** command restores this default.

Example:

In the following example, a serial interface is set to broadcast routing updates every two minutes.

```
interface serial 0
decnet routing-timer 120
```

Specifying the Designated Router

The *designated* router is that router to which all end nodes on an Ethernet communicate if they do not know where else to send a packet. The designated router is chosen through an election process in which the router with the highest priority gets the job. When two or more routers in a single area share the same, highest priority, the unit with the highest node number is elected. You can reset a router's priority to help ensure that it is elected designated router in its area.

Priority may be changed with the **decnet router-priority** interface subcommand, as shown below:

decnet router-priority *value*

The argument *value* can range from zero through 127; the default priority is 64.

Example:

In the following example, interface Ethernet 1 is set to a priority of 110.

```
!
interface ethernet 1
decnet router-priority 110
!
```

Configuring DECnet on Token Ring

The Cisco routers support DECnet on Token Ring interfaces. This protocol is fully supported between Cisco routers, including the Address Translation Gateway. Since DEC has not defined a standard for encapsulating DECnet Phase IV datagrams on Token Ring, it is unlikely that the Cisco implementation will operate with other DECnet implementations on Token Ring. The intent of the Cisco implementation is to use Token Ring as a transit network medium, especially for backbones.

Configuring a Cisco router for DECnet on Token Ring is very similar to configuring DECnet on Ethernet. The only difference is the specification of a Token Ring rather than an Ethernet interface. The syntax for the interface command in shown below:

interface tokenring *number*

The parameter *number* refers to the interface number.

Example:

The example below sets interface Token Ring 0 for DECnet routing.

```
!
interface tokenring 0
decnet cost 4
!
```

Managing Traffic Using DECnet Access Lists

There are two forms of DECnet access lists: one that specifies a single address (a standard list) and one that specifies two addresses (an extended list). See the section "Configuring IP Access Lists" in Chapter 13 for general information about setting up access lists.

Configuring DECnet Access Lists

Use the **access-list** global configuration command to create an access list:

access-list *list* {**permit** | **deny**} *address mask*

no access-list *list*

The argument *list* is an integer you choose between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments.

The standard form of the DECnet access list has a DECnet *address* followed by a *mask,* also in DECnet address format, with bits set wherever the corresponding bits in the address should be ignored. DECnet addresses are written in the form *area.node* (for example, 50.4 is node 4 in area 50). All addresses and masks are in decimal.

Example:

This example sets up access list 300 to deny packets going out to the destination node 4.51 and permit packets destined to 2.31.

```
!
access-list 300 deny 4.51 0.0
access-list 300 permit 2.31 0.0
!
```

Configuring Extended Access Lists

Use this global configuration command to create extended access lists:

access-list *list* {**permit** | **deny**} *source source-mask destination destination-mask*

no access-list *list*

The extended form of the DECnet access list has a source DECnet address and mask pair followed by a destination DECnet address and mask pair.

The argument *list* is an integer you choose between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments.

Example:

In the example below, access list 301 is configured to allow traffic from any host in networks 1 and 3. It implies no other traffic will be permitted. (The end of a list contain an implicit "deny all else" statement.)

```
!
access-list 301 permit 1.0 0.1023 0.0 63.1023
access-list 301 permit 3.0 0.1023 0.0 63.1023
!
```

Configuring Access Groups

The **decnet access-group** interface subcommand applies an access list to an interface.

decnet access-group *list*

The argument *list* can be either a standard or extended DECnet access list. A standard DECnet access list applies to destination addresses in this case.

Example:

The following example applies access list 389 to interface Ethernet 1.

```
!
interface ethernet 1
decnet access-group 389
!
```

Configuring In- and Out-Routing Filters

The **decnet in-routing-filter** interface subcommand provides access control to HELLO messages, or routing information received on this interface. Addresses that fail this test are treated as unreachable. The full syntax of the command follows.

decnet in-routing-filter *list*

no decnet in-routing-filter

The argument *list* is a standard DECnet access list.

The **no decnet in-routing-filter** command removes access control.

Example:

In the following example, interface Ethernet 0 is set up with a DECnet in-routing filter of 321, which means that any routing of HELLO messages sent from addresses that are denied in list 321 will be ignored. Additionally, all node addresses listed in received routing messages on this interface will be checked against the access list, and only routes passing the filter will be considered usable.

```
!
interface ethernet 0
decnet in-routing-filter 321
!
```

The **decnet out-routing-filter** interface subcommand provides access control to routing information being sent out on this interface. Addresses that fail this test are shown in the update message as unreachable.

decnet out-routing-filter *list*

no decnet out-routing-filter

The argument *list* is a standard DECnet access list.

The **no decnet out-routing-filter** command removes access control.

Example:

In the following example, interface Ethernet 1 is set up with a DECnet in-routing filter of 351, which means that any addresses that are denied in list 351 will not be sent out in routing updates on this interface.

```
!
interface ethernet 1
decnet out-routing-filter 351
!
```

DECnet Phase IV to Phase V Conversion

DECnet Phase V is OSI-compatible and conforms to the ISO 8473 (CLNP/CLNS) and ISO 9542 (ES-IS) standards. See Chapter 15 for an explanation of configuring OSI CLNP routing and for a review of the terminology. The following guidelines apply to the Phase IV/Phase V conversion:

- DECnet Phase IV hosts can communicate with other DECnet Phase IV hosts, and with any DECnet Phase V host whose address conforms to DEC's Phase IV address standard.
- Phase V End Systems can detect that they are communicating with a Phase IV system and that Phase IV mode is required.
- DECnet Phase IV hosts can use Phase IV routers as Phase V routers.
- DECnet Phase V hosts can use Phase IV routers as Phase V routers.
- If a Phase IV host is using a Phase V router, it is necessary to turn on DECnet Phase IV/Phase V conversion.
- DEC has defined an algorithm for mapping a subset of the Phase V address space onto the Phase IV address space, and also an algorithm for converting Phase IV and Phase V packets back and forth. This allows a network administrator to tunnel Phase IV packets in a Phase V network, or the other way around. Cisco supports the DEC algorithm.

Tunneling

A process known as *tunneling* allows Phase IV packets to traverse a Phase V backbone and vice versa. Each Phase IV host that needs to tunnel must be behind a router that is running DECnet Phase IV and CLNS and that has DECnet conversion enabled, as shown in Figure 12-3. There must be a path of Cisco routers with CLNS enabled between each pair of Phase IV hosts that need to communicate. However, all of these routers do not need to be running Phase IV.

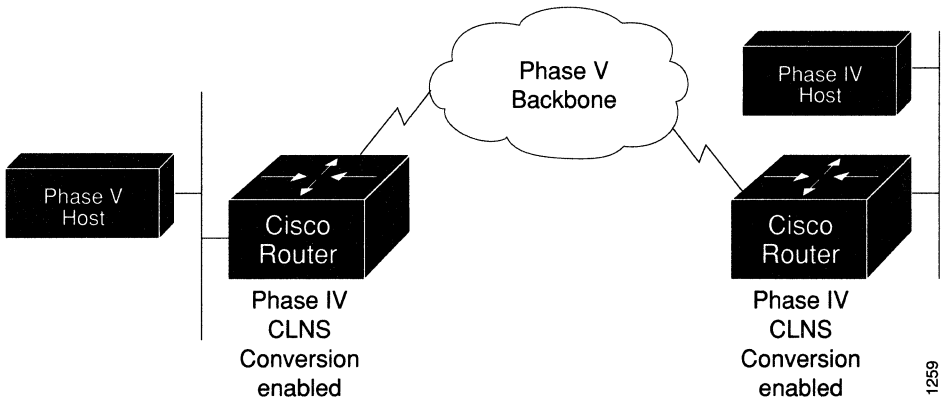


Figure 12-3 DECnet Tunneling

When a Phase IV ES HELLO message is received, the router makes a neighbor entry in both the Phase IV and Phase V ES databases. When a Phase IV packet comes in, the router looks in the Phase IV database for the destination address. If it cannot find it, it will look in the Phase V database. If it is in the Phase V database, it will be converted and sent across the backbone as a CLNS packet. Before the packet is sent on to its final destination, the router will check to see if the packet should be delivered as a CLNS or a Phase IV packet and do the conversion if needed. You may also tunnel Phase V packets across a Phase IV backbone. Phase V packets must conform to Phase IV addresses if you want to tunnel them across a Phase IV backbone. See the section "Phase V Addresses That Conform to Phase IV Addresses."

Configuring DECnet Conversion

Use the **decnet conversion igrp** global configuration command to enable the conversion for the router.

decnet conversion igrp *area-tag*

The argument *area-tag* is an ISO CLNS area name you defined for the CLNS network. Check Chapter 15 for more information on CLNS area tags.

Example:

To enable DECnet conversion on a Cisco router with the CLNS area tag *Field*, enter the following configuration commands:

```
!
decnet conversion igrp Field
!
```

Phase V Addresses That Conform to Phase IV Addresses

Phase V addresses (in the form of Network Service Access Points or NSAPs) are interpreted in the following way:

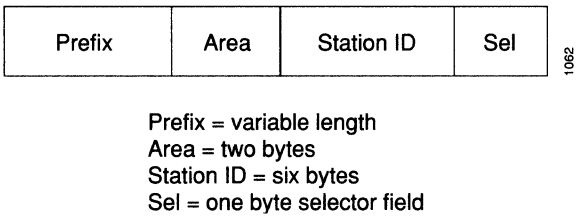


Figure 12-4 Phase V Address Format

In order for a Phase V address to be Phase IV-conforming, it must obey the following restrictions:

- The prefix must match the prefix for the CLNS router specified with the **decnet conversion** command.
- The area must be in the range of 0 to 63.
- The high-order portion of the station ID must be AA-00-04-00 (hexadecimal).

When using DECnet Phase IV routing to configure a network and pure Phase IV routers (that is, non-Cisco routers, or Cisco routers that are not CLNS-configured) are in the path between two hosts (Phase IV or Phase V), the only addresses that can pass through the Phase IV router are Phase IV-conforming addresses.

An example of a configuration would look like this:

```
!
decnet routing 32.6
!
clns routing
clns router igrp willow net 47.0004.004d.0020.AA00.0400.0680.00
decnet conversion igrp willow
!
interface ethernet 0
decnet cost 10
clns router igrp willow
!
```

Remember to convert the DECnet area to a CLNS area. DECnet areas are in decimal; CLNS areas are in hexadecimal. In this example, a DECnet area of 32 translates into a hexadecimal area of 20. Use the **no decnet conversion** command to disable this feature.

DECnet Configuration Examples

This section includes configuration examples, showing many common DECnet configuration activities.

Establishing Routing; Setting Interfaces; Maximum Address Space

The configuration subcommands in the example below establish DECnet routing on a Cisco router. The first line establishes DECnet routing for a specific address. The second line sets the maximum address space at 1023 addresses. The second section sets a cost of 4 for the Ethernet 0 interface. The third section sets a cost of 10 for the serial 1 interface.

Example:

```
!
decnet routing 4.27
decnet max-address 1023
!
interface ethernet 0
decnet cost 4
!
interface serial 1
decnet cost 10
!
```

Level 1 and Level 2 Routing; Designated Router

In the first part of this configuration, the router that is being set up with an area and node address in the first line, then it is being designated a Level 2 (area) router. In the lines that follow, the two Ethernet interfaces are given costs of four.

Example 1:

```
!
decnet routing 6.10
decnet node area
!
interface ethernet 0
decnet cost 4
interface ethernet 1
decnet cost 4
!
```

In the second example, you want the router to be the designated router, so assign it the highest possible node address. You designate it as an area router and you assign a cost of 4 and a router priority of 127 to the Ethernet, and a cost of 20 to the serial connection.

Example 2:

```
!
decnet routing 6.1023
decnet node area
!
interface ethernet 0
decnet cost 4
decnet router-priority 127
!
interface serial 0
decnet cost 20
!
```

In the third example, we want the router to be a Level 1 router and we need to give it an address in area 7. The serial link is slower than in the previous example (9.6 vs. 56 Kbps) so it has a higher cost.

Example 3:

```
!
decnet routing 7.12
decnet node routing-iv
!
interface ethernet 0
decnet cost 4
interface ethernet 1
decnet cost 4
interface serial 0
decnet cost 25
!
```

Phase IV to Phase V Conversion

This example begins by enabling DECnet routing with a specific address of *54.6*. It then specifies the area the name *Field* (as in Field Offices) with the **clns router igrp** command. Remember that this command must be typed on one line, with the complete address following the **net** keyword.You should already have a CLNS area named *Field* before you use this command. This is the last time you will have have to use the full hexadecimal address; from now on all your commands can simply specify *Field* as the area.

At this point you have set the stage for the **decnet conversion** command, which specifies that you will be using IGRP as the routing protocol in the area named *Field*.

After you have enabled the conversion, you need to name the specific interfaces that you want to route DECnet packets. In this example, the interface Ethernet 0 is enabled, with a cost of ten. The **clns router igrp** command with the *Field* area is needed to specify that the interface will be using IGRP and Phase V CLNS and that it is part of area *Field*.

You could follow this interface specification with other interface specifications such as Ethernet 1, serial 0, and so on, with the same three commands. You could also go on to specify access lists and other special commands for these specific interfaces.

Example:

```
!
decnet routing 54.6
clns router igrp Field NET 47.0006.02.000000.0000.0100.0036.AA00040006D8
decnet conversion igrp Field
interface ethernet 0
decnet cost 10
clns router igrp Field
!
```

Note: Make sure that the area you specify in the **decnet conversion** command is the same as the area you specified for the CLNS network (*Field*, in this case). Also note that the DECnet area is specified in *decimal*, and the CLNS area is specified in *hexadecimal*.

The Address Translation Gateway

The Address Translation Gateway (ATG) allows a Cisco router to route traffic for multiple independent DECnet networks and to establish a user-specified address translation for selected nodes between networks. This allows connectivity between DECnet networks which might be otherwise not connectable due to address conflicts between the networks. The ATG allows you to define multiple DECnet networks and map between them. This may be done over all media types.

ATG Command Syntax

The ATG configuration commands are basically a modification to the standard DECnet global configuration commands.

The general syntax of the DECnet ATG command follows:

decnet *network-number keywords*

The argument *network-number* specifies the network number in the range 0 through 3, and the argument *keywords* is one of the configuration keywords (**area-max-cost,** for example). Commands without the *network-number* modifier apply to *network 0.*

You may also establish a translation entry to translate a virtual DECnet address to a real DECnet address by using this global configuration command:

decnet *first-network* **map** *virtual-address second-network real-address*

The arguments *first-network* and *second-network* are DECnet network numbers in the range 0 through 3. The arguments *virtual-address* and *real-address* are specified as numeric DECnet addresses (10.5, for example).

ATG Configuration Examples

In Figure 12-5, the Cisco router is connected to two DECnet networks using Ethernet. The examples following Figure 12-5 refer to the configuration in the figure.

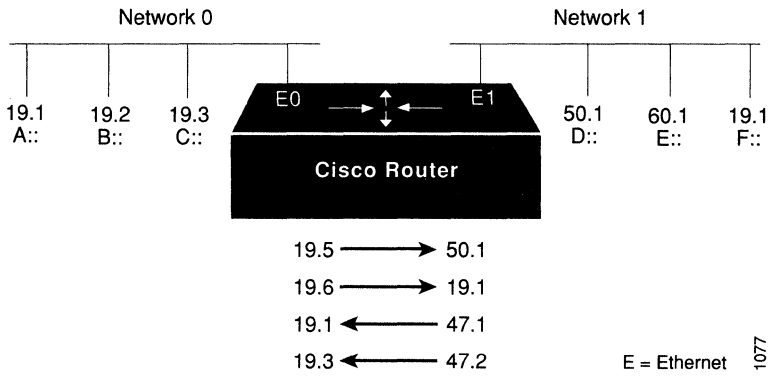


Figure 12-5 ATG Configuration Example

Example:

In Network 0, the router is configured at address *19.4* and is a Level 1 router. In Network 1, the router is configured at address *50.5* and is an area router. At this point, no routing information is exchanged between the two networks. Each network in the router has a separate routing table.

```
!
decnet 0 routing 19.4
decnet 0 node routing-iv
interface ethernet 0
decnet 0 cost 1
!
decnet 1 routing 50.5
decnet 1 node area
interface ethernet 1
decnet 1 cost 1
!
```

To establish a translation map, enter these commands:

```
decnet 0 map 19.5 1 50.1
decnet 0 map 19.6 1 19.1
decnet 1 map 47.1 0 19.1
decnet 1 map 47.2 0 19.3
```

Packets in Network 0 sent to address *19.5* will be routed to Network 1 and the destination address will be translated to *50.1*. Similarly, packets sent to address *19.6* in Network 0 will be routed to Network 1 as *19.1*; packets sent to address *47.1* in Network 1 will be routed to Network 0 as *19.1*; and packets sent to *47.2* in Network 1 will be sent to Network 0 as *19.3*.

The following table depicts a packet exchange between nodes A and D:

	Source		Dest.	
a packet addressed as:	19.1	→	19.5	received on Ethernet0
translates to:	47.1	→	50.1	and is transmitted out Ethernet1
a reply packet:	50.1	→	47.1	received on Ethernet1
translates to:	19.5	→	19.1	and is transmitted on Ethernet0

1290

Network 0 uses a block of addresses from its area to map the remote nodes. In network 0, the router will advertise nodes 19.5 and 19.6. These nodes must not already exist in network 0.

Network 1 uses another area for the address translation. Since the router will be advertising the availability of area 47, that area should not already exist in network 1 because DECnet area fragmentation could occur.

Only nodes that exist in the maps on both networks will be able to communicate directly. Network 0 node 19.1 will be able to communicate with Network 1 node 50.1 (as 19.5), but will not be able to communicate directly with Network 1 node 60.1.

When naming nodes, use the appropriate address in each network. See the following lists for examples.

Network 0 VMS NCP Command File Sample

```
$ MCR NCP
define node 19.1 name A
define node 19.2 name B
define node 19.3 name C
define node 19.4 name GS
define node 19.5 name D
define node 19.6 name F
```

Network 1 VMS NCP Command File Sample

```
$ MCR NCP
define node 50.1 name D
define node 50.5 name GS
define node 60.1 name E
define node 19.1 name F
define node 47.1 name A
define node 47.2 name C
```

As an additional feature and security caution, DECnet Poor Man's Routing may be used between nodes outside of the translation map as long as those nodes have access to nodes that are in the map, so that a user on node B could issue the following VMS command:

```
$ dir A::D::E::
```

When a Poor Man's Routing connection is made between two networks, only the two adjacent nodes between the networks will have any direct knowledge about the other network. Application-level network access may then be specified to route through the connection.

Note: Cisco does not support Poor Man's Routing directly; the intermediate nodes must be VMS systems with Poor Man's Routing enabled in FAL.

Limitations of the ATG

Keep the following limitations in mind when configuring the Address Translation Gateway:

- Both nodes that wish to communicate across the ATG must exist in the translation map. Other nodes outside of the map will see route advertisements for the mapped address, but will not be able to communicate with them. An unmapped node trying to communicate with a mapped node will always get the message "`Node unreachable.`" This can be confusing if another nearby node can communicate with mapped nodes because it is also a mapped node.
- Managing a large map can be tedious. Configuration errors will likely cause unpredictable network behavior.
- Third-party DECnet applications could fail if they pass node number information in a data stream (most likely a sign of a poorly designed application).
- Routing information for mapped addresses is static, and does not reflect the reachability of the actual node in the destination network.

DECnet Monitoring Commands

Use the EXEC commands described in this section to obtain displays of activity on the DECnet network.

Displaying DECnet Status

Use the **show decnet interface** command to display the DECnet status and configuration for all interfaces. Enter this command at the EXEC prompt:

show decnet interface [*interface unit*]

When the optional arguments *interface* and *unit* are specified, the relevant information for that particular interface are displayed.

In the following sample output, no specific interface was named, so you see information on all interfaces.

```
Global DECnet parameters for network 0:
  Local address is 19.15, node type is area
  Maximum node is 350, maximum area is 63, maximum visits is 63
  Maximum paths is 1, mode is normal
  Local maximum cost is 1022, maximum hops is 30
  Area maximum cost is 1022, maximum hops is 30
Ethernet 0 is up, line protocol is up
  Interface cost is 2, priority is 126, DECnet network: 0
  We are the designated router
  Sending HELLOs every 15 seconds, routing updates 40 seconds
  Smallest router blocksize seen is 576 bytes
  Routing input list is not set, output list is not set
  Access list is not set
  DECnet fast switching is enabled
Serial 0 is up, line protocol is up
  Interface cost is 5, priority is 126, DECnet network: 0
  Sending HELLOs every 15 seconds, routing updates 40 seconds
  Smallest router blocksize seen is 1498 bytes
  Routing input list is not set, output list is not set
  Access list is not set
  DECnet fast switching is enabled
Ethernet 1 is up, line protocol is up
  DECnet protocol processing disabled
```

Displaying the DECnet Address Mapping Information

Use the **show decnet map** command to display the address mapping information used by the DECnet Address Translation Gateway. Enter this command at the EXEC prompt:

show decnet map

Displaying the DECnet Routing Table

Use the **show decnet route** command to display the DECnet routing table. Enter this command at the EXEC prompt:

show decnet route [*decnet-address*]

The optional argument *decnet-address* is a DECnet address and, when specified, the first hop route to that address is displayed. This command may show several routes for a destination when equal cost paths have been set with the **decnet max-paths** command, and when there is more than one equal cost path to a destination. The currently selected route is indicated by an asterisk in the first column of the output. In interim mode, the selected route will never appear to change.

In the following sample output, a DECnet address name was not specified, so the entire routing table is displayed:

```
  Node       Cost  Hops   Next Hop to Node       Expires  Prio
*(Area)        0     0     (Local) ->19.15
*19.16         2     1   Ethernet0 ->19.16           44     64    V
*19.17         1     1   Ethernet2 ->19.17           31    125    VA
 19.17         2     1   Ethernet0 ->19.17           31    125    VA
*19.22         2     1   Ethernet0 ->19.22           41
```

In the displays:

- The `Expires` field displays how many seconds from now this entry expires.
- The `Prio` field is the router priority of this node.
- The `V` indicates that this is an adjacent Level 1 router; `VA` or `A` indicates that this is an adjacent Level 2 (area) router.
- An area node exists on the same local (0 hops) cable.

Displaying DECnet Traffic Statistics

The **show decnet traffic** command shows the DECnet traffic statistics, including datagrams sent, received, and forwarded. Enter this command at the EXEC prompt:

show decnet traffic

Following is sample output:

```
Total: 92275748 received, 758 format errors, 0 unimplemented
       0 not a gateway, 0 no memory, 689 no routing vector
Hellos: 13113448 received, 26 bad, 15042 other area, 1842481 sent
Level 1 routing: 3919281 received, 0 bad, 580109 other area, 1485567 sent
Level 2 routing: 794130 received, 0 not primary router, 1140858 sent
Data: 73868022 received, 0 not long format, 68 too many visits
      73852256 forwarded, 0 mapped, 10880 returned, 0 converted
      0 access control failed, 10880 no route, 0 encapsulation failed
      0 inactive network, 0 incomplete map
```

In the displays:

- `Total:` displays the totals of packet types received.
  - — The `received` field is the total of all types of DECnet packets received.
  - — The `format errors` field lists the number of packet that appeared to be DECnet, but were formatted incorrectly. The number in the `received` field includes these packets.

— The `unimplemented` field reports the number of incoming packets that are DECnet control packets, and how many specify a service that the router does not implement, including services implemented to forward Level 1 and Level 2 routing information, and router and end-system HELLO packets.

— The field labeled `not a gateway` reports the total number of packets received while not routing DECnet.

— The field labeled `no memory` is a catch-all that records transaction attempts when the system has run out of memory.

— The field labeled `no routing vector` indicates that either a routing update came in from another router when the router did not have an adjacency for it, or it had no routing vector for the type of routing update. Execute the **debug decnet-routing** command (in the section "Debugging DECnet") to display additional information.

- `Hellos`: displays the number of HELLO messages received and sent.
  — The `received` field displays the total number of HELLO messages received. All protocol types are included.
  — The `bad` field displays the total number of "bad" HELLO messages received. Invoke the EXEC command **debug decnet** to display more information about why the HELLO message was judged as bad.
  — The `other area` field displays the total number of HELLO messages received from nodes on other areas when the router is a Level 1 router only.
  — The `sent` field displays the total number of HELLO messages sent.

- `Level 1 routing`: displays the Level 1 routing updates received and sent.
  — The `received` field displays the total number of Level 1 routing updates received.
  — The `bad` field displays the total number of Level 1 updates received that were judged to be bad.
  — The `other area` field displays the total number of Level 1 updates from nodes in other areas.
  — The `sent` field displays the total number of Level 1 updates sent.

- `Level 2 routing`: displays the Level 2 routing updates received and sent.
  — The `received` field displays the total number of Level 2 updates received.
  — The `field` labeled `not primary router` should always be zero.
  — The `sent` field displays the total number of Level 2 updates sent.

- `Data`: displays the number of data packets received and sent.
  — The `received` field displays the total number of noncontrol (data) packets received.
  — The field labeled `not long format` displays the number of packets received which are not in the long DECnet format. This number should always be zero. If it is not, investigate the source of the improperly formatted packets.

— The field labeled `too many visits` lists the number of packets received which have visited too many routers and have been flushed.

— The `forwarded` field lists the total number of packets forwarded.

— The `mapped` field displays the total number of ATG packets mapped.

— The `returned` field lists the total number of packets returned to the sender at the senders' request.

— The `converted` field displays the number of Phase IV packets converted to Phase V packets.

— The field labeled `access control failed` lists the packets dropped because access control required it.

— The `no route` field lists the total packets dropped because the router did not know where to forward them.

— The field labeled `encapsulation failed` lists the number of packets that could not be encapsulated. This usually happens when there are entries missing in a map for a public data network, such as X.25 or Frame Relay. This can also occur if an interface is set for an encapsulation for which there is no defined DECnet encapsulation (such as PPP on Serial Interfaces).

— The field labeled `inactive network` displays the number of packets that appear to come from an unknown interface, or that ATG returned because they did not make sense.

— The field labeled `incomplete map` counts the number of packets that failed address translation. This usually means a node that is not in the ATG map is trying to access a node in another network advertised by the ATG.

Debugging DECnet

Use the EXEC commands described in this section to troubleshoot and monitor the DECnet network transactions. For each **debug** command, there is a corresponding **undebug** command that turns the message logging off. Generally, you enter these commands with Cisco customer engineers during troubleshooting sessions.

debug decnet-packets

The **debug decnet-packets** command enables logging of all DECnet routing updates and HELLO packets.

debug decnet-routing

The **debug decnet-routing** command enables logging of all changes made to the DECnet routing table, that is, new routes, routes that change cost, routes that expire, and so on.

DECnet Global Configuration Command Summary

This section provides an alphabetically arranged summary of all the DECnet global interface commands. These commands may appear any place in the configuration file.

access-list *list* {**permit** | **deny**} *address mask*
no access-list

Creates an access lists. The argument *list* is an integer between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments. The **no** form of the command deletes the access list.

decnet area-max-cost *value*

Sets the maximum cost specification value for *inter*-area routing. The argument *value* determines the maximum cost for a route to a distant area that the router may consider usable; the router treats as unreachable any route with a cost greater than the value you specify. A valid range for cost is from 1 to 1,022; the default is 1,022. This parameter is only valid for area routes.

decnet area-max-hops *value*

Sets the maximum hop count specification value for *inter*-area routing. The argument *value* determines the maximum number of hops for a route to a distant area that the router may consider usable; the router treats as unreachable any route with a count greater than the value you specify. A valid range for the hop count is from 1 to 30; the default is 30. This parameter is only valid for area routes

[**no**] **decnet conversion igrp** *area-tag*

Enables the DECnet Phase IV/Phase V conversion for the router. The argument *area-tag* is an ISO CLNS area name. The **no** form of the command disables conversion.

decnet *network-number keywords*

Specifies ATG. The argument *network-number* specifies the network number in the range 0 through 3, and the argument *keywords* is one of the configuration keywords. Commands without the *network-number* modifier apply to "network 0."

decnet *first-network* **map** *virtual-address second-network real-address*

Establishes a translation entry to translate a virtual DECnet address to a real DECnet address. The arguments *first-network* and *second-network* are DECnet network numbers in the range zero through three. The arguments *virtual-address* and *real-address* are specified as numeric DECnet addresses.

decnet max-address *value*

Determines the largest node number specification allowed in the current area. The argument *value* is a node number from 1 to 1,023; the default is 255. This parameter controls the sizes of internal routing tables and of messages sent to other nodes.

decnet max-area *value*

Sets the largest area number specification that the router can handle. The max-area keyword takes as its value an area number from 1 to 63; the default is 63.

decnet max-cost *value*

Sets the maximum cost specification for *intra*-area routing. The router ignores routes within the local area that have a cost greater than the corresponding value of this parameter. The argument *value* is a cost from 1 to 1,022; the default is 1,022.

decnet max-hops *value*

Sets the maximum hop count specification value for *intra*-area routing. The router ignores routes within the local area that have a hop count greater than the corresponding value of this parameter. The argument *value* is a hop count from 1 to 30; the default is 30.

decnet max-paths *value*

Defines the maximum number of equal cost paths to a destination that may be kept by the router. The argument *value* specifies the maximum number of equal cost paths, which is limited to 31. The default value is one, which specifies no multiple paths.

decnet max-visits *value*

Sets the limit on the number of times a packet can pass through a router. The argument *value* is a number from 1 to 63; the default value is 63.

decnet node-type {**area** | **routing-iv**}

Specifies the node type for the router. This command takes another keyword, **area** or **routing-iv,** as its value. If you specify **area,** the router exchanges traffic directly with routers in other areas, and participates in the inter-area (Level 2) routing protocol, as well as acting as a intra-area (Level 1) router for its local area. If you specify **routing-iv** (the default), the router acts as an intra-area router, and routes packets out of the area by taking the least cost path to an inter-area router.

decnet path-split-mode {**normal** | **interim**}

Sets the mode for splitting the routes between equal cost paths. The keyword **normal** selects the normal mode, where equal cost paths are selected on a round-robin basis. The normal mode is the default. The keyword **interim** selects an interim mode, where traffic for any particular higher level session is always routed over the same path. This mode supports older implementations of DECnet (VMS versions 4.5 and earlier) that do not support out-of-order packet caching.

[**no**] **decnet routing** *decnet-address*

Enables or disables DECnet routing. The argument *decnet-address* takes as its value an address in DECnet format X.Y, where X is the area number and Y is the node number. There is no default router address; you must specify this parameter for DECnet operation.

DECnet Interface Subcommand Summary

This section provides an alphabetically arranged summary of the DECnet interface subcommands. These commands follow an **interface** command.

[**no**] **decnet access-group** *list*

Applies or removes an access list. The argument *list* can be either a standard or extended DECnet access list. A standard DECnet access list applies to destination addresses in this case.

[**no**] **decnet cost** *cost-value*

Sets or removes a cost value for an interface. The argument *cost-value* is an integer from 1 to 63. There is no default cost for an interface, although a suggested cost for Ethernets is 4, and all hosts on the same cable must share the same value. Use the **no decnet cost** subcommand to disable DECnet routing for an interface.

[no] decnet hello-timer *value*

Specifies how often the router sends HELLO messages. This keyword takes as its value a time from 1 to 8,191 seconds; the default is 15 seconds. The **no** form of the command restores the default.

[no] decnet in-routing filter *list*

Provides access control to HELLO messages or routing information received on this interface. Addresses that fail this test are treated as unreachable. The argument *list* is a standard DECnet access list. The **no** form of the command removes access control.

[no] decnet out-routing-filter *list*

Provides access control to routing information being sent out on this interface. Addresses that fail this test are shown in the update message as unreachable. The argument *list* is a standard DECnet access list. The **no** form of the command removes access control.

[no] decnet route-cache

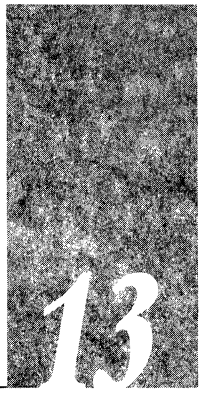
Fast switching and the route cache are normally enabled. If you want to disable fast switching, use the **no** form of the command.

[no] decnet router-priority *value*

Sets a priority value for use in determining the default router. Argument *value* is a number from 0 to 127; the default is 64. The **no** form of the command restores the default.

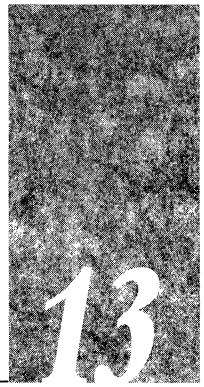
[no] decnet routing-timer *value*

Specifies how often the router sends routing messages. Argument *value*s a time from 1 to 65,535 seconds; the default is 40 seconds. The **no** form of the command restores the default.



Chapter 13
Routing IP

Chapter 13
Routing IP



This chapter begins with an introduction to Cisco's implementation of the IP protocol for its line of routing products, and continues with an in-depth view of configuration options, IP addressing and its various protocols, and examples of well-designed networks. These tasks and topics are covered in this chapter:

- Configuring IP
- Assigning IP addresses, address resolution, and broadcast addresses
- Configuring access and security
- Configuring accounting

See Chapter 14, "The IP Routing Protocols" for information on the various routing protocols, how they have evolved, and how they are best used in complex internetworks.

Cisco's Implementation of IP

Cisco's implementation of TCP/IP provides all major services contained in the various protocol specifications. Cisco routers also provide the TCP and UDP *little services* called Echo service and Discard service. These services are described in RFC 862 and RFC 863.

Cisco supports both TCP and UDP at the Transport Layer, for the maximum flexibility in services. Some Cisco global and interface commands require UDP packets to be sent (see the section "Configuring ICMP and Other IP Services"). Cisco supports all standards for IP broadcasts.

Configuring IP

The process of configuring your router for IP routing differs from the procedures for configuring other protocols in that you don't have to initially enable IP routing. All Cisco routers are shipped with IP already enabled. The **ip routing** global configuration command is described later in this chapter to allow you to re-enable IP routing if you have disabled it. You should follow the steps one the next page to configure individual interfaces and other options.

Step 1: Enter an address for the interface on which you will be routing IP using the **ip address** interface subcommand.

Step 2: Consider addressing options and broadcast packet handling, using commands described in the "Setting IP Interface Addresses" and "Broadcasting In the Internet" sections.

Step 3: Optionally, configure packet sizes and other performance parameters as well as ICMP and other IP services. Information for these tasks are in the section "Configuring ICMP and Other IP Services."

Step 4: Configure access lists and other security options, if desired.

Step 5: Configure routing. The IP routing protocols are discussed in Chapter 14.

Each task is described in the following sections, and are followed by descriptions of the EXEC commands to maintain, monitor and debug an IP network. Summaries of the global configuration commands and interface subcommands described in this section appear at the end of this chapter.

Enabling IP Routing

The **ip routing** global configuration command enables IP routing for the router. Its full syntax follows.

ip routing

no ip routing

If the system is running bridging software, the **no ip routing** subcommand turns off IP routing when setting up a system to bridge (as opposed to route) IP datagrams. (See the explanations on bridging options in the chapters in Part Five.) The default setting is to perform IP routing.

Assigning IP Addresses

The official description of Internet addresses is found in RFC 1020, "Internet Numbers." The Network Information Center (NIC), which maintains and distributes the RFC documents, also assigns Internet addresses and network numbers. Upon application from an organization, NIC assigns a network number or range of addresses appropriate to the number of hosts on the network.

Address Classes and Formats

As described in RFC 1020, Internet addresses are 32-bit quantities, divided into five classes. The classes differ in the number of bits allocated to the *network* and *host* portions of the address. For this discussion, consider a network to be a collection of devices (hosts) that have the same network field value in their Internet addresses.

Note: When discussing IP, all network-attached devices are referred to as *hosts*.

The Class A Internet address format allocates the highest eight bits to the network field and sets the highest-order bit to 0 (zero). The remaining 24 bits form the host field. Only 128 Class A networks can exist, but each Class A network can have almost 17 million hosts. Figure 13-1 illustrates the Class A address format.

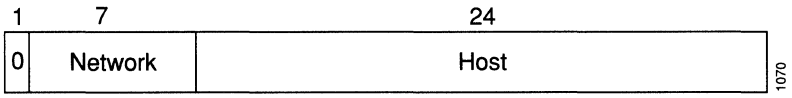


Figure 13-1 Class A Internet Address Format

The Class B Internet address format allocates the highest 16 bits to the network field and sets the two highest-order bits to 1,0. The remaining 16 bits form the host field. Over 16,000 Class B networks can exist, and each Class B network can have over 65,000 hosts. Figure 13-2 illustrates the Class B address format.

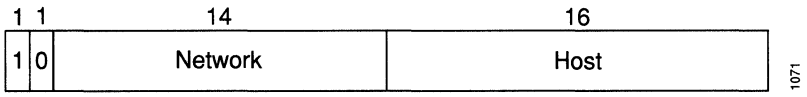


Figure 13-2 Class B Internet Address Format

The Class C Internet address format allocates the highest 24 bits to the network field and sets the three highest-order bits to 1,1,0. The remaining eight bits form the host field. Over two million Class C networks can exist, and each Class C network can have up to 254 hosts. Figure 13-3 illustrates the Class C address format.

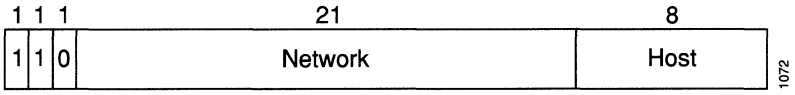


Figure 13-3 Class C Internet Address Format

The Class D Internet address format is reserved for multicast groups, as discussed in RFC 988. In Class D addresses, the four highest-order bits are set to 1,1,1,0.

The Class E Internet address format is reserved for future use. In Class E addresses, the four highest-order bits are set to 1,1,1,1. The router currently ignores Class D and Class E Internet addresses, except the global broadcast address *255.255.255.255*.

Internet Address Notation

The notation for Internet addresses consists of four numbers separated by dots (periods). Each number, written in decimal, represents an 8-bit octet. When strung together, the four octets form the 32-bit Internet address. This notation is called dotted decimal.

These samples show 32-bit values expressed as Internet addresses:

```
192.31.7.19
10.7.0.11
255.255.255.255
0.0.0.0
```

Note that 255, which represents an octet of all ones, is the largest possible value of a field in a dotted-decimal number.

Allowable Internet Addresses

Some Internet addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. Table 13-1 lists ranges of Internet addresses and shows which addresses are reserved and which are available for use.

Table 13-1 Reserved and Available Internet Addresses

Class	Address or Range	Status
A	0.0.0.0	Reserved
	1.0.0.0 through 126.0.0.0	Available
	127.0.0.0	Reserved
B	128.0.0.0	Reserved
	128.1.0.0 through 191.254.0.0	Available
	191.255.0.0	Reserved
C	192.0.0.0	Reserved
	192.0.1.0 through 223.255.254	Available
	223.255.255.0	Reserved
D, E	224.0.0.0 through 255.255.255.254	Reserved
	255.255.255.255	Broadcast

Internet Address Conventions

To create an address that refers to a specific network, the bits in the host portion of the address must all be zero. For example, the Class C address *192.31.7.0* refers to a particular network (no local component).

Conversely, if you want a local address only, without a network portion, all the bits in the network portion of an address must be 0. For example, the Class C address *0.0.0.234* refers to a particular host (local address).

If you want to send a packet to all hosts on the network specified in the network portion of the address, the local address must be all ones. For example, the Class B address *128.1.255.255* refers to all hosts on network *128.1.0.0*. This is called a *broadcast*, which is described in the section "Broadcasting in the Internet." You can also find general information on broadcasts in Chapter 14, "The IP Routing Protocols."

Note: Because of these conventions, do not use an Internet address with all zeros or all ones in the host portion for your router address.

You can manually configure the router's routing table, or you can specify that a high-level routing protocol dynamically build the routing table. In both cases, the routing table is based on the network portion of addresses. Consequently, the addresses of hosts on a single physical network must have the same network number to permit automatic routing. If a network does not meet this requirement, the routers will be unable to communicate with all of the hosts on that network. (The one exception to this general rule is the use of secondary addresses, which is described in the section "Setting IP Interface Addresses.")

Subnetting and Routing

Subnetting is a scheme for imposing a simple two-level hierarchy on host addresses, allowing multiple, logical networks to exist within a single Class A, B, or C network. The usual practice is to use a few of the leftmost bits in the host portion of the network addresses for a subnet field. For example, Figure 13-4 shows a Class B address with five bits of the host portion used as the subnet field. The official description of subnetting is contained in RFC 950, "Internet Standard Subnetting Procedure."

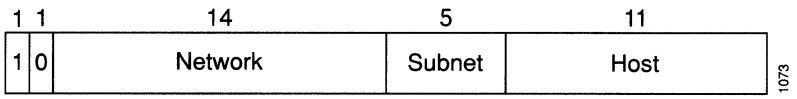


Figure 13-4 A Class B Address with a 5-Bit Subnet Field

As with the host portion of an address, do not use all zeros or all ones in the subnet field.

Routers and hosts can use the subnet field for routing. The rules for routing on subnets are identical to the rules for routing on networks. However, correct routing requires that all subnets of a network be physically contiguous. In other words, the network must be set up such that it does not require traffic between any two subnets to cross another network. The current Cisco implementation requires that all subnets of a network have the same number of subnet bits.

Creating a Single Network from Separated Subnets

You can create a single network from subnets that are physically separated by another network by using a *secondary address*. An example is shown in the section "Setting IP Interface Addresses."

Note: A subnet cannot appear on more than one active interface of the router at a time.

Subnet Masks

A *subnet mask* identifies the subnet field of network addresses. All subnets of a given class, A, B, or C, should use the same subnet mask. This mask is a 32-bit Internet address written in dotted-decimal notation with all ones in the network and subnet portions of the address. For the example shown in Figure 13-4, the subnet mask is *255.255.248.0*. Table 13-2 shows the subnet masks you can use to divide an octet into subnet and host fields. The subnet field can consist of any number of the host field bits; you do not need to use multiples of eight. However, you should use three or more bits for the subnet field—a subnet field of two bits yields only four subnets, two of which are reserved (the 1,1 and 0,0 values).

Table 13-2 Subnet Masks

Subnet Bits	Host Bits	Hex Mask	Decimal Mask
0	8	0	0
1	7	0x80	128
2	6	0xC0	192
3	5	0xE0	224
4	4	0xF0	240
5	3	0xF8	248
6	2	0xFC	252
7	1	0xFE	254
8	0	0xFF	255

Setting IP Interface Addresses

Use the **ip address** interface subcommand to set an IP address for an interface. The full command syntax follows:

ip address *address net-mask* [**secondary**]

no ip address *address net-mask* [**secondary**]

The two required arguments are an IP address and the network mask for the associated IP network. The subnet mask must be the same for all interfaces connected to subnets of the same network. Hosts can determine subnet masks using the Internet Control Message Protocol (ICMP) *Mask Request* message. Routers respond to this request with an ICMP *Mask Reply* message. (See the section "Configuring ICMP and Other IP Services" for more details.)

You can disable IP processing on a particular interface by removing its IP address with the **no ip address** subcommand. If the router detects another host using one of its IP addresses, it will print an error message on the console. The software supports multiple IP addresses per interface.

You may use this command to specify additional secondary IP addresses by including the keyword **secondary** after the IP address and subnet mask.

Example:

In the sample below, *131.108.1.27* is the primary address and *192.31.7.17* is a secondary address for Ethernet 0.

```
interface ethernet 0
ip address 131.108.1.27 255.255.255.0
ip address 192.31.7.17 255.255.255.0 secondary
```

Using Subnet Zero

Subnetting with a subnet address of zero is generally not allowed, because of the confusion inherent in having a network and a subnet with indistinguishable addresses. For example, if network *131.108.0.0* is subnetted as *255.255.255.0*, subnet zero would be written as *131.108.0.0*—which is identical to the network address.

To enable or disable the use of subnet zero for interface addresses and routing updates, use the global configuration command **ip subnet-zero**. Its full command syntax follows:

ip subnet-zero

no ip subnet-zero

Example:

In the example below, we enable subnet-zero for the router:

```
ip subnet-zero
```

Local and Network Addresses: Address Resolution

A device in the Internet may have both a local address, which uniquely identifies the device on its local segment or LAN, and a network address which identifies the network the device belongs to. The local address is more properly known as a data link address because it is contained in the data link layer (Layer 2 of the OSI Model) part of the packet header and is read by data link devices (bridges and all device transceivers, for example). The more technically inclined will refer to local addresses as MAC addresses because the Media Access Control (MAC) sublayer within the data link layer processes addresses for the layer.

To communicate with a device on Ethernet, the router must first determine the 48-bit MAC or local data link address of that device. The process of determining the local data link address from an Internet address is called *address resolution*. The process of determining the Internet address from a local data link address is called *reverse address resolution*. The router uses three forms of address resolution: Address Resolution Protocol (ARP), proxy ARP, and Probe (which is similar to ARP). The router also uses the Reverse Address Resolution Protocol (RARP). The ARP, proxy ARP, and RARP protocols, which are used on Ethernets, are defined in RFCs 826, 1027, and 903, respectively. Probe is a protocol developed by the Hewlett-Packard Company for use on IEEE-802.3 networks.

Address Resolution Using ARP

To send an Internet data packet to a local host with which it has not previously communicated, the router first broadcasts an ARP Request packet. The ARP Request packet requests the MAC local data link address corresponding to an Internet address. All hosts on the network receive this request, but only the host with the specified Internet address will respond.

If present and functioning, the host with the specified Internet address responds with an ARP Reply packet containing its local data link address. The router receives the ARP Reply packet, stores the local data link address in the ARP cache for future use, and begins exchanging packets with the host.

The EXEC command **show arp** may be used to examine the contents of the ARP cache. The **show ip arp** command will show IP entries.

Tailoring ARP: Static Entries and Timing

The function of ARP is to provide a dynamic mapping between 32-bit IP addresses and 48-bit local hardware (Ethernet, FDDI, Token Ring) addresses. ARP may also be used for protocols other than IP and media that have other than 48-bit addresses.

Because most hosts support *dynamic resolution*, you generally do not need to specify static ARP cache entries. If you do need to define static arp cache entries, you can do so globally.

When used as a global configuration command, the **arp** command installs a permanent entry in the ARP cache. The router uses this entry to translate 32-bit Internet Protocol addresses into 48-bit hardware addresses. The full syntax follows:

arp *internet-address hardware-address type* [**alias**]

no arp *internet-address*

The argument *internet-address* is the Internet address in dotted decimal format corresponding to the local data link address specified by the argument *hardware-address.*

The argument *type* is an encapsulation description. This is typically the **arpa** keyword for Ethernets and is always **snap** for FDDI and Token Ring interfaces, and **ultra** for the Ultranet interfaces. See the discussions of the individual interface types for more information on possible encapsulations.

The optional keyword **alias** indicates that the router should respond to ARP requests as if it were the owner of the specified IP address.

Example:

The following is a sample of a static ARP entry for a typical Ethernet host.

```
arp 192.31.7.19 0800.0900.1834 arpa
```

The **no arp** subcommand removes the specified entry from the ARP cache. To remove all non-static entries from the ARP cache, use the privileged EXEC command **clear arp-cache.**

When used as an interface subcommand, the **arp** command controls the interface-specific handling of IP address resolution into 48-bit Ethernet hardware addresses. The full syntax of the **arp** interface subcommand follows:

arp {**arpa** | **probe** | **snap**}

no arp {**arpa** | **probe** | **snap**}

The keyword **arpa**, which is the default, specifies standard Ethernet style ARP (RFC 826), **probe** specifies the HP-proprietary Probe protocol for IEEE-802.3 networks, and **snap** specifies ARP packets conforming to RFC 1042. The **show interfaces** monitoring command displays the type of ARP being used on a particular interface. Probe is described more in a later section in this chapter.

Note: Unlike most commands that take multiple arguments, arguments to the **arp** command are not mutually exclusive. Each command enables or disables a specific type of ARP. For example, if you enter the **arp arpa** command followed by the **ip probe** command, the router would send two packets each time it needed to discover a MAC address.

To set the number of seconds an ARP cache entry will stay in the cache, use the **arp timeout** interface subcommand. The full syntax of this command follows:

arp timeout *seconds*

no arp timeout

The value of the argument *seconds* is used to age an ARP cache entry related to that interface. By default, the *seconds* argument is set to four hours (14,400 seconds). A value of zero seconds sets no timeout.

Use the **no arp timeout** command to return to the default value.

This command is ignored when issued on interfaces that do not use ARP. Use the EXEC command **show interfaces** to display the ARP timeout value. The value follows the `Entry Timeout:` heading, as seen in this sample display:

```
ARP type: ARPA, HP-PROBE, Entry Timeout: 14400 sec
```

Example:

The following example illustrates how to set the ARP timeout to 12000, to allow entries to time out more quickly than the default.

```
arp timeout 12000
```

Address Resolution Using Proxy ARP

The router uses proxy ARP, as defined in RFC 1027, to help hosts with no knowledge of routing determine the hardware addresses of hosts on other networks or subnets. Under proxy ARP, if the router receives an ARP Request for a host that is not on the same network as the ARP Request sender, and if the router has the best route to that host, then the router sends an ARP Reply packet giving its own local data link address. The host that sent the ARP Request then sends its packets to the router, which forwards them to the intended host.

The **no ip proxy-arp** interface subcommand disables proxy ARP on the interface. The full command syntax for this command follows.

ip proxy-arp

no ip proxy-arp

The default is to perform proxy ARP; the **no ip proxy-arp** command disables this default.

Address Resolution Using Probe

By default, the router uses the HP-proprietary Probe protocol (in addition to ARP) whenever it attempts to resolve an IEEE-802.3 or Ethernet local data link address. The subset of Probe that performs address resolution is called *Virtual Address Request and Reply.* Using Probe, the router can communicate transparently with Hewlett-Packard IEEE-802.3 hosts that use this type of data encapsulation.

The **arp probe** commands enable or disable Probe protocol for IEEE-802.3 networks, as follows.

arp probe

no arp probe

The other options of the **arp** command are discussed under ARP, earlier in this chapter.

Reverse Address Resolution Using RARP and BootP

Reverse ARP (RARP) was defined in RFC 903. If a router does not know the IP address of one of its Ethernet interfaces, it will try RARP during start up processing to attempt to determine the Internet address, based on its interface local data link address. Diskless hosts also use RARP at boot time to determine their protocol addresses. RARP works in the same way as ARP, except that the RARP Request packet requests an Internet address instead of a local data link address. Use of RARP requires a RARP server on the same network segment as the router interface.

A router without non-volatile memory uses both Reverse ARP (RARP) and Boot Protocol (BootP) messages when trying to obtain its interface address from network servers.

BootP, defined in RFC 951, specifies a method for determining the Internet address of a host from its Ethernet local data link address. The basic mechanism is similar to that used by Reverse ARP, but it is UDP-based rather than a distinct Ethernet protocol. The main advantage of BootP is that its messages can be routed through routers, whereas RARP messages cannot leave the local Ethernet-based network.

Broadcasting in the Internet

A broadcast is a data packet destined for all hosts on a particular physical network. Network hosts recognize broadcasts by special addresses. This section describes the meaning and use of Internet broadcast addresses. For detailed discussions of broadcast issues in general, see RFC 919, "Broadcasting Internet Datagrams," and RFC 922, "Broadcasting Internet Datagrams in the Presence of Subnets." The router support for Internet broadcasts generally complies with RFC 919 and RFC 922; however, the router does not support multisubnet broadcasts as defined in RFC 922.

The current standard for an Internet broadcast address requires that the host portion of the address consist of all ones. If the network portion of the broadcast address is also all ones, the broadcast applies to the local network only. If the network portion of the broadcast address is not all ones, the broadcast applies to the network or subnet specified.

Cisco routers support two kinds of broadcasting: *directed broadcasting* and *flooding*. A directed broadcast is a packet sent to a specific network or series of networks, while a flooded broadcast packet is sent to every network, as shown in Figure 13-5. The packet that is incoming from interface E0 is flooded to interfaces E1, E2 and Serial 0. A directed-broadcast address includes the network or subnet fields.

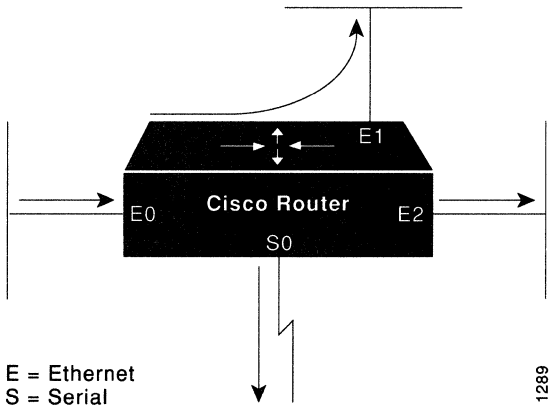


Figure 13-5 IP Flooded Broadcast

For example, if the network address is *128.1.0.0*, then the address *128.1.255.255* indicates all hosts on network *128.1.0.0*. This would be a directed broadcast. If network *128.1.0.0* has a subnet mask of *255.255.255.0* (the third octet is the subnet field), then the address *128.1.5.255* specifies all hosts on subnet 5 of network *128.1.0.0*, another directed broadcast.

The **no ip directed-broadcast** interface subcommand disables forwarding of directed broadcasts on the interface. The full syntax of this command follows.

ip directed-broadcast

no ip directed-broadcast

The default is to forward directed broadcasts. You re-enable forwarding of directed broadcasts with the **ip directed-broadcast** subcommand.

Internet Broadcast Addresses

The router supports Internet broadcasts on both local and wide area networks. There are at least four popular standard ways of indicating an Internet broadcast address. You can configure a router host to generate any form of Internet broadcast address. The router can also receive and understand any form of Internet broadcast address. By default, a router uses all ones for both the network and host portions of the Internet broadcast address (255.255.255.255). You can change the Internet broadcast address by using the **ip broadcast-address** interface subcommand. Following is the full command syntax:

ip broadcast-address [*address*]

no ip broadcast-address [*address*]

The argument *address* is the desired IP broadcast address for a network. If a broadcast address is not specified, the system defaults to a broadcast address of all ones or 255.255.255.255.

Use the **no ip broadcast-address** command to remove the broadcast address or addresses.

If the router does not have nonvolatile memory, and you want the specify the broadcast address to use before it has its configuration, you can change the Internet broadcast address by setting jumpers in the processor configuration register. Setting bit 10 causes the router to use all zeros. Bit 10 interacts with bit 14, which controls the network and subnet portions of the broadcast address. Setting bit 14 causes the router to include the network and subnet portions of its address in the broadcast address. Table 13-3 shows the combined effect of setting bits 10 and 14.

Table 13-3 Configuration Register Settings for Broadcast Address Destination

Bit 14	Bit 10	Address (<net><host>)
out	out	<ones><ones>
out	in	<zeros><zeros>
in	in	<net><zeros>
in	out	<net><ones>

For more information about the configuration register, see the Cisco Systems hardware reference guide for your system.

Forwarding of Broadcast Packets and Protocols

There are circumstances in which you want to control which broadcast packets and which protocols are forwarded. You do this with helper addresses and the **forward-protocol** commands.

The **ip helper-address** interface subcommand tells the router to forward UDP broadcasts, including BootP, received on this interface. (UDP is the connectionless alternative to TCP at the Transport Layer.) Use the **ip helper-address** interface subcommand to specify the destination address for forwarding broadcast packets. Full command syntax follows.

ip helper-address *address*

no ip helper-address *address*

The *address* argument specifies a destination broadcast or host address to be used when forwarding such datagrams. You can have more than one helper address per interface. You remove the list with **no ip helper-address**.

If you do not specify a **helper address** command, the router will not forward UDP broadcasts.

Example:

This example defines an address that act as a helper address.

```
ip helper-address 121.24.43.2
```

The **ip forward-protocol** interface subcommand allows you to specify which protocols and ports the router will forward. Its full syntax is listed next.

ip forward-protocol {**udp** | **nd**} [*port*]

no ip forward-protocol {**udp** | **nd**} [*port*]

The keyword **nd** is the ND protocol used by older diskless SUN workstations. The keyword **udp** is the UDP protocol. A UDP destination port can be specified to control which UDP services are forwarded. By default both UDP and ND forwarding are enabled if a helper address has been defined for an interface. If no ports are specified, these datagrams are forwarded, by default:

- Trivial File Transfer (TFTP)
- Domain Name System
- IEN-116 Name Server
- Time service
- NetBios Name Server
- NetBios Datagram Server
- Boot Protocol (BootP) client and server datagrams
- TACACS service

Use the **no ip forward-protocol** command with the appropriate keyword and argument to remove the protocol.

Example:

The example below first defines a helper address, then uses the **ip forward-protocol** command to specify forwarding of UDP only.

```
interface ethernet 1
ip helper-address 131.120.1.0
ip forward-protocol udp
```

Flooding IP Broadcasts

To permit IP broadcasts to be flooded throughout the internetwork in a controlled fashion, use the global configuration command **ip forward-protocol spanning-tree** (full command syntax follows):

ip forward-protocol spanning-tree

no ip forward-protocol spanning-tree

This command is an extension of the **ip helper-address** interface command, in that the same packets that may be subject to the helper address and forwarded to a single network may now be flooded. Only one copy of the packet will be put on each network segment in the network.

The **ip forward-protocol spanning-tree** command uses the database created by the bridging spanning tree protocol. The transparent bridging option must be in the routing software, and bridging must be configured on each interface which is to participate in the flooding. If an interface does not have bridging configured, it will still be able to receive broadcasts, but it will never forward broadcasts received on that interface, and it will never use that interface to send broadcasts received on a different interface. If no bridging is desired, then a type-code bridging filter may be configured which will deny all packet types from being bridged. The spanning-tree database is still available to the IP forwarding code to use for the flooding.

Packets must meet the following criteria to be considered for flooding (these are the same conditions for IP helper addresses):

- Packets must be MAC-level broadcasts.
- Packets must be IP-level broadcasts.
- Packets must be a TFTP, DNS, IEN-116, Time, NetBios, ND, or BootP packet, or a UDP protocol specified by the command **ip forward-protocol udp.**
- The packets' time to live (TTL) value must be at least two.

A flooded UDP datagram is given the destination address specified by the **ip broadcast** command on the output interface. This can be set to any desired address. Thus, the destination address may change as the datagram propagates through the network. The source address is never changed. The TTL value is decremented.

After a decision has been made to send the datagram out on an interface (and the destination address possibly changed), the datagram is handed to the normal IP output routines and is therefore subject to access lists, if they are present on the output interface.

Use the **no ip forward-protocol spanning-tree** command to prevent flooding of IP broadcasts.

Limiting Broadcast Storms

Several early TCP/IP implementations do not use the current broadcast address standard. Instead, they use the old standard, which calls for all zeros instead of all ones to indicate broadcast addresses. Many of these implementations do not recognize an all-ones broadcast address and fail to respond to the broadcast correctly. Others forward all-ones broadcasts, which causes a serious network overload known as a broadcast storm. Implementations that exhibit these problems include UNIX systems based on versions of BSD UNIX prior to Version 4.3.

Routers provide some protection from broadcast storms by limiting their extent to the local cable. Bridges, because they are Layer 2 devices, even intelligent router/bridges, forward broadcasts to all network segments, thus propagating all broadcast storms.

The best solution to the broadcast storm problem is to use a single broadcast address scheme on a network. Most modern TCP/IP implementations allow the network manager to set the address to be used as the broadcast address. Many implementations, including that on the Cisco router, can accept and interpret all possible forms of broadcast addresses.

UDP Broadcasts

Network hosts occasionally employ UDP broadcasts to determine address, configuration, and name information. (UDP stands for User Datagram Protocol, an alternative to TCP for connectionless networks. UDP is defined in RFC 768.) If such a host is on a network segment that does not include a server host, UDP broadcasts fail.

To correct this situation, configure the interface of your Cisco router to forward certain classes of UDP broadcasts to a helper address. See the description of the **ip helper-address** and the **ip forward-protocol** subcommands in this chapter for more information.

Configuring ICMP and Other IP Services

The Internet Control Message Protocol (ICMP) is a special protocol within the IP protocol suite that focuses exclusively on control and management of IP connections. ICMP messages are generated by routers that discover a problem with the IP part of a packet's header; these messages could be alerting another router, or they could be sent to the source or destination device (host). Characteristics of the ICMP messages follow.

- The router listens to ICMP *Destination Unreachable* messages for packets that it originated.
- If the value in the time-to-live (TTL) field of a packet falls to zero, the router sends an ICMP *Time Exceeded* message to the source of the packet and discards the packet.
- If the router receives the ICMP *Information Request* or ICMP *Timestamp Request* message, it responds with an ICMP *Information Reply* or *Timestamp Reply* message.
- During the process of obtaining configuration information from network servers, the router sends broadcast ICMP *Mask Request* messages to determine subnet definitions for the local networks.

The **ip mask-reply** interface subcommand tells the router to respond to mask requests. The full syntax of this command follows.

ip mask-reply

no ip mask-reply

The default is not to send *a Mask Reply*, and this default is restored with the **no ip mask-reply** command.

Each router interface has an output hold queue with a limited number of entries that it can store. Upon reaching this limit, the interface sends an ICMP *Source Quench* message to the source host of any additional packets and discards the packet. When the interface empties the hold queue by one or more packets, the interface can accept new packets again. The router limits the rate at which it sends *Source Quench* and *Unreachable* messages to one per second.

Generating Unreachable Messages

If the router receives a nonbroadcast packet destined for itself that uses a protocol the router does not recognize, it sends an ICMP *Protocol Unreachable* message to the source.

If the router receives a datagram which it is unable to deliver to its ultimate destination because it knows of no route to the destination address, it replies to the originator of that datagram with an ICMP *Host Unreachable* message. Use the **ip unreachables** interface subcommand to enable or disable the sending of these messages. The full syntax for this command follows.

ip unreachables

no ip unreachables

The **no ip unreachables** subcommand disables sending ICMP unreachable messages on an interface.

Generating Redirect Messages

The Cisco router sends an ICMP *Redirect* message to the originator of any datagram that it is forced to resend through the same interface on which it was received, since the originating host could presumably have sent that datagram to the ultimate destination without involving the router at all. The router ignores *Redirect* messages that have been sent to it by other routers. Use the **ip redirects** interface subcommand to enable or disable the sending of these messages, as follows:

ip redirects

no ip redirects

Setting and Adjusting Packet Sizes

All interfaces have a default maximum packet size or MTU. You can set the IP MTU to a smaller unit by using the **ip mtu** interface subcommand. If an IP packet exceeds the MTU set for the router's interface, the router will fragment it. The full command syntax follows.

ip mtu *bytes*

no ip mtu

The default and maximum MTU depends on the interface medium type. The minimum MTU is 128 bytes. The **no ip mtu** subcommand restores the default MTU for that interface.

Example:

In the following example, you are setting the maximum IP packet size for the first serial interface to 300 bytes.

```
interface serial 0
ip mtu 300
```

MTU Path Discovery

All Cisco routers running software release 8.3 or later have the IP MTU Path Discovery mechanism running by default. IP Path MTU Discovery allows a host to dynamically discover and cope with differences in the maximum allowable MTU size of the various links along the path. Sometimes a router is unable to forward a datagram because fragmentation of the datagram is required (the packet is larger than the MTU you set for the interface with the **ip mtu** command), but the "Don't fragment" bit is set. If you have Path Discovery enabled, the router sends a message to the sending host, alerting it to the problem. The host will have to replicate packets destined for the receiving interface so that they fit the smallest packet size of all the links along the path. This technique is defined by RFC 1191 and shown in Figure 13-6.

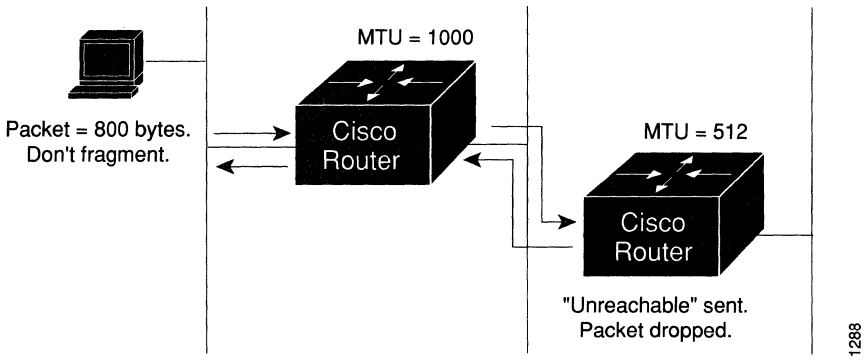


Figure 13-6 MTU Path Discovery

MTU Path Discovery is useful when a link in a network goes down, forcing use of another, different MTU-sized link (and different routers). As an example, suppose one were trying to send IP packets over a network where the MTU in the first router is set to 1500 bytes, but then reaches a router where the MTU is set to 512 bytes. If the datagram's "Don't fragment" bit is set, the datagram would be dropped because the 512-router is unable to forward it. The router returns an ICMP *Destination Unreachable* message to the source of the datagram with its Code field indicating "Fragmentation needed and DF set." To support Path MTU Discovery, it would also include the MTU of the next-hop network link in the low-order bits of an unused header field.

MTU Path Discovery is also useful when a connection is first being established and the sender has no information at all about the intervening links. It is always advisable to use the largest MTU that the links will bear; the larger the MTU, the fewer packets the host needs to send.

You can test that MTU Path Discovery is functioning using the **ping** command; see the following section and the section "The IP Ping Command" later in this chapter for more information.

The Ping Function

When you use the privileged EXEC command **ping** (IP packet internet groper function), the router sends ICMP *Echo* messages to check host reachability and network connectivity. If the router receives an ICMP *Echo* message, it sends an ICMP *Echo Reply* message to the source of the ICMP *Echo* message. See the section "The IP Ping Command" later in this chapter for more information about the use of the **ping** command.

Configuring Internet Header Options

The router supports the Internet header options *Strict Source Route*, *Loose Source Route*, *Record Route*, and *Time Stamp*.

The router examines the header options to every packet that passes through it. If it finds a packet with an invalid option, the router sends an ICMP *Parameter Problem* message to the source of the packet and discards the packet.

You can use the extended command mode of the **ping** command to specify several Internet header options. To see the list of the options you can specify, type a question mark at the extended commands prompt of the **ping** command.

Configuring IP Host-Name-to-Address Conversion

The router maintains a cache of host-name-to-address mappings for use by the EXEC **connect** or **telnet** commands and related Telnet support operations. This cache speeds the process of converting names to addresses.

Defining Static Name-to-Address Mappings

To define a static host-name-to-address mapping in the host cache, use the **ip host** global configuration command, as shown below:

ip host *name address*

The argument *name* is the host name, and the argument *address* is the associated IP address. Additional addresses may be bound to a host name by repeated use of the **ip host** subcommand.

Example:

The following example uses the **ip host** command to define two static mappings.

```
ip host croff 192.31.7.18
ip host bisso-gw 10.2.0.2 192.31.7.33
```

Configuring Dynamic Name Lookup

You can specify that the Domain Name System (DNS) or IEN-116 Name Server automatically determines host-name-to-address mappings. Use these global configuration commands to establish different forms of dynamic name lookup:

- **ip name-server**
- **ip domain-name**
- **ip ipname-lookup**
- **ip domain-lookup**

Name Server

To specify one or more hosts that supply name information, use the **ip name-server** global configuration command, as follows:

ip name-server *server-address1* [*server-address2 . . . server-address6*]

The arguments *server-address* are the Internet addresses of up to six name servers.

Example:

This command specifies host *131.108.1.111* as the primary name server, and host *131.108.1.2* as the secondary server.

```
ip name-server 131.108.1.111   131.108.1.2
```

Domain Name

The global configuration command **ip domain-name** defines a default domain name the router uses to complete unqualified host names (names without a dotted domain name appended to them). The full syntax of this command follows:

ip domain-name *name*

no ip domain-name

The argument *name is* the domain name; do not include the initial period that separates an unqualified name from the domain name. The **no ip domain-name** command disables use of the Domain Name System.

Example:

This command defines *cisco.com* to be used as the default name.

```
ip domain-name cisco.com
```

Any IP host name that does not contain a domain name, that is, any name without a dot (.), will have the dot and cisco.com appended to it before being added to the host table.

Domain Lookup

By default, the IP Domain Name System-based host-name-to-address translation is enabled. To enable or disable this feature, use the **ip domain-lookup** global configuration command as follows:

ip domain-lookup

no ip domain-lookup

IP Name Lookup

To specify the IP IEN-116 Name Server host-name-to-address translation, use the **ip ip name-lookup** global configuration command as follows:

ip ipname-lookup

no ip ipname-lookup

This command is disabled by default; the **no ip ipname-lookup** command restores the default.

HP Probe Proxy Support

HP Probe Proxy support allows a router to respond to HP Probe Proxy Name requests. This will typically be used at sites which have HP equipment and are already using HP Probe. Use the interface subcommand **ip probe proxy**, to enable or disable HP Proxy Probe, as follows:

ip probe proxy

no ip probe proxy

To use the proxy service, you must first enter the host name of the HP host into the host table through the configuration command **ip hp-host**. Full syntax follows:

ip hp-host *hostname ip-address*

no ip hp-host *hostname ip-address*

The *hostname* argument specifies the host's name and the argument *ip-address* specifies it's IP address. Use the **no ip hp-host** command with the appropriate arguments to remove the host name.

Example:

The following example specifies an HP host's name and address, and then enables Probe proxy.

```
ip hp-host BCWjo 131.108.1.27
interface ethernet 0
ip probe proxy
```

Commands that will help you to maintain and debug your HP-based network are listed in the sections "Monitoring the IP Network" and "Debugging the IP Network" at the end of this chapter.

Establishing Domain Lists

To define a list of default domain names to complete unqualified host names, use the **ip domain-list** global configuration command. The full syntax of this command follows.

ip domain-list *name*

no ip domain-list *name*

The **ip domain-list** command is similar to the **ip domain-name** command, except that with **ip domain-list** you can define a list of domains, each to be tried in turn.

The argument *name* is the domain name; do not enter an initial period. Specify only one *name* when you enter the **ip domain-list** command.

Use the **no ip domain-list** command with the appropriate argument to delete a name from the list.

Example 1:

In the example below, several domain names are added to a list:

```
ip domain-list martinez.com
ip domain-list stanford.edu
```

Example 2:

The example below adds a name to, and then deletes a name from the list:

```
ip domain-list sunya.edu
no ip domain-list stanford.edu
```

Note: If there is no domain list, the default domain name is used.

Configuring IP Access Lists

An *access list* is a sequential collection of permit and deny conditions that apply to Internet addresses. The router tests addresses against the conditions in an access list one by one. The first match determines whether the router accepts or rejects the address. Because the router stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the router rejects the address.

The two steps involved in using access lists are:

- Create a list
- Apply it to implement a policy

You apply access lists in several ways:

- Controlling the transmission of packets on an interface
- Controlling virtual terminal line access
- Restricting contents of routing updates

The Cisco software supports two styles of access lists for IP. The standard IP access lists have a single address for matching operations. Extended IP access lists have two addresses with optional protocol type information for matching operations.

Note: Keep in mind when making the access list that, by default, the end of the access list contains an implicit deny statement for *everything* that has not been permitted. Plan your access conditions carefully and be aware of this implicit deny.

Configuring Standard Access Lists

To create an access list, use the **access-list** global configuration command. Full command syntax follows:

access-list *list* {**permit** | **deny**} *address wildcard-mask*

no access-list *list*

The argument *list* is an integer from 1 through 99 that you assign to identify one or more permit/deny conditions as an access list. Access list 0 (zero) is predefined; it permits any address and is the default access list for all interfaces.

The router compares the address being tested to *address,* ignoring any bits specified in *wildcard-mask*. If you use the keyword **permit,** a match causes the address to be accepted. If you use the keyword **deny,** a match causes the address to be rejected.

The arguments *address* and *wildcard-mask* are 32-bit quantities written in dotted-decimal format. Address bits corresponding to wildcard mask bits set to 1 are ignored in comparisons; address bits corresponding to wildcard mask bits set to zero are used in comparisons. See the examples later in this section.

An access list can contain an indefinite number of actual and wildcard addresses. A wildcard address has a non-zero address mask and thus potentially matches more than one actual address. The router examines first the actual address, then the wildcard addresses. The order of the wildcard addresses is important because the router stops examining access-list entries after it finds a match.

The **no access-list** subcommand deletes the entire access list. To display the contents of all access lists, use the EXEC command **show access-lists.**

Example:

The following access list allows access for only those hosts on the three specified networks. It assumes that subnetting is not used; the masks apply to the host portions of the network addresses.

```
access-list 1 permit 192.5.34.0  0.0.0.255
access-list 1 permit 128.88.1.0  0.0.255.255
access-list 1 permit 36.0.0.0  0.255.255.255
```

To specify a large number of individual addresses more easily, you can omit the address mask that is all zeros from the **access-list** configuration command. Thus, the following two configuration commands are identical in effect:

```
access-list 2 permit 36.48.0.3
access-list 2 permit 36.48.0.3  0.0.0.0
```

Configuring Extended Access Lists

Extended access lists allow finer granularity of control. They allow you to specify both source and destination addresses and some protocol and port number specifications.

To define an extended access list, use the extended version of the **access-list** subcommand.

access-list *list* {**permit** | **deny**} *protocol source source-mask destination destination-mask* [*operator operand*] [**established**]

The argument *list* is an integer from 100 through 199 that you assign to identify one or more extended permit/deny conditions as an extended access list. Note that a list number in the range 100 to 199 distinguishes an extended access list from a standard access list. The condition keywords **permit** and **deny** determine whether the router allows or disallows a connection when a packet matches an access condition. The router stops checking the extended access list after a match occurs.

The argument *protocol* is one of the following keywords:

- **ip**
- **tcp**
- **udp**
- **icmp**

Use the keyword **ip** to match any Internet protocol, including TCP, UDP, and ICMP.

The argument *source* is an Internet source address in dotted-decimal format. The argument *source-mask* is a mask, also in dotted-decimal format, of source address bits to be ignored. The router uses the *source* and *source-mask* arguments to match the source address of a packet. For example, to match any address on a Class C network 192.31.7.0, the argument *source-mask* would be 0.0.0.255. The arguments *destination* and *destination-mask* are dotted-decimal values for matching the destination address of a packet.

To differentiate further among packets, you can specify the optional arguments *operator* and *operand* to compare destination ports, service access points, or contact names. Note that the **ip** and **icmp** protocol keywords do not allow port distinctions.

For the **tcp** and **udp** protocol keywords, the argument *operator* can be one of these keywords:

- **lt**—less than
- **gt**—greater than
- **eq**—equal
- **neq**—not equal

The argument *operand* is the decimal destination port for the specified protocol.

For the TCP protocol there is an additional keyword, **established**, that does not take an argument. A match occurs if the TCP datagram has the ACK or RST bits set, indicating an established connection. The non-matching case is that of the initial TCP datagram to form a connection; the software goes on to other rules in the access list to determine if a connection is allowed in the first place.

Ethernet to Internet Example

For an example of using an extended access list, suppose you have an Ethernet-to-Internet routing network, and you want any host on the Ethernet to be able to form TCP connections to any host on the Internet. However, you do not want Internet hosts to be able to form TCP connections into the Ethernet except to the mail (SMTP) port of a dedicated mail host.

To do this, you must ensure that the initial request for an SMTP connection is made on TCP destination port 25 from port *X* where *X* is a number greater than 1023. The two port numbers continue to be used throughout the life of the connection, with the originator always using port 25 as the destination, and the acceptor always using port *X* as the destination. The fact that the secure system behind the router will always be accepting mail connections on port 25, with a foreign port number greater than 1023, is what makes it possible to separately allow/disallow incoming and outgoing services. Also remember that the access list used is that of the interface on which the packet would ordinarily be transmitted.

Example:

In the following example, the Ethernet network is a Class B network with the address 128.88.0.0 and the mail host's address is 128.88.1.2.

```
access-list 101 permit tcp  128.88.0.0 0.0.255.255 0.0.0.0 255.255.255.255
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255 estab-
lished
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 eq 25
interface serial 0
access-group 101
interface ethernet 0
access-group 102
```

This is a complex example, designed to show the power of all the options we've just discussed. The **access group** interface subcommand will be described in detail shortly.

Controlling Line Access

To restrict incoming and outgoing connections between a particular virtual terminal line and the addresses in an access list, use the **access-class** line configuration subcommand. Full command syntax for this command follows:

access-class *list* {**in** | **out**}

no access-class *list* {**in** | **out**}

This command restricts connections on a line or group of lines to certain Internet addresses.

The argument *list* is an integer from 1 through 99 that identifies a specific access list of Internet addresses.

The keyword **in** applies to incoming connections, such as virtual terminals. The keyword **out** applies to outgoing Telnet connections.

The **no access-class** line configuration subcommand removes access restrictions on the line for the specified connections.

Example 1:

The following example defines an access list that permits only hosts on network *192.89.55.0* to connect to the virtual terminals on the router.

```
access-list 12 permit 192.89.55.0  0.0.0.255
line 1 5
access-class 12 in
```

Use the **access-class** keyword **out** to define the access checks made on outgoing connections. (A user who types a host name at the router prompt to initiate a Telnet connection is making an outgoing connection.)

Note: Set identical restrictions on all the virtual terminal lines, because a user may connect to any of them.

Example 2:

The following example defines an access list that denies connections to networks other than network *36.0.0.0* on terminal lines 1 through 5.

```
access-list 10 permit 36.0.0.0   0.255.255.255
line 1 5
access-class 10 out
```

To display the access lists for a particular terminal line, use the EXEC command **show line** and specify the line number.

Controlling Interface Access

To control access to an interface, use the **access-group** interface subcommand, as shown below:

access-group *list*

The argument *list* is an integer from 1 through 199 that specifies an access list.

After receiving and routing a packet to a controlled interface, the router checks the destination address of the packet against the access list. If the access list permits the address, the router transmits the packet. If the access list rejects the address, the router discards the packet and returns an ICMP *Destination Unreachable* message. Access lists are applied on *outbound* interfaces, to *outbound* traffic.

Note: If this statement is made without the access list number, the implicit deny default takes precedence and will deny all access.

Example:

The following example applies list 101:

```
interface ethernet 0
access-group 101
```

Configuring the IP Security Option (IPSO)

All aspects of the IP Security Option (IPSO) are set up using configuration commands. The Cisco IPSO support addresses both the Basic and Extended security options described in a draft of the IPSO circulated by the Defense Communications Agency. This draft document superceded RFC1038, but has not yet been published as an RFC.

The following list describes some of the abilities of the IP security option (IPSO).

- Defines security level on a per-interface basis.
- Defines single level or multilevel interfaces.
- Provides a label for incoming datagrams.
- Strips labels on a per-interface basis.
- Re-orders options to put any Basic Security Option first.
- Accepts or rejects messages with extended security options.

IPSO Definitions

The following definitions apply to the descriptions of IPSO in this section.

- **level**—The degree of sensitivity of information. For example, data marked TOPSECRET is more sensitive than data marked SECRET. Table 13-4 lists the level keywords used by the Cisco software and their corresponding bit patterns.
- **authority**—An organization that defines the set of security levels that will be used in a network. For example, the Genser authority consists of level names defined by the Defense Communications Agency (DCA). Table 13-5 lists the authority keywords used by the Cisco software and their corresponding bit patterns.
- **label**—A combination of a security level and an authority or authorities.

Table 13-4 IPSO Level Keywords and Bit Patterns

Level Keyword	Bit Pattern
Reserved4	0000 0001
TopSecret	0011 1101
Secret	0101 1010
Confidential	1001 0110
Reserved3	0110 0110
Reserved2	1100 1100
Unclassified	1010 1011
Reserved1	1111 0001

Table 13-5 IPSO Authority Keywords and Bit Patterns

Authority Keyword	Bit Pattern
Genser	1000 0000
Siop-Esi	0100 0000
SCI	0010 0000
NSA	0001 0000

Disabling IPSO

The **no ip security** interface subcommand resets an interface to its default state, dedicated, unclassified Genser; no extended state is allowed.

no ip security

Use one of the **ip security** commands to enable other kinds of security.

Setting Security Classifications

The **ip security dedicated** interface subcommand sets the interface to the requested classification and authorities.

ip security dedicated *level authority* [*authority . . .*]

All traffic entering the system on this interface must have a security option that exactly matches this label. Any traffic leaving via this interface will have this label attached to it. The levels and authorities were listed previously in tables.

Example:

In the following example, we set a confidential level with Genser authority:

```
ip security dedicated confidential Genser
```

Setting a Range of Classifications

The **ip security multilevel** interface subcommand sets the interface to the requested range of classifications and authorities. All traffic entering or leaving the system must have a security option that falls within this range. The levels are set with this command:

ip security multilevel *level1* [a*uthority* **...**] **to** *level2 authority2* [*authority2...*]

Being within range requires that the following two conditions be met:

- The classification level must be greater than or equal to *level1,* and less than or equal to *level2.*
- The authority bits must be a superset of *authority1,* and a subset of *authority2.* That is, *authority1* specifies those authority bits that are required on a datagram, while *authority2* specifies the required bits plus any optional authorities that can also be included. If the *authority1* field is the empty set, then a datagram is required to specify any one or more of the authority bits in *authority2.*

Example:

In the example below, we specify levels Unclassified to Secret and NSA authority.

```
ip security multilevel unclassified to secret nsa
```

Modifying Security Levels

IPSO allows you to choose from several interface subcommands if you decide you need to modify your security levels.

Ignore Authority Field

The **ip security ignore-authorities** interface subcommand ignores the authorities field of all incoming datagrams. The value used in place of this field will be the authority value declared for the given interface. Full syntax for this command follows.

ip security ignore-authorities

no ip security ignore-authorities

This action is only allowed for single-level interfaces. Enter the **no ip security ignore-authorities** command to turn this function off.

Accept Unlabeled Datagrams

The **ip security implicit-labelling** interface subcommand accepts datagrams on the interface, even if they do not include a security option. If your interface has multilevel security set, you must use the second form of the command (because it specifies the precise level and authority to use when labeling the datagram, just like your original **ip security multilevel** subcommand.) The full syntax of the **ip security implicit-labelling** command follows.

ip security implicit-labelling
no ip security implicit-labelling

ip security implicit-labelling *level authority* [*authority ...*]
no ip security implicit-labelling *level authority* [*authority ...*]

Enter the **ip security implicit-labelling** command (optionally, with the appropriate arguments) to turn these functions off.

Example:

In the example below, an interface is set for security and will accept unlabeled datagrams.

```
ip security dedicated confidential genser
ip security implicit-labelling
```

Accept Datagrams with Extended Security Option

The **ip security extended-allowed** interface subcommand accepts datagrams on the interface that have an extended security option present. Full syntax is shown below:

ip security extended-allowed

no ip security extended-allowed

The default condition rejects the datagram immediately; the **no ip security extended-allowed** command restores this default.

Adding or Removing Security Option by Default

The **ip security add** interface subcommand ensures that all datagrams leaving the router on this interface contain a basic security option. Its full syntax is as follows.

ip security add

no ip security add

If an outgoing datagram does not have a security option present, this subcommand will add one as the first IP option. The security label added to the option field is the label that was computed for this datagram when it first entered the router. Because this action is performed after all the security tests have been passed, this label will either be the same as or will fall within the range of the interface. This action is always enforced on multilevel interfaces.

The **ip security strip** interface subcommand removes any basic security option that may be present on a datagram leaving the router through this interface. The full syntax of this command follows.

ip security strip

no ip security strip

This is performed after all security tests in the router have been passed and is not allowed for multilevel interfaces.

Prioritizing the Presence of a Security Option

The **ip security first** interface subcommand prioritizes the presence of security options on a datagram. The full syntax of this command is as shown:

ip security first

no ip security first

If a basic security option is present on an outgoing datagram, but it is not the first IP option, then it is moved to the front of the options field when this subcommand is used.

Default Values for Minor Keywords

In order to fully comply with IPSO, the default values for the minor keywords have become complex:

- The default for all of the minor keywords is *off*, with the exception of **implicit-labelling** and **add.**
- The default value of **implicit-labelling** is *on*, if the interface is unclassified genser, and otherwise *off.*
- The default value for **add** is *on* if the interface is not unclassified genser, and otherwise *off.*

Table 13-6 provides a list of all default values.

Table 13-6 Default Security Keyword Values

Type	Level	Authority	Implicit	Add
none	(none)	(none)	on	off
dedicated	Unclassified	Genser	on	off
dedicated	any	any	off	on
multilevel	any	any	off	on

The default value for an interface is "dedicated, unclassified genser." Note that this implies implicit labeling. This may seem unusual, but it makes the system entirely transparent to datagrams without options. This is the setting generated when the **no ip security** subcommand is given.

IPSO Configuration Examples

In this first example, three Ethernet interfaces are presented. These interfaces are running at security levels of Confidential Genser, Secret Genser, and Confidential to Secret Genser as shown in Figure 13-7.

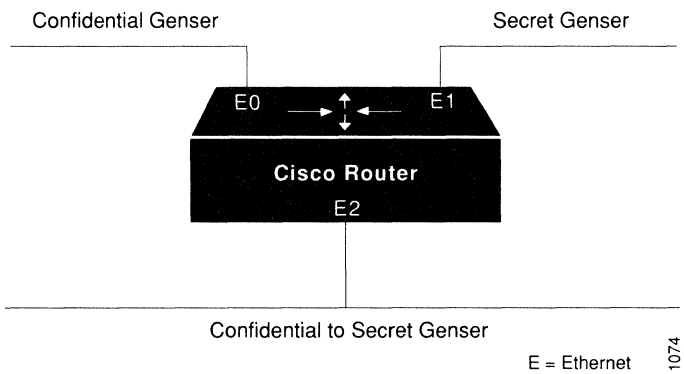


Figure 13-7 IPSO Security Levels

Example 1:

The following commands set up interfaces for the configuration in Figure 13-7.

```
interface ethernet 0
ip security dedicated confidential genser
interface ethernet 1
ip security dedicated secret genser
interface ethernet 2
ip security multilevel confidential genser to secret genser
end
```

It is possible for the set up to be much more complex.

Example 2:

In this next example, there are devices on Ethernet 0 that cannot generate a security option, and so must accept datagrams without a security option. These hosts also crash when they receive a security option, therefore, never place one on such interfaces. Furthermore, there are hosts on the other two networks that are using the extended security option to communicate information, so you must allow these to pass through the system. Finally, there is also a host on Ethernet 2 that requires the security option to be the first option present, and this condition must also be specified. The new configuration follows.

```
interface ethernet 0
ip security dedicated confidential genser
ip security implicit-labelling
ip security strip
interface ethernet 1
ip security dedicated secret genser
ip security extended-allowed
!
interface ethernet 2
ip security multilevel confidential genser to secret genser
ip security extended-allowed
ip security first
```

Debugging IPSO

Debugging of security-related problems can be performed by using the EXEC command **debug ip-packet**. Each time a datagram fails any security test in the system, a message is logged describing the exact cause of failure.

Security failure is also reported to the sending host when allowed by the specification. This calculation on whether to send an error message can be somewhat confusing. It depends upon both the security label in the datagram and the label of the incoming interface. First, the label contained in the datagram is examined for anything obviously wrong. If nothing is wrong, it should be assumed to be correct. If there is something wrong, then the datagram should be treated as *unclassified genser.* Then this label is compared to the interface range, and the appropriate action is taken.

Table 13-7 Security Actions

Classification	Authorities	Action Taken
Too low	Too low	No Response
	Good	No Response
	Too high	No Response
In range	Too Low	No Response
	Good	Accept
	Too high	Send Error
Too high	Too Low	No Response
	In range	Send Error
	Too high	Send Error

The range of ICMP error messages that can be generated by the security code is very small. The only possible error messages are:

- `ICMP Parameter problem, code 0` — Error at pointer
- `ICMP Parameter problem, code 1` — Missing option
- `ICMP Parameter problem, code 2` — See Note, below
- `ICMP Unreachable, code 10` — Administratively prohibited

Note: The message `ICMP Parameter problem, code 2` identifies a very specific error that occurs in the processing of a datagram. This message indicates that a datagram containing a maximum length IP header, but no security option, was received by the router. After being processed and routed to another interface, it is discovered that the outgoing interface is marked with "add a security label." Since the IP header is already full, the system cannot add a label and must drop the datagram and return an error message.

Configuring IP Accounting

IP accounting is enabled on a per-interface basis. The IP accounting support records the number of bytes and packets switched through the system on a source and destination IP address basis. Only transit IP traffic is measured and only on an outbound basis; traffic generated by the router, or terminating in the router, is not included in the accounting statistics.

Enabling IP Accounting

The interface subcommand **ip accounting** enables or disables IP accounting for transit traffic outbound on an interface. Full syntax of this command follows.

ip accounting

no ip accounting

It does not matter whether or not IP fast-switching or IP access lists are being used on that interface; the numbers will be accurate; however, IP accounting does not keep statistics if autonomous switching is set.

Defining Maximum Entries

The global configuration command **ip accounting-threshold** enables or disables IP accounting for transit traffic outbound on an interface, as follows.

ip accounting-threshold *threshold*

no ip accounting-threshold *threshold*

The accounting threshold defines the maximum number of entries (source and destination address pairs) that the router accumulates, preventing IP accounting from possibly consuming all available free memory. This level of memory consumption could occur in a router that is switching traffic for many hosts. The default threshold value is 512 entries. Overflows will be recorded; see the monitoring commands for display formats.

Example:

The following example sets the IP accounting threshold to only 500 entries.

```
ip accounting-threshold 500
```

Specifying Accounting Filters

Use the **ip accounting-list** global configuration command to filter accounting information for hosts. The full syntax for this command follows.

ip accounting-list *ip-address mask*

no ip accounting-list *ip-address mask*

The source and destination address of each IP datagram is logically ANDed with the *mask* and compared with *ip-address*. If there is a match, the information about the IP datagram will be entered into the accounting database. If there is no match, then the IP datagram is considered a *transit* datagram and will be counted according to the setting of the **ip accounting-transits** command described next.

Use the **no ip accounting-list** command with the appropriate argument to remove this function.

Controlling the Number of Transit Records

The **ip accounting-transits** global configuration command controls the number of transit records that will be stored in the IP accounting database. The full syntax of this command is as follows.

ip accounting-transits *count*

no ip accounting-transits *count*

Transit entries are those that do not match any of the filters specified by **ip accounting-list** commands. If you do not define filters, the router will not maintain transit entries. To maintain accurate accounting totals, the router software maintains two accounting databases: an active and a checkpointed database.

Use the **no ip accounting-transits** command to remove this function.

Example:

The following example specifies that no more than 100 transit records are stored.

```
ip accounting-transit 100
```

Use the EXEC command **show ip accounting** to display the active accounting database. The EXEC command show ip accounting checkpoint displays the checkpointed database. The EXEC command **clear ip accounting** clears the active database and creates the checkpointed database. See the sections "Maintaining the IP Network" and "Monitoring the IP Network" later in this chapter for more options on monitoring your network's accounting.

Special IP Configurations

This section discusses how to configure static routes, source routing, how to control IP processing on serial interfaces, and how to manage fast-switching.

Configuring Source Routing

The command **no ip source-route** causes the system to discard any IP datagram containing a source-route option. The **ip source-route** global configuration subcommand allows the router to handle IP datagrams with source routing header options.

ip source-route

no ip source-route

The default behavior is to perform the source routing.

IP Processing on a Serial Interface

The **ip unnumbered** interface subcommand enables IP processing on a serial interface, but does not assign an explicit IP address to the interface. The full command syntax is shown below:

ip unnumbered *interface-name*

no ip unnumbered *interface-name*

The argument *interface-name* is the name of another interface on which the router has an assigned IP address. The interface may not be another unnumbered interface, or the interface itself.

Whenever the unnumbered interface generates a packet (for example, for a routing update), it uses the address of the specified interface as the source address of the IP packet. It also uses the address of the specified interface in determining which routing processes are sending updates over the unnumbered interface. Restrictions include:

- Only those serial interfaces using HDLC encapsulation may be unnumbered. It is not possible to use this subcommand with X.25 interfaces.
- You cannot use the **ping** command to determine if the interface is up, since the interface has no address. SNMP (Simple Network Management Protocol) may be used to remotely monitor interface status.
- You cannot netboot a runnable image over an unnumbered serial interface.
- You cannot support IP security options on an unnumbered interface.
- The argument *interface-name* is the name of another interface in the network server which has an IP address, not another unnumbered interface.

Example:

In the example below, the first serial interface is given Ethernet 0's address.

```
interface ethernet 0
ip address 131.108.6.6 255.255.255.0
interface serial 0
ip unnumbered ethernet 0
```

Configuring Simplex Ethernet Interfaces

The **transmit-interface** interface subcommand assigns a transmit interface to a receive-only interface.

transmit-interface *interface-name*

When a route is learned on this receive-only interface, the interface designated as the source of the route is converted to *interface-name*. This is useful in setting up dynamic IP routing over a simplex circuit, that is, a circuit that receives only or transmits only. When packets are routed out *interface-name*, they are sent to the IP address of the source of the routing update. To reach this IP address on a transmit-only Ethernet link, a static ARP entry mapping this IP address to the hardware address of the other end of the link is required.

Example:

This example illustrates how to configure IP on two routers sharing transmit only and receive only Ethernet connections.

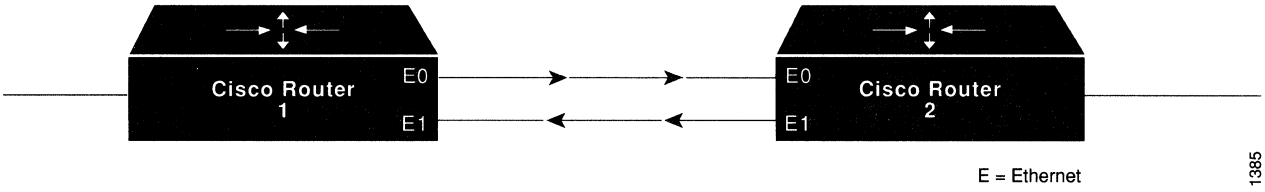


Figure 13-8 Simplex Ethernet Connections

Example for Router 1:

```
interface ethernet 0
ip address 128.9.1.1
!
interface ethernet 1
ip address 128.9.1.2
transmit-interface ethernet 0
!
!use show interfaces command to find router2-MAC-address-E0
arp router2-MAC-address-E0 128.9.1.4 arpa
```

Example for Router 2:

```
interface ethernet 0
ip address 128.9.1.3
transmit-interface ethernet 1
!
interface ethernet 1
ip address 128.9.1.4
!
!use show interfaces command to find router1-MAC-address-E1
arp router1-MAC-address-E1 128.9.1.1 arpa
!
```

Enabling Fast Switching

The **ip route-cache** interface subcommand controls the use of outgoing packets on a high-speed switching cache for IP routing. The route cache is enabled by default and allows load-balancing on a *per-destination* basis.

ip route-cache

no ip route-cache

To enable load-balancing on a *per-packet* basis, use the **no ip route-cache** command to disable fast-switching.

Cisco routers generally offer better packet transfer performance when fast switching is enabled with one exception. On networks using slow serial links (56K and below) disabling fast switching to enable the per-packet load-sharing is usually the better choice.

Enabling IP Autonomous Switching

Autonomous switching gives a router faster packet processing by allowing the cBus to switch packets independently, without interrupting the system processor. It works only in AGS+ systems with high-speed network controller cards, such as the CSC-MEC and CSC-FCI, and with a cBus controller card running microcode version 1.4 or later. (See the "Microcode Revisions" section in the release notes accompanying this publication for other microcode revision requirements.)

Autonomous switching is enabled by adding the **cbus** keyword to the existing **ip route-cache** interface subcommand. The syntax to enable and disable this function follows.

ip route-cache [cbus]

no ip route-cache [cbus]

By default, IP autonomous switching is not enabled. The **ip route-cache** command sets up fast switching, and by default, fast switching is enabled on all MCI/cBus interfaces.

To turn *on* both fast switching and autonomous switching use this syntax:

ip route-cache cbus

To turn *off* both fast and autonomous switching on an interface, add the **no** keyword as shown below:

no ip route-cache

To turn off autonomous switching only on an interface, use this syntax:

no ip route-cache cbus

To return to the default, use the standard ip route-cache command:

ip route-cache

This turns fast switching on and autonomous switching off.

TCP Header Compression

You can compress the TCP headers of your Internet packets in order to reduce the size of your packets. TCP header compression is only supported on serial lines using HDLC encapsulation. (RFC1144 specifies the compression process.) Compressing the TCP header can speed up Telnet connections dramatically. In general, TCP header compression is advantageous when your traffic consists of many small packets, not for traffic that consists of large packets. Transaction-processing (using terminals, usually) tends to use small packets while file

transfers use large packets. This feature only compresses the TCP header, of course, so it has no effect on UDP packets or other protocol headers.

The **ip tcp header-compression** interface subcommand enables header compression. Full command syntax for this command follows:

ip tcp header-compression [**passive**]

no ip tcp header-compression [**passive**]

If you use the optional **passive** keyword, outgoing packets are only compressed if TCP incoming packets on the same interface are compressed. Without the **passive** keyword, the router will compress all traffic. The **no ip tcp header-compression** command (the default) disables compression. You must enable compression on both ends of a serial connection.

When compression is enabled, fast switching is disabled which means that fast interfaces like T-1 can overload the router. Think about your network's traffic characteristics before using this command. See the section "Monitoring the IP Network" for more explanation of commands for monitoring your compressed traffic.

The **ip tcp compression-connection**s interface subcommand specifies the total number of header compression connections that can exist on an interface. Each connection sets up a compression cache entry, so you are, in effect, specifying the maximum number of cache entries and the size of the cache.

ip tcp compression-connections *number*

The argument *number* specifies the number of connections the cache will support. The default is 16; *number* can vary between 3 and 256, inclusive. Too few cache entries for the specific interface can lead to degraded performance while too many cache entries leads to wasted memory.

Note: Both ends of the serial connection must use the same number of cache entries.

Example:

In the following example, the first serial interface is set for header compression with a maximum often cache entries.

```
interface serial 0
ip tcp header-compression
ip tcp compression-connections 10
```

IP Configuration Examples

This section shows complete configuration examples for the most common configuration situations.

Configuring Serial Interfaces

In the example below, the second serial interface is given ethernet 0's address. The serial interface is unnumbered.

Example:

```
interface ethernet 0
ip address 145.22.4.67 255.255.255.0
interface serial 1
ip unnumbered ethernet 0
```

Flooding of IP Broadcasts

In this example, flooding of IP broadcasts is enabled on all interfaces (two Ethernet and two serial). No bridging is permitted. (The access list denies all protocols.) No specific UDP protocols are listed by a separate **ip forward-protocol udp** command, so the default protocols (TFTP, DNS, IEN-116, Time, NetBios, and BootP) will be flooded.

Example:

```
ip forward-protocol spanning-tree
bridge 1 protocol dec
access-list 201 deny 0x0000 0xFFFF
interface ethernet 0
bridge-group 1
bridge-group 1 input-type-list 201
interface ethernet 1
bridge-group 1
bridge-group 1 input-type-list 201
interface serial 0
bridge-group 1
bridge-group 1 input-type-list 201
interface serial 1
bridge-group 1
bridge-group 1 input-type-list 201
```

Creating a Network from Separated Subnets

In the example below, networks 132 and 196 are separated by a backbone as shown in Figure 13-9. The two networks are brought into the same logical network through the use of secondary addresses.

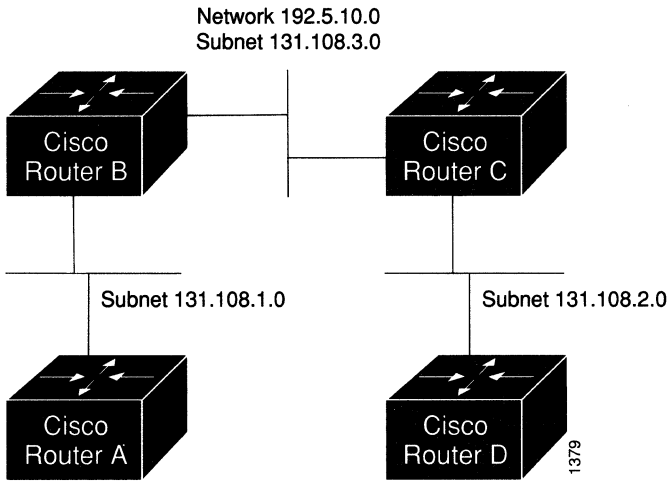


Figure 13-9 Creating a Network from Separated Subnets

Example—Router B:

```
interface ethernet 2
ip address 192.5.10.1 255.255.255.0
ip address 131.108.3.1 255.255.255.0 secondary
```

Example—Router C:

```
interface ethernet 1
ip address 192.5.10.2 255.255.255.0
ip address 131.108.3.2 255.255.255.0 secondary
```

Customizing ICMP Services

The example below changes some of the ICMP defaults for the first Ethernet interface. Disabling the sending of redirects could mean you don't think your routers on this segment will ever have to send a redirect. Lowering the error processing load on your router would be an efficiency move in this case. Disabling the unreachables messages will have a secondary effect—it will also disable MTU path discovery because path discovery works by having routers send unreachables messages. If you have a network segment with a small number of devices and an absolutely reliable traffic pattern—which could easily happen on a segment with a small number of little-used user devices—this would disable options your router would be unlikely to need to use anyway.

Example:

```
interface ethernet 0
no ip unreachables
no ip redirects
```

Helper Addresses

In this example, one server is on network *191.24.1.0.* and the other is on network *110.44.0.0*, and you want to permit IP broadcasts from all hosts to reach these servers. The example below illustrates how to configure the router that connects network 110 to network 191.

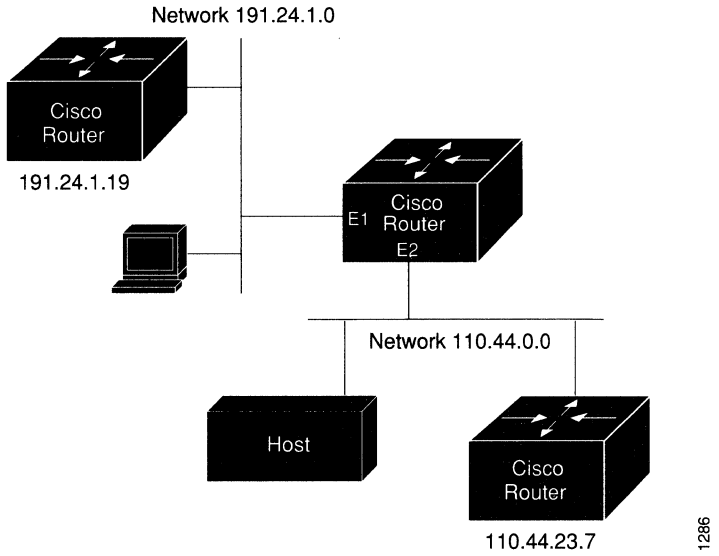


Figure 13-10 IP Helper Addresses

Example:

```
!
ip forward-protocol udp
!
interface ethernet 1
ip helper address 110.44.23.7
interface ethernet 2
ip helper address 191.24.1.19
```

HP Hosts on a Network Segment

The following example has a network segment with Hewlett-Packard devices on it. The commands listed customize the router's first Ethernet port to accommodate the HP devices.

Example:

```
ip hp-host bl4zip 131.24.6.27
interface ethernet 0
arp probe
ip probe proxy
```

Establishing IP Domains

The example below establishes a domain list with several alternate domain names.

Example:

```
ip domain-list cisco.com
ip domain-list telecomprog.edu
ip domain-list merit.edu
```

Configuring Access Lists

In the next example, network *36.0.0.0* is a Class A network whose second octet specifies a subnet; that is, its subnet mask is *255.255.0.0*. The third and fourth octets of a network *36.0.0.0* address specify a particular host. Using access list 2, the router would accept one address on subnet 48 and reject all others on that subnet. The router would accept addresses on all other network *36.0.0.0* subnets; that is the purpose of the last line of the list.

Example:

```
access-list 2 permit 36.48.0.3  0.0.0.0
access-list 2 deny 36.48.0.0  0.0.255.255
access-list 2 permit 36.0.0.0  0.255.255.255
interface ethernet 0
access-group 2
```

Configuring Extended Access Lists

In the example below, the first line permits any incoming TCP connections with destination port greater than 1023. The second line permits incoming TCP connections to the SMTP port of host *128.88.1.2*. The last line permits incoming ICMP messages for error feedback.

Example:

```
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255 gt 1023
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 0.0.0.0 eq 25
access-list 102 permit icmp 0.0.0.0 255.255.255.255 128.88.0.0 255.255.255.255
interface ethernet 0
access-group 102
```

Maintaining the IP Network

Use the EXEC commands described in this section to maintain IP routing caches, tables, and databases.

Removing Dynamic Entries from the ARP Cache

The **clear arp-cache** EXEC command removes all dynamic entries from the Address Resolution Protocol (ARP) cache, and clears the fast-switching cache. Enter this command at the EXEC prompt:

clear arp-cache

Removing Entries from the Host-Name-and-Address Cache

Use the EXEC command **clear host** to removes one or all entries from the host-name-and-address cache, depending upon the argument you specify.

clear host {*name* | ★}

To remove a particular entry, use the argument *name* to specify the host. To clear the entire cache, use the asterisk (★) argument.

Clearing the Checkpointed Database

Use the **clear ip accounting** command to clear the active database when IP accounting is enabled. Use the **clear ip accounting checkpoint** command to clear the checkpointed database when IP accounting is enabled. You may also clear the checkpointed database by issuing the **clear ip accounting** command twice in succession. Enter one of these commands at the EXEC prompt.

clear ip accounting

clear ip accounting [checkpoint]

Removing Routes

Use the **clear ip route** command to remove a route from the IP routing table. Enter this command at the EXEC prompt:

clear ip route {*network* | ★}

The optional argument *network* is the network or subnet address of the route that you want to remove. Use the asterisk (★) argument to clear the entire routing table.

Monitoring the IP Network

Use the EXEC commands described in this section to obtain displays of activity on the IP network.

Displaying the IP Show Commands

Use the **show ip ?** command to display a list of all the available EXEC commands for monitoring the IP network. Following is sample output:

```
accounting <checkpoint>    Accounting statistics
arp                        IP ARP table
bgp <address>              Border Gateway Protocol
cache                      Fast switching cache
egp                        EGP peers
interface <name>           Interface settings
protocols                  Routing processes
route <network>            Routing table
tcp <keyword>              TCP information, type "show ip tcp ?" for list
traffic                    Traffic statistics
```

Displaying the ARP Cache

To display the ARP cache, use the following EXEC command:

show ip arp

This command displays the contents of the ARP cache. ARP establishes correspondences between network addresses (an IP address, for example) and LAN hardware addresses (Ethernet addresses). A record of each correspondence is kept in a cache for a predetermined amount of time and then discarded. Following is sample output. Table 13-8 describes the fields seen.

```
Protocol  Address          Age (min)     Hardware Addr   Type      Interface
AppleTalk 4.57                   0     aa00.0400.6408  ARPA      Ethernet0
Internet  131.108.1.140        137     aa00.0400.6408  ARPA      Ethernet0
Internet  131.108.1.111        156     0800.2007.8866  ARPA      Ethernet0
AppleTalk 4.128                  0     aa00.0400.6508  ARPA      Ethernet0
AppleTalk 4.129                  -     aa00.0400.0134  ARPA      Ethernet0
Internet  131.108.1.115         33     0000.0c01.0509  ARPA      Ethernet0
Internet  192.31.7.24            5     0800.0900.46fa  ARPA      Ethernet2
Internet  192.31.7.26           41     aa00.0400.6508  ARPA      Ethernet2
Internet  192.31.7.27            -     aa00.0400.0134  ARPA      Ethernet2
Internet  192.31.7.28           67     0000.0c00.2c83  ARPA      Ethernet2
Internet  192.31.7.17           67     2424.c01f.0711  ARPA      Ethernet2
Internet  192.31.7.18           64     0000.0c00.6fbf  ARPA      Ethernet2
Internet  192.31.7.21          114     2424.c01f.0715  ARPA      Ethernet2
Internet  131.108.1.33          15     0800.2008.c52e  ARPA      Ethernet0
Internet  131.108.1.55          44     0800.200a.bbfe  ARPA      Ethernet0
Internet  131.108.1.6           89     aa00.0400.6508  ARPA      Ethernet0
Internet  131.108.7.1            -     0000.0c00.750f  ARPA      Ethernet3
Internet  131.108.1.1            -     aa00.0400.0134  ARPA      Ethernet0
Internet  131.108.1.27          75     0800.200a.8674  ARPA    Ethernet0
```

Table 13-8 Show IP Arp Field Displays

Field	Description
Protocol	Protocol for network address in the Address field
Address	The network address that corresponds to Hardware Addr
Age (min)	Age, in minutes, of the cache entry
Hardware Addr	LAN hardware address that corresponds to network address
Type	Type of ARP (Address Resolution Protocol): ARPA = Ethernet-type ARP SNAP = RFC 1042 ARP Probe = HP Probe Protocol

Displaying IP Accounting

The **show ip accounting** command displays the active accounting database. The **show ip accounting checkpoint** command displays the checkpointed database.

show ip accounting

show ip accounting checkpoint

Following is sample output for the **show ip accounting** and **show ip accounting checkpoint** commands:

```
  Source           Destination      Packets               Bytes
131.108.19.40    192.67.67.20           7                 306
131.108.13.55    192.67.67.20          67                2749
131.108.2.50     192.12.33.51          17                1111
131.108.2.50     130.93.2.1             5                 319
131.108.2.50     130.93.1.2           463               30991
131.108.19.40    130.93.2.1             4                 262
131.108.19.40    130.93.1.2            28                2552
131.108.20.2     128.18.6.100          39                2184
131.108.13.55    130.93.1.2            35                3020
131.108.19.40    192.12.33.51        1986               95091
131.108.2.50     192.67.67.20         233               14908
131.108.13.28    192.67.67.53         390               24817
131.108.13.55    192.12.33.51      214669             9806659
131.108.13.111   128.18.6.23        27739             1126607
131.108.13.44    192.12.33.51       35412             1523980
192.31.7.21      130.93.1.2            11                824
131.108.13.28    192.12.33.2           21                1762
131.108.2.166    192.31.7.130         797              141054
131.108.3.11     192.67.67.53           4                 246
192.31.7.21      192.12.33.51       15696              695635
192.31.7.24      192.67.67.20          21                 916
131.108.13.111   128.18.10.1           16                1137
```

The output lists the source and destination addresses, as well as total number of packets and bytes for each address pair.

Displaying Host Statistics

The **show hosts** command displays the default domain name, the style of name lookup service, a list of name server hosts, and the cached list of host names and addresses.

show hosts

Enter **show hosts** at the user-level prompt.

Following is sample output:

```
show hosts
Default domain is CISCO.COM
Name/address lookup uses domain service
Name servers are 255.255.255.255
Host                    Flags       Age Type   Address(es)
SLAG.CISCO.COM          (temp, OK)  1   IP     131.108.4.10
CHAR.CISCO.COM          (temp, OK)  8   IP     192.31.7.50
CHAOS.CISCO.COM         (temp, OK)  8   IP     131.108.1.115
DIRT.CISCO.COM          (temp, EX)  8   IP     131.108.1.111
DUSTBIN.CISCO.COM       (temp, EX)  0   IP     131.108.1.27
DREGS.CISCO.COM         (temp, EX) 24   IP     131.108.1.30
```

In the display:

- A `temp` entry in the Flags field is entered by a name server; the router removes the entry after 72 hours of inactivity.
- A `perm` entry is entered by a configuration command and is not timed out. Entries marked OK are believed to be valid. Entries marked ?? are considered suspect and subject to revalidation. Entries marked EX are expired.
- The `Age` field indicates the number of hours since the router last referred to the cache entry. The Type field identifies the type of address, for example, IP, CLNS, or X.121.
- The `Address(es)` field shows the address of the host. One host may have up to eight addresses.

If you have used the **ip hp-host** configuration command (see the section "HP Probe Proxy Support"), the **show hosts** command will display these host names as type `HP-IP`.

Displaying the Route Cache

The **show ip cache** command displays the routing table cache that is used to rapidly switch Internet traffic. Enter this command at the EXEC prompt:

show ip cache

Following is sample output:

```
IP routing cache version 435, entries 19/20, memory 880

 Hash   Destination     Interface  MAC Header
*6D/0   128.18.1.254    Serial0    0F000800
*81/0   131.108.1.111   Ethernet0  00000C002C83AA00040002340800
*8D/0   131.108.13.111  Ethernet0  AA0004000134AA00040002340800
 99/0   128.18.10.1     Serial0    0F000800
*9B/0   128.18.10.3     Serial0    0F000800
*B0/0   128.18.5.39     Serial0    0F000800
*B6/0   128.18.3.39     Serial0    0F000800
*C0/0   131.108.12.35   Ethernet0  AA0004000134AA00040002340800
*C4/0   131.108.2.41    Ethernet0  00000C002C83AA00040002340800
*C9/0   192.31.7.17     Ethernet0  2424C01F0711AA00040002340800
*CD/0   192.31.7.21     Ethernet0  2424C01F0715AA00040002340800
*D5/0   131.108.13.55   Ethernet0  AA0004006508AA00040002340800
*DC/0   130.93.1.2      Serial0    0F000800
*DE/0   192.12.33.51    Serial0    0F000800
*DF/0   131.108.2.50    Ethernet0  AA0004000134AA00040002340800
*E7/0   131.108.3.11    Ethernet0  00000C002C83AA00040002340800
*EF/0   192.12.33.2     Serial0    0F000800
*F5/0   192.67.67.53    Serial0    0F000800
*F5/1   131.108.1.27    Ethernet0  AA0004006508AA00040002340800
*FE/0   131.108.13.28   Ethernet0  AA0004006508AA00040002340800
```

In the display:

- The `*` designates valid routes.
- The `Destination` field shows the destination IP address.
- The `Interface` field specifies the interface type and number (serial 1, Ethernet 2, etc.).
- The `MAC Header` field displays the MAC header.

Displaying Interface Statistics

To display the usability status of interfaces, use the EXEC command **show interfaces.** If the interface hardware is usable, the interface is marked "up." If the interface can provide two-way communication, the line protocol is marked "up." For an interface to be usable, both the interface hardware and line protocol must be up.

show ip interface [*interface unit*]

If you specify an optional interface type, you will see only information on that specific interface.

If you specify no optional parameters you will see information on all the interfaces.

The following sample output was obtained by specifying the serial 0 interface:

```
Serial 0 is up, line protocol is up
  Internet address is 192.31.7.129, subnet mask is 255.255.255.240
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is 131.108.1.255
  Outgoing access list is not set
  Proxy ARP is enabled
  Security level is default
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is enabled
  Gateway Discovery is disabled
  IP accounting is enabled, system threshold is 512
  TCP/IP header compression is disabled
  Probe proxy name replies are disabled
```

In the display:

- The `Broadcast Address` field shows the broadcast address.
- The `Helper Address` field specifies a helper address, if one has been set.
- The `Outgoing Access List` field indicates whether or not the interface has an outgoing access list set.
- The `Proxy ARP` field indicates whether Proxy ARP is enabled for the interface.
- The `Security Level` field specifies the IPSO security level set for this interface.
- The `ICMP redirects` field specifies whether redirects will be sent on this interface.
- The `ICMP unreachables` field specifies whether unreachable messages will be sent on this interface.
- The `ICMP mask` replies field specifies whether mask replies will be sent on this interface.
- The `IP fast switching` field specifies whether fast switching has been enabled for this interface. It is generally enabled on serial interfaces, such as this one.
- The `Gateway Discovery` field specifies whether the discovery process has been enabled for this interface. It is generally disabled on serial interfaces, such as this one.
- The `IP accounting` field specifies whether IP accounting is enabled for this interface and what the threshold (maximum number of entries) is.
- The `TCP/IP header compression` field indicates whether compression is enabled or disabled.
- The `Probe proxy name` field indicates whether the function is enabled or disabled.

Displaying the Routing Table

The **show ip route** command displays the IP routing table. Enter this command at the EXEC prompt:

show ip route [*network*]

A specific network in the routing table is displayed when the optional *network* argument is entered.

Following is sample output with the optional network argument:

```
Routing entry for 131.108.1.0
  Known via "igrp 109", distance 100, metric 1200
  Redistributing via igrp 109
  Last update from 131.108.6.7 on Ethernet0, 35 seconds ago
  Routing Descriptor Blocks:
  * 131.108.6.7, from 131.108.6.7, 35 seconds ago, via Ethernet0
      Route metric is 1200, traffic share count is 1
      Total delay is 2000 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 0
```

This display is the result of the **show ip route** command without the network number:

```
Codes: I - IGRP derived, R - RIP derived, H - HELLO derived
       C - connected, S - static, E - EGP derived, B - BGP derived
       * - candidate default route

Gateway of last resort is 131.108.6.7 to network 131.119.0.0

I*Net 128.145.0.0 [100/1020300] via 131.108.6.6, 30 sec, Ethernet0
I Net 192.68.151.0 [100/160550] via 131.108.6.6, 30 sec, Ethernet0
I Net 128.18.0.0 [100/8776] via 131.108.6.7, 58 sec, Ethernet0
                            via 131.108.6.6, 31 sec, Ethernet0
E Net 128.128.0.0 [140/4] via 131.108.6.64, 130 sec, Ethernet0
C Net 131.108.0.0 is subnetted (mask is 255.255.255.0), 54 subnets
I     131.108.144.0 [100/1310] via 131.108.6.7, 78 sec, Ethernet0
C     131.108.91.0 is directly connected, Ethernet1
```

The output begins by showing the address of the gateway of last resort for this network. In the rest of the display:

- The first field specifies how the route was derived. The options are listed above the routing table.
- The second field specifies a remote network/subnet to which a route exists. The first number in brackets is the administrative distance of the information source; the second number is the metric for the route.
- The third field specifies the IP address of a router that is the next hop to the remote network.
- The fourth field specifies the number of seconds since this network was last heard.
- The final field specifies the interface through which you can reach the remote network via the specified router.

Displaying Protocol Traffic Statistics

The **show ip traffic** command displays IP protocol statistics. Enter this command at the EXEC prompt:

show ip traffic

Following is sample output:

```
IP statistics:
  Rcvd:  98 total, 98 local destination
         0 format errors, 0 checksum errors, 0 bad hop count
         0 unknown protocol, 0 not a gateway
         0 security failures, 0 bad options
  Frags: 0 reassembled, 0 timeouts, 0 too big
         0 fragmented, 0 couldn't fragment
  Bcast: 38 received, 52 sent
  Sent:  44 generated, 0 forwarded
         0 encapsulation failed, 0 no route

ICMP statistics:
  Rcvd: 0 checksum errors, 0 redirects, 0 unreachable, 0 echo
        0 echo reply, 0 mask requests, 0 mask replies, 0 quench
        0 parameter, 0 timestamp, 0 info request, 0 other
  Sent: 0 redirects, 3 unreachable, 0 echo, 0 echo reply
        0 mask requests, 0 mask replies, 0 quench, 0 timestamp
        0 info reply, 0 time exceeded, 0 parameter problem

UDP statistics:
  Rcvd: 56 total, 0 checksum errors, 55 no port
  Sent: 18 total, 0 forwarded broadcasts

TCP statistics:
  Rcvd: 0 total, 0 checksum errors, 0 no port
  Sent: 0 total

EGP statistics:
  Rcvd: 0 total, 0 format errors, 0 checksum errors, 0 no listener
  Sent: 0 total

IGRP statistics:
  Rcvd: 73 total, 0 checksum errors
  Sent: 26 total

HELLO statistics:
  Rcvd: 0 total, 0 checksum errors
  Sent: 0 total

ARP statistics:
  Rcvd: 20 requests, 17 replies, 0 reverse, 0 other
  Sent: 0 requests, 9 replies (0 proxy), 0 reverse

Probe statistics:
  Rcvd: 6 address requests, 0 address replies
        0 proxy name requests, 0 other
  Sent: 0 address requests, 4 address replies (0 proxy)
        0 proxy name replies
```

In the display:

- A format error is a gross error in the packet format, such as an impossible Internet header length.
- A bad hop count occurs when a packet is discarded because its time-to-live (TTL) field was decremented to zero.
- An encapsulation failure usually indicates that the router received no reply to an ARP request and therefore did not send a datagram.
- A no route occurrence is counted when the router discards a datagram it did not know how to route.
- A proxy reply is counted when the router sends an ARP or Probe Reply on behalf of another host. The display shows the number of probe proxy requests that have been received and the number of responses that have been sent.

Monitoring TCP Header Compression

The **show ip tcp header-compression** command shows statistics on compression. Enter this command at the EXEC prompt:

show ip tcp header-compression

Following is sample output:

```
TCP/IP header compression statistics:
  Interface Serial1: (passive, compressing)
    Rcvd:    4060 total, 2891 compressed, 0 unknown type, 0 errors
             0 dropped, 1 buffer copies, 0 buffer failures
    Sent:    4284 total, 3224 compressed,
             105295 bytes saved, 661973 bytes sent
             1.15 efficiency improvement factor
    Connect: 16 slots, 1543 long searches, 2 misses, 99% hit ratio
             Five minute miss rate 0 misses/sec, 0 max misses/sec
```

In the display:

- The `buffer copies` are the number of packets that had to be copied into bigger buffers for decompression.
- The report `buffer failures` is the number of packets dropped due to a lack of buffers.
- The `efficiency improvement factor` is the improvement in line efficiency because of TCP header compression.
- The number of `slots` is the size of the cache.
- The `long searches` field indicates the number of times the software had to look "hard" to find a match.
- The `misses` field indicates the number of times a match could not be made. If your output shows a large miss rate, then the number of allowable simultaneous compression connections may be too small.

- The `hit ratio` is the percentage of times the software found a match and was able to compress the header.
- The `Five minute miss rate` calculates the miss rate over the previous five minutes for a longer-term (and more accurate) look at miss rate trends.

The IP Ping Command

The EXEC command **ping** allows the administrator to diagnose network connectivity by sending ICMP *Echo Request* messages and waiting for ICMP *Echo Reply* messages. The following sample session shows **ping** command output for IP:

Sample Session 1:

```
Protocol [ip]:
Target IP address: 131.108.1.27
Repeat count [5]:
Datagram size [100]: 1000
Timeout in seconds [2]:
Extended commands [n]: yes
Source address:
Type of service [0]:
Set DF bit in IP header? [no]: yes
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 131.108.2.27, timeout is 2 seconds:
M.M.M
Success rate is 60 percent, round-trip min/avg/max = 4/6/12 ms
```

The **ping** command uses the following notation to indicate the responses it sees:

Table 13-9 Ping Test Characters

Char	Meaning
!	Each exclamation point indicates receipt of a reply.
.	Each period indicates the network server timed out while waiting for a reply.
U	Destination unreachable error PDU received.
N	Network unreachable.
P	Protocol unreachable.
Q	Source quench.
M	Could not fragment.
?	Unknown packet type.

To abort a ping session, type the escape sequence (by default, Ctrl-^, X).

The IP **ping** command, in verbose mode, accepts a data pattern. The pattern is specified as a 16-bit hexadecimal number. The default pattern is 0xABCD. Patterns such as all ones or all zeros can be used to debug data sensitivity problems on CSU/DSUs.

Note: If the IP version of the **ping** command is used on a directly connected interface, the packet is sent out the interface and should be forwarded back to the router from the far end. The time travelled reflects this round trip route. This feature can be useful for diagnosing serial line problems. By placing the local or remote CSU/DSU into loopback mode and pinging your own interface, you can isolate the problem to the router or leased line.

Sample Session 2:

You can also specify the router address to use as the source address for ping packets, which here is *131.108.105.62*.

```
Sandbox#ping
Protocol [ip]:
Target IP address: 131.108.1.111
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: yes
Source address: 131.108.105.62
Type of service [0]:
Set DF bit in IP header? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 131.108.1.111, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/4 ms
```

The IP Trace Command

The EXEC command **trace** allows you to discover the routing path your router's packets are taking through your network. It sends probe packets to destination hosts and routers and takes advantage of the ICMP message packets that are generated when a packet exceeds its time-to-live (TTL) value.

The **trace** command offers default and settable parameters for specifying a simple or extended trace mode.

How Trace Works

The **trace** command works by taking advantage of the error messages generated by routers when a datagram exceeds its time-to-live (TTL) value.

The **trace** command starts by sending probe datagrams with a TTL value of one. This causes the first router to discard the probe datagram and send back an error message. The **trace** command sends several probes at each TTL level and displays the round trip time for each.

The **trace** command sends out one probe at a time. Each outgoing packet may result in one or two error messages. A *time exceeded* error message indicates that an intermediate router has seen and discarded the probe. A *destination unreachable* error message indicates that the destination node has received the probe and discarded it because it could not deliver the packet. If the timer goes off before a response comes in, **trace** prints an asterisk (*).

The **trace** command terminates when the destination responds, when the maximum TTL was exceeded, or when the user interrupts the trace with the escape sequence, by default Ctrl-^,X).

Common Trace Problems

Due to bugs in the IP implementations of various hosts and routers, the **trace** command may behave in odd ways.

Not all destinations will correctly respond to a *probe* message by sending back an *ICMP port unreachable* message. A long sequence of TTL levels with only asterisks, terminating only when the maximum TTL has been reached, may indicate this problem.

There is a known problem with the way some hosts handle an *ICMP TTL exceeded* message. Some hosts generate an *ICMP* message but they re-use the TTL of the incoming packet. Since this is zero, the ICMP packets do not make it back. When you trace the path to such a host, you may see a set of TTL values with asterisks (*). Eventually the TTL gets high enough that the *ICMP* message can get back. For example, if the host is six hops away, **trace** will timeout on responses 6 through 11.

Tracing IP Routes

When tracing IP routes, the following **trace** command parameters may be set:

- `Target IP address`. You must enter a host name or an IP address. There is no default.
- `Source Address`. One of the interface addresses of the router to use as a source address for the probes. The router will normally pick what it feels is the best source address to use.
- `Numeric Display`. The default is to have both a symbolic and numeric display; however, you may suppress the symbolic display.
- `Timeout in seconds`. The number of seconds to wait for a response to a probe packet. The default is three seconds.
- `Probe count`. This is the number of probes to be sent at each TTL level. The default count is 3.
- `Minimum Time to Live [1]`: The TTL value for the first probes. The default is 1, but may be set to a higher value to suppress the display of known hops.

- `Maximum Time to Live [30]`: This is the largest TTL value which may be used. The default is 30. The **trace** command terminates when the destination is reached or when this value is reached.
- `Port Number`. This is the destination port used by the UDP probe messages. The default is 33,434.
- `Loose, Strict, Record, Timestamp, Verbose`. These are IP header options. You may specify any combination. The **trace** command issues prompts for the required fields. Note that **trace** will place the requested options in each probe; however, there is no guarantee that all routers (or end-nodes) will process the options.
- `Loose Source Routing`. You may specify a list of nodes which must be traversed when going to the destination.
- `Strict Source Routing`. You may specify a list of nodes which must be the *only* nodes traversed when going to the destination.
- `Record`. You may specify the number of hops to leave room for.
- `Timestamp`. You may specify the number of time stamps to leave room for.
- `Verbose`. If you select any option, the verbose mode is automatically selected and **trace** prints the contents of the option field in any incoming packets. You can prevent verbose mode by selecting it again, toggling its current setting.

The following table describes the output from this test.

Table 13-10 Trace Test Characters

Char	Meaning
nn msec	The probe was successfully returned in *nn* milliseconds.
*	The probe timed out.
?	Unknown packet type.
Q	Source quench.
P	Protocol unreachable.
N	Network unreachable.
U	Host unreachable.

Sample Session 1:

The following is an example of the simple use of **trace**.

```
chaos#trace ABA.NYC.mil
Type escape sequence to abort.
Tracing the route to ABA.NYC.mil (26.0.0.73)
  1 DEBRIS.CISCO.COM (131.108.1.6) 1000 msec 8 msec 4 msec
  2 BARRNET-GW.CISCO.COM (131.108.16.2) 8 msec 8 msec 8 msec
  3 EXTERNAL-A-GATEWAY.STANFORD.EDU (192.42.110.225) 8 msec 4 msec 4 msec
  4 BB2.SU.BARRNET.NET (131.119.254.6) 8 msec 8 msec 8 msec
  5 SU.ARC.BARRNET.NET (131.119.3.8) 12 msec 12 msec 8 msec
  6 MOFFETT-FLD-MB.in.MIL (192.52.195.1) 216 msec 120 msec 132 msec
  7 ABA.NYC.mil (26.0.0.73) 412 msec 628 msec 664 msec
```

Sample Session 2:

Following is an example of going through the extended dialog of the **trace** command.

```
chaos#trace
Protocol [ip]:
Target IP address: mit.edu
Source address:
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to MIT.EDU (18.72.2.1)
  1 DEBRIS.CISCO.COM (131.108.1.6) 1000 msec 4 msec 4 msec
  2 BARRNET-GW.CISCO.COM (131.108.16.2) 16 msec 4 msec 4 msec
  3 EXTERNAL-A-GATEWAY.STANFORD.EDU (192.42.110.225) 16 msec 4 msec 4 msec
  4 NSS13.BARRNET.NET (131.119.254.240) 112 msec 8 msec 8 msec
  5 SALT_LAKE_CITY.UT.NSS.NSF.NET (129.140.79.13) 72 msec 64 msec 72 msec
  6 ANN_ARBOR.MI.NSS.NSF.NET (129.140.81.15) 124 msec 124 msec 140 msec
  7 PRINCETON.NJ.NSS.NSF.NET (129.140.72.17) 164 msec 164 msec 172 msec
  8 ZAPHOD-GATEWAY.JVNC.NET (128.121.54.72) 172 msec 172 msec 180 msec
  9 HOTBLACK-GATEWAY.JVNC.NET (130.94.0.78) 180 msec 192 msec 176 msec
 10 CAPITAL1-GATEWAY.JVNC.NET (130.94.1.9) 280 msec 192 msec 176 msec
 11 CHEESESTEAK2-GATEWAY.JVNC.NET (130.94.33.250) 284 msec 216 msec 200 msec
 12 CHEESESTEAK1-GATEWAY.JVNC.NET (130.94.32.1) 268 msec 180 msec 176 msec
 13 BEANTOWN2-GATEWAY.JVNC.NET (130.94.27.250) 300 msec 188 msec 188 msec
 14 NEAR-GATEWAY.JVNC.NET (130.94.27.10) 288 msec 188 msec 200 msec
 15 IHTFP.MIT.EDU (192.54.222.1) 200 msec 208 msec 196 msec
 16 E40-03GW.MIT.EDU (18.68.0.11) 196 msec 200 msec 204 msec
 17 MIT.EDU (18.72.2.1) 268 msec 500 msec 200 msec
```

Debugging the IP Network

Use the EXEC commands described in this section to troubleshoot and monitor the IP network transactions. For each **debug** command there is an corresponding **undebug** command that turns the display off. In general, you need use these commands only during troubleshooting sessions with Cisco personnel, as display of debugging messages can impact the operation of the router.

debug arp

The **debug arp** command enables logging of ARP and Probe protocol transactions.

debug ip-icmp

The **debug ip-icmp** command enables logging of ICMP transactions. Refer to the ICMP section for an in-depth look at the various ICMP messages.

debug ip-packet [*list*]

The **debug ip-packet** command enables logging of general IP debugging information as well as IPSO security transactions. IP debugging information includes packets received, generated, and forwarded. This command can also be used to debug IPSO security-related problems. Each time a datagram fails a security test in the system, a message is logged describing the cause of failure. An optional IP access *list* may be specified. If the datagram is not permitted by that access list, then the related debugging output is suppressed.

debug ip-routing

The **debug ip-routing** command enables logging of routing table events such as network appearances and disappearances.

debug ip tcp

The **debug ip tcp** command enables logging of significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug ip-tcp-packet *list*

The **debug ip-tcp-packet** command enables logging of each TCP packet that meets the permit criteria specified in the access list.

debug ip-udp

The **debug ip-udp** command enables logging of UDP-based transactions.

debug probe

Debugging information, including information about HP Probe Proxy Requests, is available through **debug probe**.

debug ip-tcp-header-compression

The **debug ip-header-compression** command enables logging of TCP header compression statistics.

IP Global Configuration Command Summary

This section lists and summarizes all the commands you can use to configure your IP router. Commands are listed in alphabetical order.

[no] access-list *list* {**permit** | **deny**} *address wildcard-mask*

Creates or removes an access list. The argument *list* is an IP list number from 1 to 99. The keywords **permit** and **deny** specify the security action to take. The argument *address* is a 32-bit, dotted decimal notation IP address to which the router compares the address being tested. The argument *wildcard-mask* are wildcard mask bits for the address in 32-bit, dotted decimal notation.

[no] access-list *list* {**permit** | **deny**} *protocol source source-mask destination destination-mask* [*operator operand*] [**established**]

Creates or removes an extended access list. The argument *list* is an IP list number from 100 to 199. The keywords **permit** and **deny** specify the security action to take. The argument **protocol** is one of the supported protocol keywords—**ip**, **tcp**, **udp**, **icmp**. The argument *source* is a 32-bit, dotted decimal notation IP address. The argument *source-mask* are mask bits for the source address in 32-bit, dotted decimal notation. The arguments *destination* and *destination-mask* the destination address and mask bits for the destination address in 32-bit, dotted decimal notation. Using TCP and UDP, the optional arguments *operator* and *operand* can be used to compare destination ports, service access points, or contact names. The optional **established** keyword is for use in matching certain TCP datagrams (see "Configuring Extended Access Lists").

[no] arp *internet-address hardware-address type* [**alias**]

Installs a permanent entry in the ARP cache. The router uses this entry to translate 32-bit Internet Protocol addresses into 48-bit hardware addresses. The argument *internet-address* is the Internet address in dotted decimal format corresponding to the local data link address specified by the argument *hardware-address*. The argument *type* is an encapsulation description—**arpa** for Ethernets; **snap** for FDDI and Token Ring interfaces; **ultra** for the Ultranet interfaces. The optional keyword **alias** indicates that the router should respond to ARP requests as if it were the owner of the specified IP address.

[no] ip domain-list *name*

Defines a list of default domain names to complete unqualified host name. The argument *name* is the domain name.

[no] ip domain-name *name*

Defines the default domain name, which is specified by the argument name. The router uses the default domain name to complete unqualified domain names—names without a dotted domain name.

[no] ip accounting-list *ip-address mask*

Specifies a set of filter to control the hosts for which IP accounting information is kept. The source and destination address of each IP datagram is logically ANDed with the *mask* and compared with *ip-address*. If there is a match, the information about the IP datagram will be entered into the accounting database. If there is no match, then the IP datagram is considered a transit datagram and will be counted according to the setting of the **ip accounting-transits** command.

[no] ip accounting-threshold *threshold*

Sets the maximum number of accounting entries to be created.

[no] ip accounting-transits *count*

Controls the number of transit records that will be stored in the IP accounting database. Transit entries are those that do not match any of the filters specified by **ip accounting-list** commands. If no filters are defined, no transit entries are possible.

[no] ip default-network *network*

Flags networks as candidates for default routes. The argument network specifies the network number.

[no] ip domain-lookup

Enables or disables IP Domain Name System-based host-name-to-address translation. Enabled by default. The **no** variation of the command disables the feature.

[no] ip forward-protocol spanning-tree

Permits IP broadcasts to be flooded throughout the internetwork in a controlled fashion. This command is an extension of the **ip helper-address** command, in that the same packets that may be subject to the helper address and forwarded to a single network may now be flooded.

[no] ip host *name address*

Defines a static host-name-to-address mapping in the host cache. The argument *name* is the host name and the argument *address* is the associated IP address.

[no] ip hp-host *hostname ip-address*

Enables the use of the proxy service. You enter the *hostname* of the HP host into the host table, along with its IP address.

[no] ip ipname-lookup

Specifies or removes the IP IEN-116 Name Server host-name-to-address translation. This command is disabled by default; the **no** variation of the command restores the default.

[no] ip name-server *address*

Specifies the address of name server to use for name and address resolution. By default, the router uses the all-ones broadcast address (*255.255.255.255*).

[no] ip routing

Controls the system's ability to do IP routing. If the system is running optional bridging-enabled software, the **no ip routing** subcommand will turn off IP routing when setting up a system to bridge (as opposed to route) IP datagrams. The default setting is to perform IP routing.

[no] ip source-route

Controls the handling of IP datagrams with source routing header options. The default behavior is to perform the source routing. The **no** keyword causes the system to discard any IP datagram containing a source-route option.

[no] ip subnet-zero

Enables or disables the ability to configure and route to "subnet zero" subnets. The default condition is disabled.

IP Interface Subcommand Summary

This section lists and summarizes all the commands in the interface subcommand list for your IP router. Preceding any of these commands with a **no** keyword undoes their effect or restores the default condition. Commands are listed in alphabetical order.

[no] arp {**arpa** | **probe** | **snap**}

Controls the interface-specific handling of IP address resolution into 48-bit Ethernet, FDDI, and Token Ring hardware addresses. The keyword **arpa**, which is the default, specifies standard Ethernet style ARP (RFC 826), **probe** specifies the HP-proprietary Probe protocol for IEEE-802.3 networks, and **snap** specifies ARP packets conforming to RFC 1042.

[no] arp timeout *seconds*

Sets the number of seconds an ARP cache entry will stay in the cache. The value of the argument *seconds* is used to age an ARP cache entry related to that interface, and by default is set to14,400 seconds. A value of zero seconds sets no timeout. The **no** form of the command returns the default.

[no] access-group *list*

Defines an access group. This subcommand takes a standard or extended IP access list number as an argument.

[no] ip accounting

Enables or disables IP accounting on an interface.

[no] ip address *address subnet-mask* [**secondary**]

Sets an IP address for an interface. The two required arguments are an IP address and the subnet mask for the associated IP network. The subnet mask must be the same for all interfaces connected to subnets of the same network.

[no] ip broadcast-address *address*

Defines a broadcast address. The *address* argument is the desired IP broadcast address for a network. If a broadcast address is not specified, the system will default to a broadcast address of all ones or *255.255.255.255*

[no] ip directed-broadcast

Enables or disables forwarding of directed broadcasts on the interface. The default is to forward directed broadcasts.

[no] ip forward-protocol {udp | nd} *[port]*

Controls forwarding of physical and directed IP broadcasts. This command controls which protocols and ports are forwarded for an interface on which an **ip helper-address** command has been specified. The keyword **nd** is the ND protocol used by older diskless SUN workstations. The keyword **udp** is the UDP protocol. By default both UDP and ND forwarding are enabled if a helper address has been given for an interface.

[no] ip helper-address *address*

Defines a helper-address for a specified address. The helper-address defines the selective forwarding of UDP broadcasts, including BootP, received on the interface. The *address* argument specifies a destination broadcast or host address to be used when forwarding such datagrams.

[no] ip mask-reply

Sets the interface to send ICMP *Mask Reply* messages. The default is not to send *Mask Reply* messages.

[no] ip mtu *bytes*

Sets the maximum transmission unit (MTU) or size of IP packets sent on an interface. The argument *bytes* is the number of bytes with a minimum of 128 bytes. The **no** form of the command restores the default.

[no] ip probe proxy

Enables or disables HP Probe Proxy support, which allows a router to respond to HP Probe Proxy Name requests.

[no] ip proxy-arp

Enables or disables proxy ARP on the interface. The default is to perform proxy ARP.

[no] ip redirects

Disables sending ICMP redirects on the interface. ICMP redirects are normally sent.

[no] ip route-cache

Controls the use of outgoing packets on a high-speed switching cache for IP routing. The cache is enabled by default and allows load-balancing on a per-destination basis. To enable load-balancing on a per-packet basis, use the **no ip route-cache** to disable fast-switching.

[no] ip security *arguments*

Controls the use of the Internet IP Security Option.

[no] ip security add

Adds a basic security option to all datagrams leaving the router on the specified interface. The **no** form of the command disables the function.

ip security dedicated *level authority* [*authority...*]

Sets or unsets the requested level of classification and authority on the interface. See Tables 13-4 and 13-5 for the *level* and *authority* arguments.

[no] ip security extended-allowed

Allows or rejects datagrams with an extended security option on the specified interface.

[no] ip security ignore-authorities

Sets or unsets an interface to ignore the authority fields of all incoming datagrams.

[no] ip security implicit-labelling [*level authority* [*authority...*]]

In the simplest form, sets or unsets the interface to accept datagrams, even if they do not include a security option. With the arguments level and authority, a more precise condition is set. See Tables 13-4 and 13-5 for the *level* and *authority* arguments.

ip security multilevel *level1* [*authority...*] **to** *level2 authority2* [*authority2...*]

Sets or unsets the requested range of classification and authority on the interface. Traffic entering or leaving the system must have a security option that falls within the specified range. See Tables 13-4 and 13-5 for the *level* and *authority* arguments.

[no] ip security strip

Removes any basic security option on all datagrams leaving the router on the specified interface. The **no** form of the command disables the function.

[no] ip tcp compression-connections *number*

Sets the maximum number of connections per interface that the compression cache can support. Default is 16; *number* can vary from 3 to 256.

[no] ip tcp header-compression [passive]

Enables TCP header compression. The **no** keyword disables (the default) compression. The optional keyword **passive** sets the interface to only compress outgoing traffic on the interface for a specific destination if incoming traffic is compressed.

[no] ip unnumbered *interface-name*

Enables IP processing on a serial interface, but does not assign an explicit IP address to the interface. The argument *interface-name* is the name of another interface on which the router has assigned an IP address. The interface may not be another unnumbered interface, or the interface itself.

[no] ip unreachables

Enables or disables (with the no argument) sending ICMP unreachable messages on an interface. ICMP unreachables are normally sent.

transmit-interface *interface-name*

Assigns a transmit interface to a receive-only interface. When a route is learned on this receive-only interface, the interface designated as the source of the route is converted to *interface-name.*

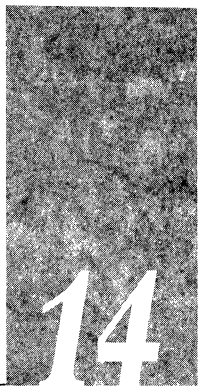
IP Line Subcommand Summary

This section contains a list of the line subcommands used to configure IP routing.

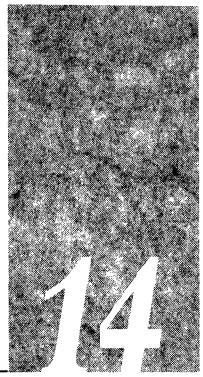
[no] access-class *list* {**in**|**out**}

Restricts incoming and outgoing connections between a particular virtual terminal line and the addresses in an access list. Serves to restrict connections on a line or group of lines to certain Internet addresses. The argument *list* is an integer from 1 through 99 that identifies a specific access list of Internet addresses. The keyword **in** applies to incoming connections; the keyword **out** applies to outgoing Telnet connections.

Chapter 14
The IP Routing Protocols



Chapter 14
The IP Routing Protocols



This chapter describes routing protocol options for the Internet Protocol (IP) suite. Chapter 13, "Routing IP," contains all the information you need for configuring IP. This chapter focuses on IP routing protocols. Other protocol stacks—DECnet, Novell, Apollo, and so on—are described in their own chapters. Topics in this chapter include:

- An introduction to the IP routing protocols
- Starting the routing process for a particular protocol
- Configuring static and dynamic routing
- Configuring the supported IP protocols—IGRP, RIP, HELLO, EGP, and BGP.
- Configuring multiprotocol operations, including redistribution of information from one protocol to another
- Filtering incoming and outgoing updates on the interface

Cisco-Supported Routing Protocols

Routing is the process of determining where to send data packets destined for addresses outside the local network. Routers gather and maintain routing information to enable the transmission and receipt of such data packets. Conceptually, routing information takes the form of entries in a routing table, with one entry for each identified route. The router can create and maintain the routing table dynamically to accommodate network changes whenever they occur.

Note: It is traditional when discussing IP routing protocols to refer to routers as *gateways*. For this reason, many IP routing protocols contain the word *gateway* as part of their name. Keep in mind that a gateway is a generic layer 3 and above device that connects one software stack to another. So an X.25 gateway, an electronic mail (layer 7) gateway and a router are all—in a computer science sense—gateways.

Interior and Exterior Protocols

The routing protocols are broadly divided into two classes, interior gateway protocols (IGPs), and exterior gateway protocols (EGPs). The interior routing protocols supported by Cisco include the Routing Information Protocol (RIP), HELLO, and the Interior Gateway Routing Protocol (IGRP). Interior protocols are used for routing networks that are under a common network administration. The exterior routing protocols include the Exterior Gateway Protocol (EGP) and the Border Gateway Protocol (BGP). Exterior protocols are used to exchange routing information between networks that do not share a common administration.

- IGRP, developed by Cisco Systems, focuses on large networks with complex topology and segments having different bandwidth and delay characteristics.
- RIP is the routing protocol used by the routed process on Berkeley-derived UNIX systems. Many networks use RIP; it works well for small, isolated, and topologically simple networks.
- HELLO is an older interior routing protocol used in the early National Science Foundation (NSF) backbone network.
- EGP is the original exterior protocol and is still used primarily in the DDN (Defense Data Network) and NSFnet (National Science Foundation Network).
- BGP is a more recent exterior routing protocol that solves some of EGP's failings.

The Cisco routers offer many protocol-independent routing features. For example, subnetting lets you divide a network into logical subparts. Load-balancing lets you split network traffic over parallel paths, which provides greater overall throughput and reliability. Because the router can avoid routing loops, you can implement general network topologies. Notification of disabled interfaces eliminates network *black holes*. Static routing table entries can provide routing information when dynamically obtained entries are not available. Most protocol-independent routing capabilities are discussed in Chapter 13, "Routing IP." This chapter focuses on the routing protocols themselves.

Autonomous Systems

An autonomous system (AS) is a collection of networks under a common administration sharing a common routing strategy (see Figure 14-1). An autonomous system may comprise one or many networks, and each network may or may not have an internal structure (subnetting). The autonomous system number, which is assigned by the DDN Network Information Center, is a 16-bit decimal number that uniquely identifies the autonomous system. An assigned AS number is required when running EGP or BGP. All routers that belong to an autonomous system must be configured with the same autonomous system number.

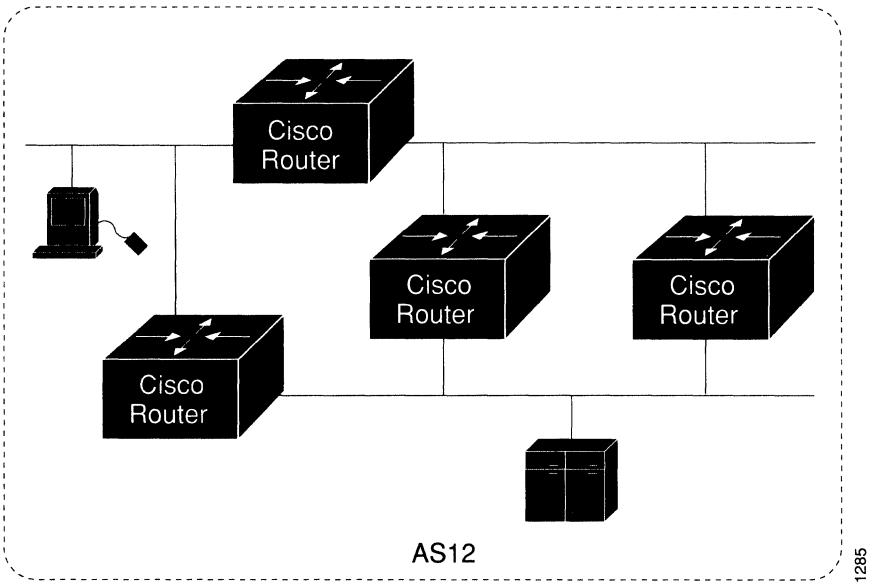


Figure 14-1 Autonomous System 12 Contains Four Routers

Multiple Routing Protocols

The multiple routing protocol support in the Cisco routers was designed for connecting networks that might be using different routing protocols. It is possible, for example, to run RIP on one subnetted network, IGRP on another subnetted network, and to exchange the routing information in a controlled fashion. The routing protocols available today were not designed to interoperate with one another, so each protocol collects different types of information and reacts to topology changes in its own way. For example, RIP uses a hop count metric and IGRP uses a five-valued vector of metric information. In the case where routing information is being exchanged between different networks that use different routing protocols, there are many configuration options to enable and to filter the exchange of routing information. See the section "Redistributing Routing Information" later in this chapter.

Configuration Overview

Each routing protocol must be configured separately. Because of this need, we have separated most of the configuration information into protocol-specific subsections. The interior routing protocols will be listed first, followed by the exterior protocols. With any routing protocol, you must follow these basic steps:

Step 1: Create the routing process with one of the **router** commands.

Step 2: Configure the protocol specifics.

The next sections provide a review of the two protocol classes and how they are configured, followed by sections that explain how to configure each of the routing protocols. EXEC-level commands for monitoring the IP routing operations are also provided, and these are explained at the end of this chapter, along with alphabetical summaries of the configuration commands.

Configuring the Interior Routing Protocols

The interior routing protocols IGRP, RIP, and HELLO must have a list of networks specified by the **network** router subcommand before routing activities can begin. The routing process will listen to updates from other routers on these networks and will broadcast its own routing information on those same networks. The IGRP routing protocol has the additional requirement of an autonomous system number, usually assigned by the DDN NIC.

Configuring the Exterior Routing Protocols

The exterior routing protocols require three sets of information before routing can begin:

- A list of neighbor (or peer) routers with which to exchange routing information. This list is created with the **neighbor** router subcommand.
- A list of networks to advertise as directly reachable, created with the **network** router subcommand.
- The AS number of the local router.

The following sections in this chapter describe the protocols, beginning with the interior routing protocols: IGRP, RIP, and HELLO.

Configuring the IGRP Protocol

Cisco Systems designed the Interior Gateway Routing Protocol (IGRP) for routing in an autonomous system having arbitrarily complex topology and consisting of media with diverse bandwidth and delay characteristics. The IGRP protocol advertises all connected and IGRP-derived networks for a particular autonomous system. A single router can service up to four autonomous systems and keep the routing information for each system separate.

Interior, System, and Exterior Routes

IGRP advertises three types of routes: interior, system, and exterior as shown in Figure 14-2. Interior routes are routes between subnets in the network attached to a router interface. If the network attached to a router is not subnetted, IGRP does not advertise interior routes.

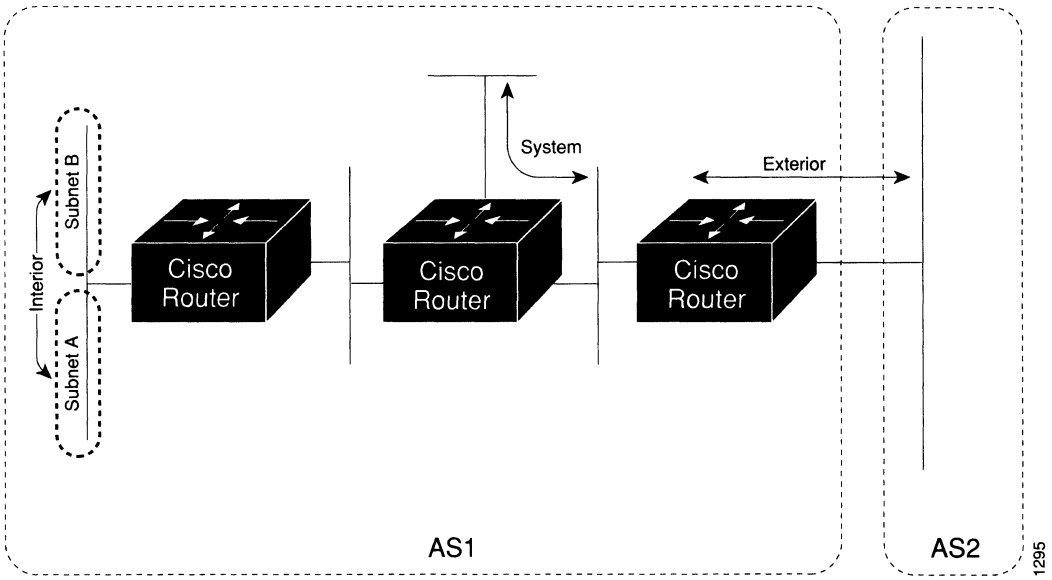


Figure 14-2 Interior, System, and Exterior Routes

System routes are routes to networks within the autonomous system. The router derives system routes from directly connected network interfaces and system route information provided by other IGRP-speaking routers. System routes do not include subnetting information. Exterior routes are routes to networks outside the autonomous system.

Creating the IGRP Routing Process

To create the routing process, use the **router** global configuration command. The full syntax of this command follows:

router igrp *autonomous-system*

no router igrp *autonomous-system*

The argument *autonomous-system* identifies the routes to other IGRP routers, and is used to tag the routing information passed along.

Use the **no router igrp** command to shut down the routing process on the AS specified by the *autonomous-system* argument.

Next, specify the list of networks with the **network** router configuration subcommand. The full syntax of this command is listed below.

network *network-number*

no network *network-number*

The argument *network-number* is a network number in dotted IP notation. Note that this number must *not* contain subnet information. You may specify multiple **network** subcommands.

When an IGRP routing process is configured, an AS number must be specified. This number may or may not be assigned by the DDN NIC. The AS number is used to tag updates belonging to one of up to four IGRP routing processes.

Use the **no network** command with the network number to remove a network from the list.

Example:

In this example, a router is configured for IGRP and assigned to AS 109. In the next two lines, two network commands assign the two networks to a list of networks to receive IGRP updates, as shown.

```
router igrp 109
network 131.108.0.0
network 192.31.7.0
```

Choosing the Gateway of Last Resort

The router chooses a *gateway of last resort* from the list of exterior routers that IGRP provides. The router uses the gateway (router) of last resort, if it does not have a better route for a packet. If the autonomous system has more than one connection to an external network, different routers may choose different exterior routers as the gateway of last resort.

IGRP Metric Information

IGRP uses several types of metric information. For each path through an autonomous system, IGRP records the segment with the lowest bandwidth, the accumulated delay, the smallest MTU (Maximum Transmission Unit), and the reliability and load.

The IGRP metric is a 32-bit quantity that is a sum of the segment delays and the lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (FDDI, Ethernets, and serial lines running from 9,600 baud to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

IGRP Updates

A router running IGRP sends an IGRP update broadcast every 90 seconds. It declares a route inaccessible if it does not receive an update from the first router in the route within three update periods (270 seconds). After five update periods (450 seconds), the router removes the route from the routing table. IGRP uses flash update and poison reverse to speed up the convergence of the routing algorithm.

Configuring the RIP Protocol

The Routing Information Protocol (RIP) uses broadcast User Datagram Protocol (UDP) data packets to exchange routing information. Each router sends routing information updates every 30 seconds; this process is termed *advertising.* If a router does not receive an update from another router for 90 seconds or more, it marks the routes served by the nonupdating router as being unusable. If there is still no update after 450 seconds, the router removes all routing table entries for the nonupdating router.

The measure, or metric, that RIP uses to rate the value of different routes is the *hop count.* The hop count is the number of routers that may be traversed in a route. A directly connected network has a metric of zero (see Figure 14-3); an unreachable network has a metric of 16. This small range of metrics makes RIP unsuitable as a routing protocol for wide area networks. If the router has a default network path, RIP advertises a route that links the router to the pseudo-network *0.0.0.0.* The network *0.0.0.0* does not exist; RIP treats *0.0.0.0* as a network to implement the default routing feature.

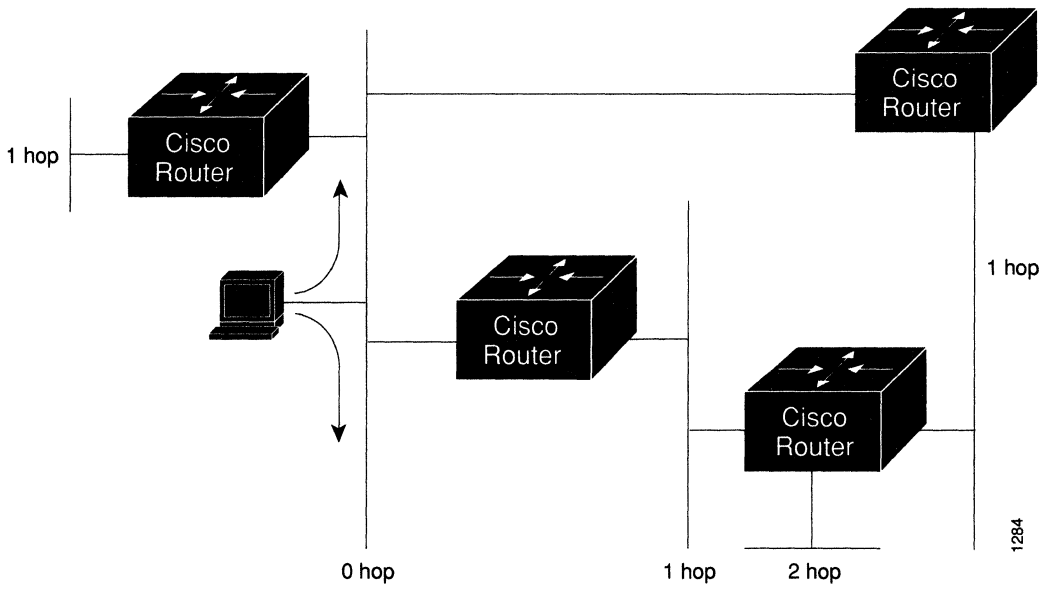


Figure 14-3 Hop Count in RIP

Creating the RIP Routing Process

To create a routing process for RIP, use the **router rip** global configuration command:

router rip

no router rip

Use the **no router rip** command to shut down the routing process.

Specifying the List of Networks

Next, specify the list of networks with the **network** router configuration subcommand. The full syntax of this command follows.

network *network-number*

no network *network-number*

The argument *network-number* is a network number in dotted IP notation. Note that this number must *not* contain subnet information. You may specify multiple **network** subcommands. RIP routing updates will be sent and received only through interfaces on this network.

Example:

The following example configuration defines RIP as the routing protocol to be used on all interfaces connected to networks *128.88.0.0* and *192.31.7.0*.

```
router rip
network 128.99.0.0
network 192.31.7.0
```

To remove a network from the list, use the **no network** router subcommand followed by the network address.

Configuring the HELLO Protocol

The HELLO protocol, described in RFC 891, was developed for the Fuzzball gateways of the Distributed Computer Network project and was used extensively in the early NSFnet backbone network. HELLO is an interior routing protocol.

The Cisco Systems implementation of HELLO does not implement the extensive timekeeping and delay measurement features. Specifically, the router sets the invalid bit in the HELLO date field and clears the time and timestamp fields.

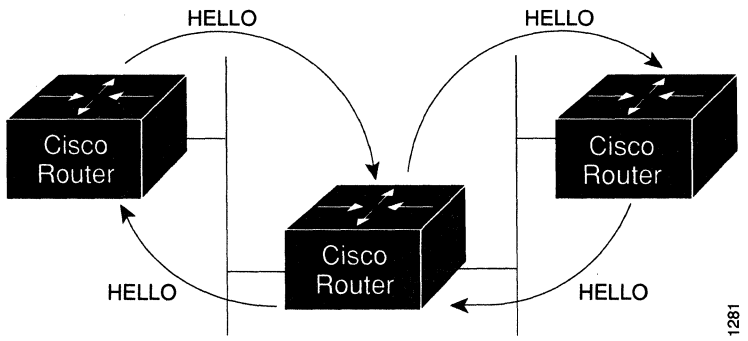


Figure 14-4 The HELLO Protocol

The metric used in HELLO is a delay value measured in milliseconds (see Figure 14-2). This metric can range from 0 to 30,000 milliseconds, making HELLO a good candidate for routing larger networks. A network with a 30,000-millisecond delay is considered unreachable. The Cisco Systems implementation uses a delay of 100 milliseconds for all routes, regardless of their actual delay characteristics.

Creating the HELLO Routing Process

The first step is to create the routing process with the **router** global configuration command:

router hello

no router hello

Use the **no router hello** command to shut down the routing process.

Specifying the List of Networks

The next step is to specify the list of networks. This list is specified with the **network** router configuration subcommand:

network *network-number*

no network *network-number*

The argument *network-number* is a network number in dotted IP notation. Note that this number must not contain subnet information. You can specify multiple **network** subcommands.

Example:

In the following example, the network *160.1.1.0* is being set up for HELLO.

```
router hello
network 160.1.1.0
```

Configuring the BGP Protocol

BGP, as defined in RFC 1163 and RFC 1164, allows you to set up a distributed routing core that automatically guarantees the loop-free exchange of routing information on an autonomous system (AS) basis.

Creating the BGP Routing Process

To configure BGP, use the **router bgp** global configuration command:

router bgp *autonomous-system*

no router bgp *autonomous-system*

The *autonomous-system* number is used to identify the router to other BGP routers, and to tag the routing information passed along.

Example:

In the following example, a router is assigned to AS 120.

```
router bgp 120
```

Specifying the List of Networks

Use the **network** router subcommand to specify those networks that are to be advertised as originating within the current AS. These networks can be learned from connected, dynamic routing, and static route sources. The command syntax follows:

network *network-number*

The argument *network-number* is the dotted IP address of the network that will be included in the router's BGP updates.

Example:

In the following command, the network *131.108.0.0* is set up to be included in the routers BGP updates.

```
network 131.108.0.0
```

Specifying the List of Neighbors

BGP supports two different kinds of neighbors: internal and external. Internal neighbors are in the same autonomous system. External neighbors are in other autonomous systems.

BGP routing includes several related router subcommands that specify "neighbor" routers and manage your router's relationship with its neighbors. Use the **neighbor** router subcommands to:

- Identify BGP peers and their AS numbers.
- Assign access lists.
- Set up various routing policies.

Basic Neighbor Specification

In the simplest case, you simply want to specify that another router is a neighbor. Use the simple neighbor command, as shown below:

neighbor *address* **remote-as** *autonomous-system*

no neighbor *address* **remote-as** *autonomous-system*

Using the **no** keyword removes the router as a neighbor.The arguments *address* and *autonomous-system* are the neighbor's address and AS number.

Example 1:

This example specifies that the router at the address *131.108.0.0* is a neighbor.

```
neighbor 131.108.1.2 remote-as 109
```

Example 2:

In the following example, a BGP router is assigned to AS 109, and two networks are listed as originating in the AS.Then the addresses of two remote routers (and their ASes) are listed. The router being configured will share information about networks *131.108.0.0* and *192.31.7.0* with the two neighbor routers, as shown:

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.240.67  remote-as  99
```

Routing Weights

The **neighbor weight** router subcommand specifies a weight to assign to a specific neighbor connection. Its full syntax is as follows:

neighbor *address* **weight** *weight*

no neighbor *address* **weight** *weight*

The argument *address* is the address of the neighbor connection.The argument *weight* is the weight value to assign.The route with the highest weight will be chosen as the preferred route when multiple routes are available to a particular network.

Use the **no neighbor** command with the appropriate arguments and keywords to remove this function.

Example:

In the example below, the neighbor at address *151.23.12.1* is assigned a weight of 50.

```
neighbor 151.23.12.1 weight 50
```

Filtering BGP Advertisements

You can filter BGP advertisements in two ways: by using access lists and by using AS-path filters. Access lists in IP are discussed in Chapter 13.

You can apply access lists to BGP updates with the **neighbor distribute-list** router subcommand. Its full syntax follows.

neighbor *address* **distribute-list** *list* {**in** | **out**}

no neighbor *address* **distribute-list** *list*

The argument *address* is the address of the neighbor connection. The argument *list* is a predefined access list number. The keywords **in** and **out** specify whether you are applying the access list to incoming or outgoing advertisements to that neighbor.

Use the **no neighbor** command with the appropriate arguments and keywords to remove this function.

Example:

In the example below, list 41 is applied to outgoing advertisements to neighbor *120.23.4.1.*

```
neighbor 120.23.4.1 distribute-list 41 out
```

You can also filter neighbor updates coming from specific neighbors by AS number using these variations of the **neighbor** command.

neighbor *address* **filter-as** *number* **deny**
no neighbor *address* **filter-as** *number* **deny**

neighbor *address* **filter-as** *number* **permit** *weight*
no neighbor *address* **filter-as** *number* **permit** *weight*

The **permit** keyword assigns a weight to routes that include the specified AS in the path. The **deny** keyword causes the information learned about that network using the specified AS to be ignored.

Example:

This example assigns the weight 60 to all routes learned from the neighbor at address *120.23.4.1* which traverse AS 20.

```
neighbor 120.23.4.1 filter-as 20 permit 60
```

Adjusting the BGP Timers

To adjust the default BGP timers, use the **timers bgp** router subcommand. The full syntax of this command follows.

timers bgp *keepalive holdtime*

no timers bgp

The argument *keepalive* is the frequency in seconds with which the router sends *keepalive* messages to its peer (default 60 seconds), and *holdtime* is the interval in seconds after not receiving a *keepalive* message that the router declares a peer dead (default 180 seconds). The **no timers bgp** command restores the default.

Example:

In the example below, the keepalive timer is changed to 70 seconds and the holdtime is changed to 210 seconds.

```
timers bgp 70 210
```

BGP and IGP Routing Information

This section discusses the issues of BGP interacting with the various interior gateway protocols (referred to generically as IGPs), such as IGRP, RIP, and HELLO. BGP maintains its own routing table, separate from the main IP routing table used to make datagram switching decisions. The BGP routing table is organized by network, and contains information referred to as *attributes,* such as the list of ASs that a datagram must transit to reach a particular network. Information from the BGP routing table is periodically entered into the main IP routing table and is aged appropriately (see Figure 14-5). In most cases, the BGP information should override IGP information.

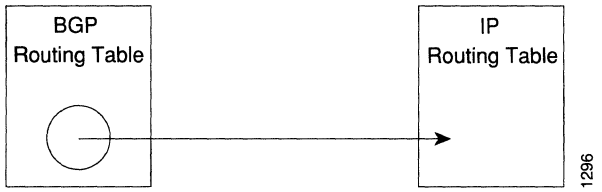


Figure 14-5 BGP and IGP Routing

Networks that originate in the local AS are indicated with the **network** router subcommand for the BGP process. Such networks, referred to as local networks, will have a BGP origin attribute of IGP. They appear in the main IP routing table and may have any source, for example, directly connected, static route, learned from an IGP, and so forth. The BGP routing process periodically scans the main IP routing table to detect the presence or absence of local networks, updating the BGP routing table as appropriate.

Since all networks originating within an AS are flagged by the **network** command, information from an IGP about a nonlocal network that conflicts with external BGP information will be suppressed on the grounds that BGP is supplying better information. It is possible, however, to indicate which networks are reachable using a back-door route that the border router should use instead. Back-door networks are permitted in the main IP routing table,

but are not entered into the BGP routing table, except when they are learned from another BGP-speaking router.

Use this variation of the **network** router subcommand to specify a back-door route:

network *address* **backdoor**

The argument *address* is the network that you wish to set up a back-door route to.

Example:

In the example below, network *131.108.0.0* is a local network and network *192.31.7.0* is a backdoor network.

```
router bgp 109
network 131.108.0.0
network 192.31.7.0  backdoor
```

Using the **redistribute** router subcommand, you can inject BGP routing information into the IGP. This creates a situation where BGP is potentially deriving information about local networks from the IGP, and then sending such information back into the IGP. This is another reason to suppress nonlocal networks from the IP routing table—you do not want to hear echoes of your routing updates.

It is also possible to inject IP routing table information into the BGP routing table using the **redistribute** router subcommand. EGP-derived information will have a BGP origin attribute of EGP; all other nonlocal routes will have a BGP origin attribute of incomplete. All IGP and EGP information will now override BGP-derived IP routing table entries. If you are also redistributing information from BGP into an IGP, you must set up appropriate filtering using the **distribute-list** command to ensure that routing information does not loop. A configuration such as this is fairly risky, requiring careful attention to filtering. Filtering is described in more detail later in this chapter.

BGP Route Selection Rules

The BGP process selects a single AS path to pass along to other BGP-speaking routers. It is important for routing stability that each BGP-speaking router in an AS use the same set of rules so that all BGP-speaking routers arrive at a consistent view of the AS topology. To this end, the Cisco BGP implementation has a reasonable set of factory defaults that may be overridden by administrative configuration, as follows:

- An AS path for a network sourced by this BGP-speaking router has the highest preference. The Cisco router uses the sourced path with the lowest origin code.
- Administrative weighting is then considered. Larger weight takes precedence.
- Prefer the shorter AS path. All succeeding rules assume equal length paths.
- Prefer external links over internal links.
- Prefer the lowest origin code (IGP <EGP <INCOMPLETE).
- If INTER_AS metric attributes are present, prefer path with lowest metric.
- Final determinant is the peer with the largest value for the IP address.

BGP Path Attributes

The Cisco BGP implementation supports all path attributes defined in RFC 1163. This section describes some details of that implementation.

The **default-metric** router subcommand may be used to configure the value for the INTER_AS metric attribute. The same metric value will be sent with all BGP updates originating from the router. The default is to not include an INTER_AS metric in BGP updates.

The Cisco implementation will use a third party next hop router address in NEXT_HOP attribute regardless of the AS of that third party router.

Transitive, optional path attributes are passed along to other BGP-speaking routers. The Cisco BGP implementation does not currently generate such attributes.

Configuring the EGP Protocol

The Exterior Gateway Protocol (EGP), specified in RFC 904, is used for communicating with certain routers in the Defense Data Network (DDN) that the U.S. Department of Defense designates as core routers. EGP is also extensively used when attaching to the NSFnet (National Science Foundation Network) and other large backbone networks as shown in Figure 14-6. An exterior router uses EGP to advertise its knowledge of routes to networks within its autonomous system. It sends these advertisements to the core routers, which then re-advertise their collected routing information to the exterior router. A neighbor or peer router is any router with which the router communicates using EGP.

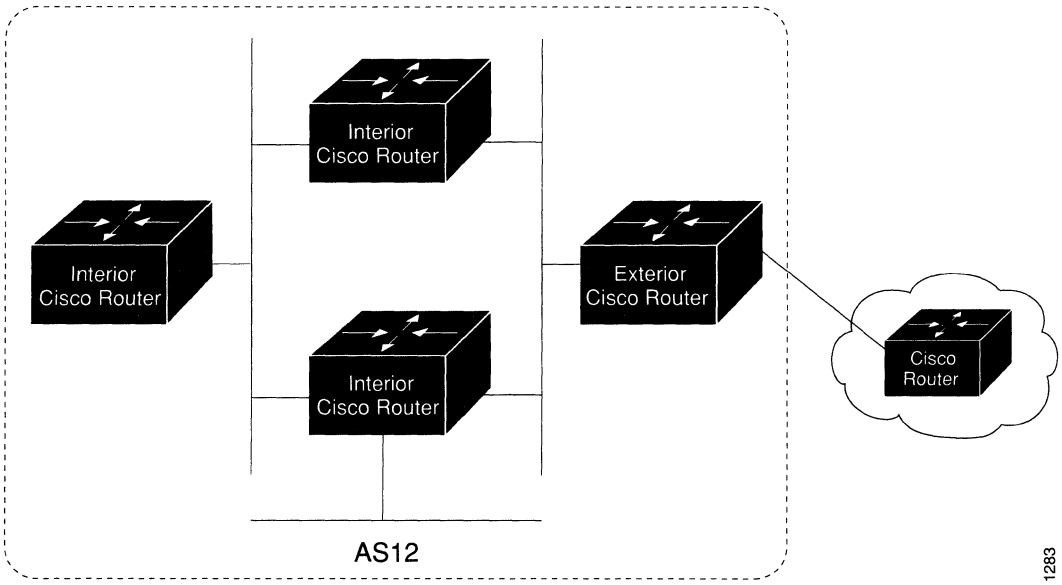


Figure 14-6 EGP and Interior and Exterior Routers

Specifying the Autonomous System Number

Before you can set up EGP routing, you must specify an autonomous system number using the **autonomous-system** global configuration command. The syntax for this command follows:

autonomous-system *local-AS*

no autonomous-system *local-AS*

The argument *local-AS* is the local autonomous system (AS) number to which the router belongs. The local AS number will be included in EGP messages sent by the router. To remove the AS number, use the **no autonomous-system** global configuration command.

Creating the EGP Routing Process

After the local AS number has been specified, start the EGP routing process with a **router egp** global configuration command:

router egp *remote-AS*

no router egp *remote-AS*

The argument *remote-AS* is the AS number the router expects its peers to be advertising in their EGP messages. The Cisco software does not insist that the actual remote AS number match the configured remote AS numbers. (The output from **debug ip-egp** EXEC command will advise of any discrepancies, however. See the section "Debugging IP Routing" for more information.) You can create up to four different EGP routing processes. Turn off your EGP routing process with the **no router egp** subcommand.

Specifying the List of Neighbors

A router using EGP cannot dynamically determine its neighbor or peer routers. You must provide a list of neighbor routers using the **neighbor** router subcommand:

neighbor *ip-address*

no neighbor *ip-address*

The argument *ip-address* is the IP address of a peer router with which routing information will be exchanged. Multiple **neighbor** subcommands may be used to specify additional neighbors or peers. The **no neighbor** subcommand followed by an IP address removes a peer from the list.

Specifying the Network to Advertise

Use the **network** router subcommand to specify the network to be advertised to the EGP peers of an EGP routing process.

network *network-number*

no network *network-number*

The argument *network-number* is the IP address of the network. Such networks are advertised with a distance of zero. There is no restriction on the network number other than that the network must appear in the routing table. The network may be connected, may be statically configured, or may be redistributed into EGP from other routing protocols.

Multiple **network** subcommands may be used to specify additional networks. The **no network** subcommand followed by the network number removes a network from the list.

Example:

The following is an example configuration for an EGP router process. The router is in autonomous system 109 and is peering with routers in AS 164, as shown in Figure 14-7. It will advertise the networks *131.108.0.0* and *192.31.7.0* to the router in AS 164, *10.2.0.2*. The information sent and received from peer routers may be filtered in various ways, including blocking information from certain routers and suppressing the advertisement of specific routes.

```
autonomous-system 109
router egp 164
network 131.108.0.0
network 192.31.7.0
neighbor 10.2.0.2
```

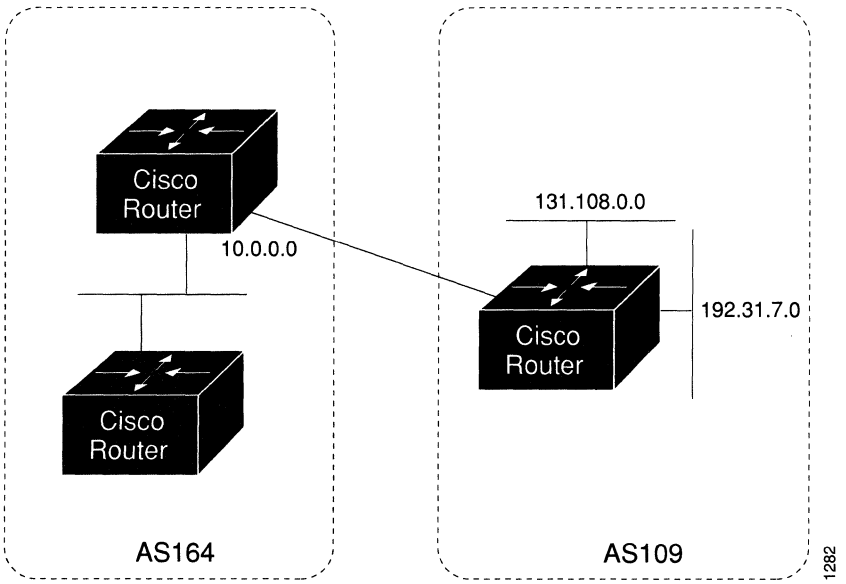


Figure 14-7 Router in AS 164 Peers with Router in AS 109

Adjusting Timers

The HELLO and polltime timers for EGP are adjustable. To adjust the EGP timers, use the subcommand:

timers egp *hello polltime*

no timers egp

The argument *hello* is the frequency in seconds with which the router sends HELLO messages to its peer. The default is 60 seconds.

The argument *polltime* is the interval in seconds after not receiving a *hello* message that the router declares a peer dead. The default is 180 seconds, and the **no timers egp** restores this default.

Example:

This command changes the EGP timers to two minutes and five minutes respectively.

```
timers egp 120 300
```

To change the invalid time or flush time for EGP routes, use the **timers basic** command as explained in the section "Special Routing Configuration Techniques" later in this chapter.

Configuring Third-Party EGP Support

EGP supports what is termed a *third-party mechanism*. In this circumstance EGP tells its peer that another router (the third party) on the shared network is the appropriate router for some set of destinations. If updates mentioning third party routers are desired, they may be configured using a variation of the **neighbor** subcommand (full syntax follows):

neighbor *address* **third-party** *third-party-ip-address* [**internal** | **external**]

no neighbor *address* **third-party** *third-party-ip-address* [**internal** | **external**]

The argument *third-party-ip-address* is the address of another router (the third party) on the network shared by the Cisco router and the EGP peer specified by the *address* argument. All networks reachable through that third party router will be listed in the Cisco EGP updates as reachable via that router. Any other networks will be listed as reachable via the Cisco router. The optional keyword **internal** or **external** indicates whether the third party router should be listed in the internal or external section of the EGP update. Normally, all networks are mentioned in the internal section. You may use the **neighbor** *address* **third-party** router subcommand multiple times to specify additional third party routers.

Example 1:

In the following example, routes learned from router *131.108.6.99* will be advertised to *131.108.6.5* as third-party internal routes.

```
neighbor 131.108.6.5  third-party 131.108.6.99 internal
```

Example 2:

In the following example, routes learned from *131.108.6.100* will be advertised to *131.108.6.5* as third-party external routes.

```
neighbor 131.108.6.5  third-party 131.108.6.100 external
```

Configuring a Backup EGP Router

It may be desirable to have a second router belonging to a different AS act as a backup to the EGP router for your AS. To differentiate between the primary and secondary EGP routers, the two routers will advertise network routes with differing EGP distances or metrics. A network with a low metric is generally favored over a network with a high metric.

Networks flagged with the **network** router subcommand are always announced with a metric of zero. Networks that are redistributed will be announced with a metric specified by the **default-metric** router subcommand. If no metric is specified, redistributed routes will be advertised with a metric of three. All redistributed networks will be advertised with the same metric. The redistributed networks may be learned from static or dynamic routes. See the section "Redistributing Routing Information" for details about the **redistribute** router subcommand. A complete configuration example is contained in the section "IP Routing Protocols Configuration Examples."

Example:

The following example configuration illustrates that networks learned by RIP are being advertised with a distance of five. (This is *not* a complete configuration.)

```
redistribute rip
default-metric 5
```

Filtering Routing Information

The information sent and received on the various networks may be filtered in various ways, including blocking information from certain routers, not sending updates onto a particular subnet, and suppressing the advertisement of specific routes. This section reviews the options for filtering incoming and outgoing information.

Filtering Outgoing Information

This section describes the options you may use to control outgoing information.

Suppressing Updates on an Interface

The **passive-interface** router subcommand disables sending routing updates on an interface.

passive-interface *interface*

no passive-interface *interface*

The argument *interface* specifies a particular interface. The particular subnet will continue to be advertised to other interfaces. Updates from other routers on that interface continue to be received and processed.

The **no passive-interface** command re-enables sending routing updates on the specified interface.

Example:

In the following example, IGRP updates are sent to all interfaces on network *131.108.0.0* except interface Ethernet 1. Figure 14-8 shows this configuration.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 1
```

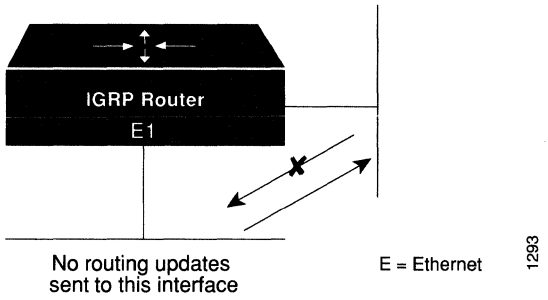


Figure 14-8 Filtering IGRP Updates

Filtering Outband Updates

To suppress networks from being sent in updates, use the **distribute-list** router subcommand. Full syntax for this command follows.

distribute-list *access-list-number* **out** [*interface-name* | *routing-process*]

no distribute-list *access-list-number* **out** [*interface-name* | *routing-process*]

The argument *access-list-number* is a standard IP access list number as described in the section "Configuring IP Access Lists" in the Chapter 13. The list explicitly specifies which networks are to be sent and which are to be suppressed.

Use the keyword **out** to apply the access list to outgoing routing updates.

When redistributing networks, a routing process name may be specified as an optional trailing argument to the **distribute-list** subcommand. This causes the access list to be applied to only those routes derived from the specified routing process. After the process-specific access list is applied, any access list specified by a **distribute-list** subcommand without a process name argument will then be applied.

Use the **no distribute-list** command with the appropriate access list number and keyword to disable or change this function.

Note: To filter networks received in updates, use the distribute-list command with the **in** keyword, as explained in the section "Filtering Received Updates."

Example 1:

The following example set of configuration subcommands would cause only two networks to be advertised by a RIP routing process, network *0.0.0.0* (the RIP default) and network *131.108.0.0.*

```
access-list 1 permit 0.0.0.0
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
router rip
network 131.108.0.0
distribute-list 1 out
```

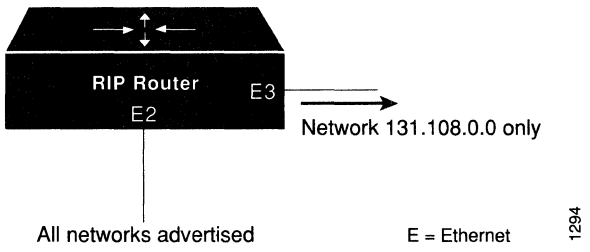


Figure 14-9 Filtering RIP Updates

Example 2:

To filter a routing update sent on a specific interface, you may, optionally, specify the interface. If the last line of the previous example were written as follows, the access list would be applied to updates sent on Ethernet 3.

```
distribute-list 1 out ethernet 3
```

Example 3:

In the following example, access list 3 is applied to networks derived from process IGRP 109 that are being redistributed by process EGP 164. Networks suppressed by that access list will not be advertised by EGP 164.

```
router egp 164
network 131.108.0.0
redistribute igrp 109
distribute-list 3 out igrp 109
```

Point-to-Point Updates

The **neighbor** router subcommand defines a neighboring router with which to exchange routing information, as discussed under the specific protocols:

neighbor *address*

The argument *address* is the neighboring router address.

For exterior routing protocols such as EGP and BGP, this command specifies routing peers. For normally broadcast protocols such as IGRP or RIP, this subcommand permits the point-to-point (nonbroadcast) exchange of routing information. When used in combination with the **passive-interface** subcommand, routing information may be exchanged between a subset of routers on a LAN.

Adjusting Metrics

The **offset-list** router subcommand can be used to add a positive offset to incoming and outgoing metrics for networks matching an access list. Full syntax for this command follows.

offset-list *list* {**in** | **out**} *offset*

no offset-list *list* {**in** | **out**}

If the argument *list* is zero, the argument supplied to *offset* is applied to all metrics. If *offset* is zero, no action is taken. For IGRP, the offset is added to the delay component only. This subcommand is implemented for the RIP and HELLO routing protocols as well.

The **no offset-list** command with the appropriate keyword removes the offset list.

Example:

In the following example, a router using IGRP applies an offset of 10 to its delay component for all outgoing metrics.

```
offset-list 0 out 10
```

In the next example, the router applies the same offset only to access list 121:

```
offset-list 121 out 10
```

Filtering Incoming Information

This section describes the options you may use to control incoming information.

Filtering Received Updates

Use the router subcommand **distribute-list** to filter networks received in updates.

distribute-list *access-list-number* **in** [*interface-name*]

no distribute-list *access-list-number* **in** [*interface-name*]

The argument *access-list-number* is a standard IP access list number as described in the section "Configuring IP Access Lists" in Chapter 13. The list explicitly specifies which networks are to be sent and which are to be suppressed.

Use the keyword **in** to suppress incoming routing updates.

The optional argument *interface-name* specifies the interface (for example, Ethernet 0) on which the access list should be applied to incoming updates. If no interface is specified, the access list will be applied to all incoming updates.

Use the **no distribute-list** command with the appropriate access list number and keyword to disable or change this function.

Example:

The following set of example configuration subcommands would cause only two networks to be accepted by a RIP routing process, network *0.0.0.0* (the RIP default) and network *131.108.0.0.*

```
access-list 1 permit 0.0.0.0
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
router rip
network 131.108.0.0
distribute-list 1 in
```

Filtering Sources of Routing Information

In a large network, some routing protocols and some routers can be more reliable than others as sources of routing information. By specifying administrative distance values, you enable the router to intelligently discriminate between sources of routing information.

An administrative distance is a rating of the trustworthiness of a routing information source, such as an individual router or a group of routers. Numerically, an administrative distance is an integer between 0 and 255. In general, the higher the value, the lower the trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored.

The router always uses the best routing source available: the routing source with the lowest administrative distance. For example, consider a router using IGRP and RIP. Suppose you

trust the IGRP-derived routing information more than the RIP-derived routing information. If you set the administrative distances accordingly, the router uses the IGRP-derived information and ignores the RIP-derived information. However, if you lose the source of the IGRP-derived information (say, to a power shutdown in another building), the router uses the RIP-derived information until the IGRP-derived information reappears.

You can also use administrative distance to rate the routing information from routers running the same routing protocol. This application is generally discouraged, however, since it can result in inconsistent routing information including forwarding loops. Example 1, below, shows how to do this safely.

To define an administrative distance, use the **distance** router subcommand.

distance *weight* [[*ip-source-address ip-address-mask*] [*access-list-number*]]

no distance *weight* [[*ip-source-address ip-address-mask*] [*access-list-number*]]

The argument *weight* is an integer from 10 to 255 that specifies the administrative distance. (Values 0 through 9 are reserved for internal use.) Used alone, the argument *weight* specifies a default administrative distance that the router uses when no other specification exists for a routing information source. Weight values are subjective; there is no quantitative method for choosing weight values.

The optional argument pair *ip-source-address* and *ip-address-mask* specifies a particular router or group of routers to which the weight value applies. The argument *ip-source-address* is an Internet address that specifies a router, network, or subnet. The argument *ip-address-mask* (in dotted-decimal format) specifies which bits, if any, to ignore in the address value; a set bit in the *mask* argument instructs the router to ignore the corresponding bit in the address value. The optional argument *access-list-number* is the number of a standard IP access list. When used, it will apply the access list number when a network is being inserted into the routing table. This allows filtering of networks according to the IP address of the router supplying the routing information. This could be used, as an example, to filter out possibly incorrect routing information from routers not under your administrative control.

To remove an administrative distance value, use the **no distance** subcommand with the appropriate arguments and keywords.

Example 1:

In the example below, the **router igrp** global configuration command sets up IGRP routing in AS number 109. The network subcommands specify routing on networks *192.31.7.0* and *128.88.0.0*. The first **distance** router subcommand sets the default administrative distance to 255, which instructs the router to ignore all routing updates from routers for which an explicit distance has not been set. The second **distance** subcommand sets the administrative distance for all routers on the Class C network *192.31.7.0* to 90. The third **distance** subcommand sets the administrative distance for the router with the address *128.88.1.3* to 120.

```
router igrp 109
network 192.31.7.0
network 128.88.0.0
distance 255
distance 90  192.31.7.0  0.0.0.255
distance 120  128.88.1.3  0.0.0.0
```

Example 2:

The order in which you enter **distance** router subcommands can affect the assigned administrative distances in unexpected ways. For example, the following subcommands assign the router with the address *192.31.7.18* an administrative distance of 100, and all other routers on subnet *192.31.7.0* an administrative distance of 200.

```
distance 100 192.31.7.18  0.0.0.0
distance 200 192.31.7.0  0.0.0.255
```

Example 3:

However, if you reverse the order of these subcommands, all routers on subnet *192.31.7.0* are assigned an administrative distance of 200, even the router at address *192.31.7.18*.

```
distance 200 192.31.7.0  0.0.0.255
distance 100 192.31.7.18  0.0.0.0
```

Assigning administrative distances is a problem unique to each network and is done in response to the greatest perceived threats to the connected network. Even when general guidelines exist, the network manager must ultimately determine a reasonable matrix of administrative distances for the network as a whole. Table 14-1 below shows the default administrative distance for various sources of routing information.

Table 14-1 Default Administrative Distances

Route Source	Default Distance
connected interface	0
static route	1
IGRP	100
RIP	120
HELLO	130
EGP	140
BGP	200
unknown	255

Directly Connected Routes

Directly connected routes are routes to the networks specified by the interface addresses of the router. An interface may have multiple IP addresses.

Treatment of Directly Connected Routes

The router automatically enters a directly connected route in the routing table if the interface is usable. A "usable" interface is one through which the router can send and receive packets. If the router determines that an interface is not usable, it removes the directly connected

routing entry from the routing table. Removing the entry allows the router to use dynamic routing protocols to determine backup routes to the network (if any).

To display the usability status of interfaces, use the EXEC command **show interfaces.** If the interface hardware is usable, the interface is marked "up." If the interface can provide two-way communication, the line protocol is marked "up." For an interface to be usable, both the interface hardware and line protocol must be up.

Multiple Interface Addresses

The software supports multiple IP addresses per interface. In addition to the primary address specified by the **ip address** interface subcommand, an unlimited number of secondary addresses may be specified by adding the optional keyword **secondary**, as shown:

ip address *address mask* [**secondary**]

Example:

In the example below, *131.108.1.27* is the primary address and *192.31.7.17* is a secondary address for Ethernet 0.

```
interface ethernet 0
ip address 131.108.1.27 255.255.255.0
ip address 192.31.7.17 255.255.255.0 secondary
```

Secondary addresses are treated like primary addresses, except that the system never generates datagrams other than routing updates with secondary source addresses. IP broadcasts and ARP requests are handled properly, as are interface routes in the IP routing table.

Secondary IP addresses can be used in a variety of situations. The following are the most common applications:

- There may not be enough host addresses for a particular network segment. For example, your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you need to have 300 host addresses. Using secondary IP addresses on the routers allows you to have two logical subnets using one physical subnet.
- Many older networks were built using Level 2 bridges. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can be easily made aware that there are many subnets on that segment.
- Two subnets of a single network might otherwise be separated by another network. This situation not permitted when subnets are in use. In these instances, the first network is *extended*, or layered on top of the second network using secondary addresses.

Note: If any router on a network segment uses a secondary address, all other routers on that same segment must also use a secondary address from the same network or subnet. An inconsistent use of secondary addresses on a network segment can very quickly lead to routing loops.

Overriding Static Routes with Dynamic Protocols

A static routing entry remains in effect until you remove it. This section describes how a static route can be overridden by dynamic routing information from one of the IP routing protocols. The **ip route** global configuration command for static routes is described in Chapter 13. The full syntax of the command follows:

ip route *network router* [*distance*]

The argument *network* is the Internet address of the target network or subnet, and the argument *router* is the Internet address of a router that can reach that network. The *distance* argument specifies an administrative distance.

If you specify an administrative distance, you are flagging a static route that may be overridden by dynamic information. For example, IGRP-derived routes have a default administrative distance of 100. To have a static route that would be overridden by an IGRP dynamic route, specify an administrative distance greater than 100.

Example:

In the example below, an administrative distance of 110 was chosen.

```
ip route 10.0.0.0  131.108.3.4   110
```

This implies that packets for network *10.0.0.0* will be routed to the router at *131.108.3.4.* if dynamic information about network *10.0.0.0* is not available.

Default Routes

A router may not be able to determine the routes to all other networks. To provide complete routing capability the common practice is to use some routers as "smart routers" and give the remaining routers default routes to the smart router. These default routes may be passed along dynamically or may be configured into the individual routers.

Generating Default Routes

Most dynamic interior routing protocols include a mechanism for causing a "smart router" to generate dynamic default information that is then passed along to other routers.

On the Cisco router, use this global configuration command:

ip default-network *network-number*

The argument *network-number* is a network number.

If the router has a directly connected interface onto that network, the dynamic routing protocols running on that router will generate or source a default route. In the case of RIP and HELLO, this is the mention of the pseudo-network *0.0.0.0.* In the case of IGRP, it is the network itself, flagged as an exterior route.

A router that is generating the default for a network may also need a default of its own. This may be done by specifying a static route to the network *0.0.0.0* via the appropriate router.

Picking a Default Route

When default information is being passed along through the dynamic routing protocol, no further configuration is required. The system will periodically scan its routing table to choose the optimal default network as its default route. In the case of RIP and HELLO, there will be only one choice, network *0.0.0.0*. In the case of IGRP, there may be several networks that can be candidates for the system default. The router uses both administrative distance and metric information to determine the default route. The selected default route appears in the gateway of last resort display of the EXEC command **show ip route**.

If dynamic default information is not being passed to the router, candidates for the default route may be specified with the **ip default-network** subcommand. In this usage, **ip default-network** takes a nonconnected network as an argument. If this network appears in the routing table from any source (dynamic or static), then it is flagged as a candidate default route and is subject to being chosen as the default route for the router. Multiple **ip default-network** commands may be given. All candidate default routes, both static (that is, flagged by **ip default-network**) and dynamic, appear in the routing table preceded by an asterisk.

Example:

In the following example, a static route to network *10.0.0.0* is defined as the static default route.

```
ip route 10.0.0.0  131.108.3.4
ip default-network 10.0.0.0
```

If the following global configuration command was issued on a router not connected to network *129.140.0.0*, then the router might choose the path to that network as a default route when the network appeared in the routing table.

```
ip default-network 129.140.0.0
```

Subnet Defaults

A default subnet may be specified for a network using the **ip default-network** *network* global configuration command. The *network* argument must have a subnet address value (where the host portion is zero and the subnet portion is not zero).

If the router is unable to route a datagram to a subnet of a network and a default subnet exists, then the datagram is delivered to that subnet (if directly connected) or to another router along the route to the default subnet. A default subnet is different than a default network; it applies only to subnets of a particular network. Suppose, for example, that network *131.108.0.0* was subnetted on the third octet and that subnet 45, or 131.108.45.0 was to be the default subnet. The command specifying the default would be default network *131.108.45.0*. The EXEC command **show ip route** *network* displays the default subnet for the specified network.

Redistributing Routing Information

In addition to running multiple routing protocols simultaneously, the router can redistribute information from one routing protocol to another. For example, you can instruct the router to re-advertise IGRP-derived routes using the RIP protocol, or to re-advertise static routes using the IGRP protocol.

The metrics of one routing protocol do not necessarily translate into the metrics of another routing protocol. For example, the RIP metric is a hop count, the HELLO metric is a delay, and the IGRP metric is a combination of five quantities. In such situations, an artificial metric is assigned to the redistributed route. Because of this unavoidable tampering with dynamic information, the careless exchange of routing information between different routing protocols can create routing loops, which can seriously degrade network operation.

Supported Metric Translations

This section describes the few supported automatic metric translations between the routing protocols. These descriptions assume that you have not defined a default redistribution metric, which replaces metric conversions (see the section "Setting Default Metrics" in this chapter).

RIP can automatically redistribute static routes and HELLO-derived routes. RIP assigns static routes a metric of 1 (directly connected) and converts HELLO metrics in accordance with Table 14-2.

Values are derived from the mapping function defined by Dave Mills and the gated developers at the Cornell University Theory Center.

EGP can automatically redistribute static routes and RIP-, HELLO-, and IGRP-derived routes. EGP assigns the metric three to all static and derived routes.

BGP does not normally send metrics in its routing updates.

HELLO can automatically redistribute static routes and RIP- and IGRP-derived routes. HELLO assigns static routes a metric of 100 (directly connected) and converts RIP metrics in accordance with Table 14-2. HELLO advertises IGRP-derived routes with a metric equal to the delay portion of the IGRP metric or to 100, whichever is larger. Ethernets have a delay of 1 millisecond and serial links have a phase delay of 20 milliseconds; HELLO assigns a metric of 100 to routes using these media.

IGRP can automatically redistribute static routes and information from other IGRP-routed autonomous systems. IGRP assigns static routes a metric that identifies them as directly connected. IGRP does not change the metrics of routes derived from IGRP updates from other autonomous systems.

Note that any protocol can redistribute other routing protocols if a default metric is in effect (see the section "Setting Default Metrics" in this chapter).

Table 14-2 RIP and HELLO Metric Transformations

From HELLO	To RIP	From RIP	To HELLO
0	0	0	0
1-100	1	1	100
101-148	2	2	200
149-219	3	3	300
220-325	4	4	325
326-481	5	5	481
482-713	6	6	713
714-1057	7	7	1057
1058-1567	8	8	1567
1568-2322	9	9	2322
2323-3440	10	10	3440
3441-5097	11	11	5097
5098-7552	12	12	7552
7553-11190	13	13	11190
11191-16579	14	14	16579
16580-24564	15	15	24564
24565-30000	16	16	30000

Passing Routing Information Among Protocols

By default, the router does not exchange information among different routing protocols. If you want to pass routing information among routing protocols, use the **redistribute** router subcommand. Full syntax for this command follows.

redistribute *process-name* [*AS-number*]

no redistribute *process-name* [*AS-number*]

The argument *process-name* specifies a routing information source using one of the following keywords:

- **static**
- **rip**
- **bgp**
- **egp**
- **hello**
- **igrp**

Use the optional argument *AS-number* when you specify the **igrp** or **egp** keyword, to specify the autonomous system number.

Use the **no redistribute** command with the appropriate arguments to remove this function.

Example:

To redistribute RIP-derived information using the HELLO protocol, enter these commands:

```
router hello
redistribute rip
```

To end redistribution of information from a routing protocol, use the **no redistribute** router subcommand, supplying the appropriate arguments.

Redistributed routing information should always be filtered by the **distribute-list out** router subcommand described in the section "Filtering Outgoing Information" in this chapter. This ensures that only those routes intended by the administrator are passed along to the receiving routing protocol.

When redistributing information between IGRP processes, use the **default-information** router subcommand. The full syntax of this command follows:

default-information {**in** | **out**}

no default-information {**in** | **out**}

This subcommand controls the handling of default information between multiple IGRP processes. The **no default-information in** subcommand causes IGRP exterior or default routes to be suppressed when received by an IGRP process. Normally exterior routes are always accepted. The **no default-information out** subcommand causes IGRP exterior routes to be suppressed in updates. Default information is normally passed between IGRP processes when doing redistribution.

The default network of *0.0.0.0* used by RIP and HELLO cannot be redistributed by IGRP.

Setting Default Metrics

The **default-metric** router subcommand, used in conjunction with the **redistribute** router subcommand, causes the current routing protocol to use the same metric value for all redistributed routes. (Redistributed routes are those routes established by other routing protocols.) A default metric helps solve the problem of redistributing routes with incompatible metrics; whenever metrics do not convert, using a default metric provides a reasonable substitute and enables the redistribution to proceed.

The **default-metric** router subcommand has two forms, depending on the routing protocol specified in the **redistribute** subcommand.

For RIP, EGP, BGP, and HELLO, which use scalar, single-valued metrics, the subcommand has this syntax:

default-metric *number*

The argument *number* is the default metric value (an unsigned integer) appropriate for the specified routing protocol.

For IGRP, the **default-metric** router subcommand has this syntax:

default-metric *bandwidth delay reliability loading mtu*

- The argument *bandwidth* is the minimum bandwidth of the route in kilobits per second.
- The argument *delay* is the route delay in tens of microseconds.
- The argument *reliability* is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100x25 reliability).
- The argument *loading* is the effective bandwidth of the route in kilobits per second.
- The argument *mtu* is the minimum Maximum Transmission Unit (MTU) of the route.

Example:

For example, consider a router in autonomous system 109 using both the HELLO and IGRP routing protocols. To advertise IGRP-derived routes using the HELLO protocol and to assign the IGRP-derived routes a HELLO metric of 10,000, the configuration subcommands are:

```
router hello
default-metric 10000
redistribute igrp 109
```

Figure 14-10 shows this type of redistribution.

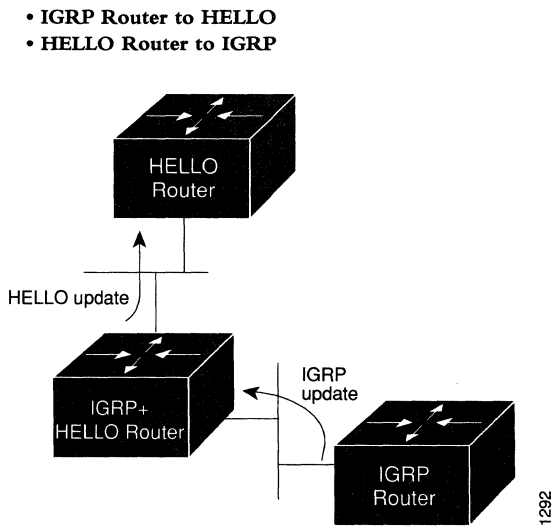


Figure 14-10 Assigning Metrics for Redistribution

Use the **no default metric** command to return the routing protocol to using the built-in, automatic metric translations.

Special Routing Configuration Techniques

This section describes configuration techniques for special situations and requirements.

Configuring Static Routes

The **ip route** interface subcommand is used to establish static routes. A static route is appropriate if the router cannot dynamically build a route to the destination.

ip route *network address* [*distance*]

The argument *network* is the Internet address of the target network or subnet, and the argument *address* is the Internet address of a router that can reach that network. The optional *distance* argument specifies an administrative distance.

If you specify an administrative distance, you are flagging a static route that may be overridden by dynamic information.

Example 1:

In the example below, packets for network *10.0.0.0* will be routed to the router at 131.108.3.4:

```
ip route 10.0.0.0  131.108.3.4
```

Example 2:

Occasionally, you will need to set up static routes to particular destinations. In this example, the router is configured for static routes to several destinations: specific hosts and specific addresses.

```
ip route 131.22.3.21 192.31.7.19
ip route 145.42.6.63 192.31.7.19
ip route 145.45.3.24 192.31.7.19 150
```

IGRP Metric Adjustments

The following **metric** router subcommands alter the default behavior of IGRP routing and metric computation, and allow the tuning of the IGRP metric calculation for a particular Type of Service (TOS).

metric weights *TOS K1 K2 K3 K4 K5*

no metric weights

The *TOS* parameter currently must always be zero. Parameters *K1* through *K5* are constants in the equation that converts an IGRP metric vector into a scalar quantity.

The composite IGRP metric is computed according to the following formula:

metric = [K1*bandwidth + (K2*bandwidth)/(256 - load) + K3*delay] *
[K5/(reliability + K4)]

If K5 == 0, then there is no reliability term.

The default version of IGRP has K1 == K3 == 1, K2 == K4 == K5 == 0 .

Delay is in units of 10 microseconds. This gives a range of 10 microseconds to 168 seconds. A delay of all ones indicates that the network is unreachable.

Bandwidth is inverse minimum bandwidth of the path in bits per second scaled by a factor of 10e10. The range is from a 1200 bps line to 10 Gbps.

Because of the somewhat unusual units used for bandwidth and delay, some examples seem in order. These are the default values used for several common media.

	Delay	Bandwidth
Satellite	200,000 (2 sec)	20 (500 Mbit)
Ethernet	100 (1 ms)	1,000
1.544 Mbit	2000 (20 ms)	6,476
64 Kbit	2000	156,250
56 Kbit	2000	178,571
10 Kbit	2000	1,000,000
1 Kbit	2000	10,000,000

Reliability is given as a fraction of 255. That is, 255 is 100% reliability, or a perfectly stable link.

Load is given as a fraction of 255. A load of 255 indicates a completely saturated link.

Use the **no metric weights** command to return these five constants to their default values.

The IGRP-only router subcommand **metric holddown** can be used to disable hold down. The syntax is as follows:

metric holddown

no metric holddown

Note: This command assumes that the entire AS is running Version 8.2(5) or more recent software since the hop count is used to avoid information looping. Using it with earlier software will cause problems.

The IGRP-only router subcommand **metric maximum-hops** sets a maximum hop count:

metric maximum-hops *hops*

no metric maximum-hops *hops*

This command causes the IP routing software to advertise as unreachable routes with a hop count greater than the value assigned to the *hops* argument. This is a safety mechanism that breaks any potential count-to-infinity problems. The default value is 100 hops; the maximum value is 255.

Example:

In the following example, a router in AS 71, attached to network *15.0.0.0*, wants a maximum hop count of 200, doubling the default. The network administrators decided to do this because they have a complex WAN that may generate a large hop count under normal (non-looping) operations. (Other commands are needed between the network command and the metric command in a real-world situation; this is not a complete configuration example).

```
router igrp 71
network 15.0.0.0
metric maximum-hops 200
```

Keepalive Timers

It is possible to tune the IP routing support in the Cisco software to enable faster convergence of the various IP routing algorithms and hence quicker fall-back to redundant routers. The total effect is to minimize disruptions to end users of the network in situations where quick recovery is essential.

The network administrator can configure the keepalive interval, the frequency at which the Cisco router sends messages to itself (Ethernet and Token Ring) or to the other end (serial) to ensure a network interface is alive. The interval in previous software versions was ten seconds; it is now adjustable in one second increments down to one second. An interface is declared down after three update intervals have passed without receiving a keepalive packet.

The syntax for the **keepalive** interface subcommand is:

keepalive [*seconds*]

no keepalive

If the optional argument *seconds* is not specified, a default of ten seconds is assumed.

Example:

In the following example, the keepalive interval is set to three seconds.

```
interface ethernet 0
keepalive 3
```

Setting the keepalive timer to a low value is very useful for rapidly detecting Ethernet interface failures (transceiver cable disconnecting, cable unterminated, and so on).

A typical serial line failure involves losing Carrier Detect (CD). Since this sort of failure is typically noticed within a few milliseconds, adjusting the keepalive timer for quicker routing recovery is generally not useful.

Note: When adjusting the keepalive timer for a very low bandwidth serial interface, it is possible to have large datagrams delay the smaller keepalive packets long enough to cause the line protocol to spuriously go down. Be careful when using this feature.

Adjustable Routing Timers

The basic timing parameters for IGRP, EGP, RIP, and HELLO are adjustable. Since these routing protocols are executing a distributed, asynchronous routing algorithm, it is important that these timers be the same for all routers in the network.

To adjust timers, use the **timers basic** router subcommand. Full syntax for this command follows:

timers basic *update invalid holddown flush*

no timers basic

- The argument *update* is the rate at which updates are sent. This is the fundamental timing parameter of the routing protocol.
- The argument *invalid* is an interval of time after which a route is declared invalid; it should be three times the value of *update.*
- The argument *holddown* is the interval during which routing information regarding better paths is suppressed. It should be at least three times the value of *update.*
- The argument *flush* is the amount of time that must pass before the route is removed from routing table; it must be at least the sum of *invalid* and *holddown.*

Use the **no timers basic** command to reset the defaults.

Note: The current and default timer values can be seen by inspecting the output of the EXEC command **show ip protocols**. The relationships of the various timers should be preserved as described above.

Example 1: IGRP

In the following example, updates are broadcast every five seconds. If a router is not heard from in 15 seconds, the route is declared unusable. Further information is suppressed for an additional 15 seconds. At the end of the suppression period, the route is flushed from the routing table.

```
router igrp 109
timers basic 5  15  15  30
```

Note that by setting a short update period, you run the risk of congesting slow-speed serial lines; however, this is not much of a concern on faster-speed Ethernets and T1-rate serial lines. Also, if you have many routes in your updates, you can also cause the routers to spend an excessive amount of time processing updates.

Example 2: EGP

When **timers basic** is used with EGP, the update time and holddown time are ignored. For example, the commands below will set the invalid time for EGP to 100 seconds and the flush time to 200 seconds.

```
router egp 47
timers basic 0 100 0 200
```

Gateway Discovery Protocol (GDP)

The Gateway Discovery Protocol (GDP) allows hosts to dynamically detect the arrival of new routers as well as determine when a router goes down. Host software is needed to take advantage of this protocol. Unsupported, example GDP clients can be obtained from Cisco Systems.

GDP is not a standard; it is a protocol designed by Cisco Systems to meet the customers' needs. Work is in progress to establish GDP or similar protocol as a standard means of discovering routers. The current GDP implementation is described next in more detail.

For ease of implementation on a variety of host software, GDP is based on the User Datagram Protocol (UDP). The UDP source and destination ports of GDP datagrams are both set to 1997 (decimal).

There are two types of GDP messages: *Report* and *Query*. On broadcast media *Report* message packets are periodically sent to the IP broadcast address announcing that the router is present and functioning. By listening for these *Report* packets, a host can detect a vanishing or appearing router. If a host issues a *Query* packet to the broadcast address, the routers each respond with a *Report* sent to the host's IP address. On nonbroadcast media, routers send *Report* message packets only in response to *Query* message packets. The protocol provides a mechanism for limiting the rate at which *Query* messages are sent on nonbroadcast media.

Figure 14-11 shows the format of the GDP *Report* message packet format. A GDP *Query* message packet has a similar format, except that the Count field is always zero and no address information is present.

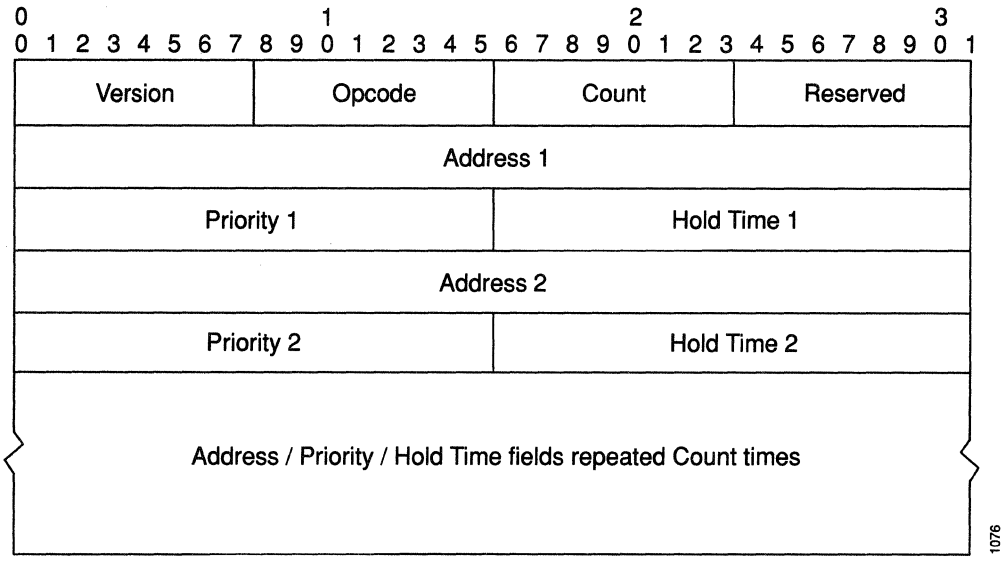


Figure 14-11 GDP Report Message Packet Format

The fields in the *Report* and *Query* messages are described next.

- **Version**—8-bit field containing the protocol version number. The current GDP version number is 1. If an unrecognized version number is found, the GDP message must be ignored.
- **Opcode**—8-bit field that describes the GDP message type. Unrecognized opcodes must be ignored. Opcode 1 is a *Report* message and opcode 2 is a *Query* message.
- **Count**—8-bit field that contains the number of address, priority, and hold time tuples in this message. A *Query* message has a Count field value of zero. A *Report* message has a Count field value of 1 or greater.
- **Reserved**—8-bit reserved field; it must be set to zero.
- **Address**—32-bit fields containing the IP address of a router on the local network segment. There are no other restrictions on this address. If a host encounters an address that it believes is not on its local network segment, then the host should quietly ignore that address.
- **Priority**—16-bit fields that indicate the relative quality of the associated address. The numerically larger the value in the Priority Field, the *better* the address should be considered.
- **Hold Time**—16-bit fields. On broadcast media, the number of seconds the associated address should be used as a router without hearing further *Report* messages regarding that address. On nonbroadcast media, such as X.25, this is the number of seconds the requester should wait before sending another *Query* message.

Numerous actions can be taken by the host software listening to GDP packets. One possibility is to flush the host's ARP cache whenever a router appears or disappears. A more complex possibility is to update a host routing table based on the coming and going of routers. The particular course of action taken depends on the host software and the customer's requirements.

Using GDP Commands

The **ip gdp** interface subcommand enables GDP processing on an interface. Full syntax for this command follows:

ip gdp

no ip gdp

If you use this form of the **ip gdp** subcommand, you use only the default parameters. The default parameters are:

- A reporting interval of five seconds for broadcast media such as Ethernets, and zero seconds (never) for nonbroadcast media such as X.25
- A priority of 100
- A hold time of 15 seconds

If you want to alter some of these parameters, you can use one of the following subcommands:

ip gdp priority *number*

ip gdp reporttime *seconds*

ip gdp holdtime *seconds*

The **priority** keyword takes a *number* parameter and alters the priority from its default of 100. A larger number signifies a higher priority.

The **reporttime** keyword takes a time parameter in seconds.

The **holdtime** keyword also takes a time parameter in seconds.

When enabled on an interface, GDP updates report the primary and secondary IP addresses of that interface.

Example:

In the example below, GDP is enabled on interface Ethernet 1 with a report time of ten seconds, and priority and hold time set to their defaults (because none are specified).

```
ip gdp reporttime 10
```

IP Routing Protocols Configuration Examples

This sections contains complete configuration examples of the IP routing protocols.

Static Routing Redistribution

In the example below, three static routes are specified, two of which wish to have the IGRP process advertise. Do this by specifying the **redistribute static** subcommand, then specifying an access list that allows only those two networks to be passed to the IGRP process. Any redistributed static routes should be sourced by a single router to minimize the likelihood of creating a routing loop.

Example:

```
ip route 192.1.2.0  192.31.7.65
ip route 193.62.5.0  192.31.7.65
ip route 131.108.0.0  192.31.7.65
access-list 3 permit 192.1.2.0
access-list 3 permit 193.62.5.0
router igrp 109
network 192.31.7.0
redistribute static
distribute-list 3 out static
```

RIP and HELLO Redistribution

Consider a wide area network at a university that uses RIP as an interior routing protocol. Assume the university wants to connect its wide area network to a regional network, *128.1.1.0*, which uses HELLO as the routing protocol. The goal in this case is to advertise the networks in the university network to the routers on the regional network. The commands for the interconnecting router are:

Example:

```
router hello
network 128.1.1.0
redistribute rip
default-metric 10000
distribute-list 10 out rip
```

In this example, the **router** command starts a HELLO routing process. The **network** subcommand specifies that network *128.1.1.0* (the regional network) is to receive HELLO routing information. The **redistribute** subcommand specifies that RIP-derived routing information be advertised in the HELLO routing updates. The **default-metric** subcommand assigns a HELLO delay of 10,000 to all RIP-derived routes.

The **distribute-list** router subcommand instructs the router to use access list 10 (not defined in this example) to limit the entries in each outgoing HELLO update. The access list prevents unauthorized advertising of university routes to the regional network.

This example could have specified automatic conversion between the RIP and HELLO metrics. However, in the interest of routing table stability, it is not desirable to do so. Instead, this example limits the routing information exchanged to availability information only.

IGRP Redistribution

Each IGRP routing process can provide routing information to only one autonomous system; the router must run a separate IGRP process and maintain a separate routing database for each autonomous system it services. However, you can transfer routing information between these routing databases.

Suppose the router has one IGRP routing process for network *15.0.0.0* in autonomous system 71 and another for network *192.31.7.0* in autonomous system 109, as the following commands specify:

Example 1:

```
router igrp 71
network 15.0.0.0
router igrp 109
network 192.31.7.0
```

To transfer a route to *192.31.7.0*, to the database of the first routing process (without passing any other information about autonomous system 109), use the following commands:

Example 2:

```
router igrp 71
redistribute igrp 109
distribute-list 3 out igrp 109
access-list 3 permit 192.31.7.0
```

Third-Party EGP Support

In this example configuration, the Cisco router is in AS 110 communicating with an EGP neighbor in AS 109 with address *131.108.6.5*. Network *131.108.0.0* is advertised as originating within AS 110. The configuration specifies that two routers, *131.108.6.99* and *131.108.6.100*, should be advertised as third-party sources of routing information for those networks that are accessible through those routers. The global configuration commands also specify that those networks should be flagged as internal to AS 110.

Example:

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
neighbor 131.108.6.5  third-party 131.108.6.99 internal
neighbor 131.108.6.5  third-party 131.108.6.100 internal
```

Backup EGP Router

The following example configuration illustrates a router that is in AS 110 communicating with an EGP neighbor in AS 109 with address 131.108.6.5. Network *131.108.0.0* is advertised with a distance of zero, and networks learned by RIP are being advertised with a distance of five. Access list 3 filters which RIP-derived networks are allowed in outgoing EGP updates.

Example:

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
redistribute rip
default-metric 5
distribute-list 3 out rip
```

Maintaining IP Routing Operations

The IP routing protocols have one command to help you maintain the IP routing operations.

Use the privileged EXEC command **clear ip route** to remove a dynamic route from the routing table. Enter this command at the EXEC prompt:

clear ip route {*network* | *}

The argument *network* is the network address portion of the routing table entry to be removed. If you specify * all routes are removed. Note that routing entries you remove with the **clear ip route** command may reappear because of dynamic routing, or because they are floating static routes.

To remove a static route from the routing table, use the **no ip route** global configuration command with the appropriate arguments for the type of static route.

Monitoring IP Routing Operations

This section describes the EXEC commands you may use to monitor IP routing operations configured on your router.

Displaying the BGP Routing Table

Use the **show ip bgp** EXEC command to display a particular network in the BGP routing table. Enter this command at the EXEC prompt:

show ip bgp [*network*]

The optional argument *network* is a network number, and is entered to display a particular network in the BGP routing table.

Following is sample output:

```
BGP Table Version is 19
Status codes: * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, * - incomplete

   Network          Next Hop            BGP Peer         Metric Weight Path
*> 128.145.0.0      0.0.0.0             0.0.0.0                 0      *
*> 192.68.151.0     0.0.0.0             0.0.0.0                 0      *
*> 151.142.0.0      0.0.0.0             0.0.0.0                 0      *
*> 150.136.0.0      0.0.0.0             0.0.0.0                 0      *
*> 192.91.180.0     0.0.0.0             0.0.0.0                 0      *
*> 132.249.0.0      0.0.0.0             0.0.0.0                 0      *
*> 128.18.0.0       0.0.0.0             0.0.0.0                 0      *
*> 192.53.65.0      0.0.0.0             0.0.0.0                 0      *
*> 192.53.66.0      0.0.0.0             0.0.0.0                 0      *
*> 192.67.67.0      0.0.0.0             0.0.0.0                 0      *
*> 192.53.48.0      0.0.0.0             0.0.0.0                 0      *
*> 192.31.7.0       0.0.0.0             0.0.0.0                 0      *
*> 130.93.0.0       0.0.0.0             0.0.0.0                 0      *
*> 192.12.33.0      0.0.0.0             0.0.0.0                 0      *
*  131.108.0.0      131.108.6.68        131.108.6.68            0      68 i
*>                  0.0.0.0             0.0.0.0                 0      i
```

In the display:

- The `Table Version` is the internal version number for the table. This is incremented any time the table changes.
- The first three characters indicate the status. The * (asterisk) indicates that the table entry is valid. The > character indicates that that table entry is the best entry to use for that network. The lowercase `i` indicates that the table entry was learned via an internal BGP session.
- The `Next Hop` entry is the IP address of the next system to use when forwarding a packet to the destination network. An entry of *0.0.0.0* indicates that the local router has some nonBGP route to this network.
- The `BGP peer` entry is the IP address of the BGP peer that we learned the route from. An entry of *0.0.0.0* indicates that the local router injected the route into BGP.

- The `Metric` field, if any, is the value of the inter-autonomous system metric. This is frequently not used.
- The `Weight` field is set through the use of AS filters.
- The path is the autonomous system path to the destination network. At the end of the path is the origin code for the path. The lowercase `i` indicates that the entry was originated with the local IGP and advertised with a **network** subcommand. A lowercase `e` indicates that the route originated with EGP. An asterisk (`*`) indicates that the origin of the path is not clear. Usually this a path that is redistributed into BGP from an IGP.

Displaying BGP Neighbors

Use the **show ip bgp neighbors** EXEC command to display detailed information on the TCP and BGP connections to individual neighbors. Enter this command at the EXEC prompt:

show ip bgp neighbors

Following is sample output:

```
BGP neighbor is 131.108.6.68, remote AS 68, external link
  BGP state = 3, table version = 19, up for 0:12:38
  Last read 0:00:37, hold time is 180, keepalive interval is 60 seconds
  Received 48 messages, 0 NOTIFICATIONS
  Sent 51 messages, 0 NOTIFICATIONS
  Connections established 1; dropped 0
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 131.108.6.69, 12288   Foreign host: 131.108.6.68, 179

Enqueued packets for retransmit: 0, input: 0, saved: 0

Event Timers (current time is 835828):
Timer:       Retrans   TimeWait    AckHold    SendWnd  KeepAlive
Starts:           20          0         18          0          0
Wakeups:           1          0          2          0          0
Next:              0          0          0          0          0

iss:      60876  snduna:      62649  sndnxt:      62649     sndwnd:   1872
irs:   95187024  rcvnxt:   95188733  rcvwnd:       1969  delrcvwnd:    271

SRTT: 364 ms, RTTO: 1691 ms, RTV: 481 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 340 ms, ACK hold: 300 ms
Flags: higher precedence

Datagrams (max data segment is 1450 bytes):
Rcvd: 36 (out of order: 0), with data: 18, total data bytes: 1708
Sent: 40 (retransmit: 1), with data: 36, total data bytes: 1817
```

In the display:

- The first line lists the IP address of the BGP neighbor and its AS number. If the neighbor is in the same AS as the local router, then the link between them is internal. Otherwise, it is considered external.
- The `BGP state` indicates the internal state of this BGP connection. The `table version` indicates that the neighbor has been updated with this version of the primary BGP routing table. The `up time` indicates the amount of time that the underlying TCP connection has been in existence.
- The `last read` time is the time that BGP last read a message from this neighbor. The `hold time` is the maximum amount of time that can elapse between messages from the peer. The `keepalive interval` is the time period between sending keepalive packets, which help insure that the TCP connection is up.
- The number of `Received messages` indicates the number of total BGP messages received from this peer, including keepalives. The number of notifications is the number of error messages that we have received from the peer.
- The number of `Sent messages` indicates the total number of BGP messages that have been sent to this peer, including keepalives. The number of notifications is the number of error messages that we have sent to this peer.
- The number of `Connections established` is a count of the number of times that we have established a TCP connection and the two peers have agreed speak BGP with each other. The number of dropped connections is the number of times that a good connection has failed or been taken down.
- The remainder of the display describes the status of the underlying TCP connection. See the description of **show ip tcp** for more information.

Displaying EGP Statistics

Use the **show ip egp** command to display detailed statistics on EGP connections. Enter this command at the EXEC prompt:

show ip egp

The command output includes detailed information about neighbors. Sample output follows. Table 14-3 describes the fields seen.

```
Local autonomous system is 109
  EGP Neighbor FAS/LAS State   SndSeq RcvSeq Hello Poll j/k Flags
  10.3.0.27      1/109 IDLE       625  61323    60  180   0 Perm, Act
* 10.2.0.37      1/109 UP 12:29   250  14992    60  180   3 Perm, Act
* 10.7.0.63      1/109 UP  1d19   876  10188    60  180   4 Perm, Pass
```

Table 14-3 Show IP EGP Field Descriptions

Field	Description
EGP Neighbor	Address of the EGP neighbor.
FAS	Foreign Autonomous System number
LAS	Local Autonomous System number
State	State of the connection between peers
SndSeq	Send sequence number
RcvSeq	Receive sequence number
Hello	Interval between HELLO/I-HEARD-YOU packets
Poll	Interval between POLL/UPDATE packets
j/k	Measure of reachability; 4 is perfect
Flags	• Perm—Permanent (currently no alternatives) • Act—Active, controlling the connection • Pass—Passive, neighbor controls the connection

Displaying Routing Protocol Parameters and Status

Use the EXEC command **show ip protocols** to display the parameters and current state of the active routing protocol process. Enter this command at the EXEC prompt:

show ip protocols

Following is sample output:

```
Routing Protocol is "igrp 109"
  Sending updates every 90 seconds, next due in 88 seconds
  Invalid after 270 seconds, hold down for 280, flushed after 630
  Outgoing update filter list for all routes is not set
  Incoming update filter list for all routes is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  IGRP maximum hopcount 100
  Redistributing: igrp 109
  Routing for Networks:
    131.108.0.0
    192.31.7.0
  Routing Information Sources:
    Gateway         Distance  Last Update
    131.108.2.201        100      0:00:08
    131.108.200.2        100      5d15
    131.108.2.200        100      0:00:09
    131.108.2.203        100      0:00:11
```

The information displayed by **show ip protocols** is useful in debugging routing operations. Information in the `Routing Information Sources` field of the **show ip protocols** output can help you identify a router suspected of delivering bad routing information.

In the display:

- The `Routing Protocol` field specifies the routing protocol used.
- `Sending updates` field specifies the time between sending updates, as well as precisely when the next is due to be sent.
- The `Invalid` field specifies the value of the invalid parameter.
- The `hold down` field specifies the current value of the holddown parameter.
- The `flushed` field specifies the time in seconds after which the individual routing information will be thrown (flushed) out.
- The `outgoing` update field specifies whether the outgoing filtering list has been set.
- The `incoming` update field specifies whether the incoming filtering list has been set.
- The `Default networks` field specifies how these networks will be handled in both incoming and outgoing updates.
- The `IGRP metric` field specifies the value of the K0-K5 metrics as well as the maximum hopcount.
- The `Redistributing` field lists the protocol that is being redistributed.
- The `Routing` field specifies the networks that the routing process is currently injecting routes for.
- The `Routing Information Sources` field lists all the routing sources the router is using to build its routing table. For each source, you will see displayed:
  - The IP address
  - The administrative distance
  - The time the last update was received from this source

Displaying the Routing Table

Use the EXEC command **show ip route** to display the current state of the routing table. Enter this command at the EXEC prompt:

show ip route [*address*]

When entered with the optional *address* argument, the command displays detailed routing information for the specified network or subnet.

Following is sample output from this command when entered without an address:

```
Codes: I - IGRP derived, R - RIP derived, H - HELLO derived
       C - connected, S - static, E - EGP derived, B - BGP derived
       * - candidate default route

Gateway of last resort is 131.108.1.6 to network 129.140.0.0

I*Net 10.0.0.0 [100/28476] via 192.31.7.130, 11 sec, Serial2
I*Net 129.140.0.0 [100/9576] via 131.108.1.6, 21 sec, Ethernet0
                             via 131.108.2.6, 21 sec, Ethernet1
                             via 192.31.7.26, 21 sec, Ethernet2
I Net 128.18.0.0 [100/8576] via 192.31.7.130, 11 sec, Serial2
C Net 192.31.7.0 is subnetted (mask is 255.255.255.240), 3 subnets
I     192.31.7.144 [100/1002100] via 192.31.7.114, 30 sec, Serial6
C     192.31.7.16 is directly connected, Ethernet2
C     192.31.7.48 is directly connected, Serial0
I Net 192.12.33.0 [100/8576] via 192.31.7.130, 11 sec, Serial2
C Net 131.108.0.0 is subnetted (mask is 255.255.255.0), 1 subnet
I     131.108.205.0 [100/1004200] via 131.108.1.6, 21 sec, Ethernet0
```

In the displays, the * (asterisk) indicates a candidate default route; NET indicates a network rather than subnetwork entry. Other fields contain this information:

- The first column lists the protocol that derived the route.
- The second column lists the address of the remote network The first number in the brackets is the administrative distance of the information source; the second number is the metric for the route.
- The third column specifies the address of the router that can build a route to the specified remote network.
- The fourth column specifies the cost in seconds associated with the specified route.
- The final column specifies the interface through which the specified network can be reached.

Directly connected routers may include subnet information where appropriate.

Following is sample output from this command when entered with the address *10.0.0.0*:

```
Routing entry for 10.0.0.0 (mask 255.0.0.0)
  Known via "igrp 109", distance 100, metric 28476, candidate default
path
  Redistributing via igrp 109
  Last update from 192.31.7.130 on Serial2, 29 seconds ago
  Routing Descriptor Blocks:
  * 192.31.7.130, from 192.31.7.130, 29 seconds ago, 15 uses, via Serial2
      Route metric is 28476, traffic share count is 1
      Total delay is 220000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 128/255, Hops 0
```

When you specify that you want information about a specific network displayed, more detailed statistics are shown, as the following sample output indicates:

- The protocol that provided the information
- The administrative distance

- The metric as provided by the protocol (IGRP, in this case)
- The redistribution protocol
- The address of the source of this routing information, along with the following:
  - Time of the last incoming update for the route and
  - The interface that the information arrived on
- Much of this information is repeated in the Routing Descriptor Block, along with:
  - Number of times this route has been used since it was added to the table
  - Total round-trip delay in seconds
  - Minimum bandwidth on the route (the smallest pipe you will encounter along the way to the remote network, in other words)
  - `Reliability`, which is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100% reliability)
  - Minimum MTU
  - `Loading` (which is the effective bandwidth of the route in kilobits per second)
  - Hop count

Debugging IP Routing

The EXEC command **debug** and the global configuration command **logging** enable you to record useful routing information. Using the privileged EXEC command **debug,** you can instruct the router to log any combination of RIP, EGP, HELLO, BGP, and IGRP routing events as well as routing table events to the console terminal. In general, these commands are entered during troubleshooting sessions with Cisco engineers. For each **debug** command, there is a corresponding **undebug** command that disables the command output.

debug ip-bgp

The EXEC command **debug ip-bgp** displays debugging information on BGP transactions.

debug ip-egp

The **debug ip-egp** command displays EGP transactions.

debug ip-egp-events

The **debug ip-egp-events** command enables logging of major EGP transactions, such as an EGP peer being acquired or discontinued. If both **debug ip-egp** and **debug ip-egp-events** are enabled, the mention of individual networks in updates is suppressed. This reduction in the logging output permits easier debugging of EGP update problems.

debug ip-hello

The **debug ip-hello** command enables logging of HELLO routing transactions.

debug ip-igrp

The **debug ip-igrp** command enables logging of IGRP routing transactions.

debug ip-rip

The **debug ip-rip** command enables logging of RIP routing transactions.

debug ip-routing

The **debug ip-routing** command enables logging of routing table events such as network appearances and disappearances.

debug ip-tcp

The **debug ip-tcp** command enables logging of significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug ip-tcp-packet *line*

The **debug ip-tcp-packet** command enables logging of each TCP packet associated with the specified line number.

debug ip-udp

The **debug ip-udp** command enables logging of UDP-based transactions.

Global Configuration Command Summary

This section provides an alphabetical list of the global commands used with the IP Routing Protocols.

[no] autonomous-system *local-AS*

Specifies an autonomous system number. The argument *local-AS* is the local autonomous system (AS) number to which the router belongs. To remove the AS number, use the **no autonomous-system** configuration command.

[no] ip default-network *network-number*

Provides a mechanism for causing a smart router to generate dynamic default information that is then passed along to other routers. The argument *network-number* is a network number.

Router Subcommand Summary

This section provides an alphabetical list of the router subcommands used with the IP routing protocols.

[no] default-information {in | out}

Controls the handling of default information between multiple IGRP processes. The **no default-information in** subcommand causes IGRP exterior or default routes to be suppressed when received by an IGRP process. Normally exterior routes are always accepted. The **no default-information out** subcommand causes IGRP exterior routes to be suppressed in updates. Default information is normally passed between IGRP processes.

default-metric *bandwidth delay reliability loading mtu*

Sets metrics for IGRP only. The argument *bandwidth* is the minimum bandwidth of the route in kilobits per second. The argument *delay* is the route delay in tens of microseconds. The argument *reliability* is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100% reliability). The argument *loading* is the effective bandwidth of the route in kilobits per second. The argument *mtu* is the minimum Maximum Transmission Unit (MTU) of the route.

default-metric *number*

Sets metrics for RIP, EGP, BGP, and HELLO, which use scalar, single-valued metrics. The argument *number* is the default metric value (an unsigned integer) appropriate for the specified routing protocol.

no default-metric

Causes the current routing protocol to return to using the built-in, automatic metric translations.

[**no**] **distance** *weight* [[*ip-source-address ip-address-mask*] [*access-list-number*]]

Defines or deletes an administrative distance. The argument *weight* is an integer from 10 to 255 that specifies the administrative distance. Used alone, the argument *weight* specifies a default administrative distance that the router uses when no other specification exists for a routing information source. The optional argument pair *ip-source-address* and *ip-address-mask* specifies a particular router or group of routers to which the weight value applies. The argument *ip-source-address* is an Internet address that specifies a router, network, or subnet. The argument *ip-address-mask* (in dotted-decimal format) specifies which bits, if any, to ignore in the address value; a set bit in the *mask* argument instructs the router to ignore the corresponding bit in the address value. The optional argument *access-list-number* is the number of a standard IP access list.

[**no**] **distribute-list** *access-list-number* **in** [*interface-name*]

Filters networks received in updates. The argument *access-list-number* is a standard IP access list number. Use the keyword **in** to suppress incoming routing updates. The optional argument *interface-name* specifies the interface on which the access list should be applied to incoming updates. If no interface is specified, the access list will be applied to all incoming updates.

[**no**] **distribute-list** *access-list-number* **out** [*interface-name* | *routing-process*]

Suppresses networks from being sent in updates. The argument *access-list-number* is a standard IP access list number. Use the keyword **out** to apply the access list to outgoing routing updates. To filter a routing update sent on a specific interface, you may, optionally, specify the interface. When redistributing networks, a routing process name may be specified.

[**no**] **metric holddown**

Disables or re-enables holddown (IGRP only). This assumes that the entire AS is running Version 8.2(5) or more recent software since the hop count is used to avoid information looping. Using it with Version 7.1 or earlier software will cause problems.

[no] metric maximum-hops *hops*

Causes the IP routing software to advertise as unreachable routes with a hop count greater than the value assigned to the *hops* argument (IGRP only). The default value is 100 hops; the maximum value is 255.

[no] metric weights *TOS K1 K2 K3 K4 K5*

Allows the tuning of the IGRP metric calculation for a particular Type of Service (*TOS*). The TOS parameter currently must always be zero. Parameters *K1* through *K5* are constants in the equation that converts an IGRP metric vector into a scalar quantity.

[no] neighbor *address*

Creates a list of neighbor routers. The argument *address* is the IP address of a peer router with which routing information will be exchanged. The **no** form of the command removes an entry. The **no** form of the command returns the default.

[no] neighbor *address* **distribute-list** *list* {**in** | **out**}

Distributes neighbor information as specified in an access list *list* for BGP. You specify the access list to be applied to incoming or outgoing updates with the **in** and **out** keywords.

[no] neighbor *address* **filter-as** *number* **deny**

Filters neighbor information for an address in a specific AS for BGP. The **deny** keyword causes the information learned about that network using the specified AS to be ignored.

[no] neighbor *address* **filter-as** *number* **permit** *weight*

Filters neighbor information for an address in a specific AS for BGP. The **permit** keyword allows incoming information to be added to the routing table with a weight specified by the *weight* argument.

[no] neighbor *address* **remote-as** *autonomous-system*

Adds a neighbor entry to the routing table for BGP. The keywords address and autonomous system specify the IP address and AS of the neighbor router.

[no] neighbor *address* **third-party** *third-party-ip-address* **[internal | external]**

Adds third-party information to routing updates. The argument *third-party-ip-address* is the address of the third party on the network shared by the Cisco router and the EGP peer specified by the *address* argument. The optional keyword **internal** or **external** indicates whether the third party router should be listed in the internal or external section of the EGP update. Normally, all networks are mentioned in the internal section. You can enter this command multiple times.

[no] neighbor *address* **weight** *weight*

Specifies a weight to assign to a specific neighbor connection indicated by the argument *address*. The route with the highest weight is chosen as the preferred route when multiple routes are available to a particular network.

[no] network *address* **backdoor**

Specifies a back-door route to a BGP border router that will provide better information about the network.

[no] network *network-number*

Specifies a list of networks to be advertised as originating within an AS, or for EGP, the network to be advertised to the EGP peers of an EGP routing process. The argument *network-number* is a network number. The **no** form of the command removes an entry from the list.

[no] offset-list *list* {**in** | **out**} *offset*

Adds or removes a positive offset to incoming and outgoing metrics (as indicated by the **in** and **out** keywords) for networks matching an access list (for IGRP, RIP and HELLO only). If the argument *list* is zero, the argument supplied to *offset* is applied to all metrics. If *offset* is zero, no action is taken. For IGRP, the offset is added to the delay component only.

[no] passive-interface *interface*

Disables or enables sending routing updates on an interface. The argument *interface* specifies a particular interface.

[no] redistribute *process-name* [*AS-number*]

Passes routing information among routing protocols. The argument *process-name* specifies a routing information source using one of the keywords: **static, rip, egp, hello, igrp**. Use the optional argument *AS-number* when you specify the **igrp** or **egp** keyword, to specify the autonomous system number.

[no] router *protocol* [*autonomous-system*]

Creates an IP routing process. The argument *protocol* is one of these protocol-type keywords —**rip, egp, hello, bgp, igrp.** The IGRP, BGP, and EGP protocols use the optional argument *autonomous-system* to supply the number of an autonomous system.

[no] timers basic *update invalid holddown flush*

Adjusts timers. The argument *update* is the rate at which updates are sent. This is the fundamental timing parameter of the routing protocol. The argument *invalid* is an interval of time after which a route is declared invalid; it should be three times the value of *update.* The argument *holddown* is the interval during which routing information regarding better paths is suppressed. It should be at least three times the value of *update.* The argument *flush* is the amount of time that must pass before the route is removed from routing table; it must be at least the sum of *invalid* and *holddown.* The **no** form of the command restores the default.

[no] timers bgp *keepalive holdtime*

Adjusts BGP timers. The argument *keepalive* is the frequency in seconds with which the router sends *keepalive* messages to its peer (default 60 seconds), and where *holdtime* is the interval in seconds after not receiving a *keepalive* message that the router declares a peer dead (default 180 seconds). The **no** form of the command restores the default.

[no] timers egp *hello poll*

Adjusts the EGP timers. The argument *hello* adjusts the interval at which hellos are sent. The default is set to 60 seconds. The argument *poll* adjusts the interval at which polling is performed. The default is set to 180 seconds; the **no** form of the command restores the default.

IP Routing Interface Subcommands

This section provides alphabetical lists of the interface subcommands used with the IP routing protocols.

ip address *address mask* [**secondary**]

Specifies the IP address on an interface. The argument *address* supplies the address; the argument **mask** the subnet mask. The optional keyword **secondary** allows multiple IP addresses per interface.

[no] ip gdp

Enables or disables GDP routing, with all default parameters. Reporting interval is 5 seconds for Ethernet and zero seconds for nonbroadcast media. The priority is 100 and the hold time is 15 seconds.

[no] ip gdp holdtime *seconds*

Enables or disables GDP routing, specifying *holdtime* in seconds and keeping all other parameters (priority and reporting interval) at their default settings.

[no] ip gdp priority *number*

Enables or disables GDP routing with a priority number you specify. Report time remains at five seconds for Ethernets and the holdtime remains 15 seconds.

[no] ip gdp reporttime *seconds*

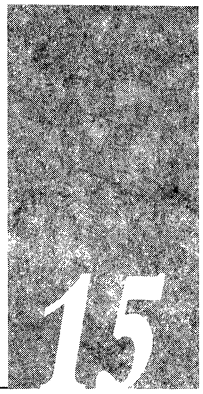
Enables or disables GDP routing, with a report time you specify. The priority remains 100 and the hold time remains 15 seconds.

[no] ip route *network address*

Establishes static routes.The argument network address is the network address for which you are establishing a static route.

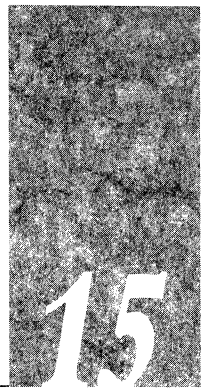
[no] keepalive [*seconds*]

Adjusts the keepalive timer for a specific interface. If the optional argument *seconds* is not specified, a default of ten seconds is assumed.



Chapter 15
Routing ISO CLNS

Chapter 15
Routing ISO CLNS



This chapter describes Cisco Systems' implementation of the International Organization for Standardization (ISO) Connectionless Network Services (CLNS) protocol, which is a standard for the Network Layer of the Open Systems Interconnection (OSI) model. This chapter includes a section reviewing ISO terminology, as well as sections focusing on these tasks and topics:

- An overview of CLNS addresses and the basics of the configuration process.
- How to configure dynamic and static routing.

For both static and dynamic routing, you have choices of intra-domain and inter-domain routing, locally-sourced packets and various lower-layer protocols.

This chapter also explains how Cisco's implementation differs from the standards (where they do), and how to optimize and fine-tune your CLNS-based internet. Configuration examples are grouped in a separate section, with configuration command lists and illustrations.

Cisco's Implementation of ISO CLNS

The Cisco routing software supports packet forwarding and routing for ISO CLNS for networks using a variety of lower layer protocols: Ethernet, Token Ring, FDDI, and serial.

You can use CLNS routers on serial interfaces with either HDLC, LAPB, or X.25 encapsulation. To use HDLC encapsulation, you must have a Cisco router at both ends of the link. If you use X.25 encapsulation, you must manually enter the NSAP-to-X.121 mapping. The LAPB and X.25 encapsulations will interoperate with other vendors.

In addition, the Cisco CLNS implementation is compliant with the Government Open Systems Interconnection Profile (GOSIP) Version 2.

- As part of its CLNS support, Cisco routers fully support these ISO and ANSI standards:
- ISO 9542—Documents the End System-Intermediate System (ES-IS) routing exchange protocol.
- ISO 8473—Documents the ISO Connectionless Network Protocol (CLNP).
- ISO 8348/Ad2—Documents Network Service Access Points (NSAPs).

Cisco supports the Interior Gateway Routing Protocol (IGRP) for dynamic routing of ISO CLNS. IGRP discovers routes to all End Systems (hosts) in the routing domain, provides dynamic recovery from failures, determines if there are alternate paths available, and supports load balancing across equivalent routes.

In addition, Cisco supports static routing for addresses (NSAPs) that do not conform to the addressing constraints. You must also choose static routing if you want to mix Cisco and nonCisco routers.

ISO Routing Terminology

The ISO network architecture uses terminology and concepts that differ from IP networks. In other network architectures, CLNS would be known as a *datagram service*; in the TCP/IP architecture, for example, datagram service is provided by IP.

The lowest level of the routing hierarchy is the *area*—a set of networks connected by routers (see Figure 15-1). The network itself has no separate identity, no network number; instead, the next addressable entity above the End System or Intermediate System (router) is the area.

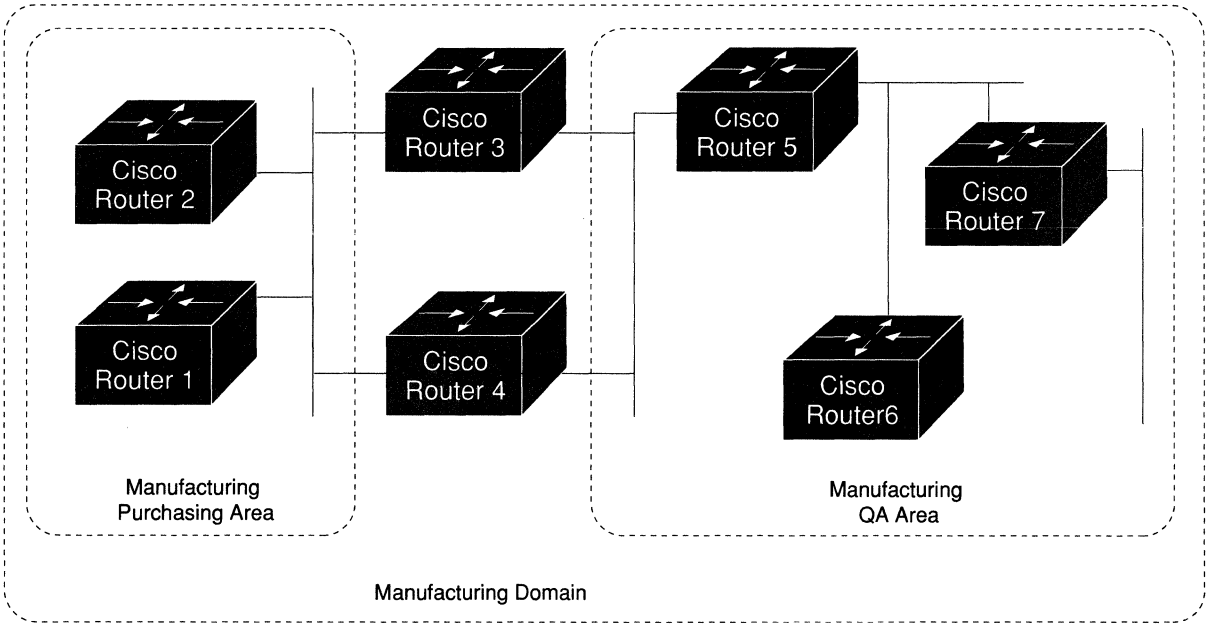


Figure 15-1 ISO CLNS Areas

Areas are connected to other areas to form *routing domains.* Each domain is a separately administered region, similar in concept to an autonomous system in IP networks.

Some intermediate systems keep track of how to communicate with all of the End Systems in their area, and thereby functions as *Level 1 routers.* Other intermediate systems keep track of how to communicate with other areas in the domain, functioning as *Level 2 routers.* Cisco routers are always Level 1 and Level 2 routers.

End Systems communicate with Intermediate Systems using the ES-IS protocol. Level 1 and Level 2 Intermediate Systems communicate with each other using the Cisco ISO IGRP protocol.

Routing across domains (inter-domain routing) may be done either statically or dynamically.

Configuring CLNS Routing

Follow these steps to configure your router for CLNS routing:

Step 1: Enable CLNS routing with the **clns routing** global configuration command.

Step 2: Create dynamic or static routing processes with the **clns router** global configuration command and either the **igrp** or **static** keyword.

Step 3: For each interface, specify which CLNS routers should be active with the **clns router** interface subcommand.

These steps enable CLNS routing. Optional commands are also available for customizing the routing environment, and you may follow these steps to complete configuration:

Step 4: Redistribute routing information.

Step 5: Map NSAP addresses to media addresses.

Step 6: Adjust ES-IS parameters, as necessary.

Each task is described in the following sections, and are followed by descriptions of the EXEC commands to maintain, monitor and debug the ISO CLNS network. Summaries of the global configuration commands and interface subcommands described in these sections appear at the end of this chapter.

CLNS Addresses

Addresses in the ISO network architecture are referred to as Network Entity Titles (NETs) and Network Service Access Points (NSAPs). Each node in an ISO network has one or more NETs. In addition, each node has many NSAPs. Each NSAP differs from one of the NETs for that node in only the last byte. This byte is called the *selector byte*, see Figure 15-2. Its function is similar to the port number in other protocol suites.

Cisco's implementation supports all NSAPs that are defined by ISO 8348/Ad2; however, Cisco provides dynamic routing only for those NSAPs that conform to the address constraints defined in the draft proposal of ISO IS-IS (ANSI DP 10589). Figure 15-2 illustrates conforming NSAP.

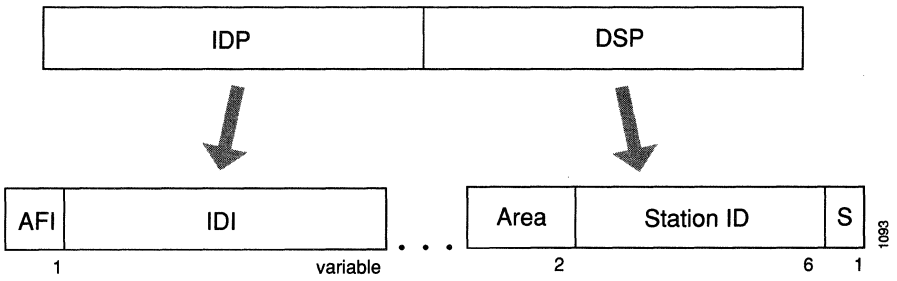


Figure 15-2 Conforming NSAP Addressing Structure

The NSAP address is formed by the Initial Domain Part (IDP), followed by the Domain Specific Part (DSP).

The IDP is made up of a one-byte Authority and Format Identifier (AFI), and a variable length Initial Domain Identifier (IDI).

The DSP is a nine-byte structure that contains a two-byte Area identifier, six-byte station ID, and one-byte selector (S) field.

Cisco's CLNS implementation interprets the bytes from the AFI up to the Area field in the DSP as a *domain identifier.* The Area field specifies the *area,* and the station ID specifies the *station.*

The domain address uniquely identifies the routing domain. Within each routing domain you can set up one or more areas. The area address uniquely identifies the area.

Addressing Rules

All NSAPs must obey the following constraints:

- No two nodes may have addresses with the same NET; that is, addresses that match all but the selector (S) field in the DSP.
- There must be at least nine bytes in the DSP portion of the address.
- No two nodes residing within the same area may have addresses in which the station ID fields are the same.

Examples:

Following are examples of OSInet and GOSIP network addresses.

OSInet:

```
47.0004.004D.0003.0000.0C00.62E6.00
|     Domain|Area|    Station ID| S|
```

GOSIP Network:

```
47.0005.80.FFFF.0000.00.FFFF.0004.0000.0C00.62E6.00
                            |Area|    Station ID| S|
```

Enabling CLNS Routing

Conceptually, each End System (ES) lives in one area. It discovers the nearest Level 1 IS (router) by listening to ES-IS packets. Each ES must be able to communicate directly with a Level 1 IS in its area.

When an ES wants to communicate with another ES, it sends the packet to the Level 1 IS for its area. The IS will look up the destination NSAP and forward the packet along the best route. If the destination NSAP is for an ES in another area, the Level 1 IS will send the packet to the nearest Level 2 IS. The Level 2 IS will forward the packet along the best path for the destination area until it gets to a Level 2 IS which is in the destination area. This IS will forward the packet along the best path inside the area until it is delivered to the destination ES.

The configuration process begins with enabling CLNS routing. You enable routing of CLNS packets using the **clns routing** global configuration command. The full syntax for this command follows.

clns routing

no clns routing

Use the **no clns routing** command to disable CLNS routing.

You need to decide now if you want to use static or dynamic routing. You must use static routing if you are inter-operating with nonCisco routers. In addition, you may want to use static routing over an X.25 link.

Configuring CLNS Static Routing

Static routing is used when it is not possible or desirable to use dynamic routing: when routing with both Cisco and nonCisco routers or when using an X.25 connection. Of course, an interface that is configured for static routing cannot reroute around failed links or load share.

Routes are entered by specifying pairs (NSAP-prefix, next-hop-NET). NET is a Network Entity Title, similar in function to an NSAP. In the routing table, the best match means the longest NSAP-prefix entry that matches the beginning of the destination NSAP. In the following static routing table the next-hop-NETs are listed for completeness, but are not necessary to understand the routing algorithm. Table 15-2 offers examples of how the routing entries in Table 15-1 can be applied to various NSAPs.

Table 15-1 Sample Routing Table Entries

Entry #	NSAP Prefix	Next Hop NET
1	47.0005.000c.0001	47.0005.000c.0001.0000.1234.01
2	47.0004	47.0005.000c.0002.0000.0231.01
3	47.0005.0003	47.0005.000c.0001.0000.1234.01
4	47.0005.000c	47.0005.000c.0004.0000.0011.01
5	47.0005	47.0005.000c.0002.0000.0231.01

Table 15-2 Hierarchical Routing Examples

Datagram Destination NSAP	Table entry # used
47.0005.000c.0001.0000.3456.01	1
47.0005.000c.0001.6789.2345.01	1
47.0004.1234.1234.1234.1234.01	2
47.0005.0003.4321.4321.4321.01	3
47.0005.000c.0004.5678.5678.01	4
47.0005.000c.0005.3456.3456.01	4
47.0005.0023.9876.9876.9876.01	5

Octet boundaries must be used for the internal boundaries of NSAPs and NETs.

You enter a specific static route by using the **clns route** global configuration command. The full syntax of this command follows:

clns route *nsap-prefix next-hop-net*

no clns route *nsap-prefix*

NSAPs that start with *nsap-prefix* are forwarded to *next-hop-net.*

This variation of the **clns route** command uses the **discard** keyword to explicitly tell a router to discard packets with the specified *nsap-prefix* (full syntax listed).

clns route *nsap-prefix* **discard**

no clns route *nsap-prefix*

Example:

This example sets a static route for a router.

```
clns route 47.0004.000c  47.0005.0001.0000.0001.0000.00
```

Defining Areas

You must enter the **clns router** global configuration command for each area the router is in. The command has this syntax:

clns router static *area-tag* **NET** *net*

no clns router static *area-tag*

The keyword **static** specifies that the router will use statically entered routes. The argument *area-tag* defines a meaningful name for an area. For example, you could define an area named *Finance* for the Finance department, and another area named *Marketing* for the Marketing department. Creating a name for an area means that you use names when configuring routing in the area, instead of having to enter the NET. Following the keyword **NET** you specify the Network Entity Title.

Use the **no clns router static** command with the appropriate arguments to remove the areas.

Example:

In the following example, you are configuring a router in the area *sales*:

```
clns router static sales NET 47.0005.0001.0000.0001.0000.00
```

Each area an interface is in must be specified with the **clns router** interface subcommand. The full syntax of this command is listed next:

clns router static *area-tag*

no clns router static *area-tag*

The keyword **static** specifies static routing. The argument *area-tag* is the tag defined for the NET using the **clns router** global configuration command. Use the **no clns router static** command to remove the definition.

Example:

In this example, an interface is being configured for static routing in the marketing area:

```
clns router static marketing
```

Configuring CLNS Over X.25

X.25 is not a broadcast medium, and therefore ES-IS is not used to automatically advertise and record NSAP/NET (protocol address) to SNPA (media address) mappings. Operation of CLNS over X.25 requires that this mapping information be statically entered, by using the **clns is-neighbor** and/or the **clns es-neighbor** interface subcommands.

Configuring a serial line to use CLNS over X.25 requires configuring the general X.25 information and the CLNS-specific information. The general X.25 information must be configured first. Then, the **clns is-neighbor** and **clns es-neighbor** interface subcommands are issued to list all the Intermediate and End Systems that will be used. Full command syntax for these commands follows.

clns es-neighbor *nsap snpa* [*X.25-facilities-info*]
no clns es-neighbor *nsap*

clns is-neighbor *nsap snpa* [*X.25-facilities-info*]
no clns is-neighbor *nsap*

In this case, the Subnet Points of Attachment (SNPAs) are the X.25 network addresses (X.121 addresses). These are usually assigned by the X.25 network provider. Use the argument *X.25-facilities-info* to specify nondefault packet and window size, reverse charge information, and so on.

Example 1:

This command maps NSAP *47.0004.004d.3132.3334.3536.00* to X.121 address *310117*.

```
clns es-neighbor 47.0004.004d.3132.3334.3536.00 310117
```

Only one **es-neighbor** or one **is-neighbor** entry can be made for each X.121 address.

The X.25 facilities information that can be specified is exactly the same as in the **x25 map** subcommand described in the section "Setting Address Mappings" in Chapter 8.

You can specify the following information:

- Packet window size (send and receive)
- Packet size (send and receive)
- Reverse charges requested
- Reverse charges accepted
- Closed user group
- Maximum number of virtual circuits (VCs) to open
- Broadcasts

Example 2:

Following is a more complicated example that specifies nondefault packet and window size:

```
clns is-neighbor 47.0004.0021.0001.0000.0000.00 3101 windowsize 7 7 packetsize
512 512
```

Note: All of this configuration command must be given on one line.

The system does not accept an X.25 call unless the source of the call has been configured with these commands. The system will not accept a call requesting reverse charges unless the keyword **accept-reverse** is used as follows.

```
clns is-neighbor 47.0005.0000.0001.0000.00 310120249 accept-reverse
```

All of the information that is entered using the **clns is-neighbor** and **clns es-neighbor** subcommands actually goes into two places in the system: the ES-IS table, and the X.25 map table. The ES-IS table stores only the NSAP/NET and X.121 address information.

The X.25 map table stores this information but, in addition, stores the facility information. If a virtual circuit (VC) has been established, the logical channel numbers (LCNs) in use are shown.

A configuration example using an X.25 link is included in the section "CLNS Configuration Examples."

Configuring CLNS Dynamic Routing

Use the commands described in this section to configure CLNS dynamic routing. In the previous section, you turned on CLNS processing in the router. Now you need to identify the area the router will work in to create a routing process. You do this with the **clns router** global configuration command.

clns router igrp *area-tag* **NET** *net*

The keyword **igrp** specifies dynamic routing using the IGRP protocol. You can specify up to ten IGRP processes. The argument *area-tag* defines a meaningful name for an area. For example, you could define an area named *Finance* for the Finance department, and another area named *Marketing* for the Marketing department. Creating a name for an area means that you use names when configuring routing in the area, instead of having to enter the NET. Follow the keyword **NET** with the Network Entity Title.

Example:

In the following example, a router with its NET is specified in the *Marketing* area. (The command must be typed on one line.)

```
clns router igrp marketing NET 47.0004.0021.0001.0000.0000.00
```

Each interface in an area must also have a **clns router** interface subcommand entry. Following is the full command syntax for this command:

clns router igrp *area-tag* [**level2**]

no clns router igrp *area-tag*

The argument *area-tag* is the tag defined for the NET using the **clns router** global configuration command above.

If you want this interface to advertise Level 2 information only, use the **level2** keyword. The purpose of this option is to reduce the amount of router to router traffic by specifying the routers to only send out Level 2 routing updates on certain interfaces. The Level 1 information will not be passed between the routers with the Level 2 option set.

Use the **no clns router igrp** command with the appropriate area tag to disable the CLNS routing protocol on the interface.

Example:

In the following example, the interface will advertise Level 2 information only on this interface serial 0:

```
interface serial 0
clns router igrp marketing level2
```

Inter-Domain Dynamic Routing

A router may be configured to do inter-domain dynamic routing by putting it into two domains and configuring the router to redistribute the routing information between the domains. Router configured this way are referred to as *border* routers.

When configuring inter-domain dynamic routing, keep in mind that routers that *are* border routers must have at least two ISO CLNS routing processes defined in two different domains.

If you have a router which is in two routing domains, you may want to redistribute routing information between the two domains. This is done with this variation of the **clns router igrp** global configuration command. The following lists the full command syntax:

clns router igrp *tag1* **redistribute domain** *tag2*

no clns router igrp *tag1* **redistribute domain** *tag2*

The keywords **redistribute domain** enable the redistribution. The arguments *tag2* and *tag1* are the defined tags for the areas in which the routing information is to be redistributed. When enabled, this command causes the area defined by *tag1* to redistribute information learned from domain *tag2*.

The **no clns router igrp** command with the appropriate arguments and keywords stops redistribution.

Example:

In the example below, information learned in the area named *MktDevelop* will be redistributed to the *CustAdv* areas.

```
clns router igrp custadv redistribute domain mktdevelop
```

The section "CLNS Configuration Examples" later in this chapter illustrates how to configure CLNS inter- and intra-domain dynamic routing.

Redistributing Static Routes

The following variation of the **clns router igrp** global configuration command causes the router to advertise static CLNS routes in the domain. Its full syntax is as follows:

clns router igrp *area-tag* **redistribute static**

no clns router igrp *area-tag* **redistribute static**

The argument *area-tag* is the defined tag for the area in which static routing information is to be advertised.

Use the **no clns router igrp** command with the appropriate arguments and keywords to remove the router from the list.

Example:

In the following example, the router will advertise any static routes it knows about in the `ship` domain.

```
clns router igrp ship redistribute static
```

The keyword **redistribute** injects the static routes into the routing domain. This eliminates the need for each router in a domain to have a static route configured for each destination. The argument *area-tag* is the tag defined for the NET using the **clns router** global configuration command.

Note: Only the router that advertises the static route needs to have a **redistribute** configuration command defined.

CLNS Configuration Examples

This section provides configuration examples of both intra- and inter-domain static and dynamic routing.

Basic Static Routing

Configuring FDDI, Ethernets, Token Rings, and serial lines using HDLC encapsulation for CLNS can be as simple as just enabling CLNS on the interfaces. This is all that is ever required on serial lines using HDLC encapsulation. If all systems on an Ethernet or Token Ring support ISO 9542 ES-IS, then nothing else is required as well. In this case, an Ethernet and a serial line can be configured as in the following example.

Example:

```
clns routing
clns router static Ydivision NET 47.0004.004D.0055.0000.0C00.BF3B.00
interface ethernet 0
clns router static Ydivision
interface serial 0
clns router static Ydivision
```

Systems Not Using ES-IS

If there are systems on the Ethernet which do not use ES-IS, or if X.25 is being used, NSAP/NET (protocol address) to SNPA (media address) mappings must be specified, using the **clns is-neighbor** and/or the **clns es-neighbor** interface subcommands.

Example:

An example of configuring an Ethernet interface with the Ethernet address (MAC address) of systems that do not use ES-IS follows.

```
interface ethernet 0
clns es-neighbor 47.0004.004D.0055.0000.00C0.A45B.00  0000.00C0.A45B
```

In this case, the End Systems with the NSAP (or NET) listed below is configured at an Ethernet MAC address of 0000.00C0.A45B.

```
47.0004.004D.0055.0000.00C0.A45B.00
```

Note: It is only necessary to use static mapping for those End Systems that do *not* support ES-IS. The router will continue to dynamically discover those End Systems that *do* support ES-IS.

Static Routing

The following is a more complete example of CLNS static routing on a system with two Ethernet interfaces. After configuring routing, you define an NET and enable CLNS on the Ethernet 0 and Ethernet 1 interfaces. You must then define an IS-neighbor and define a static route with the **clns route** command, as shown.

Example:

```
clns routing
clns router static Xdivision net 47.0004.004D.0055.0000.0C00.3F3B.00
interface Ethernet 0
clns router static Xdivision
interface Ethernet 1
clns router static Xdivision
clns is-neighbor 47.0005.0001.0000.0001.0000.00  0000.0C00.62E7
clns route 47.0004.000c  47.0005.0001.0000.0001.0000.00
```

Static Intra-Domain Routing

This example demonstrates how to use static routing inside of a domain. Imagine a company with two branch offices in Detroit and Chicago connected with an X.25 link. These offices are both in the domain named *Sales*.

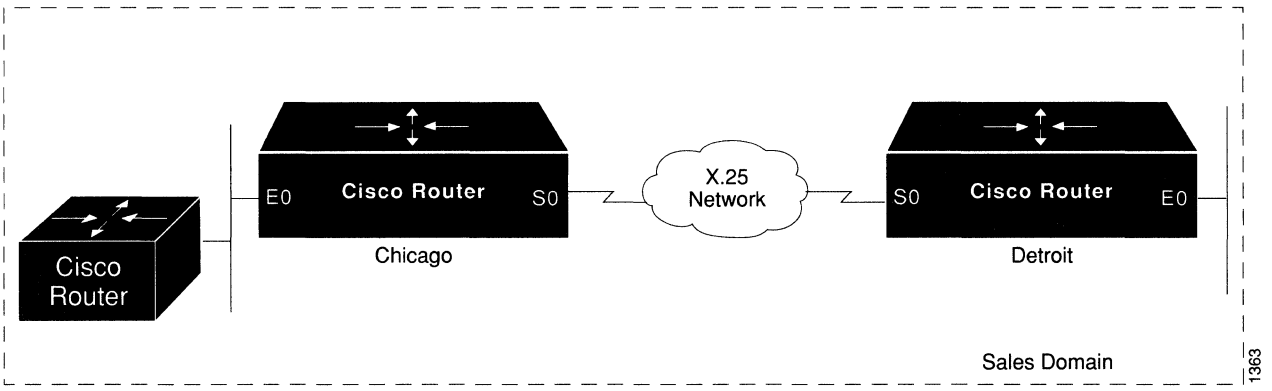


Figure 15-3 CLNS Static Intra-Domain Routing

The following is one way to configure the router in Chicago.

Example 1:

```
clns host chicago 47.0004.0050.0002.0000.0c00.243b.00
clns host detroit 47.0004.0050.0001.0000.0c00.1e12.00
clns routing
clns router igrp sales net chicago
interface ethernet 0
clns router igrp sales
interface serial 0
encapsulation x25
x25 address 031342174523156
x25 nvc 4
clns router igrp sales
clns is-neighbor detroit 031343136931281 broadcast
```

This configuration will bring up an X.25 virtual circuit between the router in Chicago and the router in Detroit. Routing updates will be sent across this link. This implies that the virtual circuit could be up continuously.

If this is undesirable, use the following configuration instead.

Example 2:

```
!
clns host chicago net  47.0004.0050.0002.0000.0c00.243b.00
clns host detroit 47.0004.0050.0001.0000.0c00.1e12.00
clns router igrp sales net chicago
!
interface ethernet 0
clns router igrp sales
!
interface serial 0
encapsulation x25
x25 address 031342174523156
x25 nvc 4
clns enable
clns is-neighbor detroit 031343136931281
!
clns route 47.0004.0050.0001 detroit
!
```

If the Chicago office should grow to contain multiple routers, it would be appropriate for each of those routers to know how to get to Detroit. Add the following command to redistribute information between the routers:

```
clns router igrp sales redistribute static
```

Static Inter-Domain Routing

The following sample illustrates how to configure two routers that distribute information across domains. In this example, Castor and Pollux communicate across a serial link.

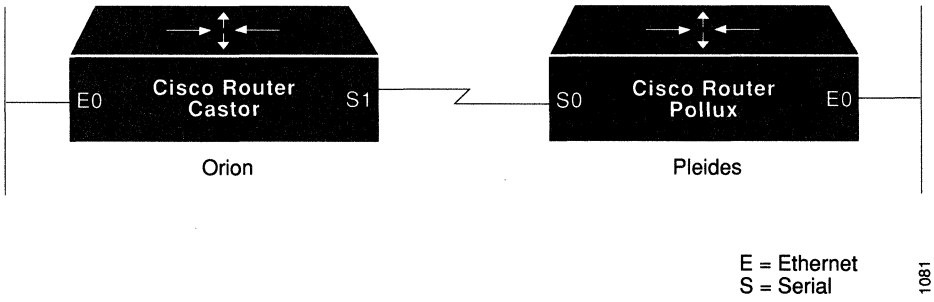


Figure 15-4 CLNS Inter-Domain Static Routing

Example for Castor:

```
clns router igrp orion NET
47.0006.02.000000.0000.0100.0001.010203040506.00
!
clns host pollux 47.0006.02.000000.0000.0200.0003.111213141516.00
interface ethernet 0
clns router igrp orion
interface serial 1
clns enable
clns is-neighbor pollux
clns route 47.0006.02.000000.0000.0200 pollux
```

Example for Pollux:

```
clns router igrp pleiades NET
47.0006.02.000000.0000.0200.0003.111213141516.00
!
clns host orion 47.0006.02.000000.0000.0100.0001.010203040506.00
interface ethernet 0
clns router igrp pleiades
interface serial 0
clns enable
clns is-neighbor orion
clns route 47.0006.02.000000.0000.0100 orion
```

CLNS routing updates will not be sent on the serial link; however, CLNS packets will be sent and received over the serial link.

Routing Within the Same Area

This example illustrates how to configure dynamic routing **within** a routing domain. The router may exist in one or more areas within the domain. The router named *Castor* exists in a single area.

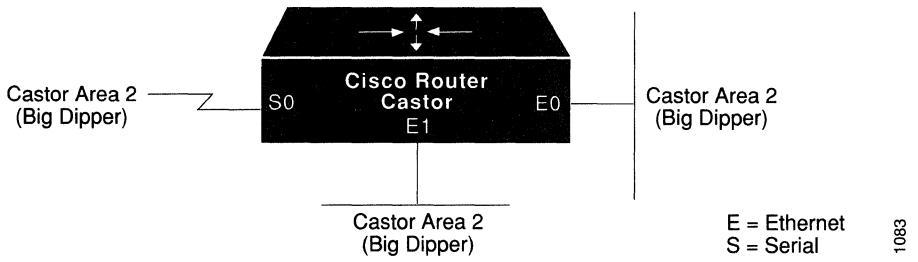


Figure 15-5 CLNS Dynamic Routing — Within a Single Area

Example:

```
clns routing
clns router igrp bigdipper NET 47.0004.004D.0002.010203040506.00
interface Ethernet 0
clns router igrp bigdipper
interface Ethernet 1
clns router igrp bigdipper
interface Serial 0
clns router igrp bigdipper
```

Routing in More Than One Area

The following example illustrates how to configure a router named Castor that exists in two areas.

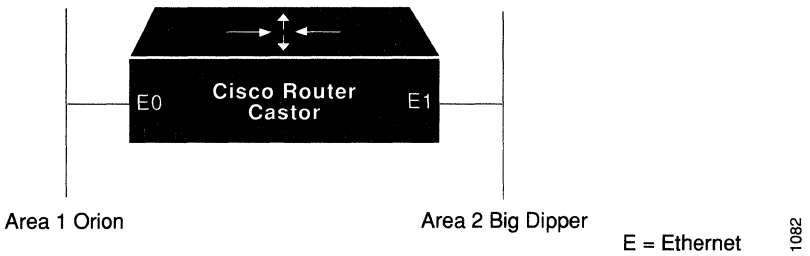


Figure 15-6 CLNS Dynamic Routing—Within Two Areas

Example:

```
!
clns routing
clns router igrp orion NET 47.0004.004D.0001.212223242526.00
clns router igrp bigdipper NET 47.0004.004D.0002.212223242526.00
interface ethernet 0
clns router igrp orion
interface ethernet 1
clns router igrp bigdipper
```

Overlapping Areas

The following example illustrates how to configure a router with overlapping areas.

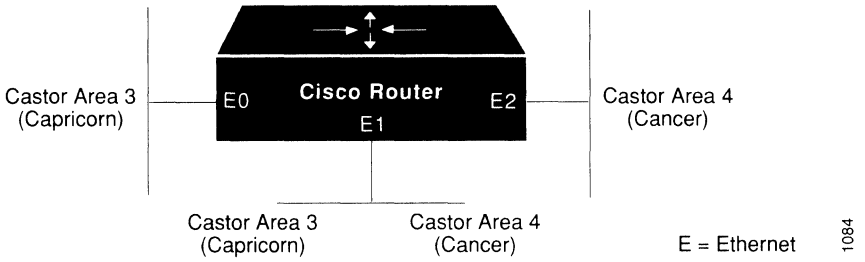


Figure 15-7 CLNS Dynamic Routing— Within Overlapping Areas

Example:

```
clns routing
clns router igrp capricorn NET 47.0004.004D.0003.010203040506.00
clns router igrp cancer NET 47.0004.004D.0004.010303040506.00
interface ethernet 0
clns router igrp capricorn
interface ethernet 1
clns router igrp capricorn
clns router igrp cancer
interface ethernet 2
clns router igrp cancer
```

Dynamic Inter-Domain Routing

The following example illustrates how to configure three domains that want to be transparently connected.

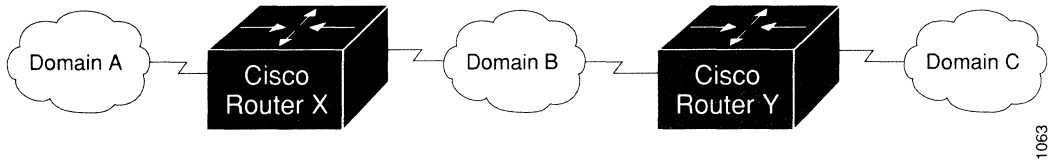


Figure 15-8 CLNS Inter-Domain Dynamic Routing

Example for Router X:

```
clns routing
clns router igrp A NET 47.0006.02.000000.0000.0100.0002.010201040506.00
clns router igrp B NET 47.0007.02.000000.0000.0100.0003.010201040506.00
clns router igrp A redistribute domain B
clns router igrp B redistribute domain A
interface serial 0
clns router igrp A
interface serial 1
clns router igrp B
```

Example for Router Y:

```
clns routing
clns router igrp B NET 47.0007.02.000000.0000.0100.0004.010201040506.00
clns router igrp C NET 47.0008.02.000000.0000.0100.0005.010201040506.00
clns router igrp B redistribute domain C
clns router IGRP C redistribute domain B
interface serial 0
clns router igrp B
interface serial 1
clns router igrp C
```

Router X will inject a prefix route for domain A into domain B. Domain B will inject this prefix route plus one for domain B into domain C.

You can also configure a border router between domain A and domain C.

Configuring ES-IS Parameters

This section describes the commands used to configure the ES-IS parameters for device-router communication. In general, however, these should be left at their default values.

When configuring an ES-IS router, be aware that:

- ES-IS does not run over X.25 links.
- ES HELLO (ESH) packets and IS HELLO (ISH) packets are sent without options. Options in received ESH and ISH packets are ignored.

Specifying HELLO Packets

The **clns configuration-time** global configuration command specifies the rate at which ESH and ISH packets are sent.

clns configuration-time *seconds*

no clns configuration-time

The default value for *seconds* is 60, and this value is restored by the **no clns configuration-time** command.

The **clns holding-time** subcommand allows the sender of an ESH or ISH to specify the length of time during which the information in the HELLO packets will be believed.

clns holding-time *seconds*

no clns holding-time

The argument *seconds* specifies the time in seconds. The default value is 300 seconds (five minutes), which is restored by the **no clns holding-time command**.

Configuring Static Configuration of ESs

End Systems need to know how to get to a Level 1 IS for their area and Level 1 ISs need to know all of the ESs that are directly reachable through each of their interfaces. To provide this information, Cisco routers support the ES-IS protocol. A Cisco router dynamically discovers all ESs running the ES-IS protocol. ESs which are not running the ISO IGRP protocol must be statically configured. This is done using the **clns es-neighbor** interface subcommand:

clns es-neighbor *nsap snpa*

no clns es-neighbor *nsap*

The argument *nsap* specifies the CLNS address. The argument *snpa* specifies the data link address. The **no** keyword deletes the ES neighbor. There must be a static **clns es-neighbor** subcommand entry for each End System that does not support ES-IS. If you have configured any ISO IGRP routers using the **clns router igrp** subcommand, the Cisco ES-IS routing software automatically turns ES-IS on.

It is sometimes desirable for a router to have a neighbor entry statically configured rather than learning it through ES-IS. The **clns is-neighbor** interface subcommand enters an IS neighbor.

clns is-neighbor *nsap snpa*

no clns is-neighbor *nsap*

The argument *nsap* specifies the NSAP address. The argument *snpa* specifies the data link Subnet Point of Attachment (SNPA) MAC-layer address. The **no clns is-neighbor** command deletes the specified IS neighbor.

Configuring Performance Parameters

Generally, you do not need to change the default settings for CLNS packet switching, but there are some modifications you can make when you decide that it is advantageous.

Specifying the MTU Size

All interfaces have a default maximum packet size. You can set the maximum transmission unit (MTU) size of the packets sent on the interface using the **clns mtu** interface subcommand. The full syntax of this command follows.

clns mtu *size*

no clns mtu

The minimum value for the *size* argument is 512; the default and maximum packet size depends on the interface type. The default value is restored by the **no clns mtu** command.

Configuring Checksums

When the ISO CLNS routing software sources a CLNS packet, by default it generates checksums. The **clns checksum** interface subcommand specifies this function. Use the **no clns checksum** command to disable checksum generation.

clns checksum

no clns checksum

Enabling Fast Switching

The **clns route-cache** interface subcommand allows fast switching through the cache, and by default, is enabled. To disable fast switching, use the **no clns route-cache** command.

clns route-cache

no clns route-cache

Note: The cache still exists and is used after the no clns route-cache command is used; the software just does not do fast switching through the cache.

Setting the Congestion Threshold

If a router configured for CLNS experiences congestion, it sets the congestion experienced bit. The congestion threshold is a per-interface parameter set by the **clns congestion-threshold** interface subcommand. The full syntax for this command follows.

clns congestion-threshold *number*

no clns congestion-threshold *number*

This subcommand causes the system to set the congestion experience. bit if the output queue has more than the specified number of packets in it. A *number* value of zero or the **no** keyword prevents this bit from being set. The default value for *number* is 4.

Use the **no clns congestion-threshold** command with the appropriate value to remove the parameter setting.

Transmitting Error PDUs

When a CLNS packet comes in, the routing software looks in the routing table for the next hop. If it does not find the next hop, the packet is discarded and an error Protocol Data Unit (ERPDU) may be sent.

Sending an Error PDU

The **clns send-erpdu** interface subcommand allows CLNS to send an error PDU when it detects an error in a data PDU, and by default, is enabled. To disable this function, use the **no clns send-erpdu** command. The syntax for both commands follows.

clns send-erpdu

no clns send-erpdu

Determining the Interval Between ERPDUs

The **clns erpdu-interval** interface subcommand determines the minimum interval time, in milliseconds, between ERPDUs. The full syntax of this command follows.

clns erpdu-interval *milliseconds*

no clns erpdu-interval *milliseconds*

A *milliseconds* value of zero or the **no clns erpdu-interval** command turns off the interval rate and effectively sets no limit to the ERPDU rate. The default rate is once every ten milliseconds.

The **clns erpdu-interval** subcommand will not send ERPDUs more frequently than one per interface per ten milliseconds. If a packet is sent out the same interface it came in on, a redirect PDU (RDPDU) may also be sent to the sender of the packet.

Redirecting PDUs

The **clns send-rdpdu** interface subcommand allows CLNS to send redirect PDUs when a better route for a given host is known, and this is the default behavior. The full syntax of the command follows.

clns send-rdpdu

no clns send-rdpdu

To disable this function, use the **no clns send-rdpdu** command.

Disabling RDPDUs

An RDPDU may be disabled on a per-interface basis using the **clns rdpdu-interval** interface subcommand. The full syntax of the command follows.

clns rdpdu-interval *milliseconds*

no clns rdpdu-interval *milliseconds*

The command determines the minimum interval time, in milliseconds, between RDPDUs. A *milliseconds* value of zero or the **no clns rdpdu-interval** command turns off the interval rate and effectively sets no limit to the RDPDU rate. The default rate is once every 100 milliseconds. An RDPDU is rated-limited, and is not sent more frequently than one per interface, per 100 milliseconds.

Disabling the RDPDU Mask

The address mask is normally present on all RDPDUs, but can be disabled with the **no clns rdpdu-mask** interface subcommand. The full syntax of the **clns rdpdu-mask** command follows.

clns rdpdu-mask

no clns rdpdu-mask

Note: SNPA masks are never sent, and RDPDUs are ignored by Cisco routers when the router is acting as an IS.

Configuring Parameters for Locally Sourced Packets

Use these commands to configure parameters for packets sourced by this router. Full command syntax for is command is listed with the descriptions. The **no** forms of the commands remove the parameters settings.

The **clns packet-lifetime** global configuration command specifies the initial lifetime for locally generated packets.

clns packet-lifetime *number*

no clns packet-lifetime *number*

The default value for *number* is 64.

The **clns want-erpdu** interface subcommand specifies whether to request error PDUs on packets sourced by the router.

clns want-erpdu

no clns want-erpdu

The default is to request error PDUs.

Header Options

The ISO CLNS routing software ignores the Record Route option, the Source Route option, and the QOS (quality of service) option other than congestion experienced. The security option causes a packet to be rejected with a bad option indication.

NSAP Shortcut Command

The clns host global configuration command can be useful for creating a name for an NSAP. This name can then be used in place of typing the long set of numbers associated with an NSAP. The command syntax follows.

clns host *name nsap*

The argument *name* is the desired name for the NSAP, which is specified by the *nsap* argument.

Maintaining a CLNS Network

Use the EXEC commands described in this section to maintain the ISO CLNS caches, tables, and databases.

clear clns cache

The EXEC command **clear clns cache** clears and re-initializes the CLNS routing cache.

clear clns route

The command **clear clns route** removes all of the dynamically derived CLNS routing information.

Monitoring a CLNS Network

Use the EXEC commands described in this section to obtain displays of activity on the ISO CLNS network.

Displaying General CLNS Information

The show clns command displays information about the CLNS network. Enter this command at the EXEC prompt:

show clns

Sample output follows.

```
Global CLNS Information:
  1 interfaces enabled for clns
 NET: 47.0004.0001.2122.2324.2526.00
  Configuration Timer: 60, default holding timer: 300, packet lifetime 64
  ERPDU's requested on locally generated packets
  Intermediate system operation enabled (forwarding allowed)
Level 1 Router: AREATWO
   Routing for domain: 47.0004 area: 0001
Level 2 Router: DOMAIN_AREATWO
   Routing for domain: 47.0004
```

In the display:

- The first line indicates how many interfaces have the CLNS routing protocol turned on.
- The second line contains the NET for this router. Note that there may be more than one NET for one router.
- The `Configuration Timer` field displays the frequency with which the router will send out IS HELLO packets. The number following the `default holding timer` field is the length of time the timer will remember ES HELLO packets. The `packet lifetime` displayed is the default value used in packets sourced by this router.
- The next line indicates whether error PDUs (ERPDUs) will be requested for packets sourced by the router.
- The last line of global information indicates whether or not this router is configured to be an End System or an Intermediate System. (It is not generally useful for a customer to configure a router to be an End System.)
- The last lines of this display list the areas and domains that this router is in.

Displaying CLNS Routes

Use the EXEC command **show clns route** to display all of the destinations to which this router knows how to route packets. Enter this command at the EXEC prompt:

show clns route [*nsap*]

Destinations are sorted by category. The optional argument *nsap* specifies the CLNS address.

Sample output follows:

```
     AREATWO: <47.0004><0001>
I B1.B2B3.B4B5.B6
    ->47.0004.0001.3132.3334.3536.00, HT 128  metric: 44C
     DOMAIN_AREATWO: <47.0004><0001>
I 0003
    ->47.0004.0002.6162.6364.6566.00, HT 119  metric: 2180
     Prefix Routes
S 47.0004.5555
    ->47.0004.0001.A1A2.A3A4.A5A6.00, <permanent>metric: 0
```

In this display:

- The ES in the same area as the router are displayed first, followed by the paths to other areas in this domain. Finally, paths to other domains are displayed. All of these are listed in a similar format. The first field lists the way that the route was discovered: `I` stands for IGRP; `S` stands for Static.
- The second field lists the destination. Station IDs are used for ESes; Area numbers are used for other areas in the domain; and prefixes are used for destinations outside of this domain.
- The next field is the first hop that packets for this destination will take.
- Next is the `HT` (hold time) field which lists the number of seconds this route will be remembered. If the route was statically entered and will not be forgotten, the string `<permanent>` will be displayed instead.
- The final field is the metric that is used for this route.

Neighbors are not included in the **show clns route** display.

Displaying the CLNS Routing Cache

Use the EXEC command **show clns cache** to display the CLNS routing cache. Enter this command at the EXEC prompt:

show clns cache

The cache contains an entry for each destination that has packet switching enabled.

Following is sample output:

```
Cache version 4
47.0004.0001.2122.2324.2526.00
47.0004.0001.3132.3334.3536.00 ->
   47.0004.0001.3132.3334.3536.00@Ethernet0:0000.0C00.62E6
47.0004.5555.3232 ->
   47.0004.0001.A1A2.A3A4.A5A6.00@Ethernet0:0000.0C00.5153
```

In the display, there will be an entry in the cache for each destination that the Cisco router has switched a packet for in the recent past. This includes the Cisco router. Each entry has the following format:

```
destination → first hop address@interface:MAC-layer address
```

Displaying CLNS Traffic

Use the **show clns traffic** command to list the CLNS packets this router has seen. Enter this command at the EXEC prompt:

show clns traffic

Sample output follows:

```
CLNS & ESIS Output: 177, Input: 431
CLNS Local: 0, Forward: 1
CLNS Discards:
  Hdr Syntax: 0, Checksum: 0, Lifetime: 0, Output cngstn: 0
  No Route: 0, Dst Unreachable 0, Encaps. Failed: 0
  NLP Unknown: 0, Not an IS: 0
CLNS Options: Packets 7, total 7, bad 0, GQOS 0, cngstn exprncd 0
CLNS Segments:  Segmented: 0, Failed: 0
CLNS Broadcasts: sent: 0, rcvd: 0
Echos: Rcvd 10 requests, 4 replies
      Sent 68 requests, 10 replies
ESIS(sent/rcvd): ESHs: 0/18, ISHs: 19/30, RDs: 1/0, QCF: 0/0
ISO IGRP:Querys (sent/rcvd): 0/0 Updates (sent/rcvd): 60/103
Router Hellos: (sent/rcvd): 68/127
```

In the display:

- The first line lists the total number of packets that this router has sent (the `Output` field) and received (the `Input` field).
- The `CLNS Local` field lists the number of packets that were generated by this router.
- The `CLNS Forward` field lists the number of packets that this router has forwarded.
- The `CLNS Discards` field lists the packets that CLNS has discarded, along with the reason for the discard.
- The `CLNS Options` field lists the options that have been seen in CLNS packets.
- The `CLNS Segments` field lists the number of packets that have been segmented and the number of failures that occurred because a packet could not be segmented.
- The `CLNS Broadcasts` field lists the number of CLNS broadcasts that have been sent and received.
- The `Echos` field lists the number of echo request packets and echo reply packets that have been received. The line following this field lists the number of echo request packets and echo reply packets that have been sent.
- The `ESIS (sent/rcvd)` field lists the Number of ESH, ISH, and Redirects sent and received.
- The `ISO IGRP` field lists the number of IGRP queries, and updates sent and received.
- The `Router Hellos` field lists the number of IGRP router HELLO packets which have been sent and received.

Displaying CLNS Redirect Information

The **show clns redirects** command displays CLNS redirect information. Enter this command at the EXEC prompt:

show clns redirects

Only ESs maintain redirect information.

Displaying Status About Specific Interfaces

The **show clns interface** command lists the CLNS-specific information about each interface, and is entered at the EXEC prompt, as follows:

show clns interface [*interface unit*]

Following is sample output:

```
Ethernet 0 is up, line protocol is up
  Checksums enabled, MTU 1500, Encapsulation ISO1
  Next ESH/ISH in 20 seconds
  ERPDUs enabled, min. interval 10 msec.
  RDPDUs enabled, min. interval 100 msec., Addr Mask enabled
  Congestion Experienced bit set at 4 packets
   Routing Domain/Area: <47.0004><0001>
  CLNS fast switching enabled
```

In the display:

- The `Checksums enabled` field may be enabled or disabled. The number following `MTU` is the maximum transmission size for a packet on this interface. The `Encapsulation` field will always be ISO1.
- The `Next` field displays when the next ESH or ISH will be sent on this interface.
- The next line displays information about the generation of error PDUs (`ERPDUs`). They may be either enabled or disabled. If they are enabled, they will be sent out no more frequently than the specified interval.
- The next line provides information about the generation of redirect PDUs (`RDPDUs`). They may be either enabled or disabled. If they are enabled, they will be sent out no more frequently than the specified interval. If the address mask is enabled, redirects will be sent out with an address mask.
- The next line tells when CLNS will turn on the congestion experienced bit. The default is to turn this bit on when there are more than four packets in a queue.
- The next line lists the areas that this interface is in. In most cases, an interface will be in only one area.
- The last line displays whether or not fast switching is supported for CLNS on this interface.

Displaying CLNS ES Neighbors

The **show clns es-neighbor** command lists the ES neighbors that this router knows about. Enter this command at the EXEC prompt:

show clns es-neighbor

Following is sample output:

```
   AREATWO: <47.0004><0001>
E ES 41.4243.4445.46, Ethernet0:0000.0C00.40AF, HT 247
S ES A1.A2A3.A4A5.A6, Ethernet0:0000.0C00.5153, <permanent>
```

In the display, the neighbors are sorted by which area they are in. The area tag, domain, and area address are printed first. Below them, all of the neighbors are listed.

Each ES neighbor entry consists of the following fields:

- The first field lists the way that this neighbor was discovered. There are several possible values for End Systems: `E` (ES-IS), `S` (Static), `D` (DECnet Phase IV).
- The second field lists the type of neighbor; `ES` is an End System.
- The next field lists the station address for this neighbor.
- The next field lists the interface which is used to communicate with this neighbor and its MAC layer address.
- The `HT` (hold time) field in the last line lists the number of seconds the neighbor information will be held. If the neighbor was statically entered, the string `<permanent>` will be displayed instead.

Displaying CLNS IS Neighbors

The **show clns is-neighbor** command displays neighbor entries sorted by which area they are in. Enter this command at the EXEC prompt:

show clns is-neighbors

Following is sample output:

```
     AREATWO: <47.0004><0001>
I IS 01.0203.0405.06, Ethernet0:0000.0C00.3E29, HT 0
I IS 31.3233.3435.36, Ethernet0:0000.0C00.62E6, HT 42
     DOMAIN_AREATWO: <47.0004>
I IS 0002:61.6263.6465.66, Ethernet0:0000.0C00.3E29, HT 48
     nonconforming neighbors
E IS 47.0005.0001.6162.6364.6566.00, Ethernet0:0000.0C00.3E29, HT 288
```

In the display:

- The first field lists the way that this neighbor was discovered. There are several possible values for this field: `E` (ESIS), `S` (Static), `I` (IGRP), `D` (DECnet Phase IV).
- The second field lists the type of neighbor; `IS` is Intermediate System.

- If the neighbor is in one of our areas, the next field lists the station address. If the neighbor is in our domain, the next field lists the area and station address. If the neighbor is a nonconforming neighbor, its entire NET is listed.
- The next field lists the interface and MAC-layer address used to communicate with that neighbor.
- The HT (hold time) field in the last line lists the number of seconds the neighbor information will be held. If the neighbor was statically entered, the string <permanent> will be displayed instead.

Some neighbors are in our domain, but not in one of our areas and these are listed under the domain. Finally, some neighbors are not in our domain at all. These are listed under the nonconforming neighbors field.

Displaying ES and IS Neighbors

The **show clns neighbors** command displays both ES and IS neighbors. Enter this command at the EXEC prompt:

show clns neighbors

This display is a composite of the **show clns es-neighbor** and **show clns is-neighbor** commands.

Displaying Protocol-Specific Information

The **show clns protocol** command lists the protocol-specific information for each IGRP routing process in this router. Enter this command at the EXEC prompt:

show clns protocol [*domain*]

There will always be at least two routing processes, a Level 1 and a Level 2, and there may be more. The optional argument *domain* specifies a particular routing domain.

Following is sample output:

```
Level 1 Router: AREATWO
   Routing for domain: 47.0004 area: 0001
   Sending Updates every 45 seconds. Next due in 42 seconds
   Invalid after 135 seconds,
   Hold down for 145 seconds
   Sending Router Hellos every 17 seconds. Next due in 10 seconds
   Invalid after 51 seconds,
   IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
   Interfaces in domain/area:
        Ethernet0
    -More-
Level 2 Router: DOMAIN_AREATWO
   Routing for domain: 47.0004
   Sending Updates every 45 seconds. Next due in 26 seconds
   Invalid after 135 seconds,
   Hold down for 145 seconds
   Sending Router Hellos every 17 seconds. Next due in 11 seconds
```

```
Invalid after 51 seconds,
IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
Interfaces in domain/area:
     Ethernet0
```

In the display:

- The first line provides the domain address and area number for Level 1 routing processes. For Level 2 routing processes, this command lists the domain address.
- The next set of fields indicate some of the protocol timers. The field labeled `Sending` updates displays when the next routing updates will be sent.
- The `Invalid` field indicates how long routing updates are to be believed.
- The `Hold Down` field indicates how long a route will be held down before new information is to be believed.
- The `Sending Router Hellos` field indicates how often the routers will send hellos to each other and when the next is due.
- The field labeled `Invalid` indicates how long a neighbor entry will be remembered.
- The `IGRP metric weight` displays lists the weights applied to the various components of the metric. These fields are followed by the list of interfaces that are in this area.

The ISO CLNS Ping Command

The OSI Connectionless Network Protocol (ISO 8473) does not specify a network-level echo protocol. The Internet Engineering Task Force (IETF) has specified and proposed such a protocol in RFC 1139. Cisco has implemented this specification using the proposed new PDU types, Echo Request Selector (1E), and Echo Reply Selector (1F). noncisco routers may or may not forward these packets, depending on whether they are specific about the packet types they will forward. End Systems will not recognize these packets, but will typically generate an error packet (ERPDU) as a response. This ERPDU is useful, as it confirms the reachability of the end system. Table 15-3 lists the characters displayed during the **ping** test and their meaning.

Table 15-3 Ping Test Characters

Char	Meaning
!	Each exclamation point indicates receipt of a reply.
.	Each period indicates the network server timed out while waiting for a reply.
U	A destination unreachable, error PDU was received.
C	A congestion experienced packet was received.
I	User interrupted test.
?	Unknown packet type.
&	Packet lifetime exceeded.

The output concludes with the success rate and minimum, average, and maximum round-trip times. To abort a **ping** session, type the escape sequence (by default, Ctrl-^, X).

Example 1:

The following example uses a name to specify the source.

```
Protocol [ip]: clns
Target CLNS address: thoth
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 100-byte CLNS Echos to
55.0006.0100.0000.0000.0001.8888.1112.1314.151
6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 112/113/116 ms
```

Example 2:

In this example, an NET address is specified.

```
Protocol [ip]: clns
Target CLNS address: 47.0004.0050.0002.0000.0c00.243b.00
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 100-byte CLNS Echos to 47.0004.0050.0002.0000.0C00.243B.00,
timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 1/4/8 ms
```

The ISO CLNS Trace Command

The ISO CLNS **trace** command allows you to discover the path packets are taking through your network. It sends Probe packets and takes advantage of the ERPDUs that are generated when a packet exceeds its time-to-live (TTL) value. The **trace** command offers default and settable parameters for specifying a simple or extended trace mode. Enter the following command at the EXEC prompt:

trace [*destination*]

To invoke a simple **trace** test, enter the destination address or host name on the command line. The default parameters for the appropriate protocol are assumed and the tracing action begins.

To use nondefault parameters and invoke an extended **trace** test, enter the command without a destination argument. You will be stepped through a dialog to select the desired parameters.

Typing the escape sequence (by default, Ctrl^, X) terminates a **trace** command.

How Trace Works

The **trace** command works by taking advantage of the error messages generated by routers when a datagram exceeds its time-to-live (TTL) value.

The **trace** command starts by sending probe datagrams with a TTL value of one. This causes the first router to discard the probe datagram and send back an error message. The **trace** command sends several probes at each TTL level and displays the round trip time for each.

The **trace** command sends out one probe at a time. Each outgoing packet may result in one of two error messages. A *time exceeded* error message indicates that an intermediate router has seen and discarded the probe. A *destination unreachable* error message indicates that the destination node has received the probe and discarded it because it could not deliver the packet. If the timer goes off before a response comes in, **trace** prints an asterisk (*).

The **trace** command terminates when the destination responds, when the maximum TTL was exceeded, or when the user interrupts the trace with the escape sequence. The information is encoded as follows:

```
hop-count name(nsap) result-of-probe
```

Tracing CLNS Routes

You may use the **trace** command to trace routes on a Cisco router configured with the ISO CLNS protocol. When stepping through the **trace** dialog for CLNS, the following parameters may be specified:

- `Protocol [ip]`. The default protocol for **trace** is IP. You must specify CLNS to begin tracing a router on a CLNS router.
- `Target CLNS address`. You may specify either an NSAP or host name.
- `Timeout in seconds`. You may specify the length of time to wait after sending each probe before giving up on getting a response.
- `Probe count`. You may specify the number of probes to be sent at each TTL level. The default is three.
- `Minimum Time to Live [1]`: You may set the TTL value for the first probes. The default is 1. Set to a higher value to suppress the display of known hops.
- `Maximum Time to Live [30]`: You may set the largest TTL value which may be used. The default is 30. The **trace** command terminates when the destination is reached or when this value is reached.

The following table describes the output.

Table 15-4 Trace Test Characters

Char	Meaning
nn msec	The probe was successfully returned in *nn* milliseconds.
&	A time-to-live-exceeded error PDU was received.
U	A destination unreachable error PDU was received.
I	The user interrupted the test.
*	The probe timed out.
C	A congestion experienced packet was received.

Following is simple **trace** output.

```
Protocol [ip]: clns
Target CLNS address: thoth
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Type escape sequence to abort.
Tracing the route to THOTH (55.0006.0100.0000.0000.0001.8888.1112.1314.1516)
  1 HORUS(55.0006.0100.0000.0000.0001.6666.3132.3334.3536) 32 msec ! 28 msec !
28 msec !
  2 ISIS(55.0006.0100.0000.0000.0001.7777.2122.2324.2526) 56 msec ! 80 msec ! 56
msec !
  3 THOTH(55.0006.0100.0000.0000.0001.8888.1112.1314.1516) 80 msec ! 80 msec ! 8
```

This example traces a more complex route.

```
Protocol [ip]: clns
Target CLNS address: cerdiwen
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Type escape sequence to abort.
Tracing the route to CERDIWEN (49.BEAD.0000.0C00.40A7.00)
  1 DEMETER(49.BEAD.0000.0C00.40AF.00) 72 msec !
     ARTEMIS(49.BEAD.0000.0C00.2D57.00) 72 msec !
     DEMETER(49.BEAD.0000.0C00.40AF.00) 76 msec !
  2 CERDIWEN(49.BEAD.0000.0C00.40A7.00) 148 msec ! 148 msec ! 144 msec !
```

The output from the **trace** command displays information for each probe that it sends out. As you can see, there were two equal cost paths to the destination. The first packet went via Demeter, the second went via Artemis, and the third went via Demeter again.

Debugging a CLNS Network

Use the EXEC commands described in this section to troubleshoot and monitor the ISO CLNS network transactions. For each **debug** command, there is a corresponding **undebug** command that turns the message logging off. Generally, you will enter these commands during troubleshooting sessions with Cisco engineers.

debug clns-esis-events

The **debug clns-esis-events** command traces the more unusual ES-IS events, including previously unknown neighbors, neighbors which have aged out, and neighbors which have changed roles (ES to IS, and so on).

debug clns-esis-packets

The **debug clns-esis-packets** command traces ES-IS activity, including sending and receiving of ESHes and ISHes, receiving Redirects (RDs), and aging out of ESH/ISH/RD entries.

debug clns-events

The **debug clns-events** command traces the more unusual CLNS events, including packet discards, sending of redirects, and so forth.

debug clns-igrp-packets

The **debug clns-igrp-packets** command causes all ISO IGRP routing activity to be displayed.

debug clns-packets

The **debug clns-packets** command causes all CLNS activity to be traced, including forwarding of packets. You can use this to circumvent a potential problem with the **ping** command in mixed Cisco and nonCisco installations; see the Ping command for more information.

debug clns-routing

The **debug clns-routing** command causes all CLNS routing table activity to be traced.

ISO CLNS Global Configuration Command Summary

This section provides an alphabetical summary of the ISO CLNS global configuration commands.

[no] clns configuration-time *seconds*

Specifies the rate at which ESHs and ISHs are sent. The default value for *seconds* is 60.

[no] clns holding-time *seconds*

Allows the sender of an ESH or ISH to specify the length of time during which the information in the HELLO packets will be believed. The argument *seconds* specifies the time in seconds. The default value is 300 seconds (five minutes).

clns host *name nsap*

Defines a name for an NSAP that can then be used in commands requiring NSAPs.

[no] clns packet-lifetime *number*

Specifies the initial lifetime for locally generated packets. The default value for *number* is 64.

[no] clns route *nsap-prefix next-hop-net*

Enters a specific static route. NSAPs that start with *nsap-prefix* are forwarded to *next-hop-nsap.*

[no] clns route *nsap-prefix* **discard**

A variation of the **clns route** command that uses the **discard** keyword to explicitly tell a router to discard packets with the specified *nsap-prefix*.

[no] clns router igrp *area-tag* **NET** *net*

Identifies the area the router will work in and lets the router know that it will be routing dynamically rather than statically. The keyword **igrp** specifies dynamic routing using the IGRP protocol. The argument *area-tag* defines a meaningful name for an area. Following the keyword **NET** you specify the Network Entity Title. The **no** form of the command turns off this IGRP router.

[no] clns router igrp *area-tag2* **redistribute domain** *area-tag1*

Redistributes routing information throughout a routing domain. The keywords **redistribute domain** enable the redistribution. The arguments *area-tag2* and *area-tag1* are the defined tags for the areas in which the routing information is to be redistributed. The **no** form disables this CLNS routing protocol on this interface.

[no] clns router igrp *area-tag* **redistribute static**

Causes the router to inject any static CLNS routes into the domain. The **no** form stops redistribution.

[no] clns router static *area-tag* **NET** *net*

Identifies the area the router will work in and lets the router know that it will be routing statically. The keyword **static** specifies that the router will use statically entered routes only. The argument *area-tag* defines a meaningful name for an area. The keyword **NET** specifies the Network Entity Title.

[no] clns routing

Enables or disables routing of CLNS packets.

ISO CLNS Interface Subcommand Summary

This section provides an alphabetical list of the ISO CLNS interface subcommands.

[no] clns checksum

Enables or disables checksum generation when ISO CLNS routing software sources a CLNS packet. By default, this function is on. Use the **no** form of the command to disable checksum generation.

[no] clns congestion-threshold *number*

Sets the congestion experience bit if the output queue has more than the specified number of packets in it. A *number* value of zero or the **no** form of the command prevents this bit from being set. The default value for *number* is four.

[no] clns erpdu-interval *milliseconds*

Determines the minimum interval time, in milliseconds, between ERPDUs. A *milliseconds* value of zero or the **no** form of the command turns off the interval rate and effectively sets no limit to the ERPDU rate. The default rate is once every ten milliseconds.

[no] clns es-neighbor *nsap snpa* [*X.25-facilities-info*]

Lists all End Systems that will be used when mapping information is statically entered. The SNPAs are the X.25 network addresses (X.121 addresses). These are usually assigned by the X.25 network provider. Use the argument *X.25-facilities-info* to specify nondefault packet and window size, reverse charge information, and so on.

[no] clns is-neighbor *nsap snpa* [*X.25-facilities-info*]

Lists all Intermediate Systems that will be used when mapping information is statically entered. The SNPAs are the X.25 network addresses (X.121 addresses). These are usually assigned by the X.25 network provider. Use the argument X.25-facilities-info to specify nondefault packet and window size, reverse charge information, and so on.

[no] clns mtu *size*

Sets the MTU packet size for the interface. The minimum value for *size* is 512. The **no** form of the command restores the default and maximum packet size.

[no] clns rdpdu-interval *milliseconds*

Determines the minimum interval time, in milliseconds, between RDPDUs. A *milliseconds* value of zero or the **no** keyword turns off the interval rate and effectively sets no limit to the RDPDU rate. The default rate is once every 100 milliseconds.

[no] clns rdpdu-mask

Enables or disables the address mask on RDPDUs. The address mask is normally present on all RDPDUs, but may be disabled with the **no clns rdpdu-mask** command.

Note: SNPA masks are never sent, and RDPDUs are ignored by Cisco routers when the router is acting as an IS.

[no] clns route-cache

Allows fast switching through the cache, and by default, is enabled. To disable fast switching, use the **no** keyword.

Note: The cache still exists and is used after the no clns route-cache command is used; the software just does not do fast switching through the cache.

---

clns router igrp *area-tag* [**level2**]

This command specifies IGRP routing. The argument *area-tag* is the tag defined for the NET using the **clns router** global configuration command. The optional **level2** keyword allows the interface to advertise Level 2 information.

clns router static *area-tag*

Specifies static routing. The argument *area-tag* is the tag defined for the NET using the **clns router** global configuration command.

[no] clns send-erpdu

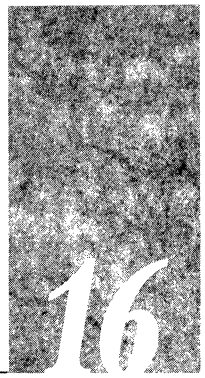
Allows or prevents CLNS to send an error PDU when it detects an error in a data PDU, and by default, is enabled. To disable this function, use the **no** keyword.

[no] clns send-rdpdu

Allows or prevents CLNS to send redirect PDUs when a better route for a given host is known, and this is the default behavior. To disable this function, use the **no** keyword.

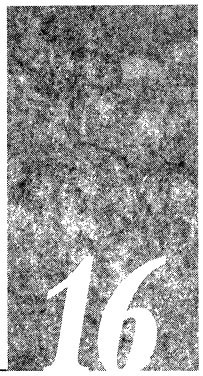
[no] clns want-erpdu

Specifies whether to request error PDUs on packets sourced by the router. The default is to request error PDUs.



Chapter 16
Routing Novell IPX

Chapter 16
Routing Novell IPX



This chapter describes Cisco's implementation of the Novell IPX routing protocol. You will find these topics and tasks described in this chapter:

- Overview of addressing in Novell IPX networks
- Basic Novell IPX routing configuration steps
- Processes for configuring optional parameters
- Steps for developing and using Novell IPX access lists
- Examples of controlling Novell broadcast messages and using various filters

Cisco's Implementation of Novell IPX

Novell IPX is a variation on Xerox Network Systems (XNS). One major difference between IPX and XNS is that they do not use the same Ethernet encapsulation format. A second difference is that IPX uses Novell's proprietary Service Advertisement Protocol (SAP) to advertise special network services. A file server is one instance of a service typically advertised.

Cisco's implementation of Novell's IPX protocol provides all of the functionality of a Novell "External Bridge" (Novell refers to their router functionality as bridging). As a Novell External Bridge, a Cisco router connects Ethernets and Token Rings, either directly or through high-speed serial lines (56 Kbps to T1 speeds) or X.25. Novell workstations on any LAN, including those without a file server, or connects to Novell file servers on any other LAN. Novell sells an X.25 and a T1 interface capability. At this time, the Cisco X.25 and T1 support is not compatible with Novell. This means that Cisco routers must be used on both ends of T1 and X.25 circuits.

Novell Addresses

Novell node IDs are 48-bit quantities, represented by dotted triplets of four-digit hexadecimal numbers. A Novell router will have interfaces on more than one physical network (Ethernet, Token Ring, serial line, and so on). Physical networks are identified by 32-bit numbers, written in hexadecimal. These network numbers must be unique throughout a Novell internet. Since both the network number and the host address are needed to deliver traffic to a host, addresses are usually given as network numbers, followed by host addresses, separated with dots. An example would be:

```
4a.0000.0c00.23fe
```

Here, the network number is *4a*, and the host address is *0000.0c00.23fe*.

Note: All numbers in the address, including the network number *4a*, are expressed in hexadecimal numbers.

Configuring Novell Routing

There are only two commands required to enable Novell IPX routing:

Step 1: Enable routing using the global configuration command **novell routing**.

Step 2: Assign Novell routing to a specific interface using the interface subcommand **novell network**.

All other configuration commands provide additional functionality or refinements. Each task is described in the following section. These descriptions are followed by applicable EXEC commands for monitoring and debugging Novell networks. Summaries of global configuration commands and interface subcommands described here appear at the end of this chapter.

Novell Configuration Restrictions

An interface takes as its Novell host address the hardware MAC address currently assigned to the interface. If later the MAC address is changed to some other value, the Novell node address automatically changes to the new address. Of course, connectivity will be lost for a while because of this change.

An optional address argument to the **novell routing** configuration command (see below) establishes the default Novell node address. This address is used as the Novell node address for any non-LAN interface, such as serial links.

Enabling Novell Routing

To enable or disable Novell routing, use the **novell routing** global configuration command. The full command syntax of this command follows.

novell routing [*host-address*]

no novell routing

The argument *host-address* is optional. If you do not specify an address, the MAC address of the first Ethernet, Token Ring, or FDDI interface is used. If there are no satisfactory interfaces present, you must specify the host address argument using the optional *host-address* argument. The address must not be multicast. The **novell routing** command enables Novell RIP routing and SAP services. Novell network numbers must still be assigned to the appropriate interfaces with the **novell network** subcommand.

Use the **no novell routing** command to disable Novell IPX routing.

Enabling Novell on an Interface

To enable Novell routing on a particular interface, use the **novell network** interface subcommand. The full syntax of this command follows.

novell network *number*

no novell network *number*

The argument *number* is the number of the Novell network to which that interface is attached. Novell packets received on an interface which do not have a Novell network number are ignored. Use the **no novell network** command with the network number to disable Novell on the interface.

Example:

```
novell network 2f
```

Repairing Corrupted Network Numbers

In some early implementations of Novell client software, it was possible for the client's network number to be corrupted. The **novell source-network-update** interface subcommand repairs corrupted network numbers by setting the source network field of any packet with a hop count of zero to the local network number. The full syntax of this command follows.

novell source-network-update

no novell source-network-update

The route cache must be disabled or this command will not work, and this is done using the **no novell source-network-update** command.

Note: This command will interfere with the proper working of OS/2 Requestors. Do not use this command in a network where OS/2 Requestors are present.

Example:

```
novell network 106A
novell source-network-update
no novell route-cache
```

Novell Encapsulation

There are two different data formats used by Novell on Ethernets. Use the **novell encapsulation** interface subcommand to select which data format or encapsulation is used on an Ethernet interface.

novell encapsulation *keyword*

The default keyword argument is **novell-ether**, which specifies Novell IPX over Ethernet using Novell's variant of IEEE 802.2 encapsulation. The keyword **arpa** is used when the Novell systems must communicate with other vendors' systems, such as DEC VAX/VMS. In this case, Ethernet-style encapsulation is used with a protocol type of 8137.

Note: On Token Rings, only one style of encapsulation exists. Some Novell nodes do not recognize Token Ring packets with the source-route bridging RIF field set. You can work around this Novell discrepancy by using the **no multiring** interface subcommand on Token Ring interfaces that are used for Novell IPX routing. See Chapter 21 in Part Five of this publication for more information.

Configuring Static Routes

Static routes for a Novell network can be specified with the **novell route** global configuration command. The full syntax of the command follows.

novell route *network network.address*

no novell route *network network.address*

The **novell route** command causes packets received for the specified network to be forwarded to the specified router, whether or not that router is sending out dynamic routing.

Use the **no novell route** command with the appropriate arguments to remove the route.

Example:

If the router that handled traffic for network 5e had the address, *3abc.0000.0c00.1ac9*, then you would enter this command:

```
novell route 5e 3abc.0000.0c00.1ac9
```

Note: Be careful when assigning static routes. When links associated with static routes are lost, traffic may stop being forwarded, although alternative paths are available.

Setting Maximum Paths

To set the maximum number of multiple paths that the router will remember and use, use the **novell maximum-paths** global configuration command. The command was designed to increase throughput by using multiple paths. It remembers higher bandwidth routes in preference to lower bandwidth routes. The full syntax of the command follows:

novell maximum-paths *paths*

no novell maximum-paths

The argument *paths* is the number of paths to be remembered. For a given destination, multiple paths of equal cost will be remembered. The default value for *paths* is 1. Output will be determined in round-robin fashion over these multiple paths at the packet level.

The **no novell maximum-paths** command restores the default.

The EXEC command **show novell routes** displays these additional routes and the maximum path value.

Setting Novell Update Timers

To allow the Novell routing update timers to be set on a per-interface basis, use the **novell update-time** interface subcommand. Full syntax follows.

novell update-time *seconds*

no novell update-time

Internal Novell timers are affected by the value set for the *seconds* argument, as follows:

- Novell routes are marked invalid if no routing updates are heard within six times the value of the update timer and are advertised with a metric of infinity.
- Novell routes are removed from the routing table if no routing updates are heard within eight times the value of the update timer.
- The default value for the **update-time** *seconds* argument is 30.

- The granularity of the update timer is determined by the lowest value defined.
- The minimum is ten seconds.

The **no novell update-time** command restores the default of 30 seconds.

Example:

In the example listed below, the granularity would be 20 because that is the lowest value specified for that protocol.

```
interface serial 0
novell update-time 40
interface ethernet 0
novell update-time 20
interface ethernet 1
novell update-time 25
```

Note: Be careful when using this command. It can be used only in an all-Cisco environment, and all timers should be the same for routers connected to the same network segment.

The EXEC command **show novell interface** displays the value of these timers.

Filtering Novell Packets

Cisco's implementation of the Novell IPX software provides three types of filtering:

- Access lists, or packet filtering, which controls whether or not packets are sent out the filtered interface.
- Routing update filtering, which controls to which Novell IPX networks the router advertises routes to, depending on the type used.
- SAP filtering, which controls the services the router knows about, and which services the router advertises.

In setting up these packet filters, care must be taken to not set up filtering conditions that result in packets falling through a "black hole." This can happen, as an example, when the software is configured to advertise services on a network with access lists configured to deny these packets, or when a network is configured to advertise services on a network that is unreachable because routing updates are filtered out by routing update filtering.

Keep these pitfalls in mind while configuring the filter types, each discussed in the following sections.

Configuring Novell IPX Access Lists

Simple Novell IPX access lists are numbered from 800 to 899 and filter on the source and destination addresses only.

The command syntax for standard Novell IPX access lists is lengthy. For typographic reasons, the command example is shown on multiple lines; it must be on a single line when given as a configuration command. The full command syntax for the Novell **access-list** global configuration command follows.

access-list *number* {**deny** | **permit**} *novell-source-network*[[.*source-address*[*source-mask*]] *novell-destination-network*[.*destination-address* [*destination-mask*]]

no access-list *number*

The only required argument for standard Novell IPX access lists is the Novell IPX source network. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered.

Use the **no access-list** command with the appropriate access list number to remove the access list.

Example:

The following example denies access from source network –1 (all Novell IPX networks) to destination network 2.

```
access-list 800 deny -1 2
```

The following example denies access from Novell IPX source address *0000.0c00.1111*.

```
access-list 800 deny 1.0000.0c00.1111
```

The following example denies access from all nodes on network 1 that have a source address beginning with *0000.0c*.

```
access-list 800 deny 1.0000.0c00.1111 0000.00ff.ffff
```

The following example denies access from source address *1111.1111.1111* on network 1 to destination address *2222.2222.2222* on network 2.

```
access-list 800 deny 1.1111.1111.1111 0000.0000.0000 2.2222.2222.2222
0000.0000.0000
```

Configuring Extended Novell Access Lists

Extended Novell IPX access lists filter on protocol information as well; numbers for the extended lists range from 900 to 999. The command syntax for extended Novell IPX access lists is again rather lengthy as a configuration command (that must be typed on one line):

access-list *number* {**deny** | **permit**} *novell-protocol source-network.*[*source-address* [*source-mask*]] *source-socket destination-network.*[*destination-address* [*destination-mask*]] *destination-socket*

no access-list *number*

The source and destination addresses and masks are optional. The protocol number *novell-protocol* is the only required parameter. A network number of –1 matches all networks; a socket number of 0 matches all sockets.

Use the **no access-list** command with the appropriate access list number to remove the access list.

Example:

The following example denies access to protocol *1* from source network *1*, source socket *1234* to destination network *2*, destination socket *1234*.

```
access-list 900 deny 1 1 1234 2 1234
```

The following example illustrates the use of all possible parameters:

```
access-list 900 deny 1 1.1111.1111.1111 0000.0000.0000 1234
  2.2222.2222.2222 0000.0000.0000 1234
```

Filtering Outgoing Traffic

The Novell IPX access list group number is assigned with the **novell access-group** interface subcommand. The syntax for this command follows:

novell access-group *number*

no novell access-group *number*

The argument *number* refers to the appropriate Novell access list number. All outgoing packets forwarded through the interface will be filtered by this access list.

Use the **no novell access-group** command with the appropriate group number to remove the access list group number from the interface.

Example of Controlling Traffic with Access Lists

Using access lists to manage traffic routing can be a powerful tool in overall network control. However, it does require a certain amount of planning and the appropriate application of several related commands. Figure 16-1 illustrates a network featuring two Cisco routers connecting a number of network segments.

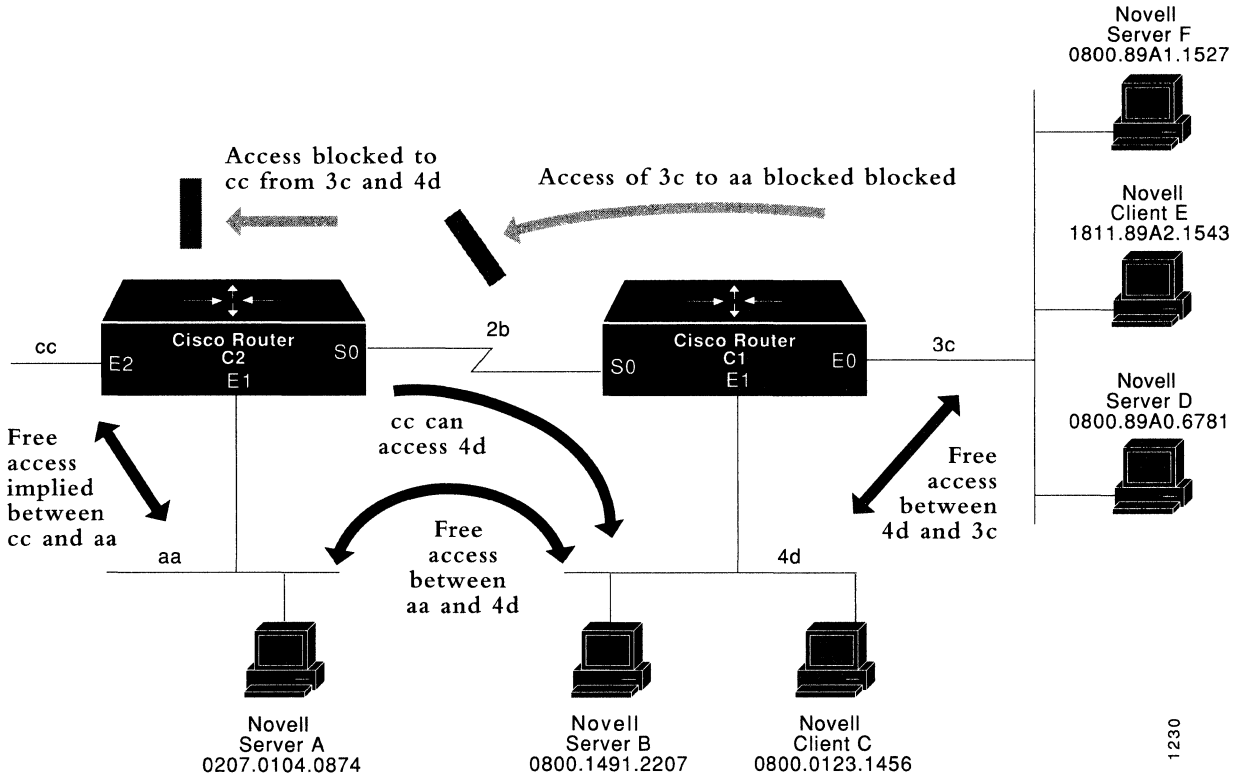


Figure 16-1 Multiple Novell Servers Requiring Access Control

For the purposes of illustrating access control, the network in Figure 16-1 has the following specific requirements:

- Resources on networks *3c* and *4d* are to be allowed free access to each other through router *C1*.
- Resources on network *4d* are to be allowed access to resources on network *aa*.
- Resources on network segment *3c* are not allowed access resources on *aa*.
- Resources on network *aa* are not allowed to access resources on *3c*.
- All networks can access resources on segment *4d*.

The configuration for this environment can be defined using simple access lists as illustrated in the following example. In this example, the global configuration command **access-list** and interface subcommands **novell network** and **novell access-group** are applied to router/bridge *C1* in Figure 16-1.

Note: The commands in configuration files are executed in a sequential, first match, top down basis. Plan the placement of command lines carefully, especially when specifying and applying access lists. Access lists are always applied to the *transmitting* interface.

Example 1:

This first example is applied to router *C1*, permitting resources on network *4d* to access resources on network aa. It implicitly denies all other traffic.

```
access-list 800 permit 4d aa
```

If you want to explicitly deny all other traffic, you add the following line:

```
access-list 800 deny -1 -1
```

Example 2:

This example assigns network number *2b* to the first serial interface, then applies access group 800 and permit resources on network *4d* to access resources on network *3c*. Again, you explicitly deny all other traffic.

```
interface serial 0
novell network 2b
novell access-group 800
access-list 800 permit 4d 3a
access-list 801 deny -1 -1
```

Example 3:

This example assigns a network number and access group 801 to interface Ethernet 1, and applies a network number to the second Ethernet address. There are no explicit permissions or denials for interface Ethernet 1.

```
access-list 801 permit 4d 3c
interface ethernet 0
novell network 3c
novell access-group 801
interface ethernet 1
novell network 4d
```

Note: This configuration example does not address access control for segment cc on router *C1*. However, given no other restrictions or specific permissions, it implies that resources on cc can access resources on *4d*, but cannot access resources on *3c*. In addition, resources on *4d* cannot access resources on network *cc*.

Filtering Novell Routing Updates

This section describes the filtering commands that use access lists to control which routing information is accepted, or passed on, within Novell networks. The commands filter incoming traffic, outgoing routing updates, and specific routers.

Each access list entry contains only one address parameter. How this address is interpreted is defined by the command that will use the list.

As with all other Cisco access lists, an implicit *deny everything* is defined at the end of the list. If this is not desired, an explicit *permit everything* definition must be included at the end of the list.

Each filter type is described in the following sections.

Establishing Input Filters

To control which networks are added to the routing table, use this interface subcommand:

novell input-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the **novell access-list** command.

Example:

In the following example, access list 876 controls which networks are added to the routing table when Novell routing updates are received. The address in the access list is a source network.

```
access-list 876 permit 1b
interface ethernet 1
novell input-network-filter 876
```

This configuration causes network *1b* to be the only network that is accepted from updates received on the defined Ethernet interface.

Establishing Output Filters

To control the list of networks that are sent in routing updates, use the interface subcommand:

novell output-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the **novell access-list** command.

Example:

In the following example, access list 896 controls which networks are sent out in routing updates. The address parameter is the desired network.

```
access-list 896 permit 2b
interface serial 1
novell output-network-filter 896
```

This configuration causes network *2b* to be the only network advertised in Novell routing updates sent on the defined serial interface.

Establishing Router Filters

To control the list of routers from which data will be accepted, use this interface subcommand:

novell router-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the **novell access-list** command.

Example:

In this example, access list 866 controls from which router data will be accepted. In this case, the address parameter is the address of the router.

```
access-list 866 permit 3c.0000.000c0.047d
interface serial 0
novell router-filter 866
```

Information from a disallowed router is ignored.

Building SAP Filters

A common source of traffic on Novell networks is the SAP-based messages generated by Novell servers and Cisco routers as they broadcast their available capabilities. Control of SAP messages can be established with Cisco routers using several facilities. Access lists and SAP filters combine to allow you to control how SAP messages from network segments or specific servers are routed among Novell networks.

Defining Access Lists for SAP Filtering

To define an access list for filtering SAP requests, use this variation of the **access-list** command:

access-list *number* **permit | deny** *network.*[*address*] [*service-type*]

The argument *number* is the SAP address lists, which must is a decimal number in the range 1000 to 1099.

Enter the keyword **permit** or **deny** to establish the type of access desired. Permit or deny access is based on the data provided.

The argument *network* is a hexadecimal Novell network number; 0 defines the local network, –1 defines all networks.

The optional *address* argument is a Novell node address.

The *service-type* argument defines the service type to filter; 0 is all services. Service types are entered in hexadecimal. Examples of the service types that may be entered are listed in Table 16-1.

Table 16-1 Sample Novell SAP Services

Description	Service Type
Unknown	0
User	1
User Group	2
Print Queue	3
File Server	4
Job Server	5
Gateway	6
Print Server	7
Archive Queue	8
Archive Server	9
Job Queue	A
Administration	B
Remote Bridge Server	24
Advertizing Printer Server	47
Wildcard	blank (no entry)

Example—Service Type Specification

```
! Deny access from all nets for service 4:
access-list 1001 deny -1 4
! Permit access from all nets to all other services:
access-list 1001 permit -1
```

Note: In the example outlined above, companion interface specification must be defined for this access list to be applied. Once applied to an interface, this access list definition blocks all access to service type 4 (file service) on directly attached network by resources on other Novell networks. The second specification explicitly allows access to all other available services on the interface.

Configuring Novell SAP Filters

Use these commands to filter Novell SAP messages.

novell input-sap-filter *access-list-number*

novell output-sap-filter *access-list-number*

novell router-sap-filter *access-list-number*

These commands take a SAP Novell access list number as their input. The range for SAP lists is 1000 to 1099.

Follow these guidelines to use SAP filtering:

- When the **novell input-sap-filter** list is enabled, use the list to determine the services that will be accepted.
- When the **novell output-sap-filter list** is enabled, use the list to determine the services that will be included in SAP updates from Cisco routers.
- When the **novell router-sap-filter** list is enabled, use the list to determine the router from which the Cisco router will hear SAP messages, and the service type.

Example SAP Input Filter

Input SAP filters are applied prior to a Cisco router accepting information about a service. In the example that follows, Cisco router *C1* (illustrated in Figure 16-2) will not accept and, consequently not advertise, any information about Novell server *F*. However, *C1* will accept information about all other servers on the network *3c*. Cisco router *C2* will receive information about servers *D* and *B* in this example.

Example:

This example configures router *C1*. The first line denies server *F*. It accepts all other servers.

```
access-list 1000 deny 3c.0800.89a1.1527
access-list 1000 permit -1
interface ethernet 0
novell network 3c
novell input-sap-filter 1000
interface ethernet 1
novell network 4d
interface serial 0
novell network 2b
```

Note: Please note that NetWare 386 servers use an internal network and node number as their address for access list commands (the first configuration command in this example).

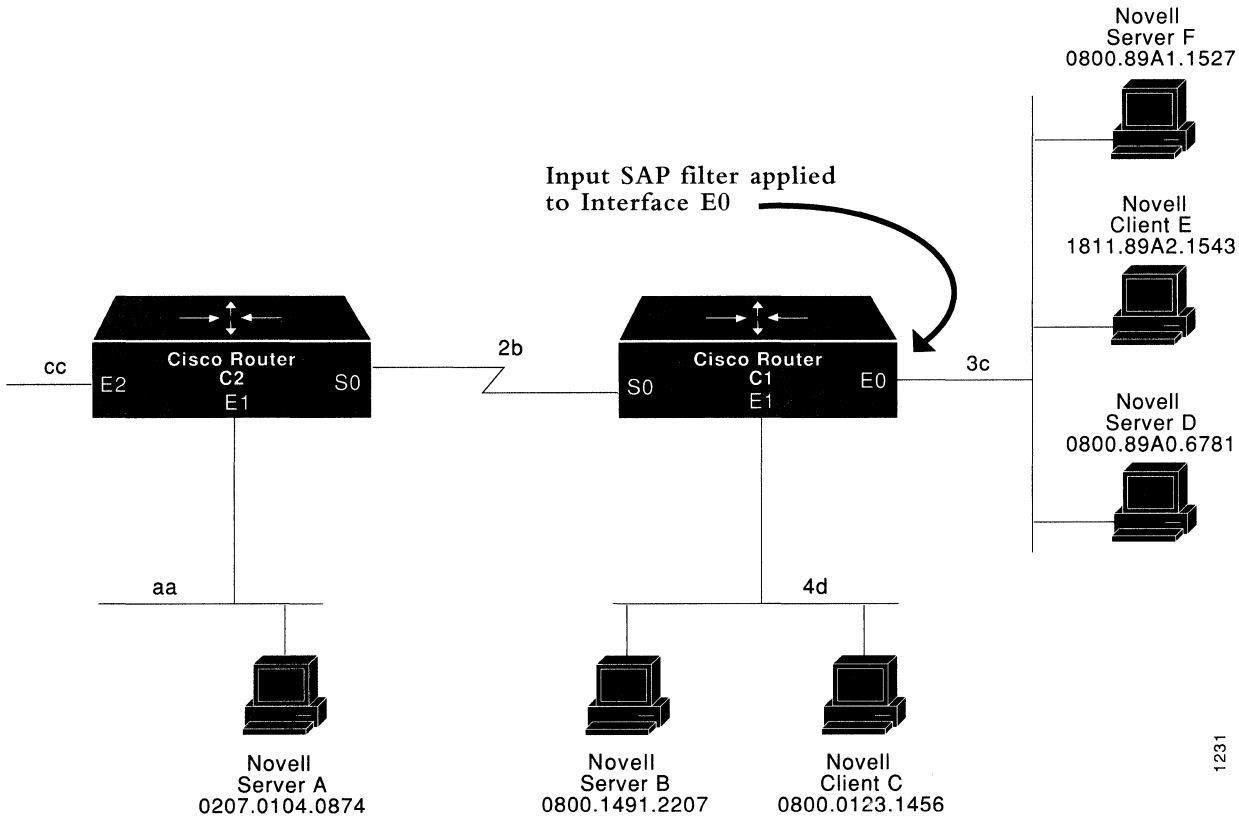


Figure 16-2 SAP Input Filter

Example SAP Output Filter

Output SAP filters are applied prior to a Cisco router sending information out a specific interface. In the example that follows, Cisco router *C1* (illustrated in Figure 16-3) is prevented from advertising information about Novell server *A* out interface Ethernet 1, but can advertise server *A* on network *3c*.

Example:

The following example refers to router *C1*. The first line denies server *A*. All other servers are permitted.

```
access-list 1000 deny aa.0207.0104.0874
access-list 1000 permit -1
interface ethernet 0
novell net 3c
interface ethernet 1
novell network 4d
novell output-sap-filter 1000
interface serial 0
novell network 2b
```

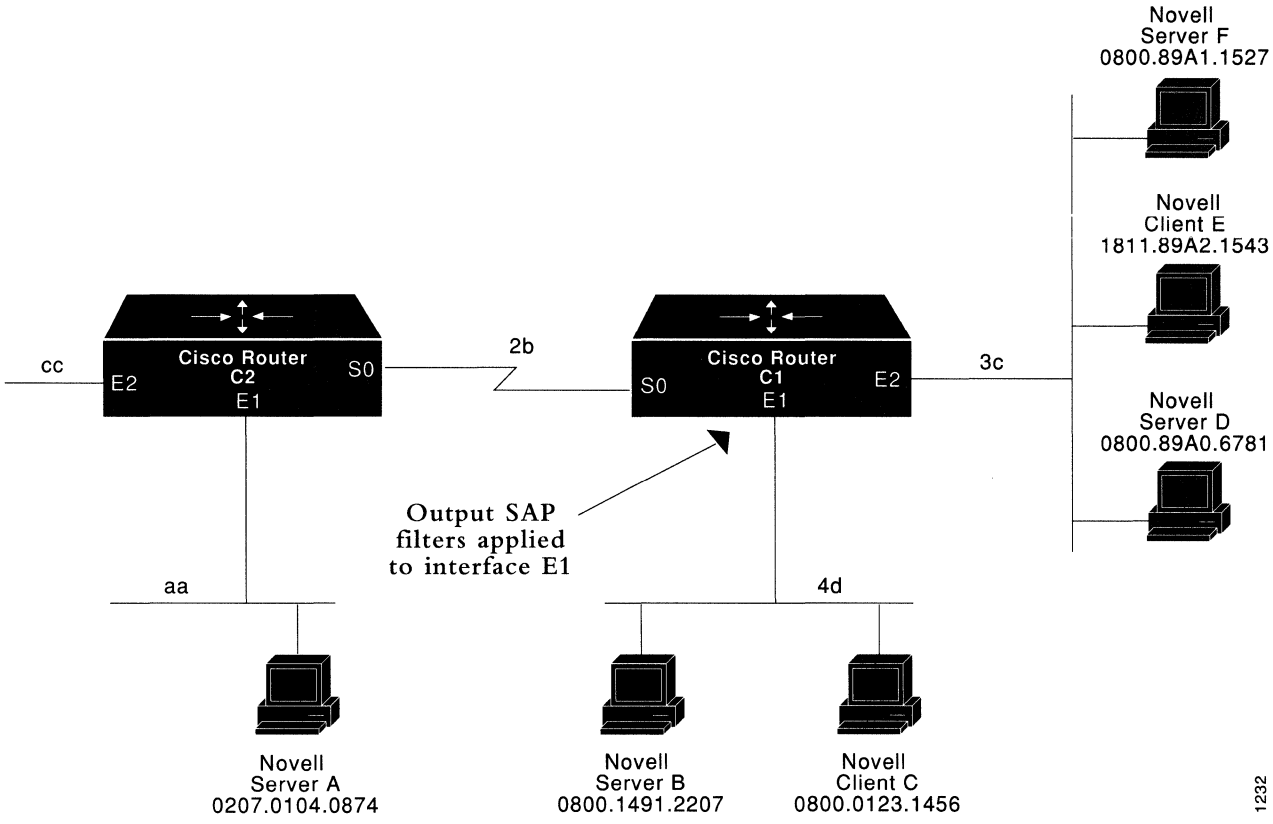


Figure 16-3 SAP Output Filter

Novell Broadcast Helper Facilities

Cisco's helper facilities provide a flexible set of tools to help you manage Novell network broadcast traffic. Several configuration options allow network administrators to tailor the way broadcasts generated by Novell clients are forwarded through a network.

If Novell clients and servers are attached to the same network segment, this basic function (blocking broadcasts) is acceptable and often preferred, since it helps reduce unwanted traffic among networks. However, when clients must broadcast through a router to a remotely-located server, several modifications to the Cisco system configuration are required. To make these modifications, use the Novell helper functions.

Cisco routers support flooding. *Flooding*, as the name suggests, forwards broadcasts to all networks.

The key to controlling Novell broadcasts (rather than simply blocking them) rests with the use of several commands specific to Cisco's Novell IPX routing implementation:

- **novell helper-address**—Explicitly specifies the address of a target Novell server (or network) to which broadcast packets are sent (applied to a specific interface).
- **novell helper-list**—Assigns access lists to interfaces to control broadcast traffic (applied to a specific interface).
- **access-list**—A general Cisco traffic-controlling tool that can be tailored to help manage broadcast traffic in a Novell network environment.

Defining a Helper List

The **novell helper-address** and **novell helper-list** interface subcommands are defined briefly below, while the global configuration command **access-list** is described in a preceding section. Following the "helper" facility definitions, several typical applications illustrate how to use Cisco's helper and access list mechanisms together within the context of Novell-based internetworking environments.

The **novell helper-list** interface subcommand specifies that only those packets that pass the specified Novell access list are forwarded to a remote Novell server. (The only exception to this rule is that all-nets flooded broadcasts (our next topic) and NetBIOS are ALWAYS forwarded, regardless of how you set the helper-list command.) The syntax for this command is:

novell helper-list *access-list-number*

The argument *access-list-number* specifies the access list. The network numbers in that list are expressed in hexadecimal values.

Note: Since the destination address of a broadcast is by definition the broadcast address, this is only useful for filtering based on the source of the broadcast packet. This can be used to prevent nodes from discovering services they should not use.

Specifying Target Novell Servers

To forward broadcast packets that match the access list specified by the **novell helper-list** subcommand, use the **novell helper-address** interface subcommand:

novell helper-address *net.host*

This subcommand causes all-nets broadcasts to be forwarded to *net.host*. The argument *net.host* is a dotted combination of the network and host addresses as explained in the **novell route** subcommand.

Incoming unrecognized broadcast packets that match the access list will be forwarded on to the address specified by the argument *net.host*.

The Cisco routers support all network (*all nets*) flooding. To configure the all nets broadcast flooding, define the Novell helper address for an interface as:

```
-1.FFFF.FFFF.FFFF
```

On systems configured for Novell routing, this helper address will be displayed as:

```
FFFFFFFF:FFFF.FFFF.FFFF
```

Note: Although care has been taken to keep traffic to a minimum, some duplicates will be unavoidable. Under certain conditions (where loops exist) flooding can propagate bursts of excess traffic that will, eventually age out, when the hop count reaches its limit. Use the broadcast address carefully and only when necessary.

Using Helper Facilities to Control Broadcasts

Use of the **helper-list** and **helper-address** tools is best illustrated with examples. The following illustrations and accompanying descriptions outline the application of access lists, helper lists, and helper addresses to forward traffic to a specific network or to a node, and to flood broadcast messages on all attached links.

Forwarding to an Address

You can direct broadcasts to a specific network or host (node) on a segment. The following examples illustrate both these forwarding options.

Figure 16-4 shows a Cisco router (C1) connected to several Ethernets. In this environment, all Novell clients are attached to segment aa, while all servers are attached to segment bb. In controlling broadcasts, the following conditions are to be applied:

- Only type 10 broadcasts are to be forwarded.
- The Novell clients on network aa are to be allowed to broadcast to any server on network *bb*.
- All-nets broadcasts are always flooded.

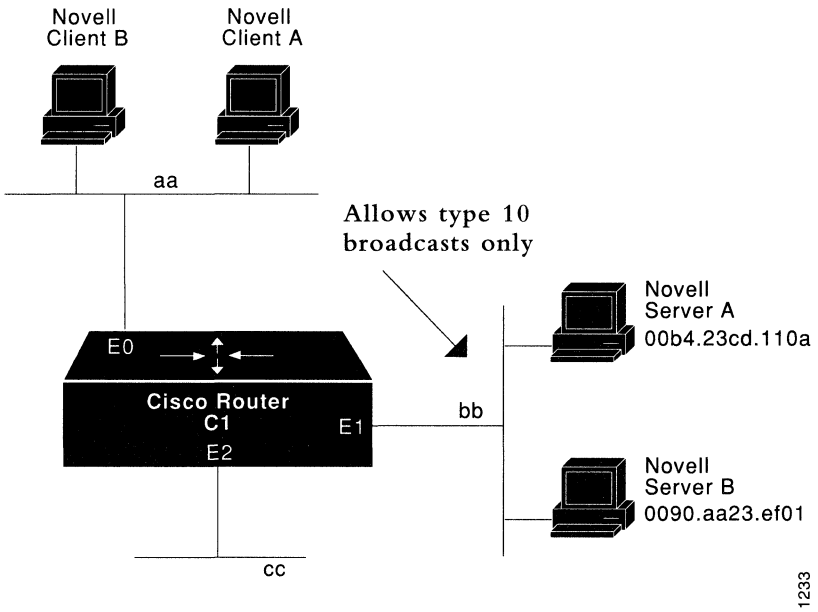


Figure 16-4 Novell Clients Requiring Server Access Through a Cisco Router

Interfaces E1 and E2 do not require application of any specific permissions to meet the conditions for this example, since broadcasts are by default blocked by the router.

Example:

This example configures the router shown in Figure 16-4. The first line permits traffic of type 10 from network *aa*. Then the interface and network commands configure a specific interface. The helper-address command permits broadcast forwarding from Network *aa* to *bb*. The last line forwards type 10 broadcasts from networks *aa* to *bb*.

```
access-list 900 permit 10 aa
interface ethernet 0
novell network aa
novell helper-address bb.ffff.ffff.ffff
novell helper-list 900
```

Any downstream network that is cascaded beyond network aa (for example, some arbitrary network *aa1*) will not be able to broadcast to network *bb* through router *C1*, unless the routers partitioning networks *aa* and *aa1* are configured to forward broadcasts with a series of configuration entries analogous to the example provided for Figure 16-4. These entries must be applied to the input interface and be set to forward broadcasts between directly connected networks. In this way, traffic can be passed along, in a directed manner, from network to network.

Example:

The example provided below rewrites the **novell helper-address** command line to direct broadcasts to server *A* in Figure 16-4.

```
novell helper-address bb.00b4.23cd.110a
! Permits node-specific broadcast forwarding to
! Server A at address 00b4.23cd.110a on network bb
```

Forwarding to All Networks

In some networks, it may be necessary to allow client nodes to broadcast to servers on multiple networks. If you configure your router to forward broadcasts to all attached networks, you are flooding the interfaces. In the environment illustrated in Figure 16-5, client nodes on network *2b1* must obtain services from Novell servers on networks *3c2*, *4a1*, and *5bb* through Cisco router *C1*. To support this requirement, use the flooding (-1.ffff.ffff.ffff) address in your **novell helper-address** interface subcommand specifications.

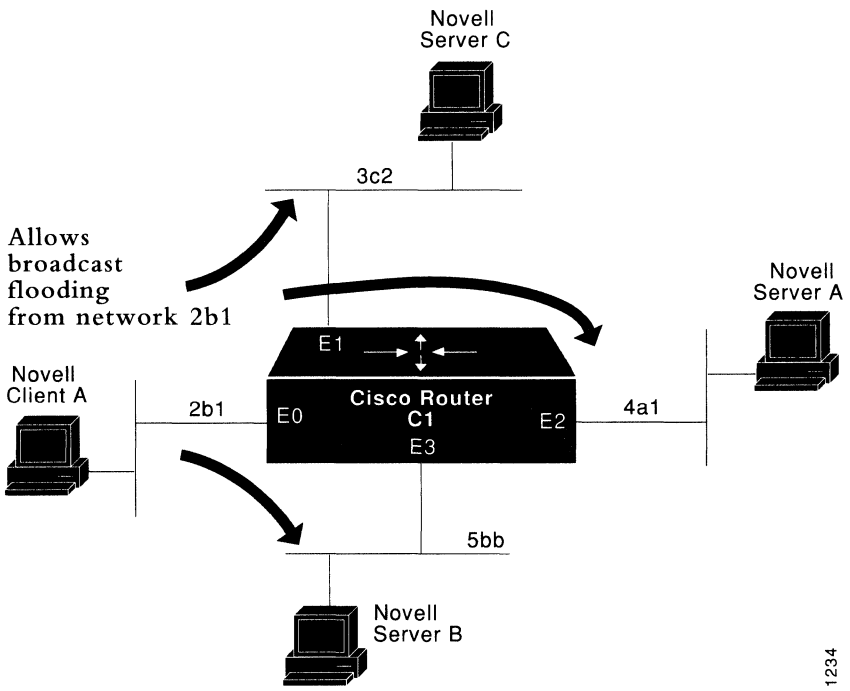


Figure 16-5 Type 10 Broadcast Flooding

As with the prior example, the configuration for this environment can be defined using an extended access list, a helper list and a helper address.

Example:

In this example, the first line permits traffic of type 10 to network *2b1*. Then the first Ethernet interface is configured with a network number. The helper address is defined and the helper list limits forwarding to type 10 traffic.

```
access-list 901 permit 10 2b1
interface ethernet 0
novell network 2b1
novell helper-address -1.ffff.ffff.ffff
novell helper-list 901
interface ethernet 1
novell network 3c2
interface ethernet 2
novell network 4a1
interface ethernet 3
novell network 5bb
```

In this example, type 10 broadcasts from network *2b1* are forwarded to all directly connected networks. All other broadcasts are blocked. If all broadcasts are to be permitted, delete the **novell helper-list** entry.

Enabling Novell Fast Switching

Novell fast switching allows higher throughput by switching the packet using a cache created by previous transit packets. Fast switching also provides load sharing on a per-packet basis.

Use the **novell route-cache** interface subcommand to enable fast switching. The full syntax for this command follows.

novell route-cache

no novell route-cache

When Novell routing is enabled, by default, Novell fast switching is enabled on the appropriate interface. Use the **no novell route-cache** command to disable fast switching.

Restricting SAP Updates

To configure less frequent SAP updates over slow links, use the interface subcommand **novell sap-interval**. This command has the following syntax:

novell sap-interval *interval*

Use the *interval* argument to set the interval between SAP updates. If *interval* is zero, periodic updates are not sent. A message is sent only when the server first appears and when it goes down. The default value for the argument *interval* is one minute. This is the value used by the Novell servers.

Example:

In this example, SAP updates are sent (and expected) on interface serial 0 every five minutes.

```
interface serial 0
novell sap-interval 5
```

All Novell servers and routers on a particular network require the same SAP interval or they are likely to decide that a server is down, even though it is actually up. Since it is impossible to change this value on most PC-based servers, you should never change the interval for an Ethernet or Token Ring that has actual servers on it. This subcommand is most useful on limited bandwidth point-to-point links or X.25 interfaces, where one prefers to use as little bandwidth as possible for sending the SAP updates.

Note: Setting the interval to zero is especially dangerous. Routers that were inaccessible for any reason at the time of a server power up or shut down will miss that event, and will either fail to learn about the existence of new servers, or will fail to detect the server shut down.

SAP Update Delays

Some slow Novell servers lose SAP updates because they cannot keep up the same brisk processing pace as the Cisco routers can. The **novell output-sap-delay** interface subcommand allows you to set a delay between SAP updates, in effect forcing the Cisco router interface to pace its output to the slower-processing needs of the Novell servers. If your server is not slow and is not losing SAP updates, you can skip this configuration command. The full syntax of the command follows:

novell output-sap-delay *delay*

no novell output-sap-delay

The parameter *delay* is measured in milliseconds.

The **no novell output-sap-delay** disables the delay mechanism.

Example:

```
novell network 106A
novell output-sap-delay 200
```

Novell Configuration Example

The following configuration commands enable Novell routing, defaulting the Novell host address to that of the first IEEE-conformance (no serial, for example) interface. Routing is then enabled on Ethernet 0 and Ethernet 1 for Novell networks *2abc* and *1def*, respectively.

Example:

```
novell routing
interface ethernet 0
novell network 2abc
interface ethernet 1
novell network 1def
```

Note: The network numbers used must match the numbers used by Novell file servers on the same cable.

Monitoring the Novell IPX Network

Use the EXEC commands described in this section to obtain displays of activity on the Novell IPX network.

Displaying the Novell Cache Entries

Use the **show novell cache** command to display a list of fast-switching cache entries. Enter this command at the EXEC prompt:

show novell cache

Following is sample output:

```
Novell routing cache version is 9
Destination                Interface                MAC Header
*1006A                     Ethernet0            00000C0062E600000C003EB0064
*14BB                      Ethernet 1           00000C003E2A00000C003EB0064
```

In the sample display, valid entries are marked by an asterisk (*).

Displaying Novell Interface Parameters

Use the **show novell interface** command to display the Novell parameters that have been configured on the interfaces. Enter this command at the EXEC prompt:

show novell interface [*interface unit*]

An optional interface name can be specified with the *interface unit* arguments to see information for a specific interface. Following is sample output:

```
Ethernet 0 is up, line protocol is up
  Novell encapsulation is NOVELL-ETHER
  Novell address is 1006A.0000.0c00.62e6
  Outgoing access list is not set
  Novell SAP update interval is 1 minute(s)
  Novell Helper access list is not set
  SAP Input filter list is not set
  SAP Output filter list is not set
  SAP Router filter list is not set
  Input filter list is not set
  Output filter list is not set
  Router filter list is not set
  Update time is 30 seconds
  NOVELL Fast switching enabled
```

Displaying the Novell Routing Table

Use the **show novell route** EXEC command to display the Novell routing table. Enter this command at the EXEC prompt:

show novell route

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 2 learned routes

Maximum allowed path(s) are/is 1
R Net 1001 [1/1] via 1006.aa00.0400.6508,  94 sec, 0 uses, Ethernet0
R Net 1003 [1/1] via 1006.aa00.0400.6508,  94 sec, 0 uses, Ethernet0
C Net 13A is directly connected, 0 uses, Ethernet1 (down)
C Net 1006A is directly connected, 0 uses, Ethernet0
```

In the display, the leading character R indicates routes learned via RIP, C indicates connected entries, and S indicates statically defined entries.

Displaying Novell Servers

Use the **show novell servers** EXEC command to list the servers discovered through SAP advertisements. Enter this command at the EXEC prompt:

show novell servers

Following is sample output:

```
Type       Name          Net Address                Port          Hops

4          SYSOP         2a.0206.00a2.41ec:0450      2          Ethernet2
4          MKTG          3c.0800.00a3.45ef:0450      2          Ethernet5
4          SERVICE       4d.0a44.008a.0220:0450                 Ethernet10
4          FINANCE       1a.0080.a246.0001:0450                 Ethernet10
```

For each known server in the network, the display lists its name, complete address, number of hops distant it is and the interface through which it can be accessed (through which it was discovered).

Displaying Novell Traffic

Use the **show novell traffic** EXEC command to display information on the number and type of Novell packets transmitted and received. Enter this command at the EXEC prompt:

show novell traffic

Following is sample output:

```
Rcvd:   68112 total, 0 format errors, 0 checksum errors, 0 bad hop count,
        68102 local destination, 0 multicast
Bcast:  68102 received, 43745 sent
Sent:   43745 generated, 0 forwarded
        0 encapsulation failed, 10 no route
SAP:    0 SAP requests, 0 SAP replies
        0 SAP advertisements received, 0 sent
 Echo: Rcvd 0 requests, 0 replies
       Sent 0 requests, 0 replies
        0 unknown
```

The following notes apply to the less-obvious statistics in this screen:

- `Format errors` are reported whenever a bad packet is detected (for example, a corrupted header).
- `Checksum errors` should not be reported, since IPX does not use a checksum.
- `Bad hop count` increments when a packets hop count exceeds 16.
- `Encapsulation failed` is registered when the router is unable to encapsulate a packet.
- The `unknown` counter increments when packets are encountered that the router is unable to forward (for example, misconfigured helper-address, or no route available).

Novell Ping Command

To execute the **ping** command on a Cisco network server configured for Novell routing, enter **novell** at the **ping** protocol prompt and the Novell routing address at the address parameter prompt. The defaults are enclosed in brackets at each prompt.

Here is a sample:

```
Protocol [ip]: novell
Target Novell Address: 1006A.0000.0c00.62e6
Repeat Count [5]:
Datagram Size [100]:
Timeout in seconds [2]:
Verbose [n]:
Type escape sequence to abort.
Sending 5 100-byte Novell echoes to 1006A,0000,0c00,62e6, timeout is 2
seconds.
!!!!!!!
Success rate is 100%, round trip min/avg/max  = 1/2/4 ms.
```

See the section "Testing Connectivity with the Ping Command" in Chapter 5 for more information.

Note: This command only works on Cisco network servers running Release 8.2 (or later) software. Novell devices will not respond to this command.

Debugging the Novell IPX Network

Use the commands described in this section to troubleshoot and monitor the Novell IPX network. For each **debug** command, these is a corresponding **undebug** command that turns off message logging.

debug novell-packet

The **debug novell-packet** command outputs information about packets received, transmitted, and forwarded.

debug novell-routing

The **debug novell-routing** command prints out information on Novell routing packets.

debug novell-sap

The **debug novell-sap** command displays additional information about Novell Service Advertisement packets.

Novell IPX Global Configuration Command Summary

The following is an alphabetical list of the Novell IPX global configuration commands. These commands specify system-wide parameters for Novell IPX support.

[no] novell routing [*host-address*]

Enables and disables Novell routing and Novell RIP routing and SAP services. You can also use this command to specify the system-wide host address to use with the optional argument *host-address*. If you do not specify an address, the MAC address of the first Ethernet, Token Ring, or FDDI interface is used. If there are no satisfactory interfaces present, you must specify the host address argument. The address must not be multicast. Assign Novell network numbers to the appropriate interfaces with the **novell network** subcommand.

[no] novell route *network network.address*

Specifies or removes static routes for a Novell network. When specified, the command causes packets received for the specified network to be forwarded to the specified router, whether or not that router is sending out dynamic routing.

[no] novell maximum-paths *paths*

Sets the maximum number of multiple paths that the router will remember and use. The argument *paths* is the number of paths to be remembered. The **no** form of the command restores the default.

[no] access-list *number* **deny | permit** *novell-source-network*[*.source-address*] *source-mask novell-destination-network.destination-address destination-mask*

Specifies standard Novell IPX access lists. Standard Novell IPX access lists are numbered from 800 to 899 and filter on the source and destination addresses only. An access list command must be completely specified on a single line when given as a configuration command. The only required parameter for standard Novell IPX access lists is the Novell IPX source network. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered. The **no** form of the command removes any access list in the current image with the specified number.

[no] access-list *number* **deny | permit** *novell-protocol source-network.*[*source-address* [*source-mask*]] *source-socket destination-network.*[*destination-address* [*destination-mask*]]*destination-socket*

Specifies extended Novell IPX access lists. The source and destination addresses and masks are optional. The protocol number *novell-protocol* is the only required parameter. A network number of –1 matches all networks; a socket number of 0 matches all sockets. Extended Novell IPX access lists filter on protocol information as well; numbers for the extended lists range from 900 to 999. The **no** form of the command removes any access list in the current image with the specified number.

[no] access-list *number* **permit | deny** *network.*[*address*] [*service-type*]

Defines an access list for filtering SAP requests. The argument *number* is a decimal number in the range of 1000 to 1099. Enter the keyword **permit** or **deny** to establish the type of access desired. Permit or deny access is based on the data provided. The argument *network* is a hexadecimal Novell network number; 0 defines the local network, –1 defines all networks. The optional *address* argument is a Novell host address. The *service-type* argument defines the service type to filter; 0 is all services. Service types are entered in hexadecimal.

novell encapsulation *keyword*

Selects which data format or encapsulation is used on an Ethernet interface. The default keyword argument is **novell-ether** which specifies Novell IPX over Ethernet using Novell's variant of IEEE 802.2 encapsulation. The keyword **arpa** is used when the Novell systems must communicate with other vendors' systems, such as DEC VAX/VMS. In this case, Ethernet-style encapsulation is used with a protocol type of 8137.

[**no**] **novell input-sap-filter** *access-list-number*
[**no**] **novell output-sap-filter** *access-list-number*
[**no**] **novell router-sap-filter** *access-list-number*

Configure Cisco routers to filter the acceptable source of Novell SAP messages; the intended destination of SAP messages; or the specific router from which SAP filters will be accepted. These commands take a SAP Novell access list number as their input. The range for SAP lists is 1000 to 1099. The **no** forms of the commands remove the filters.

Novell IPX Interface Subcommand Summary

The following is an alphabetical list of the Novell IPX interface subcommands. These commands specify line-specific parameters for Novell IPX support. These subcommands must be preceded by an **interface** command.

[**no**] **novell access-group** *number*

Assigns or removes a Novell IPX access list group number to a specific interface. The argument *number* refers to the appropriate Novell access list number. All outgoing packets forwarded through the interface will be filtered by this access list.

[**no**] **novell helper-address** *net.host*

Broadcast packets that match the access list specified by the **novell helper-list** subcommand are forwarded when this command is used. This subcommand causes all-nets broadcasts to be forwarded to *net.host*. The argument *net.host* is a dotted combination of the network and host addresses as explained in the **novell route** subcommand. Incoming unrecognized broadcast packets that match the access list will be forwarded on the address specified by the argument *net.host*. This subcommand is useful for hosts which use a protocol other than SAP for advertising their availability.

[no] novell helper-list *access-list-number*

Specifies that only those packets which pass the specified Novell access list will be forwarded to the Novell helper host. The argument *access-list-number* specifies the access list. The network numbers in that list are expressed in decimal values. The **no** form of the command disables the function.

[no] novell source-network-update

Enables the interface to provide the current network number in place of the source network number of any packet that arrives with a hop count of zero. The **no** form of the command disables the function.

[no] novell input-network-filter *access-list-number*

Explicitly specifies which networks are added to the Novell IPX routing table. The argument *access-list-number* is the access list number specified in the **novell access-list** command. The **no** form of the command disables the function.

[no] novell network *number*

Enable and disables Novell routing on a particular interface. The argument *number* is the number of the Novell network to which that interface is attached. Novell packets received on an interface which do not have a Novell network number are ignored.

[no] novell output-network-filter *access-list-number*

Explicitly specifies the list of networks that are sent in routing updates. The argument *access-list-number* is the access list number specified in the **novell access-list** command. The **no** form of the command disables the function.

[no] novell output-sap-delay *delay*

Sets the interval, measured in milliseconds, that an interface will delay, added to the usual SAP reporting interval. The **no** form of the command disables the mechanism.

[no] novell route-cache

Enables and disables Novell fast-switching. When routing is enabled, by default, Novell fast-switching is enabled on the appropriate interface. The **no** form of the command disables fast-switching.

[no] novell router-filter *access-list-number*

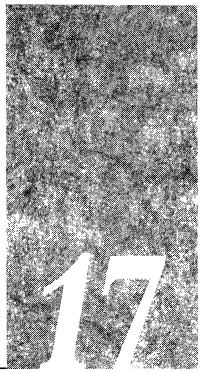
Specifies or removes the list of routers from which data will be accepted. The argument *access-list-number* is the access list number specified in the **novell access-list** command.

novell sap-interval *interval*

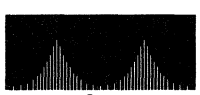
Configures less frequent SAP updates over slow links by setting the interval between SAP updates to the number of minutes specified by the *interval* argument. If *interval* is zero, periodic updates are not sent. A message is sent only when the server first appears and when it goes down. The default value for the argument *interval* is one minute. This is the value used by the Novell servers.

[no] novell update-time *seconds*

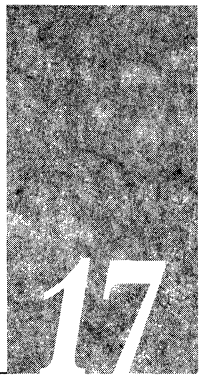
Allows the Novell routing update timers to be set on a per-interface basis.



Chapter 17
Routing PUP



Chapter 17
Routing PUP



This chapter describes routing configuration using the Xerox PARC Universal Protocol (PUP).

Cisco's Implementation of PUP

PUP was developed at Xerox's Palo Alto Research Center in the mid-1970s. PUP originally ran on the experimental three-megabits per second (mbps) Ethernet, the precursor to the IEEE 802.3 and Ethernet standards. The protocol is still used today by some Xerox workstations. The Cisco Systems routers support routing the PUP and provide a minimal set of PUP host functions, primarily the PUP Echo server and client. PUP packets can be routed over all Cisco-supported media. The Cisco PUP implementation uses the PUP GWINFO routing protocol to maintain routing information across a PUP network.

PUP Addresses

PUP addresses are 16-bit quantities written as two octal numbers separated by a pound sign. The following is an example of a PUP address:

```
10#371
```

The most significant eight bits of the address constitute the PUP network number, and the least significant eight bits constitute the PUP host number. In the example case, the network number is 10, and the host number is 371 (both in octal).

Configuring PUP Routing

PUP routing is enabled and disabled with the **pup routing** global configuration command, which has this simple syntax:

pup routing

no pup routing

When you start the PUP router process, the router begins routing PUP packets and sending out PUP routing updates.

PUP routing is enabled on a per interface basis by the **pup address** interface subcommand. The syntax of this command follows:

pup address *address*

no pup address

The argument *address* is the desired PUP address of the interface.

PUP routing can be disabled on a per interface basis by using **no pup address** subcommand.

The default state is for PUP routing to be disabled.

PUP Mapped to IP

This is the PUP support in versions of the cisco router software prior to Release 8.0. PUP addresses are automatically configured by the system. The PUP addresses are calculated from the subnet and host addresses of the interfaces connected to the IP network onto which the PUP network is mapped. The PUP network number is the least significant eight or fewer bits of the subnet address of an interface on the mapped network, and the PUP host number is the least significant eight or fewer bits of the host address of the interface. As a result of this, all of the PUP networks must be mapped onto one subnetted IP network.

To configure the mapping, use the global configuration command:

pup map *address*

The argument *address* is the network number of the major network over which the PUP network is overlaid.

All PUP routes acquired via the GWINFO routing protocol are entered into the IP routing table and labeled with a G. The reverse is *not* true; IP routes are not propagated into the PUP routing table.

PUP Miscellaneous Services

The Cisco PUP support allows broadcasts to the PUP Miscellaneous Services socket to be forwarded to another network (the helper address) where the appropriate servers can be found. This function is available in either mode of routing operation. To set the helper address, use the **pup helper-address** interface subcommand, which has this syntax:

pup helper-address *address*

no pup helper-address

The argument *address* is the PUP address of the desired server in the format described above.

The PUP Ping Command

The privileged EXEC command **ping** sends PUP Echo packets when the keyword **pup** is specified as the protocol. The **ping** menu will also prompt for a PUP protocol address; enter an address to begin the exchange.

Monitoring PUP

Use the EXEC commands described in this section to display statistics about the PUP network.

show pup arp

The command **show pup arp** displays PUP-specific ARP entries.

show pup route

The command **show pup route** displays routing entries obtained from the PUP routing protocol.

show pup traffic

The command **show pup traffic** displays statistical summaries of PUP packets when PUP routing is enabled.

Debugging PUP

Use the EXEC commands described in this section to display reports about activity on the PUP network. For each **debug** command, there is a corresponding **undebug** command that turns message logging off.

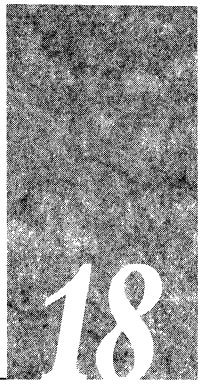
debug pup-packet

The command **debug pup-packet** enables logging of PUP routing activity to the console terminal.

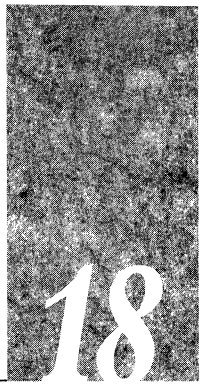
debug pup-routing

The command **debug pup-routing** enables logging of PUP GWINFO routing exchanges to the console terminal.

Chapter 18
Routing VINES



Chapter 18
Routing VINES



This chapter describes the Cisco Systems implementation of the Banyan VINES routing protocol, and provides the following information:

- Cisco's implementation of VINES
- Configuring routing in a VINES network, including special customizing procedures for such tasks as address resolution and filtering packets through the network
- Maintaining and monitoring the VINES network

Cisco's Implementation of VINES

The Banyan VINES protocol is a networking system for personal computers. The word "VINES" stands for Virtual Network System. This proprietary protocol was developed by Banyan, and is derived from Xerox's XNS protocol. The Cisco implementation of VINES has been designed in conjunction with Banyan. It is a separate effort from Cisco's XNS routing protocol support.

Cisco's implementation of VINES provides routing of VINES packets on all media types. Although software automatically determines a metric value that it uses in routing updates based on the delay set for the interface, the Cisco software implementation allows you to customize the metric.

Cisco's VINES software offers address resolution to respond to address requests, and propagation of broadcast addresses. MAC-level encapsulation is also available for Ethernet, IEEE 802.2, and Token Ring media. Name-to-address binding for VINES host names is also supported, as are access lists to filter packets to or from a specific network. As with all Cisco software, the VINES implementation also includes EXEC-level commands for maintaining, monitoring, and debugging the VINES network.

Configuring VINES

There are only two commands required to enable VINES routing:

- Use the global configuration command **vines routing** to enable routing.
- Use the interface subcommand **vines metric** to enable VINES processing on the interface.

All other configuration commands provide additional functionality or refinements. Each task is described in the following sections, and are followed by descriptions of the EXEC commands to maintain, monitor and debug the VINES network. Summaries of the global configuration commands and interface subcommands described in this chapter appear at the end of this chapter.

VINES Addressing

All VINES products automatically determine their addresses. The VINES specification guarantees interoperability. The network-level address consists of two numbers: a 32-bit network number, and a 16-bit number that Banyan refers to as a *subnet number* but that actually serves as a host number. Figure 18-1 illustrates the address format.

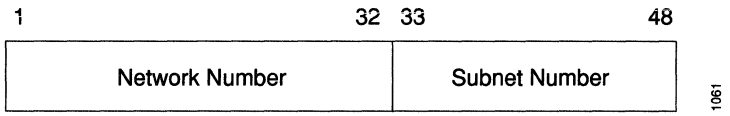


Figure 18-1 Banyan VINES Network-Level Addresses

Address conflicts are impossible since VINES servers use their Banyan-assigned unique key serial numbers as their network number, and use a subnet number of one. Since the keys are unique, the server addresses are unique. VINES clients do not have addresses, as such. The clients use a modified version of the address of the first file server found on the physical network. The clients assume the servers' network number, and are assigned a subnet number by that server.

Using this address assignment scheme, it is likely that two clients on the same physical LAN will have different addresses. This means that the router must keep a cache of local neighbors as well as of routing entries.

Cisco has been assigned a portion of the overall VINES network number space by Banyan. This portion is the set of all numbers that begin with the first 11 bits (of the 32) of 0110 0000 000. This will appear in all Cisco displays as a hexadecimal number beginning with 0x600 or 0x601. Routers attempt to automatically map themselves into the Cisco number space based upon the first nonzero Ethernet or Token Ring address found. In the unlikely

event that two routers map themselves to the same address, you may use the optional arguments to the **vines routing** command (see the section "Configuring VINES Routing") to override this selection.

Note: Older implementations of the Cisco software mapped themselves to numbers beginning with 0xFC0. This was done before Banyan made the address assignment.

The VINES Routing Table Protocol

Neighboring clients, servers, and routers are found using the Routing Table Protocol (RTP). Every VINES client sends a broadcast HELLO message at periodic intervals. This message indicates that the client is still operating and reachable from the network. For VINES Version 3, this interval is 30 seconds. It has been increased to once every 90 seconds in VINES Version 4 release. VINES servers send both *HELLO* and *update* messages periodically. The *update* message is used to indicate which remote servers and their satellite clients can be reached by routing through that server. These messages are sent by VINES servers every 90 seconds. Cisco routers also broadcast update messages once every 90 seconds; however, they do not send *HELLO* messages, as they are redundant when updates are being sent.

Configuring VINES Routing

Use the global configuration command **vines routing** to enable VINES routing. The command has this syntax:

vines routing [*address*]

no vines routing

Specify the optional *address* argument only when you are not using an Ethernet or Token Ring interface, since the router automatically maps itself into the VINES address space reserved for Cisco routers. When the optional *address* argument is used, the routers' VINES address is automatically configured as the specified address.

Use the **vines metric** interface subcommand to enable VINES processing on the defined interface. The command has this syntax:

vines metric [*number*]

no vines metric

The system automatically chooses a reasonable metric value that it uses in routing updates based upon the delay value set for the interface. These numbers are chosen to match as closely as possible to the numbers that a Banyan server would choose for the same type and speed of interface. Use the optional *number* argument to force the delay metric for the interface to a specific value. Delay metrics are used by Banyan servers to compute time-outs when communicating with other hosts. Be careful, when forcing metrics, that you do not set this number too high or too low and therefore disrupt the normal function of the Banyan servers. This number is generally inversely proportional to the speed of the interface. Some example numbers are:

Ethernet	2
16M TR	2
4M TR	4
56K Serial	45
9600 Serial	90

Example:

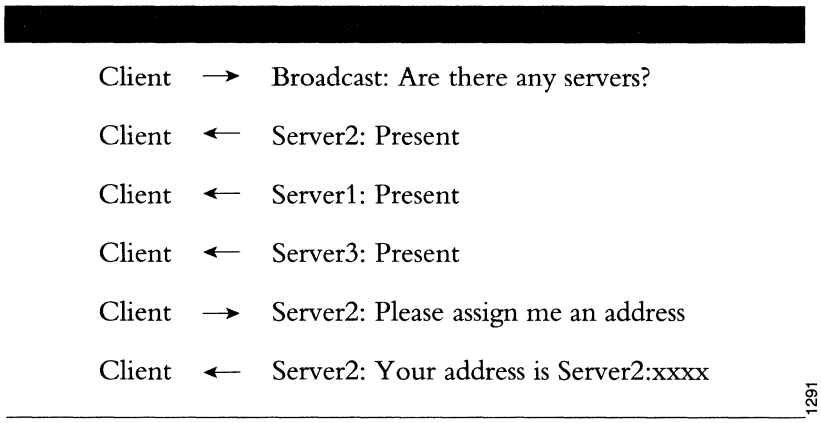
These commands enable VINES routing on the Ethernet 0 interface.

```
!
vines routing
!
interface ethernet 0
vines metric 2
!
```

Configuring Address Resolution

VINES clients boot without any knowledge of network-level addresses and preferred servers. The first thing that a client does after initializing its hardware interface is send a broadcast message looking for available servers on the network to which it is attached. The client then waits for responses from the network. Once a message is received, the client sends an address assignment request to the first server that replied to the previous message. That server computes a new, unique address based on its own network number, and assigns the address to that client.

The address assignment interaction is accomplished by sending responses directly to the MAC addresses of the client, since it does not have a network-level address yet. The following illustration depicts these events.



Cisco routers normally do not respond to any of these messages. There is only one situation where a Cisco response would be necessary, and this is when a *stub* network (an Ethernet network with only one connection to a router) has only clients and no server. The Cisco router must be explicitly configured to provide address resolution functionality. By turning on the Cisco address resolution feature, the router will begin responding to address requests and assigning addresses. The Cisco router then acts as a network communications service provider for the client. There must still be a VINES file server somewhere on the network for the client to connect to for all other services. Use the command **vines arp-enable** to enable this feature. The command has this syntax:

vines arp-enable

no vines arp-enable

The Cisco router generates a unique network number in a range assigned by Banyan, based on its VINES address.

A Banyan VINES network without a server needs to be configured with the **vines serverless** interface subcommand. This command provides special processing for certain broadcast packets and certain packets directed at the router. It is necessary for proper functioning of the clients on a network without a server. This is especially important when two networks, one with a server and one without a server, are not connected to the same router; in this situation you must use this command to build a path between the two networks. The command has this syntax:

vines serverless

no vines serverless

The default is **no vines serverless**.

Example:

Use the following example commands when interface Ethernet 1 is a stub Ethernet that does *not* contain any VINES servers.

```
interface ethernet 0
vines metric 2
!
interface ethernet 1
vines metric 2
vines arp-enable
vines serverless
!
```

See the section "VINES Configuration Examples," later in this chapter, for additional examples of use of this command.

Configuring VINES Encapsulation

Use the interface subcommand **vines encapsulation** to set the MAC-level encapsulation used for VINES broadcasts on the defined interface. The command has this syntax:

vines encapsulation [arpa | snap | vines-tr]

You can choose from these encapsulation types:

- **arpa** for Ethernet lines
- **snap** for IEEE 802.2 media
- **vines-tr** for Token Rings

The default encapsulation for Ethernet interfaces is **arpa**. The default encapsulation for Token Ring interfaces is **vines-tr.** All other media defaults to **snap.** As Banyan migrates to IEEE 802.2 encapsulation in future releases, the default will become **snap** for all media types.

Example:

This example configures the IEEE 802.2 SNAP encapsulation.

```
vines routing
!
interface ethernet 0
vines metric 2
vines encapsulation snap
```

Please note that the **vines encapsulation** command only affects broadcasts from the router. The router keeps track of which encapsulation is used by each of its neighbors, and will use the same style of encapsulation when talking directly to a neighbor.

Configuring Name-to-Address Mappings

Cisco provides only static name-to-address bindings for the VINES protocol. (This is completely separate from Banyan's Streettalk implementation.) Once entered, a VINES host name can be used anywhere that a VINES address can be used. This makes typing commands easier as it is much easier to remember and type the name of a system than it is to remember and type its address. Names are entered by using the **vines host** configuration command. The command has this syntax:

vines host *name address*

no vines host *name*

The argument *name* can be any length and sequence of characters separated by white space. The argument *address* consists of a VINES address in the form of network:subnet.

Example:

These commands define four host names.

```
!
! cisco names
vines host FARSLAYER 60002A2D:0001
vines host DOOMGIVER 60000A83:0001
! VINES PS/2 server
vines host COINSPINNER 0027AF92:0001
! PC clone client
vines host STUFF 0027AF92:8001
!
```

To examine the cache of VINES name-to-address mappings, use the EXEC command **show vines host.**

The output appears as follows.

```
VINES Name Table
            FARSLAYER                   60002A2D:0001
            DOOMGIVER                   60000A83:0001
            COINSPINNER                 0027AF92:0001
            STUFF                       0027AF92:8001
```

See the section "Monitoring the VINES Network" for more information about this command.

Configuring VINES Access Lists

An access list is a list of VINES network numbers kept by the Cisco router to control access to or from a specific network for a number of services.

Cisco's VINES access list provides network security by permitting or denying certain packets onto a network interface.

Use the global configuration command **vines access-list** to create or delete a VINES access list. The command has this syntax:

vines access-list *number* {**permit** | **deny**} **IP** *source-address source-mask dest-address dest-mask*

vines access-list *number* {**permit** | **deny**} *protocol source-address source-mask source-port dest-address dest-mask dest-port*

no vines access-list *number*

The argument *number* must be an integer between 1 and 100 for both variations of the command. The argument *protocol* may be an integer between 1 and 255, or one of these protocols:

- **SPP**—Sequence Packets Protocol
- **RTP**—Routing Table Protocol
- **ARP**—Address Resolution Protocol
- **IPC**—Interprocess Communications
- **ICP**—Internet Control Protocol

Source and destination addresses (*source-address* and *dest-address*) are VINES addresses written in the standard form.

Source and destination masks (*source-mask* and *dest-mask*) are also VINES addresses in the standard form, but they indicate the bits in the corresponding address that should be ignored.

Port numbers (*source-port* and *dest-port*) are integers in the range of 0 to 65,535.

Use the interface subcommand **vines access-group** to assign an outgoing VINES access list to the defined interface. The command has this syntax:

vines access-group *list*

no vines access-group

The argument *list* is the number of an access list defined with the **vines access-list** command.

Examples:

In this example, the first line permits any two servers (subnet of 1) to talk to each other, regardless of their network numbers, and the second line denies everything else.

```
!
vines routing
vines access-list 1 permit IP 0:1 ffffffff:0    0:1 ffffffff:0
vines access-list 1 deny   IP 0:0 ffffffff:ffff 0:0 ffffffff:ffff
!
```

In this example, the first line prohibits any communication on IPC port number 15 (hex 0F); the second line permits all other communication:

```
vines access-list 1 deny   IPC 0:0 ffffffff:ffff 0xf 0:0 ffffffff:ffff 0xf
vines access-list 1 permit IP 0:0 ffffffff:ffff     0:0 ffffffff:ffff
```

VINES Configuration Examples

Use the following configurations examples to help you configure your VINES routing network.

Propagating Broadcasts

The following examples illustrate how to configure a VINES network where the clients are all behind one router, and the servers are behind another router, as depicted in the following illustration:

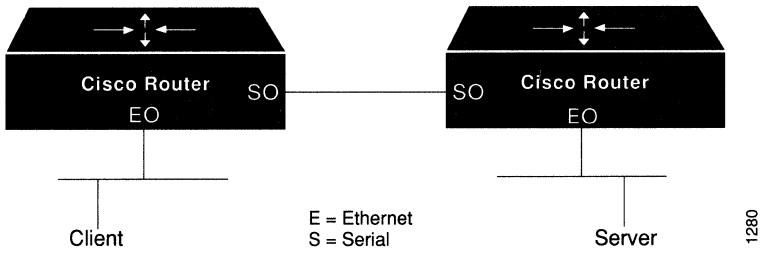


Figure 18-2 VINES Client/Server Configuration

Example for Client:

```
!
vines routing
!
interface ethernet 0
vines metric 2
vines serverless
!
interface serial 0
vines metric 15
vines serverless
!
```

Example for Server:

```
!
vines routing
!
interface ethernet 0
vines metric 2
!
interface serial 0
vines metric 15
vines serverless
!
```

Filtering Packets

The following example illustrates how to configure an access list that allows packets across two VINES networks and filters all packets across a third VINES network.

Example:

```
!
vines routing
vines access-list 1 permit IP 60000A83:0 0:FFFF 60003753:0 0:FFFF
vines access-list 1 deny   IP 0:0 FFFFFFFF:FFFF 0:0 FFFFFFFF:FFFF
!
```

Maintaining the VINES Network

Use these EXEC commands to maintain the VINES routing protocol tables.

Use the privileged EXEC command **clear vines neighbor** to delete the specified address from the neighbor table maintained by the router. This also forces the deletion of all routing table entries that have the given neighbor as the first hop in their path.

clear vines neighbor {*address* | *}

The argument *address* is the neighbor address, or may be an asterisk (*), which deletes all entries from the neighbor routing table except for the entry for the router itself.

Example:

This command clears all entries from the neighbor routing table.

```
gateway#clear vines neighbor *
```

Use the privileged EXEC command **clear vines route** to delete the specified network address from the routing table maintained by the router.

clear vines route {*address* | *}

The argument *address* is the routing address, or may be an asterisk (*), which deletes all entries from the routing table except for the entry for the router itself.

Example:

This command clears all entries from the routing table.

```
gateway#clear vines route *
```

Monitoring the VINES Network

Use these EXEC commands to monitor networks configured for VINES routing.

Displaying the VINES Name Table

Use the EXEC command **show vines host** to display the contents of the VINES name table.

show vines host [*name*]

If the optional *name* argument is included, then only that entry is printed from the table. If no name is present, then the entire table is printed.

Displaying VINES Interface Settings

Use the EXEC command **show vines interface** to display all VINES-related interface settings.

show vines interface [*interface unit*]

If no interface name is supplied, then the values for all interfaces are printed, along with the VINES global parameters.

Sample output from this command is shown below.

```
 VINES address is 60000A83:1
Next client will be 60000A83:8000
-More-
Ethernet 0 is up, line protocol is up
 VINES protocol processing disabled
-More-
Serial 0 is up, line protocol is up
 Interface metric is 27 (5.4 seconds)
 Outgoing access list is not set
-More-
Ethernet 1 is up, line protocol is up
 VINES unknown host encapsulation is ARPA
 Interface metric is 2 (0.4 seconds)
 Outgoing access list is not set
 -More-
```

```
Serial 1 is up, line protocol is up
 VINES protocol processing disabled
```

Displaying the VINES Neighbor Table

Use the EXEC command **show vines neighbors** to display the entire contents of the VINES neighbor table maintained by this router.

show vines neighbors [*address*]

If the optional *address* argument is supplied, then only that entry from the table is displayed.

Sample output from this command is shown below.

```
Network   Subnet Age   Metric Hardware Address Type      Interface
0027AF92   0001    16       2    0260.8C3C.141C   ARPA      Ethernet0
60000A83   0001    12       2    0000.0C00.0A84   ARPA      Ethernet1
0027AF92   8001    12       2    0000.C05A.0511   ARPA      Ethernet0
60002A2D  0001     -      0    -               -        -
```

Displaying the VINES Routing Table

Use the EXEC command **show vines route** to display the entire contents of the VINES routing table.

show vines route [*address*]

If the optional *address* argument is supplied, then only the routing entry for that network is displayed.

Sample output of this command is shown below.

```
Codes: R - RTP derived, C - connected, S - static, 4 routes
R Net 0027AF92 [2] via 0027AF92:1, 21 sec, 0 uses, Ethernet0
R Net 60000A83 [2.H] via 60000A83:1, 503 sec, 0 uses, Ethernet1
C Net 60002A2D is this router's network, 0 uses
```

A .H after the metric indicates that the route is currently in a hold down state.

Displaying VINES Traffic

Use the EXEC command **show vines traffic** to display the statistics kept on VINES protocol traffic.

show vines traffic

Sample output of this command is shown below.

```
     Rcvd: 4218 total, 0 format errors, 0 checksum errors, 12 bad hop count,
         3994 local destination
   Bcast: 3994 received, 2 sent, 1 forwarded
         0 not lan, 0 not gt 4800, 0 no charge
    Sent: 3512 generated, 147 forwarded
         0 encapsulation failed, 65 no route
     IPC: 0 errors received, 12 errors sent, 0 metric sent
    Echo: Received 0, Sent 2
Unknown: 0 packets
```

VINES Ping Command

The EXEC command **ping** allows the network administrator to determine network connectivity by sending datagrams to the host in question. A sample session is shown below.

```
Doomgiver# ping coinspinner
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Verbose [n]:
Type escape sequence to abort.
Sending 5, 100-byte VINES Echos to 27AF92:1,
timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/7/8 ms
```

VINES Trace Command

The EXEC command **trace** allows the administrator to determine the path that a packet takes when traversing a VINES network. This command does not produce the names of any VINES servers that are traversed. Sample output of tracing through Cisco routers to reach a Banyan file server is shown below.

```
Doomgiver#trace coinspinner
Type escape sequence to abort.
Tracing the route to COINSPINNER (27AF92:1)
 0 FARSLAYER (60002A2D:1) 0 msec 4 msec 4 msec
 1 COINSPINNER (27AF92:1) 4 msec 4 msec 8 msec
```

Debugging the VINES Network

Use these **debug** commands to obtain reports of VINES' routing functionality.

The following **debug** commands are disabled by entering the **undebug** version of the command once it is enabled.

debug vines-arp

The EXEC command **debug vines-arp** enables logging of all ARP protocol processing that occurs in the router.

debug vines-echo

The EXEC command **debug vines-echo** enables logging of all MAC-level echo processing that occurs in the router. This echo functionality is used by the Banyan interface testing programs.

debug vines-packet

The EXEC command **debug vines-packet** enables logging of general VINES debugging information. This includes packets received, generated, and forwarded, as well as failed access checks and other items.

debug vines-routing

The EXEC command **debug vines-routing** enables logging of all RTP update messages sent or received, and all routing table activities that occur in the router.

debug vines-table

The EXEC command **debug vines-table** enables logging of all modifications to the VINES routing table. This command provides a subset of the information provided by the **debug vines-routing** command.

VINES Global Configuration Command Summary

Following is an alphabetical list of the VINES global configuration commands that specify system-wide parameters for VINES support.

[no] vines access-list *number* {**permit** | **deny**} **IP** *source-address source-mask dest-address dest-mask*

[no] vines access-list *number* {**permit** | **deny**} *protocol source-address source-mask source-port dest-address dest-mask dest-port*

Creates or deletes a VINES access list. The argument *number* must be an integer between 1 and 100 for both variations of the command. The argument *protocol* is an integer between 1 and 255, or one of these protocols.

- **SPP**—Sequence Packets Protocol
- **RTP**—Routing Table Protocol
- **ARP**—Address Resolution Protocol
- **IPC**—Interprocess Communications
- **ICP**—Internet Control Protocol

Source and destination addresses (*source-address* and *dest-address*) are VINES addresses written in the standard form.

Source and destination masks (*source-mask* and *dest-mask*) are also VINES addresses in the standard form, but they indicate the bits in the corresponding address that should be ignored.

Port numbers (*source-port and dest-port*) are integers in the range of 0 to 65,535.

The **no** variation with the access list number deletes the access list.

[no] vines host *name address*

Adds or deletes an entry to the VINES name-to-address mapping table. Once entered, a VINES name can be used anywhere that a VINES addresses is requested.

[no] vines routing [*address*]

Enables or disables VINES routing. The argument *address* is not necessary if you have Ethernet interfaces, as the router will automatically map itself into the VINES address space that is reserved for Cisco. If the optional *address* argument is given, then the routers' VINES address will be forced to be the specified address.

VINES Interface Subcommand Summary

Following is an alphabetical list of the VINES interface subcommands that specify parameters for interfaces configured for VINES routing. These commands must follow an **interface** command.

[no] vines access-group *list*

Assigns or removes an outgoing VINES access list to the defined interface. The argument *list* is the number of an access list defined with the **vines access-list** command.

[no] vines arp-enable

Enables or disables the processing of ARP packets received on the defined interface. When enabled, the router responds to ARP packets and assigns network addresses to clients.

vines encapsulation [arpa | snap | vines-tr]

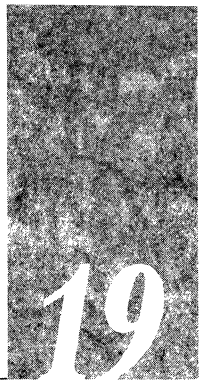
Sets the MAC-level encapsulation used for VINES broadcasts on the defined interface. The current defaults are: **arpa** for Ethernet lines, **snap** for IEEE 802.2 media, and **vines-tr** for Token Rings.

[no] vines metric [*number*]

Enables or disables the processing of VINES processing on this interface. The system automatically chooses a reasonable metric value, which is used in routing updates, and is based upon the delay value set for the defined interface. When the optional *number* argument is supplied, the system forces the VINES delay metric for this interface to the supplied number.

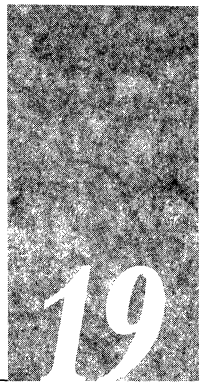
[no] vines serverless

Enter to configure a Banyan VINES network without a server. This command can be used on several routes to build a path to a network that contains servers. The default is **no vines serverless**.



Chapter 19
Routing XNS

Chapter 19
Routing XNS



This chapter describes how to configure your router to perform routing for packets following the Xerox Network Systems (XNS) stack of protocols.

You will find information about the following topics and tasks:

- How to configure a routing process for XNS routing. This includes information on the principal XNS protocols, XNS addressing, how to configure your router to route XNS traffic, managing security issues, and maximizing performance.
- How to configure helper addresses for broadcast traffic.
- How to set up routes, including setting metrics.
- How to configure options for access lists and filters.
- Configuration restrictions and requirements for encapsulation of all important lower layer protocols, including Token Ring, FDDI, Ethernet, and others.

This chapter also contains configuration information on Ungermann-Bass' and 3Com's XNS-derived protocols.

The section "XNS Configuration Examples" later in this chapter includes examples of actual configurations for working XNS networks, including Ungermann-Bass and 3Com. For additional information on configuring access lists in 3Com networks, please consult the Application Note titled *A Detailed Look at Access Lists in 3Com XNS.*

Cisco's Implementation of XNS

Cisco provides a subset of the XNS protocol stack to support XNS routing on its routers. The same Cisco routers that route XNS can also route another protocol stack like TCP/IP or DECnet. At the physical and data link layers, XNS traffic can be routed over Ethernets, FDDI, Token Rings, or point-to-point serial lines running HDLC or LAPB.

DECnet

In previous releases of Cisco's routing software, it was not possible to run DECnet Phase IV and any of the XNS family of protocols simultaneously in a router that included both Ethernet and Token Ring interfaces. This restriction was removed in Release 8.2. There are no changes to the syntax of any configuration commands.

Ethernet and Token Ring

When XNS routing is enabled, the address is either the IEEE-compliant address specified in the XNS routing configuration command, or the first IEEE-compliant address in the system. The address is also used as the node address that non-LAN media (notably serial links) use for their XNS node addresses. This address is then used as the default XNS node address on non-LAN nodes.

XNS was originally designed by the same company (Xerox) that developed Ethernet, so it was designed to run over Ethernet. If you implement an XNS stack over Token Ring at the first two layers, you have to encapsulate the lower layers differently than you would for Ethernet. Encapsulation defines the kind of envelope (layer 1 and layer 2 bits) in which three through seven of the XNS protocol and data packet are wrapped prior to being transmitted. Because there is no agreed-upon mechanism for encapsulating XNS packets on a Token Ring network, many vendors have invented their own encapsulation methods. We will discuss this in detail in the section "Configuring XNS Over Token Ring" later in this chapter.

XNS Addresses

Both Data Link (MAC) and Network Layer addressing is needed in any network that supports routing; XNS is no exception. An XNS Network Layer address is composed of three fields.

- The network number uniquely identifies a network in an internet.
- The host number uniquely identifies a host on a network.
- The socket number uniquely identifies a socket within the operating system of the host. A socket is a transport address that is the source or destination of packets.

Network and Host Numbers

The network number is expressed in decimal format in Cisco configuration files and routing tables. When configuring a Cisco router, enter the network number in decimal.

Addresses must be unique throughout an XNS internet. Since both the network number and the host address are needed to deliver traffic to a host, addresses are usually given as network numbers, followed by host addresses, separated with dots. An example address follows.

Example:

```
47.0000.0c00.23fe
```

Here, the network number is 47 (decimal), and the host address is *0000.0c00.23fe* (hex).

Socket Numbers

An XNS socket number is a 16-bit field in the IDP header. Sockets are selected by the client processes of each host before a connection is established. Certain socket numbers are considered to be well-known sockets (WKS), which means that the service performed by the software using them is statically defined. Each system element supplying a specific well-known service does so at the same WKS. Socket numbers above the well-known range are arbitrary, which means that they can be selected and reused at random.

- A socket number of zero means all.
- A socket number of all ones (0xFFFF hex) means unknown.
- Well-known socket numbers range from 1 to 0x0BB8 hex (3000 decimal). All other socket numbers may be dynamically assigned and reused.

Configuring XNS

Follow these steps to configure XNS routing:

Step 1: Enable XNS routing using the **xns routing** command.

Step 2: Assign a unique XNS network number to each interface, using the **xns network** command.

Step 3: Optionally configure performance parameters and helper addresses (which help you manage broadcast traffic).

Step 4: Optionally configure access lists and filters.

Additionally, EXEC-level commands for monitoring and debugging the XNS network are available. These commands are described in the last few sections of this chapter, along with concise summaries of the global and interface-specific configuration commands.

Enabling XNS Routing

The first step in the configuration process is to specify XNS as the protocol you are enabling. Use the **xns routing** global configuration command to enable XNS routing. The full syntax of this command follows.

xns routing [*address*]

no xns routing

The optional argument *address* is the router interface's complete address, which is expressed in hexadecimal format. The **no xns routing** disables all XNS processing.

Example:

In the example below, an interface whose address is *0123.4567.abcd* is enabled for XNS routing.

```
xns routing 0123.4567.abcd
```

If the argument *address* is omitted, the router will use the first IEEE-compliant (Token Ring, FDDI, or Ethernet) interface hardware address it finds.

Your next step is to use the **xns network** interface subcommand to assign a decimal XNS network number to an interface and enables that interface to run XNS protocols. The full syntax of the command follows.

xns network *number*

no xns network

The argument *number* is the network number, in decimal format.

Interfaces not enabled to run XNS ignore any XNS packets that they receive. Every XNS interface in a system must have a unique XNS network number.

Example:

This example starts the routing process with no specific address specified, so the router will use the first IEEE-compliant interface hardware address it finds, then specifies network number 20.

```
xns routing
xns network 20
```

Configuring Static Routing

To add a static route from your router to a remote destination in the XNS routing table, use the **xns route** global configuration command. The full syntax follows.

xns route *network host-address*

no xns route *network host-address*

The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and a hexadecimal host number, separated by a dot.

Example:

The following example sets up a host (router) address of *51.0456.acd3.1243* as the static recipient of packets destined for network 25.

```
xns network 51
xns route 25 51.0456.acd3.1243
```

Managing Throughput

You have several options for managing throughput while dynamically routing. You can set up multiple paths and send packets over these paths in a round-robin fashion. You can also enable or disable the route cache. You can even adjust how often your router uses RIP to send routing table updates to its neighbors provided all the neighbors are Cisco routers.

Setting Multiple Paths

XNS allows your router to select from multiple paths to a destination in order to increase throughput in the network. The default assumes that the router will pick one best path and send all traffic on this path. You can tell your router to compile two or more paths that have equal cost (hop count in XNS' case) and balance the traffic load across all the available paths.

To set the maximum number of multiple paths, use the **xns maximum-paths** global configuration command. The full syntax follows.

xns maximum-paths *paths*

no xns maximum-paths

The argument *paths* is the number of paths to be assigned. The default value for *paths* is 1. Packets are distributed over the multiple paths in round-robin fashion on a packet-by-packet basis. To disable multiple paths, use the command with the default value of one. The **no xns maximum-paths** command restores the default.

The EXEC command **show xns route** displays the entire routing table, so you can examine your additional routes and the maximum path cost for each. See the section "Monitoring an XNS Network" when you are ready to use this command.

Example:

The following example asks the router to send packets over two alternate paths, if available.

```
xns maximum-paths 2
```

Enabling XNS Fast Switching

XNS fast-switching achieves higher throughput by using a cache created by previous transit packets. Fast-switching also provides load sharing on a per-packet basis. As soon as you enable XNS routing, fast switching is automatically enabled as well. Use the **xns route-cache** interface command to enable or disable fast switching. The full syntax of the command follows.

xns route-cache

no xns route-cache

Use the **no xns route-cache** command to disable fast switching and then the **xns route-cache** interface subcommand to re-enable fast switching.

Adjusting Timers

RIP sends routing table updates to its neighbor routers every 30 seconds, unless you specify some other time interval. You can reset the routing update timers on a per-interface basis using the **xns update-time** interface subcommand:

xns update-time *seconds*

XNS routing timers are affected by the value set for the *seconds* argument:

- XNS routes are marked invalid if no routing updates are heard within six times the value of the update timer.
- XNS routes are removed from the routing table if no routing updates are heard within eight times the value of the update timer.
- The granularity of the update timer is determined by the lowest value defined.
- The minimum is ten seconds.

If you want to return to the default condition, you cannot use a **no** variation of the command. Use the **xns update-time** command with a value for the *seconds* argument of 30.

Note: Be careful with this command. Use it only in an all-Cisco environment. Make sure that all timers are the same for all routers attached to the same network segment.

The EXEC command **show xns route** displays the value of these timers. See the section "Monitoring an XNS Network" when you are ready to use this command.

Configuring XNS Over Token Ring

There is no standard way of packaging an XNS packet for transit across an 802.5 Token Ring. XNS was designed to exist above Ethernet at the lowest two network layers. The general process of wrapping new header information around a packet so that it can traverse an unfamiliar network is called *encapsulation*. Each Token Ring vendor has its own method of encapsulating XNS packets. Cisco supports the encapsulation methods of LAN vendors Ungermann-Bass, 3Com, and IBM.

Use the **xns encapsulation** interface subcommand to select the encapsulation method:

xns encapsulation *keyword*

If your Token Ring is an IBM installation, you use the default *keyword* **snap**. Other options are:

- **ub** for Ungermann-Bass
- **3com** for older 3Com Corporation products. Some 3COM 3+ hosts do not recognize Token Ring packets with the source-route bridging RIF field set. You can work around this discrepancy by using the **no multiring xns** interface subcommand on Token Ring interfaces that are used for 3Com XNS routing. See Chapter 21 in Part Five for more information.

Example:

In the following example, XNS is enabled, an interface is defined for Token Ring, and then Ungermann-Bass is specified as the encapsulation option, as shown:

```
xns routing
xns ub-routing
interface tokenring 0
xns network 1234
xns encapsulation ub
```

You will find more information on the Ungermann-Bass version of XNS in the following section.

Configuring Ungermann-Bass Net/One XNS

Ungermann-Bass's Net/One protocols were derived from the protocols of the Xerox Network Systems (XNS) stack and are very similar to them. However, Net/One is not equivalent to standard XNS. The following are the important differences between the Net/One implementation and standard XNS:

- Net/One routers use an Ungermann-Bass proprietary routing protocol instead of standard XNS RIP. Although they generate both Ungermann-Bass and standard RIP update packets, Net/One routers only listen to Ungermann-Bass updates. Cisco supports generation of Ungermann-Bass updates, and Cisco routers can interoperate with Ungermann-Bass routers in certain restricted configurations. Cisco routers do not presently listen to Ungermann-Bass updates, and illegal configurations can lead to routing loops.
- Net/One routers send periodic HELLO packets, which hosts use to discover the addresses of their local routers. Ordinary XNS hosts use RIP for this purpose. Cisco routers can be configured to generate Ungermann-Bass HELLO packets.
- Net/One equipment uses a non-XNS protocol for network software downloads. During the downloading process, XNS network numbers are embedded in the packets of this protocol. Ungermann-Bass routers pass the booting protocol from network to network, and modify the embedded network numbers. Cisco equipment does not understand the Net/One booting protocol, and Net/One NIUs cannot be booted through Cisco routers in XNS routing mode.

- The Ungermann-Bass Net/One Network Resource Monitor (a network management and monitoring tool) uses XNS packets whose destination host addresses are specific nodes, but whose destination network addresses are broadcast (network -1). These packets are sent as MAC-layer broadcasts, and are expected to be flooded throughout the XNS internet. Cisco does not support flooding of these packets, and this prevents the NRM from doing node discovery through a Cisco router.
- Net/One equipment uses a set of proprietary network management protocols. Cisco routers do not participate in these protocols.

Be aware that:

- The network topology must contain no Ungermann-Bass routers interconnected by media slower than Ethernets, or the topology may contain no loops which pass through both Cisco and Ungermann-Bass routers.
- The Network Resource Monitor cannot be used to manage a network through a Cisco router.
- No Cisco routers may fall in the path between a Net/One NIU and the Network Management Console from which that NIU downloads its software.

Ungermann-Bass Routing Protocol

The routing protocol used by Net/One routers is a distance-vector or Bellman-Ford protocol, similar but not identical to standard XNS RIP. The major difference between the two protocols lies in the metrics used. While standard RIP uses a hop count to determine the best route to a distant network, the Ungermann-Bass protocol uses a path delay metric. The standard RIP protocol maintains information only about hop counts, while the Ungermann-Bass protocol maintains information both about hop counts and about its own metrics. Ungermann-Bass routers generate standard RIP updates by extracting the hop-count values from the Ungermann-Bass routing protocol. Cisco routers generate Ungermann-Bass updates by computing a delay value from the hop count maintained by standard RIP, assuming that all hops are Ethernet LANs. Ungermann-Bass routers gather information only from Ungermann-Bass updates, while Cisco routers gather information only from standard RIP.

Ungermann-Bass routers do not implement the *split horizon* routing optimization; that is, they will advertize a route to a network through the interface via which packets for that network would actually be sent. Cisco software earlier than version 8.1(25) likewise did not implement split horizon for the Ungermann-Bass proprietary routing protocol. This caused packet forwarding loops in even very simple configurations. The forwarding loops typically manifested themselves as complete failures of the Ungermann-Bass routers.

Cisco software later than 8.1(25) does implement split horizon for the Ungermann-Bass routing protocol, and thus avoids forwarding loops in simpler configurations. Some stable forwarding loops are still possible, however, and transient forwarding loops caused by changes in the network topology (typically when links fail) are easy to create.

Avoiding Routing Loops

An example of a configuration in which both routing instability and a forwarding loop will occur is the following:

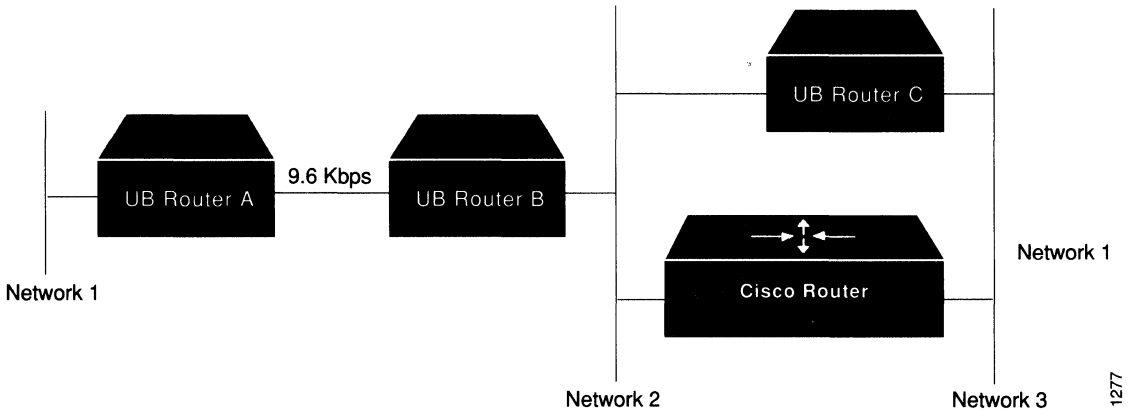


Figure 19-1 Sample Ungermann-Bass Routing Configuration

Here, Ungermann-Bass router *B* learns that it has a path to network *1* via Ungermann-Bass router A, in one hop, with a very high delay metric (introduced by the 9.6Kbps line). It re-advertises that route on network *2*, and the Cisco router learns that it can reach network *1* via Ungermann-Bass *B*. It advertises that route on network *3* using both standard RIP and the Ungermann-Bass routing protocol. Because of the metric conversions, the delay field in the Ungermann-Bass routing packets generated on network *3* by the Cisco router shows the delay of two Ethernet hops, rather than the delay of one Ethernet hop and one 9.6 Kbps hop.

Since this delay is less than the metric being advertised by Ungermann-Bass *B*, Ungermann-Bass *C* learns that the route to network 1 is via the Cisco's network *3* interface. It advertises this route onto network *2*, with a Ungermann-Bass delay metric equivalent to three Ethernet hops, and still much less than the delay of the 9.6 Kbps line. Ungermann-Bass *B* therefore switches its route to point to Ungermann-Bass *C*.

At this point, any packet destined for network *1* will be sent by Ungermann-Bass *B* to Ungermann-Bass *C*, which will send it to the Cisco router. The Cisco router will send it back to Ungermann-Bass *B*, and so forth until the packet exceeds the XNS maximum hop count of 16.

On the next update cycle, the Cisco router will learn the route through Ungermann-Bass *C* instead of through Ungermann-Bass *B* (since *B*'s hop count will now be greater than C's), and there will be a tight forwarding loop between the two. This will continue until the routes count to infinity and time out, at which point the process will begin again.

The best way to avoid problems like these is to avoid loops in the network topology which contain both Cisco and Ungermann-Bass routers. Since Cisco routers will assign Ethernet-level delays to all hops, an alternate strategy is to avoid links slower than Ethernets between Ungermann-Bass routers.

Configuring Ungermann-Bass Net/One Routing

To enable Ungermann-Bass Net/One routing, use the **xns ub-routing** global configuration command. The full syntax of this command follows:

xns ub-routing

no xns ub-routing

This command causes HELLO packets and routing updates in Ungermann-Bass format to be sent out through all the interfaces on which XNS is enabled. In addition, it changes the way some XNS broadcast packets are flooded. In normal operation, when XNS broadcast flooding is configured, packets with zero destination network numbers are flooded. With **xns ub-routing** in effect, such packets are not flooded.

There is an Ungermann-Bass configuration example in the "XNS Configuration Examples" section of this chapter.

Helper Addresses and Broadcast-Forwarding

You need to decide how you want the router to handle different kinds of broadcast packets. This is an excellent opportunity for you to reduce unnecessary network traffic, if you think about all your options carefully.

Many broadcasts occur when a node first becomes active on the network. A host will generate an Address Look-up packet when it does not know the current address of whatever other host is supposed to receive its next packet—the local server, for instance. It is generally not a good idea to place a router between users and the servers that carry their primary applications; you should minimize internet traffic. However, if you need that server configuration for some other reason, you need to ensure that users can broadcast between networks without cluttering the internet with unnecessary traffic.

Normally, a packet is sent to a specific network or series of networks. A flooded broadcast packet, with a destination network number of -1, is sent to every network. By default, flooding is off.

You can configure your routers to block all broadcasts and that may be an appropriate option in some circumstances. You have more than simply this all-or-nothing choice, however, by using the xns helper-address and forward-protocol commands.

Using Helper Addresses

The **xns helper-address** interface subcommand causes flooded (all-nets) broadcasts to be forwarded to another address that you specify. This address is called a helper address. The full syntax of this command follows:

xns helper-address *host-address*

no xns helper-address *host-address*

The argument *host-address* is a dotted combination of the network and host addresses as explained in the **xns route** command.

Here is how different kinds of broadcasts are handled:

- Broadcasts specifically sent to another network are routed to that network, and then broadcasted. This means broadcast packets with a network address that is *not* any of the following:
  - — All ones
  - — Zero
  - — The local network
- Broadcasts to all networks (network number of all ones) are forwarded regardless of protocol.
- Broadcasts directed to devices on the same connected network (network address equals the local network number or zero) are checked to see if the protocol type matches an entry in the forward protocol lists and, if a match is found, are forwarded.

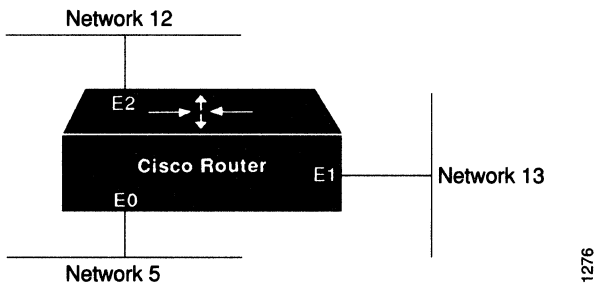


Figure 19-2 Helper Addresses

Ignore the protocol-filtering issue for a moment and just consider some examples of how helper-addresses are used. In Figure 19-2, the E0 interface has a helper address set, with the helper on network 12, available through the E2 interface. Some broadcast packets are being received on this E0 interface:

- A broadcast packet with a network address of 5 will be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 0 will also be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 13 will be sent through the E1 interface directly to network 13. It will not be sent to the helper address.

All Nets Broadcasts

To configure the interface for "all nets broadcast flooding," define the XNS **helper-address** for the interface as:

```
xns helper-address -1.FFFF.FFFF.FFFF
```

If you use this address, your router will flood packets with a destination network address of -1 coming in on that interface. Packets will flood to all networks except the source network. Although flooding always creates some duplicates, packets will not loop forever.

Forwarding Specific Protocols

The **xns forward-protocol** global configuration command allows you to specify which XNS protocols will be forwarded when the router receives a broadcast packet. The interface must already have an XNS helper address set. The full syntax of the command is shown below:

xns forward-protocol *type*

no xns forward-protocol *type*

The argument *type* is a decimal number corresponding to an appropriate XNS protocol. See the documentation accompanying your XNS implementation to determine the protocol type number.

Use the **no xns forward-protocol** command and the appropriate argument to remove this function.

In Figure 19-2, the E0 interface is set to a helper address and forwarding for protocol *1* only. (The protocol numbers will vary depending on your XNS implementation.) Broadcast packets are *arriving* on the E0 interface from network *5*:

- A broadcast packet destined for network *5* and protocol *1* will be sent to the helper address.
- A broadcast packet destined for network *5* and another protocol is discarded because it fails the protocol test.
- A broadcast packet destined for network *0* and protocol *1* is sent to the helper address.
- A broadcast packet destined for network *0* and a protocol other than *1* is discarded.

Please note that:

- A broadcast packet destined to network *12* is sent out on the E2 interface to network *12*. This has nothing to do with the helper-address or protocol forwarding.
- A broadcast packet destined to a network of all-ones is flooded to networks *12* and *13* on the E1 and E2 interfaces, regardless of protocol type. It is not sent to the broadcast address on network *5*, because it originated on network *5* and all-nets broadcasts do not flood onto the originating network.

Example:

In this example, a helper address corresponding to the local host is specified and protocol type 2 is forwarded to that host.

```
interface ethernet 0
xns helper-address  26.FFFF.FFFF.FFFF
xns forward-protocol 2
```

Configuring XNS Access Lists and Filters

You may configure XNS access lists to filter traffic on XNS interfaces. If you are specifically interested in 3Com access lists, please consult the Application Note titled *A Detailed Look at Access Lists in 3Com XNS.*

XNS access lists are numbered from 400 to 499 and filter on the source and destination addresses only. This means that they can prevent traffic from going to specific hosts or coming from specific hosts. Extended XNS access lists are numbered from 500 to 599 and filter on XNS protocol and socket fields. The extended filters can prevent entire classes of packets (SPP, Echo, and so on) from passing a router interface and they can also filter traffic going to or coming from specific processes.

Configuring XNS Access Lists

To configure an access list, use the **access-list** global configuration command, with the following syntax:

access-list *number* {**deny** | **permit**} *XNS-source-network.*[*source-address*[*source-mask*]]*XNS-destination-network.*[*destination-address*[*destination-mask*]]

no access-list *number*

Note: For typographic reasons access list command examples are shown on multiple lines; it must be on a single line when given as a configuration command.

The argument *number* must be a number between 400 and 499.

The only required parameter for standard XNS access lists is the XNS source network address. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered. (Note that XNS uses MAC addresses as the node ID; see the examples for further clarification.)

Example 1:

This example denies access from source network *–1* (all XNS networks) to destination network 2.

```
access-list 400 deny -1 2
```

Example 2:

This example denies access from XNS source address *21.0000.0c00.1111*. The destination network does not make any difference:

```
access-list 400 deny 21.0000.0c00.1111
```

Example 3:

This example denies access from all hosts on network *1* that have a source address beginning with *0000.0c*:

```
access-list 400 deny 1.0000.0c00.0000 0000.00ff.ffff
```

Example 4:

In this example, we deny access from source address *11.1622.15* on network *21* to destination address *01D3.020C.0022* on network *31*:

```
access-list 400 deny 21.011.1622.0015 0000.0000.0000 31.01D3.020C.0022 0000.0000.0000
```

Configuring Extended Access Lists

For extended access lists, the **access-list** command again must be typed on one line using this syntax (the negative form of the command is also included):

access-list *number* {**deny** | **permit**} *xns-protocol source-network.*[*source-address* [*source-mask*]] *source-socket destination-network.*[*destination-address* [*destination-mask*]] *destination-socket*

no access-list *number*

The argument *number* must be a number between 500 and 599.

The source and destination addresses and masks are optional. The protocol number *xns-protocol* is the only required parameter. A network number of –1 matches all networks; a socket number of 0 matches all sockets.

Example 1:

To deny access to packets with protocol *1* from source network *1*, source socket 1234 that are trying to be routed to destination network *2*, destination socket *1234*:

```
access-list 500 deny 1 1 1234 2 1234
```

Example 2:

This example adds masks for the source and destination networks:

```
access-list 500 deny 1 21.110011.1622.001500.0000.0000 31.01D3.020C.0022
0000.0000.0000. 1234
```

Filtering Outgoing Packets

An XNS access list group number is assigned with the **xns access-group** interface subcommand. The full syntax of this command follows.

xns access-group *number*

no xns access-group *number*

The argument *number* specifies the access list number defined by the XNS global **access list** command. This command causes all XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If the packet fails the access list, it is not transmitted, but is discarded instead.

Example:

This example uses the **xns access-list** command to deny access to protocol *1* from source network *1*, source socket *1234* to destination network *2*, destination socket *1234*, then assign number 500 to a group to be filtered:

```
access-list 500 deny 1 1 1234 2 1234
!
interface ethernet 0
xns access-group 500
```

Filtering XNS Routing Updates

This section describes the filtering commands that use access lists to control what routing information is accepted, or passed on, within XNS networks. The commands filter incoming traffic and outgoing routing information, and specific routers.

Each access list entry contains only one address parameter and the list must be in the range 400 to 499. How this address is interpreted is defined by the command that will use the list.

As with all other Cisco access lists, an implicit *deny everything* is defined at the end of the list. If this is not desired, an explicit *permit everything* definition must be included at the end of the list. Complex configuration examples are shown in the "XNS Configuration Examples" section.

Input Filters: Adding to the Routing Table

To control which networks are added to *your* router's routing table, use the **xns input-network-filter** interface subcommand.

xns input-network-filter *access-list-number*

no xns input-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In this example, access list 476 controls which networks are added to the routing table when RIP packets are received. The address in the access list is the address of the network that you want to be able to receive routing updates about.

```
access-list 476 permit 16
interface ethernet 1
xns input-network-filter 476
```

This set of configuration commands assures that network 16 is the only network whose information will be added to the routing table from Ethernet 1.

Output Filters: Controlling the List of Networks

To control the list of networks that are sent out in routing updates by your router, use this interface subcommand:

xns output-network-filter *access-list-number*

no xns output-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In the following example, access list 496 controls which networks are sent out in routing updates. The second line specifies interface serial 1, and the third line causes network 27 to be the only network advertised in routing update packets. Information about other networks will not be advertised in routing updates (for the specified interface only; other interfaces may advertise the full routing table).

```
access-list 496 permit 27
interface serial 1
xns output-network-filter 496
```

Router Filters

To control the list of routers from which data will be accepted, use the **xns router-filter** interface subcommand:

xns router-filter *access-list-number*

no xns router-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In this example, access list 466 defines the only router that data will be accepted from. In this case, the address parameter is the address of a router.

```
access-list 466 permit 26.0000.000c0.047d
interface serial 0
xns router-filter 466
```

Information from a disallowed router is ignored.

XNS Configuration Examples

This section includes examples of common configurations, designed to help you put all the specific command information into a complex, real-world configuration file. We start with basic configuration examples and move on to protocol forwarding, helper address and both simple and complex access lists. Some of the examples refer to 3Com or Ungermann-Bass XNS, rather than standard XNS; all examples are clearly marked.

Creating a Routing Process

The following example establishes XNS routing on a Cisco router (creating a routing process), then three interfaces are named and given their individual network numbers.

Example:

```
xns routing
!
interface ethernet 0
xns network 1
!
interface ethernet 1
xns network 44
!
interface serial 1
xns network 23
```

Setting Timers

This example creates a routing process by specifying a specific address. Next we specify the interfaces and give them network numbers. The update timers for the serial and the Ethernet interfaces have also been changed. The granularity for the Ethernet interface becomes 20 because that is the lowest value specified for that protocol.

Example:

```
xns routing 0000.0C53.4679
!
interface ethernet 0
xns network 1
xns update-time 20
!
interface serial 0
xns network 51
xns update-time 40
!
interface ethernet 1
xns network 27
xns update-time 25
```

Configuring for Multi-Protocol Routing

What do you do if you want to enable XNS and another protocol as well? This example illustrates one way to do this. Do a pencil copy of your proposed configuration commands and check them against the other protocol chapters in this manual before you proceed with configuring more than one protocol in a session.

Example:

```
xns routing
novell routing
interface ethernet 0
xns network 200
novell network 4e
interface ethernet 1
xns network 205
novell network 6bb
interface serial 0
xns network 301
novell network 4ad
```

Configuring for Ungermann-Bass Routing

In this example, basic Ungermann-Bass Net/One routing is enabled by first enabling XNS routing, then specifying Ungermann-Bass routing, then defining the interfaces and networks.

Example:

```
xns routing
xns ub-routing
interface ethernet 0
xns network 1234
interface ethernet 1
xns network 567
```

3Com Access List

This example permits and denies specific services between networks 1002 and 1006 in a 3Com network. Echo and error packets can go from 1002 to 1006, as well as all SPP and PEP (normal data traffic). However, all NETBios requests are denied. The final three lines are blanket permissions for RIP, SPP and PEP, because access lists will assume you wish to deny anything you do not specifically permit. These blanket permissions must come at the end of the list, as shown in the example configuration.

Example:

```
access-list 524 permit 2 1002 0x0000 1006 0x0000
!  permit Echo from 1002 to 1006
access-list 524 permit 3 1002 0x0000 1006 0x0000
!  permit Error from 1002 to 1006
access-list 524 deny 5 -1 0x0000 -1 0x046B
!  deny all NetBIOS
access-list 524 permit 4 1002 0x0000 1006 0x0000
!  permit PEP from 1002 to 1006
access-list 524 permit 5 1002 0x0000 1006 0x0000
!  permit SPP from 1002 to 1006
access-list 524 permit 1
!  permit all RIP
!
!These are needed if you want PEP and SPP to be permitted from
!networks other than 1002
access-list 524 permit 4
!  permit all PEP
access-list 524 permit 5
!  permit all SPP
```

There are additional access list examples in the Application Note titled *A Detailed Look at Access Lists in 3Com XNS.*

Monitoring an XNS Network

Use the EXEC commands described in this section to obtain displays of activity on your XNS network.

Displaying Cache Entries

Use the **show xns cache** command to displays a list of fast-switching cache entries. Enter this command at the EXEC prompt:

show xns cache

In the following sample output, the router responds to a request for cache information with a version number.

```
XNS routing cache version is 14
```

Displaying Interface Parameters

Use the **show xns interface** command to display interface-specific XNS parameters. Enter this command at the EXEC prompt:

show xns interface [*name*]

The optional argument *name* may be used to request a display a particular interface.

The following sample output shows a display of all interface statistics.

```
Ethernet 0 is up, line protocol is up
XNS address is 60.0000.0c00.1d23
xns encapsulation is ARPA
Helper address is 912.ffff.ffff.ffff
Outgoing address list is not set
INput filter list is not set
Output filter list is not set
Router filter list is not set
Update timer is not set
XNS fast-switching enabled

Ethernet 1 is administratively down, line protocol is down
XNS protocol processing disabled

Serial 1 is up, line protocol is up
XNS protocol processing disabled
```

In this display, the Ethernet 0 interface has a helper address set but all other parameters are set to the defaults. The Update Timer refers to itself as not set when it is set to default value. The Ethernet 1 interface is down and XNS processing is disabled. The serial 1 interface is up but it is not processing XNS packets.

For the Ethernet 1 and serial 1 interfaces, the **no xns network** command is in force. You can enable XNS processing by using the **xns network** configuration command.

Displaying the Routing Table

Use the EXEC command **show xns route** to display the entire XNS routing table. Enter this command at the EXEC prompt:

show xns route *network-number*

The optional network number argument specifies a particular network.

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 1 learned routes

Maximum allowed path(s) are/is 1
C Net 14 is directly connected, 0 uses, Ethernet0
C Net 15 is directly connected, 0 uses, Ethernet1
R Net 16 [1/0] via 14.0000.0c00.3e3b,  10 sec, 0 uses, Ethernet0
```

In this display, RIP-derived are indicated by the letter R. Networks that are reachable are listed in numerical order within each category—RIP or connected, in this case.

The routing table also tells you the address of the router that constitutes the first hop in the route (always the same router in this case), the round-trip delay encountered on the route and the interface that the route is available through.

Displaying Traffic Statistics

Use the EXEC command **show xns traffic** to display packet statistics, including packets sent, received, and forwarded. Enter this command at the EXEC prompt:

show xns traffic

Sample output follows. Table 19-1 describes the fields displayed.

```
Rec: 3968 total, 0 format errors, 0 checksum errors, 0 bad hop count, 3968 local des-
tination,  0 multicast
Bcast: 2912 received, 925 sent
Sent:  5923 generated, 500 forwarded, 0 encapsulation failed, 0 not routable
Errors:  10 received, 20 sent
Echo:  Recd:  100 requests, 89 replies   Sent:  20 requests, 20 replies
Unknown:  5 packets
```

Table 19-1 XNS Traffic Statistics Field Descriptions

Field	Description
Rec:	The total number of packets received on the interface.
format errors	Number of packets received with format errors in the header; they were discarded.
checksum errors	Number of packets received and discarded because of checksum error.
bad hop count	Number of packets discarded because the hop count field was equal to or greater than 16.
local destination	Number of packets received on the interface that had a local MAC address.
multicast	Number of packets received with a multicast list address.
B cast:	Number of broadcast packets received and sent (both directed and flooded).
Sent:	
generated	Total number of packets sent out on the interface.
forwarded	Number of packets sent to another router for forwarding to a remote network.
encapsulation failed	Number of packets discarded because encapsulation was needed (token ring, for example) and couldn't be accomplished.
not routable	Number of packets bridged or discarded because they did not conform to any protocol this router could route.
Errors:	Number of Error packets sent and received.
Echo:	Number of Echo packets received and sent, specifying how many replies were received.
Unknown:	Number of packets that failed for a reason not listed above; the packet conformed to nothing the router understood.

Debugging an XNS Network

Use the EXEC commands described in this section to troubleshoot and monitor your XNS network. For each **debug** command listed, there is a corresponding **undebug** command that turns off the message logging.

debug xns-packet

The command **debug xns-packet** enables logging of XNS packet traffic, including the addresses for source, destination, and next hop router of each packet.

debug xns-routing

The command **debug xns-routing** displays XNS routing transactions.

XNS Global Configuration Command Summary

Following is an alphabetically arranged list of the XNS global configuration commands.

access-list *number* {**deny** | **permit**} *XNS-source-network.*[*source-address*[*source-mask*]]*XNS-destination-network.*[*destination-address*[*destination-mask*]]

Configures an XNS access list. The argument *number* is an access list number in the range 400 and 499. The keyword **permit** or **deny** specifies the filtering action. The arguments *XNS-source-network* and *XNS-destination-network* are the only required addresses. The source and destination masks are optional.

access-list *number* {**deny** | **permit**} *XNS-protocol source-network.*[*source-address* [*source-mask*]] *source-socket destination-network.*[*destination-address* [*destination-mask*]] *destination-socket*

Configures an extended XNS access list. The argument number is an access list number in the range 500 and 599. The argument *XNS-protocol* is the XNS protocol number. See previous description for remaining argument and keyword descriptions.

no access-list *number*

Removes the specified access list.

[no] xns forward-protocol *type*

Determines which protocol types will be forwarded when a broadcast is received on an interface that has an XNS helper address set. The argument *type* is a decimal number corresponding to an appropriate XNS protocol.

[no] xns maximum-paths *paths*

Sets the maximum number of equal-cost paths the router will use. The default value of *paths* is one.

[no] xns route *network host-address*

Adds a static route to the XNS routing table. The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and the hexadecimal host number.

[no] xns routing [*address*]

Enables XNS routing. The optional argument *address* is the XNS address the router is to use. If the argument *address* is omitted, the first Token Ring, FDDI, or Ethernet interface hardware address found is used. The **no** form of the command disables all XNS packet processing.

[no] xns ub-routing

Enables or disables Ungermann-Bass Net/One routing.

XNS Interface Subcommand Summary

Following is an alphabetically arranged list of the XNS interface subcommands. These commands follow an **interface** command.

[no] xns access-group *number*

Assigns an access list number to an interface, and causes all XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If the packet fails the access list, it is not transmitted, but is discarded instead. The argument *number* specifies the access list number.

xns encapsulation *keyword*

Selects encapsulation for a Token Ring interface. Choices for *keyword* are: **snap, ub,** and **3com**.

[no] xns helper-address *host-address*

Sets a helper address to forward broadcasts. The argument *host-address* is a dotted combination of the network and host addresses.

[no] xns input-network-filter *access-list-number*

Controls which networks are added to *your router's* routing table. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

[no] xns output-network-filter *access-list-number*

Controls the list of networks that are sent out *in routing updates by your router.* The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

[no] xns network *number*

Assigns a decimal XNS network number to an interface.

[no] xns route-cache

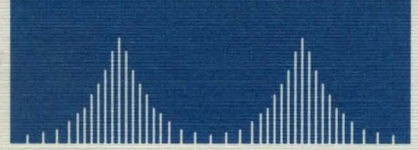
Enables fast-switching. The **no** keyword disables fast switching.

[no] xns router-filter *access-list-number*

Controls the list of routers from which data will be accepted. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

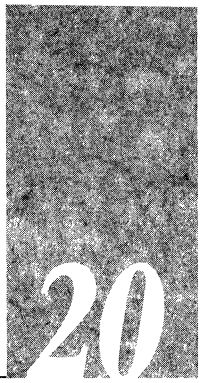
[no] xns update-time *seconds*

Sets the XNS routing update timers to the value assigned to the *seconds* argument.

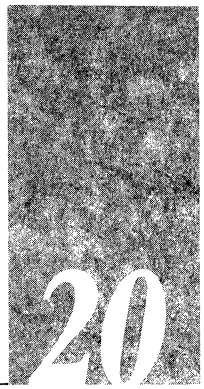


Part 5
Bridging and Connectivity

Chapter 20
Configuring Transparent Bridging



Chapter 20
Configuring Transparent Bridging



This chapter describes the transparent bridging support available in the Cisco Systems network server product line. Topics described in this chapter include:

- An overview of bridging and Cisco's implementation of transparent bridging.
- How to define a spanning tree.
- How to adjust the spanning-tree parameters.
- How to filter packets based on type; supported types include: Ethernet address, protocol type, vendor code, and LAT multicast service announcements.

Bridging Overview

The basic function of a bridge is to accept frames of data passing through a network, and then make a decision about whether to forward each frame based on information contained in the frame.

Bridges operate at the physical (Level 1) and data link (Level 2) layers of the Open Systems Interconnection (OSI) model. As defined by the IEEE 802.1 standard, the task of the physical layer is to provide the physical connection to the transmission medium. The task of the data link layer is to provide reliable transmission. To accomplish this effectively, the standards committee sublayered the data link layer into two distinct tasks, Logical Link Control (LLC) and Medium Access Control (MAC). Bridges, and specifically transparent bridges, function at the physical and MAC layers. However, the administrative filtering functions described later in this chapter make use of the LLC. Figure 20-1 illustrates these concepts.

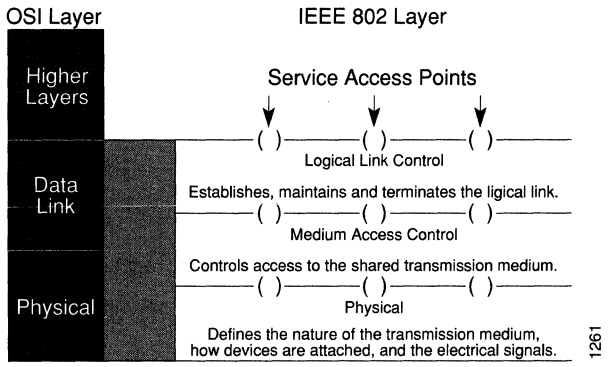


Figure 20-1 OSI Model and IEEE 802 Layers

The LLC, as its name implies, controls the logical link, that is, it serves to open, maintain, and terminate a link. It depends upon the physical layers to perform its tasks.

The MAC controls access to the media used to transmit the frames. It acts independently of the physical layer and provides a service interface to the upper layers of the network model. IEEE 802.3 and Ethernet use the same MAC layer, allowing both types of frames to coexist on the same cable. In this way, the same bridge can forward both Ethernet and IEEE 802.3 frames.

The standard identifies two basic elements to facilitate these tasks: a 48 bit address which is always at the same offset in the frame, and Service Access Points (SAPs), which can be thought of as ports through which a higher layer application may communicate with its lower layer.

Figure 20-2 illustrates the frame formats found on Ethernets.

The IEEE 802 standard allows any IEEE 802-compliant station to communicate with remote, bridged stations as if they were local. Although the networks connected by a bridge remain physically separate, they appear as one network to the rest of the devices on the internet.

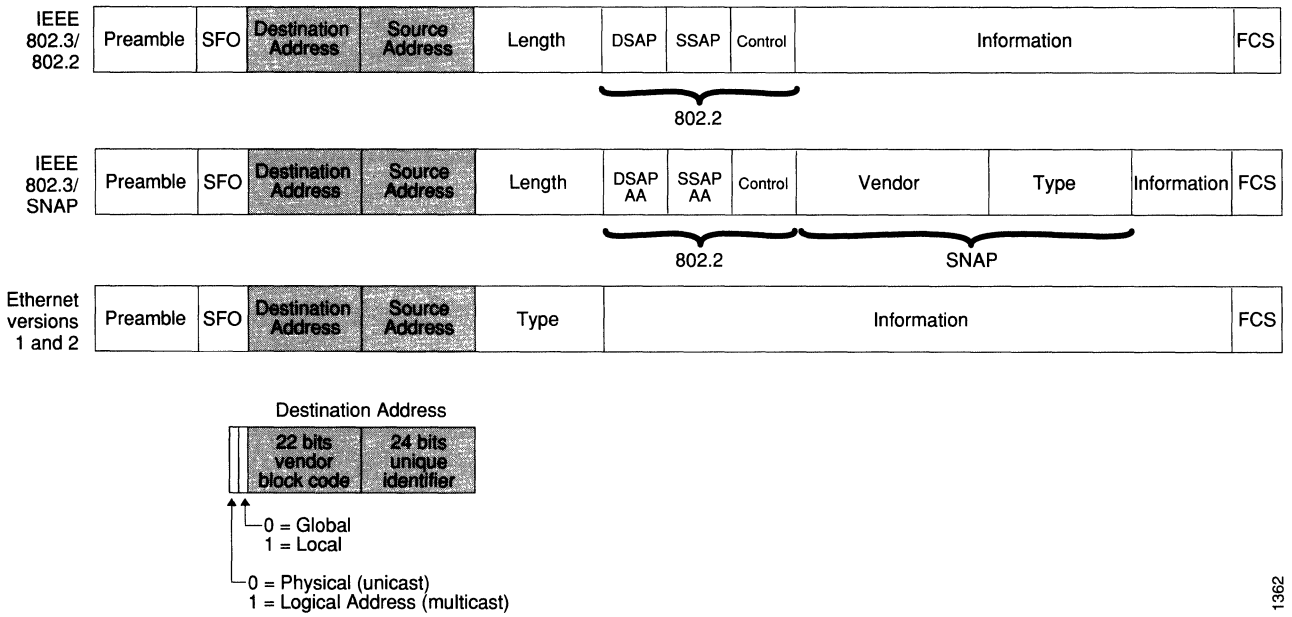


Figure 20-2 IEEE 802 Frame Formats

Bridges forward frames between networks using the same medium (two IEEE 802.3 networks, for example). Bridges can pass data using any protocol compatible with the media. Bridges receive all frames from a local area network (LAN) and examine each one to determine its destination. If the frame is destined for a station on the physical cable from which it was received, the bridge does not forward the frame. If the destination is unknown, the frame is forwarded to all ports except the one on which it was received.

Because bridges examine each frame, and because each frame contains a source address, bridges can easily learn the locations of stations on the network. However, if an address is unknown, their flooding behavior could also result in endless loops on the network, since all bridges that do not know the address will flood the frame looking for the address. To circumvent this, a tree-type algorithm has been developed that prevents loops and forces the frames into logical routes. Called a spanning-tree algorithm, it allows only one route between any two physical cables or networks.

Spanning tree bridges are based upon the concept of a *root* bridge which exchanges topology information with designated bridges to maintain the configuration. The root and designated bridges notify all other bridges in the network when topology changes are required, thereby preventing loops and providing a measure of defense against link failure.

Bridges may be further classified as *local* or *remote*. A local bridge typically connects two nearby LANs, for example, Ethernet-to-Ethernet or Token Ring-to-Token Ring. A remote bridge, on the other hand, links geographically distant LANs or LANs separated by some other media. Two Ethernets linked by a serial line, or two Token Rings linked by a TCP connection are examples of remote bridging.

Cisco's Implementation of Transparent Bridging

Cisco's transparent bridging software implementation provides these features:

- Complies with the IEEE 802 standard.
- Provides two spanning-tree algorithms—an older version that is compatible with DEC and other LAN bridges for backwards compatibility, and a recent IEEE Standard implementation that provides for multiple domains for spanning trees. The software also provides for adjustments to the spanning-tree parameters.
- Allows configuration of filters to effectively block the passing of frames based on Ethernet address, protocol type, or the vendor code. Additionally, the bridging software can be configured to selectively include or exclude LAT multicast service announcements.
- Provides load balancing and redundancy by assigning a set of serial lines between two bridges to a *circuit group.*
- Provides ability to bridge over X.25 and frame relay networks.
- Provides for the compression of LAT frames to reduce LAT traffic through the network.

Additionally, the Cisco router/bridge combines the advantages of a spanning-tree bridge and a full multiprotocol router. This combination provides the speed and protocol transparency of an adaptive spanning-tree bridge, along with the functionality, reliability, and security of routers.

Network interfaces on the Cisco Multiport Communications Interface (MCI) and on the Multiport Ethernet Controller (MEC) cards on the cBus high-speed backplane, may be configured to serve as both multiprotocol routers and MAC-level bridges, bridging any traffic that cannot otherwise be routed. For example, a router/bridge routing the Internet Protocol can also bridge DEC's Local Area Transport (LAT) protocol, or NETBIOS traffic.

Remote bridging over synchronous serial lines is also supported. Ethernet frames are encapsulated in HDLC frames for transmission between Cisco bridges/routers. As with frames received on Ethernet interfaces, the learning process and any filtering is applied to HDLC-encapsulated Ethernet frames.

The transit bridging of Ethernet frames across an FDDI ring is also supported. The term *transit* refers to the fact that the source or destination of the frame cannot be on the FDDI ring itself. This allows the FDDI ring to act as a highly efficient backbone for the interconnection of many bridged Ethernets. The configuration of FDDI transit bridging is identical to the configuration of Ethernet transparent bridging.

Configuring Transparent Bridging

Follow these steps to configure transparent bridging on a Cisco router/bridge:

Step 1: Define the spanning-tree protocol and assign a bridge group number using the **bridge** *group* **protocol** command. The Cisco bridging software supports the IEEE 802.1 spanning-tree protocol, and the earlier DEC protocol upon which the IEEE standard is based. Additionally, multiple domains are supported for the IEEE spanning tree.

Step 2: Enable each interface and configure any routed protocols. If you wish to bridge IP, you must disable IP routing.

Step 3: Assign the network interfaces to the spanning-tree group.

Step 4: Adjust the spanning-tree parameters, as necessary.

The bridging software also supports filtering of frames. Filtering can be done by Ethernet address, protocol type, vendor code, and based upon LAT group codes. Additionally, EXEC-level commands for monitoring and debugging the bridge are available. These tasks and commands are described in the following sections.

Defining the Spanning Tree Protocol

Two spanning-tree protocols are supported: the IEEE 802.1 standard, and the earlier DEC spanning-tree protocol upon which the IEEE standard is based.

To define a spanning-tree protocol, use the **bridge** *group* **protocol** global configuration command. The command has this syntax:

bridge *group* **protocol** {**ieee** | **dec**}

no bridge *group* **protocol** {**ieee** | **dec**}

The argument *group* is a number between one and nine that you choose to refer to a particular set of bridged interfaces. Frames are bridged only among interfaces in the same group. You will use the group number you assign in subsequent bridge configuration commands.

The keyword **protocol** specifies the protocol to use and is either **ieee** for the Ethernet spanning-tree protocol, or **dec** for DEC spanning-tree protocol.

The **no bridge** command with the appropriate keywords and arguments deletes the bridge group.

Note: The IEEE has not yet fully ratified the 802.1 bridging standard. It is recommended that the **dec** keyword be used.

Example:

This command defines bridge 1 as using the DECnet spanning-tree protocol.

```
!
bridge 1 protocol dec
!
```

Establishing Multiple Spanning Tree Domains

The Cisco IEEE 802 bridging software supports multiple domain spanning-trees. You can place any number of router/bridges within the domain. The devices in the domain, and only those devices, will then share spanning-tree information.

This feature is used when multiple routers share the same cable, and you wish to use only certain, discrete subsets of those routers to share spanning tree information with each other. This function is most useful when running other router applications, such as IP UDP flooding, that use the IEEE spanning tree. It can also be used to reduce the number of global reconfigurations in large bridged networks.

Note: Use with care, since bridges in different domains do not share spanning tree information. This makes it possible for bridge loops to be created if the domains are not carefully planned.

Establish a domain by assigning it a value between one to ten using this variation of the **bridge** global configuration command:

bridge *group* **domain** *domain-number*

The argument *group* is the bridge group number, and must be the same as that specified by the **protocol ieee** keywords in the previously described **bridge** *group* command.

The keyword **domain** specifies a domain; the argument *domain-number* is a domain number you choose. The default domain number is zero (0), and this is the required domain number when communicating to IEEE bridges that do not support this domain extension.

Note: This command works only when the bridge group is running the IEEE spanning-tree protocol.

Example:

This command places bridge group *1* in bridging domain *3*. Only other Cisco routers that are in domain *3* will accept spanning-tree information from this router.

```
!
bridge 1 domain 3
!
```

Assigning the Interface to a Spanning Tree Group

Assign each network interface to a spanning-tree group using the **bridge-group** *group* interface subcommand, which has this syntax:

bridge-group *group*

The simplest form of the **bridge-group** command takes just a spanning-tree group number as an argument. More complex forms of the **bridge-group** command are described later.

Note: Restrictions apply as to which interfaces can be configured for bridging. Only the Cisco MCI, MEC HSCI, and FDDI interface controller cards are capable of supporting simultaneous bridging and routing. Bridging can be configured between interfaces on different cards, although the performance is lower compared with interfaces on the same card. Also note that serial interfaces must be running with HDLC, X.25, or frame relay encapsulation.

All protocols except IP are bridged by a router/bridge unless their routing is explicitly enabled. Refer to Chapter 13, "Routing IP" in for the procedures to enable routing of individual protocols. IP is normally routed by the router/bridge.

Also note that bridging and routing are done on a per-system basis. If a protocol is being routed, it must be routed on all interfaces that are handling that protocol. This is similar for bridging. You cannot route IP on one interface and bridge it on another interface.

Example:

The following is an example of a basic bridging configuration. The system has two Ethernets and one serial line on the same card. The Internet Protocol (IP) is being routed and everything else is being bridged. The DEC-compatible bridging algorithm with default parameters is being used.

```
!
interface ethernet 0
ip address 192.31.7.26  255.255.255.240
bridge-group  1
!
interface serial 0
ip address 192.31.7.34  255.255.255.240
bridge-group  1
!
interface ethernet 1
ip address 192.31.7.65  255.255.255.240
bridge-group  1
!
bridge 1 protocol dec
```

Bridging and Routing IP

To bridge IP you must disable IP routing by giving the following global configuration command:

no ip routing

Assign an IP address (the *same* IP address) to all network interfaces to manage the system with Telnet, TFTP, SNMP, ICMP (ping) and so forth. Once bridging is enabled, all IP and ARP frames not intended for the Cisco router/bridge are handled according to standard bridging and spanning tree rules. IP routing processes (such as IGRP or RIP) must not be running.

Adjusting Spanning-Tree Parameters

Under some circumstances, adjustments to certain spanning-tree parameters may be needed. Parameters affecting the entire spanning tree are configured with variations of the **bridge** *group* global configuration command. Interface-specific parameters are configured with variations of the **bridge-group** *group* interface subcommand. Global adjustments to the entire spanning tree are described first.

Note: These adjustments must be done carefully, and only by network administrators with a good understanding of how bridges and the spanning-tree protocol work. Badly planned adjustments to these parameters can have a negative impact on performance. A good source on bridging is the IEEE 802.1d specification; see the recommended reading list at the end of this publication for other references.

Electing the Root Bridge

The priority of an individual bridge, or the likelihood that it will be selected as the root bridge, can be configured with the **bridge** *group* **priority** global configuration command. The command has this syntax:

bridge *group* **priority** *number*

The argument *number* can range from 1 to 65,000. The lower the value of *number*, the more likely the bridge will be chosen as root. The default priority is 128.

Example:

This command establishes this bridge as a likely candidate to be the root bridge.

```
!
bridge 1 priority 100
!
```

Adjusting the Interval Between HELLO BPDUs

The interval between HELLO Bridge Protocol Data Units (BPDUs) is specified with the **bridge** *group* **hello-time** global configuration command. The command has this syntax:

bridge *group* **hello-time** *seconds*

The argument *seconds* can be any value between one and ten seconds. The default value is one second.

Example:

This command sets the interval to five seconds.

```
!
bridge 1 hello-time 5
!
```

Note: Each bridge in a spanning tree adopts the **hello-time**, **forward-time**, and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

Defining the Forward Delay Interval

The forward delay interval is the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. Use the **bridge** *group* **forward-time** global configuration command to specify this interval. The command has this syntax:

bridge *group* **forward-time** *seconds*

The argument *seconds* can be any value between 10 and 200 seconds. The default value is 30 seconds.

Note: Each bridge in a spanning tree adopts the **hello-time**, **forward-time**, and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

Example:

This command sets the forward delay interval to 60 seconds.

```
!
bridge 1 forward-time 60
!
```

Defining the Maximum Idle Interval

If a bridge does not hear BPDUs from the root bridge within a specified interval, it assumes that the network has changed and recomputes the spanning-tree topology. This interval is 15 seconds by default. It can be changed with the **bridge** *group* **max-age** global configuration command. The command has this syntax:

bridge *group* **max-age** *seconds*

The argument *seconds* is the interval the bridge will wait to hear BPDUs from the root bridge.

Note: Each bridge in a spanning tree adopts the **hello-time**, **forward-time**, and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

Example:

This command increases the maximum idle interval to 20 seconds.

```
!
bridge 1 max-age 20
!
```

Assigning Path Costs

Each interface has associated with it a path cost. By convention, the path cost is 1000/data rate of the attached LAN, in mbps. Table 20-1 lists some common path cost values.

Table 20-1 Media and Path Cost Values

Media	Path Cost
Ethernet	100
FDDI	10

Use the interface subcommand **bridge-group** *group* **path-cost** to set a different path cost. The command syntax is as follows:

bridge-group *group* **path-cost** *cost*

The path cost can range from 0 to 65,535, with higher values indicating higher costs.

The default path cost is computed from the interface's bandwidth setting.

Example:

This command changes the default path cost for interface Ethernet 0.

```
!
interface ethernet 0
bridge-group 1 path cost 250
!
```

Setting an Interface Priority

A priority can also be associated with an interface. In the event that two bridges tie for position as the root bridge, this priority is used in tie-breaking when computing a network topology. Use the **bridge-group** *group* **priority** interface subcommand to do this. The command has this syntax:

bridge-group *group* **priority** *number*

The argument *number* can range from 0 to 255. The default value is zero, and the lower the number, the more likely it is that the bridge on this interface will be chosen as the root.

Example:

This command increases the likelihood that the root bridge will be the one on Ethernet 0, in spanning-tree group 1.

```
!
interface ethernet 0
bridge-group 1 priority 0
!
```

Establishing Administrative Filtering

It is the job of a bridge to examine frames and transmit them through the internet according to their destination address; that is, a bridge will not forward a frame back to its originating network segment. The Cisco bridge software allows specific administrative filters to be configured that block frames based upon information other than paths to their destinations. This administrative filtering can be done by:

- MAC-layer address
- Protocol type—Ethernet or IEEE 802.3
- Vendor code
- LAT multicast service announcement

When setting up administrative filtering, remember that there is virtually no performance penalty in filtering by Ethernet address or vendor code, but there can be a significant performance penalty when filtering by protocol type.

Administrative Filtering by MAC-layer Address

Blocking transmission of frames based on MAC-layer address is configured by variations of the **bridge** *group* global configuration command, as described in the following sections.

Filtering on Ethernet Address

To filter frames with a particular MAC-layer station source or destination address, use the global configuration command **bridge** *group* **address**. The full syntax of this command is as follows:

bridge *group* **address** *ethernet-address* [**forward** | **discard**] [*interface*]

no bridge *group* **address** *ethernet-address*

The argument *group* is the group number you assigned to the spanning tree.

The argument *ethernet-address* is a 48-bit dotted-triple hardware address such as that displayed by the EXEC **show arp** command. An example is:

```
0800.cb00.45e9
```

The argument *ethernet-address* is either a station address, the broadcast address, or a multicast destination address.

If the optional **forward** keyword is specified, a frame sent from or destined to the specified address is forwarded, as appropriate.

If the optional **discard** keyword is specified, a frame sent from or destined to the specified address is discarded without further processing.

The argument *interface* is an optional interface specification, such as *Ethernet 0*, and is added after the **discard** or **forward** keyword to indicate the interface on which that address can be reached.

Any number of addresses can be configured into the system without a performance penalty.

Use the **no bridge** *group* **address** command followed by the Ethernet address to disable the forwarding ability.

Examples:

This command enables filtering of frames with Ethernet address *0800.cb00.45e9*.

```
!
bridge 1 address 0800.cb00.45e9 forward ethernet 1
!
```

The frame is forwarded through interface Ethernet 1.

This command disables the ability to forward frames with Ethernet address *0800.cb00.45e9*.

```
!
no bridge 1 address 0800.cb00.45e9
!
```

Preventing the Forwarding of Dynamically Determined Stations

Normally the system forwards any frames for stations that it has learned about dynamically. This default can be changed with the **no bridge** *group* **acquire** global configuration command. The full syntax of this command follows:

no bridge *group* **acquire**

bridge *group* **acquire**

The bridge filters out all frames except those whose sourced-by or destined-to addresses have been statically configured into the forwarding cache.

The **bridge** *group* **acquire** global configuration command restores the default behavior.

Example:

This command prevents the fowarding of dynamically determined source and destination addresses.

```
!
no bridge 1 acquire
!
```

Fowarding the Multicast Addresses

The bridging support may be configured to allow the forwarding, but not the learning, of multicast source addresses. Use this variation of the **bridge** *group* global configuration command to configure this function (the negative form is also listed):

bridge *group* **multicast-source**

no bridge *group* **multicast-source**

List the bridge group using the *group* argument.

Administrative Filtering by Protocol Type

Filtering by protocol type is done using the access list mechanism and by specifying a protocol type code. The bridge **access-list** command specifies an element in an access list. The order in which **access-list** commands are entered affects the order in which the access conditions are checked. Each condition is tested in succession. A matching condition is then used to execute a permit or deny decision. If no conditions match, a deny decision is reached.

Establishing Protocol Type Access Lists

Type code access lists are built with this bridge **access-list** global configuration command:

access-list *list* {**permit** | **deny**} *type-code wild-mask*

The argument *list* is a user-selectable number between 200 and 299 that identifies the list.

The keyword **permit** permits the frame; the keyword **deny** denies the frame.

The argument *type-code* is a 16-bit hexadecimal number written with a leading "0x", for example, 0x6000. You may specify either an Ethernet type code for Ethernet-encapsulated packets, or a DSAP/SSAP pair for 802.3-encapsulated packets. Ethernet type codes are listed in Appendix C.

The argument *wild-mask* is another 16-bit hexadecimal number whose ones bits correspond to bits in the *type-code* argument that should be ignored when making a comparison. (A mask for a DSAP/SSAP pair should always be at least 0x0101. This is because these two bits are used for purposes other than identifying the SAP codes.)

Examples:

The following access list permits only LAT frames (type 0x6004) and filters out all other frame types.

```
!
access-list  201  permit  0x6004  0x0000
access-list  201  deny    0x0000  0xFFFF
!
```

The following access list filters out only type codes assigned to DEC (0x6000 through 0x600F) and lets all other types pass.

```
!
access-list  202  deny     0x6000  0x000F
access-list  202  permit   0x0000  0xFFFF
!
```

It is always a good idea to use the last item of an access list to specify a default action, for example, permit everything else or deny everything else. If nothing else in the access matches, the default action is normally to deny access, that is, filter out all other type codes.

Note: Type code access lists can have an impact on system performance, therefore, keep the lists as short as possible and use wild card bit masks whenever possible.

Filtering Ethernet- and SNAP-Encapsulated Packets on Input

You may filter Ethernet- and SNAP-encapsulated packets on input. The access list specifying the type codes to be filtered is given in this variation of the **bridge-group** interface subcommand:

bridge-group *group* **input-type-list** *list*

This access list is then applied to all Ethernet and SNAP frames received on that interface prior to the bridge learning process. SNAP frames must also pass any applicable IEEE802.3 DSAP/SSAP access lists.

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

Example:

The following example illustrates how to configure a system with two Ethernet interfaces and one serial interface. The system is routing both IP and DECnet. Each interface has an access list that allows only the LAT protocol to be bridged. The bridging software has also been instructed to discard frames sent to or from the address *AB00.0C00.AE35*.

```
!
decnet address 34.88
!
interface ethernet 0
ip address 192.31.7.26  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface serial 0
ip address 192.31.7.34  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface ethernet 1
ip address 192.31.7.65  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
bridge 1 protocol dec
bridge 1 address AB00.0C00.AE35 discard
!
access-list 201 permit 0x6004  0x0000
access-list 201 deny   0x0000  0xFFFF
!
```

Filtering Ethernet- and SNAP-Encapsulated Packets on Output

You may filter Ethernet- and SNAP-encapsulated packets on output. The access specifying the type codes to be filtered is given by this variation of the **bridge-group** interface subcommand:

bridge-group *group* **output-type-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

This access list is then applied just before sending out a frame to an interface.

Example:

This command specifies access list 202 on interface Ethernet 0.

```
!
interface ethernet 0
bridge-group 2 output-type-list 202
!
```

Filtering IEEE 802.3-Encapsulated Packets on Input

You may filter IEEE 802.3-encapsulated packets on input. The access list specifying the type codes to be filtered is given by this variation of the **bridge-group** interface subcommand:

bridge-group *group* **input-lsap-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

This access list is applied to all IEEE 802.3 frames received on that interface prior to the bridge-learning process. SNAP frames must also pass any applicable Ethernet type-code access list.

Example:

This command specifies access list 203 on interface Ethernet 1:

```
!
interface ethernet 1
bridge-group 3 input-lsap-list 203
!
```

Filtering IEEE 802.3-Encapsulated Packets on Output

You may filter IEEE 802.3-encapsulated packets on output. The access list specifying the type codes to be filtered is given by this variation of the **bridge-group** interface subcommand:

bridge-group *group* **output-lsap-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command. SNAP frames must also pass any applicable Ethernet type-code access list. This access list is applied just before sending out a frame to an interface.

Note: For performance reasons, it is not a good idea to have both input and output type code filtering on the same interface.

Access lists for Ethernet- and IEEE 802.3-encapsulated packets affect only bridging functions. It is not possible to use such access lists to block frames with protocols that are being routed.

Example:

This command specifies access list 204 on interface Ethernet 0.

```
!
interface ethernet 0
bridge-group 4 output-lsap-list 204
!
```

Administrative Filtering by Vendor Code

The bridging software supports administrative filtering of Ethernet addresses. These lists support filtering groups of Ethernet addresses, including those with particular vendor codes. The lists are defined with bridge **access-list** global configuration command and **bridge-group** interface subcommand. There is no noticeable performance loss in using these access lists. The lists can be of indefinite length. The following sections describe how to set these lists up.

Establishing Vendor Code Access Lists

Use the bridge **access-list** global configuration command to establish Ethernet and IEEE 802.3 address access lists. The command has the following form:

access-list *list* {**permit** | **deny**} *address mask*

no access-list *list* {**permit** | **deny**} *address mask*

The argument *list* is an integer from 700 to 799 that you select for the list.

The argument *address* and *mask* are 48-bit Ethernet addresses written in dotted triplet form. The ones bits in the *mask* argument are the bits to be ignored in *address*.

See the section "Filtering Source Addresses" for an example of use of this command.

Filtering Source Addresses

Use the **bridge-group** *group* **input-address-list** interface subcommand to assign an access list to a particular interface for filtering on the Ethernet or IEEE 802.3 source addresses of packets received on that interface. The command has this syntax:

bridge-group *group* **input-address-list** *list*

no bridge-group *group* **input-address-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

Example:

This configuration example assumes you want to disallow the bridging of Ethernet packets of all SUN workstations on Ethernet 1. Software assumes that all such hosts have Ethernet addresses with the vendor code 0800.2000.0000. The first line of the access list denies access to all SUN workstations while the second line permits everything else. You then assign the access list to the input side of Ethernet 1.

```
!
access-list 700 deny 0800.2000.0000  0000.00FF.FFFF
access-list 700 permit 0000.0000.0000  FFFF.FFFF.FFFF
interface ethernet 1
bridge-group 1 input-address-list 700
!
```

Filtering Destination Addresses

Use the **bridge-group** *group* **output-address-list** interface subcommand to assign an access list to a particular interface for filtering the Ethernet or IEEE 802.3 destination addresses of packets that would ordinarily be forwarded out that interface. The command has this syntax:

bridge-group *group* **output-address-list** *list*

no bridge-group *group* **output-address-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

Example:

This command assigns access list 703 to interface Ethernet 3.

```
!
interface ethernet 3
bridge-group 5 output-address-list 703
!
```

Administrative Filtering of LAT Service Announcements

The Cisco bridging software provides filtering of LAT frames. LAT bridge filtering allows the selective inclusion or exclusion of LAT multicast service announcements, on a per-interface basis.

In the DEC LAT protocol, a *group code* is defined as a decimal number in the range 0 to 255. Some of the Cisco LAT configuration commands take a list of group codes; this is referred to as a *group code list*. The rules for entering numbers in a group code list follow:

- Entries can be individual group code numbers separated with a space. (The DEC LAT implementation specifies that a list of numbers be separated by commas; however, Cisco's implementation expects the list to be separated by a space.)
- Entries may also specify a range of numbers. This is done by separating an ascending order range of group numbers with a hyphen.
- Any number of group codes or group code ranges can be listed in one command; just separate each with a space.

In LAT, each node transmits a periodic service advertisement message, which announces its existence and availability for connections. Within the message is a group code list; this is a mask of up to 256 bits. Each bit represents a group number.

In the traditional use of LAT group codes, a terminal server will only connect to a host system when there is an overlap between the group code list of the user on the terminal server, and the group code list in the service advertisement message.

While some believe this to be a security feature, it is in fact present to allow you to partition your physical network. This is because most DEC terminal servers do not have enough memory to cope with a large number of services. Group codes do not actually provide any real security; they are trivial to defeat.

In an environment with many bridges and many LAT hosts, the number of multicast messages with which each system has to deal becomes unreasonable. The 256 group codes may not be enough to allocate local assignment policies, such as giving each DECserver 200 device its own group code, in large bridged networks.

LAT group code filtering allows you to have very fine control over what multicast messages actually get bridged. Through a combination of input and output permit and deny lists, many different LAT control policies can be implemented. Use the EXEC command **show span** to report the group code filtering in effect.

Specifying LAT Group Code Service Filtering

Use the **bridge** *group* **lat-service-filtering** global configuration command to specify LAT group code filtering. The command has this syntax:

bridge *group* **lat-service-filtering**

The command informs the system that LAT service advertisements require special processing. Use the *group* argument to specify the bridge group in which this special processing is to take place.

Example:

This command specifies that LAT service announcements travelling across bridge group 1 require some special processing.

```
!
bridge 1 lat-service-filtering
!
```

Specifying Deny Conditions for LAT Group Codes on Input

Use the **bridge-group** *number* **input-lat-service-deny** interface subcommand to specify the group codes by which to deny access upon input The command has this syntax:

bridge-group *number* **input-lat-service-deny** *grouplist*

This command causes the system to not bridge any LAT service advertisement which has any of the specified groups set. The argument *number* is the previously chosen spanning-tree group number. Enter the list of LAT service groups with the *grouplist* argument.

Example:

This command causes any advertisements with groups 6, 8, and 14 through 20 to be dropped.

```
!
interface ethernet 0
bridge-group 1 input-lat-service-deny 6 8 14-20
!
```

Specifying Permit Conditions for LAT Group Codes on Input

Use the **bridge-group** *number* **input-lat-service-permit** interface subcommand to specify the group codes by which to permit access upon input The command has this syntax:

bridge-group *number* **input-lat-service-permit** *grouplist*

This command causes the system to bridge only those service advertisements which match at least one group in the group list specified by the *grouplist* argument.

Note: If a message specifies group codes in both the deny and permit list, the message is not bridged.

Example:

This command bridges any advertisements from groups 1, 5, and 12 through 14.

```
!
interface ethernet 1
bridge-group 1 input-lat-service-permit 1 5 12-14
!
```

Specifying Deny Conditions for LAT Group Codes on Output

Use the **bridge-group** *number* **output-lat-service-deny** interface subcommand to specify the group codes by which to deny access upon output The command has this syntax:

bridge-group *number* **output-lat-service-deny** *grouplist*

This command causes the system to not bridge onto this output interface any service advertisements which contain groups matching any of these in the group list. The LAT service advertisements are specified with the argument *grouplist.*

The argument *number* is the previously chosen spanning-tree group number.

Example:

This command prevents bridging of LAT service announcements from groups 12 through 20.

```
!
interface ethernet 0
bridge-group 1 output-lat-service-deny 12-20
!
```

Specifying Permit Conditions for LAT Group Codes on Output

Use the **bridge-group** *number* **output-lat-service-permit** interface subcommand to specify the group codes by which to permit access upon output. The command has this syntax:

bridge-group *number* **output-lat-service-permit** *grouplist*

This command causes the system to bridge onto this output interface only those service advertisements that match at least one group in the specified group code list. The service advertisements are specified with the argument *grouplist.*

The argument *number* is the previously chosen spanning-tree group number.

Note: If a message matches both a deny and a permit condition, the message will not be bridged.

Example:

This command allows only LAT service announcements from groups 5, 12, and 20 on this bridge.

```
!
interface ethernet 0
bridge-group 1 output-lat-service-permit 5 12 20
!
```

Special Bridging Configurations

This section describes some special bridging configurations, including how to configure load balancing over serial lines, and how to compress LAT traffic.

Establishing Load Balancing

In the normal operation of the spanning-tree algorithm, parallel network segments cannot all be carrying traffic at the same time. This is necessary to prevent the looping of frames. In the case of serial lines, however, there is often a desire to increase the available bandwidth by using multiple, parallel serial lines.

To modify the spanning-tree algorithm's handling of serial lines, a set of serial lines between two bridges may be grouped in an association called a *circuit group*. If the spanning-tree algorithm allows any serial interface in the circuit group to forward packets, then all interfaces can be used for forwarding traffic. Ordering problems are avoided by assigning each destination address to a particular serial interface. Reassignment is done dynamically if interfaces go down or come back up.

To establish load balancing, use the **bridge-group** *group* **circuit** interface subcommand. The command has this syntax:

bridge-group *group* **circuit** *number*

The command marks a serial interface as belonging to circuit group. The argument *group* is the spanning-tree group number. The argument *number* defines the circuit group number and is a small integer less than ten.

The parallel serial interfaces on both bridges must all be marked as being members of the same circuit group.

Example:

To load share over the two parallel serial links in this example, each router would have the configuration shown.

```
!
interface ethernet 0
bridge-group 1
!
interface serial 0
bridge-group 1
bridge-group 1 circuit 1
!
interface serial 1
bridge-group 1
bridge-group 1 circuit 1
!
bridge 1 protocol dec
!
```

Configuring X.25 Bridging

Cisco's transparent bridging software supports bridging of packets in X.25 frames. This ability is useful, as an example, for transmitting packets from proprietary protocols across an X.25 network.

To configure this capability, use this variation of the **x25 map** interface subcommand in the bridging configuration file:

x25 map bridge *X.121-address* **broadcast** [*options-keywords*]

The keyword **bridge** specifies bridging over X.25. The argument *X.121-address* is the X.121 address. The keyword **broadcast** is required for bridging over X.25. The optional argument *options-keywords* are the services that may be added to this map, and are listed in the section "Setting Address Mappings" in Chapter 8 of this publication.

The X.25 bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated in X.25 frames and transmitted across X.25 media. The command specifies Internet-to-X.121 address mapping and maintains a table of both the Ethernet and X.121 addresses.

X.25Bridging Example

The following is an example configuration illustrating three Cisco bridges connected to each other through an X.25 network.

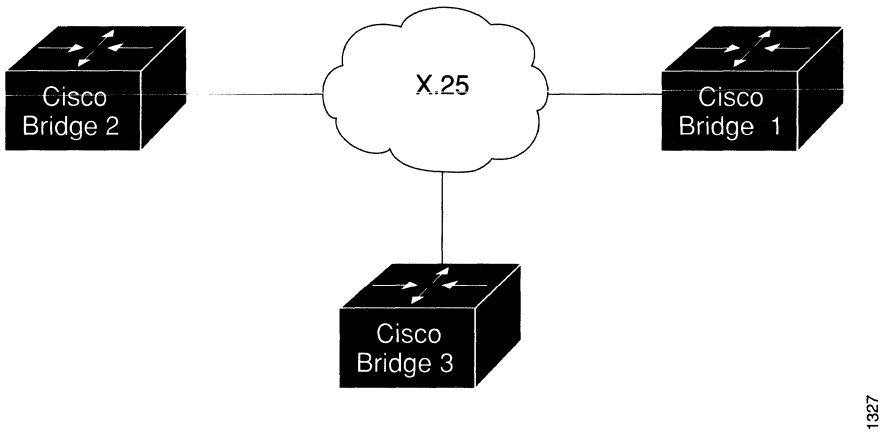


Figure 20-3 X.25 Bridging Example

Following are the configuration commands for each of the bridges depicted in Figure 20-3.

Example for Bridge 1:

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9  255.255.255.0
!
interface serial 0
encapsulation x25
x25 address 31370019027
bridge-group 5
x25 map bridge 31370019134 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 5 protocol ieee
!
```

Example for Bridge 2:

```
interface serial 1
encapsulation x25
x25 address 31370019134
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 4 protocol ieee
!
```

Example for Bridge 3:

```
interface serial 0
encapsulation x25
x25 address 31370019565
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019134 broadcast
!
bridge 5 protocol ieee
!
```

Configuring Frame Relay Bridging

Cisco's transparent bridging software supports bridging of packets over frame relay networks. This ability is useful, as an example, for transmitting packets from proprietary protocols across a frame relay network.

Bridging over a frame relay network is supported on both networks supporting a multicast facility and those that do not support such a facility. To configure bridging in a network not supporting a multicast facility, use this variation of the **frame-relay map** interface subcommand:

frame-relay map bridge *DLCI* **broadcast**

The keyword **bridge** specifies bridging over frame relay. The argument *DLCI* is the DLCI of the destination bridge. The keyword **broadcast** is required for bridging.

The frame relay bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated for transmission across a frame relay network. The command specifies Internet-to-DLCI address mapping and maintains a table of both the Ethernet and DLCIs.

Frame Relay Bridging Examples

The following is an example configuration illustrating three Cisco bridges connected to each other through a frame relay network.

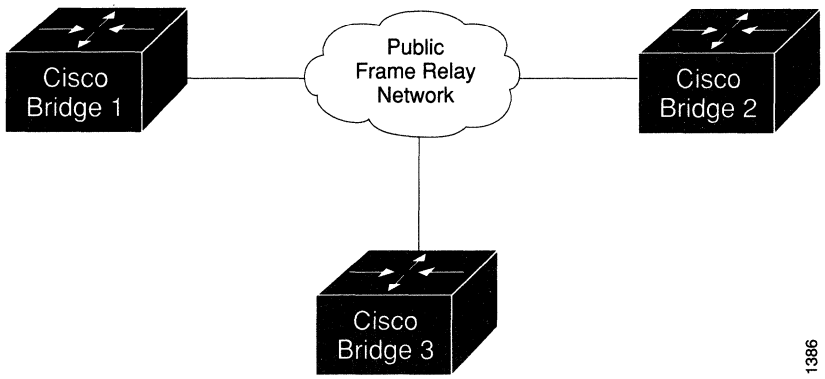


Figure 20-4 Frame Relay Bridging Example

Following are the configuration commands for each of the bridges depicted in Figure 20-4.

Example for Bridge 1:

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9  255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 134 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
!
```

Example for Bridge 2:

```
interface serial 1
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
!
```

Example for Bridge 3:

```
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 134 broadcast
!
bridge 5 protocol ieee
!
```

Bridging in a Frame Relay Network with Multicasts

The following example illustrates how to configure bridging in a frame relay network which supports the multicast facility.

Example for Bridge 1:

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9  255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
!
```

Example for Bridge 2:

```
interface serial 1
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
!
```

Example for Bridge 3:

```
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
!
```

In the above example, the multicast facility is used to learn about the other bridges on the network eliminating the need for the **frame-relay map** commands.

Configuring LAT Compression

The Local Area Transport (LAT) protocol used by DEC and DEC-compatible terminal servers is one of the common protocols that lacks a well-defined network layer (Level 3), and so must always be bridged.

To reduce the amount of bandwidth LAT traffic consumes on serial interfaces, a LAT-specific form of compression may be specified. This is done with the **bridge-group** *group* **lat-compression** interface subcommand. The command has this syntax:

bridge-group *group* **lat-compression**

The argument *group* is the spanning-tree group number.

Compression is applied to LAT frames being sent out the router/bridge through the interface in question.

LAT compression may be specified only for serial interfaces. For the most common LAT operations (user keystrokes and acknowledgment packets), LAT compression reduces LAT's bandwidth requirements by nearly a factor of two.

Example:

This command compresses LAT frames on the bridge assigned to group 1.

```
!
bridge-group 1 lat-compression
!
```

Transparent Bridging Configuration Examples

This section provides example configurations that you may use to configure your bridging environment.

Configuring Ethernet Bridging

In this example, two buildings have networks that must be connected via a T1 link. For the most part, the systems in each building use either IP or DECnet and, therefore, should be routed. There are some systems in each building that must communicate, but they can use only a proprietary protocol.

The example places two Ethernets in each building. One of the Ethernets will be used to attach to the rest of the building network that speaks IP and DECnet. The other Ethernet will be attached to the hosts that use a proprietary protocol. This Ethernet will be enabled for bridging to the serial line and to the other building. Figure 20-5 shows an example configuration.

The interfaces marked with an asterisk (*) are to be configured as part of spanning tree 1. The routers will be configured to route IP and DECnet. This configuration permits hosts on any Ethernet to communicate with hosts on any other Ethernet using IP or DECnet. In addition, hosts on Ethernet 1 in either building can communicate using protocols not supported for routing.

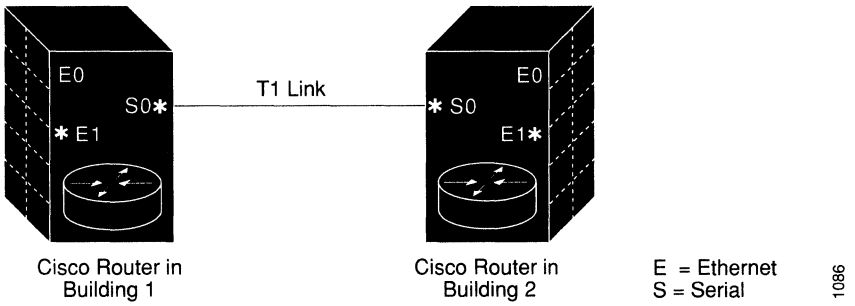


Figure 20-5 Ethernet Bridging Configuration Example

The configuration file for the router/bridge in Building 1 would be as follows. (Note that no bridging takes place over Ethernet 0. Both IP and DECnet routing are enabled on all interfaces.)

```
!
decnet address 3.34
interface ethernet 0
ip address 128.88.1.6  255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.1  255.255.255.0
bridge-group  1
decnet cost 10
!
interface ethernet 1
ip address 128.88.3.1  255.255.255.0
bridge-group  1
decnet cost 10
!
bridge 1 protocol dec
```

The configuration file for the router/bridge in Building 2 is similar.

```
!
decnet address 3.56
!
interface ethernet 0
ip address 128.88.11.9  255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.2  255.255.255.0
bridge-group  1
decnet cost 10
!
interface ethernet 1
ip address 128.88.16.8  255.255.255.0
bridge-group  1
decnet cost 10
!
bridge 1 protocol dec
```

Maintaining the Transparent Bridge

Use the **clear bridge** command to remove any learned entries from the forwarding database and zero the transmit and receive counts for any statically configured forwarding entries.The command has this syntax:

clear bridge *group*

The argument *group* is the number you chose to specify a particular spanning tree.

Monitoring the Transparent Bridge

This section describes the EXEC commands you use to obtain displays of activity on the bridged network.

Viewing Entries in the Forwarding Database

Use the **show bridge** command to view classes of entries in the bridge forwarding database. The command has this syntax:

show bridge [*group*] [*interface*]

show bridge [*group*] [*address* [*mask*]]

The optional argument *group* is the number you chose that specifies a particular spanning tree.

The optional argument *interface* is a specific interface, such as Ethernet 0.

The optional argument *address* is a 48-bit Ethernet address. This may be entered with an optional mask of bits to be ignored in the address, which is specified with the *mask* argument.

In the sample display, below, the first command would display all entries for hosts reachable via interface Ethernet 0, the second command would display all entries with the vendor code of 0000.0c00.0000, and the third command displays the entry for address 0000.0c00.0e1a. In the fourth command, all entries in the forwarding database would be displayed. In all four examples, the bridge group number has been omitted.

```
show bridge ethernet 0
show bridge 0000.0c00.0000  0000.00FF.FFFF
show bridge 0000.0c00.0e1a
show bridge
```

The following is sample output of the **show bridge** command:

```
Total of 300 station blocks, 295 free
BG Hash   Address        Action  Interface Age  RX count  TX count
 1 00/0   FFFF.FFFF.FFFF discard     -       P          0         0
 1 09/0   0000.0C00.0009 forward Ethernet0   0          2         0
 1 49/0   0000.0C00.4009 forward Ethernet0   0          1         0
 1 CA/0   AA00.0400.06CC forward Ethernet0   0         25         0
```

The first line of the **show bridge** output lists the total number of forwarding database elements in the system and the number in the free list. The total number of forwarding elements is expanded dynamically, as needed. Other field descriptions follow.

Table 20-2 Forwarding Database Display Field Descriptions

Field	Description
BG	Indicates the bridging group to which the address belongs.
Address	Is the MAC address.
Action	Is the action to be taken when that address is looked up; choices are to discard or forward the datagram.
Interface	Indicates the interface, if any, on which that address was seen.
Age	Indicates the number of minutes since a frame was received from or sent to that address. The letter "P" indicates a permanent entry. The letter "S" indicates the system as recorded by the router. On the modular systems, this is typically the broadcast address and the router's own hardware address; on the IGS, this field will also include certain multicast addresses.
RX count	Displays count of the number of frames received from that address.
TX count	Displays count the number of frames sent to that address.

Displaying the Known Spanning Tree Topology

Use the **show span** command to display the spanning-tree topology known to the router/bridge. The display includes whether or not LAT group code filtering is in effect. The command has this syntax:

show span

The following is a sample output of the **show span** command. The first part of the display lists global spanning-tree parameters, followed by port-specific parameters.

```
Bridge Group 1 is executing the DEC compatible spanning tree protocol
  Bridge Identifier has priority 127, address 0000.0c00.4369
  Configured hello time 1, max age 15, forward delay 30
  We are the root of the spanning tree
  Acquisition of new addresses is enabled
  Forwarding of multicast source addresses is disabled
  LAT service filtering is disabled
  Topology change flag not set, detected flag not set
  Times:  hold 1, topology change 30, notification 30
          hello 1, max age 15, forward delay 30
  Timers: hello 1, topology change 0, notification 0
 --More--
Port 5 (Ethernet0) of bridge group 1 is forwarding. Path cost 10, pri-
ority 0
   Designated root has priority 127, address 0000.0c00.4369
   Designated bridge has priority 127, address 0000.0c00.4369
   Designated port is 5, path cost 0
   Timers: message age 0, forward delay 0, hold 1
   LAT compression is not set
   Input LAT service deny group code list is not set
   Input LAT service permit group code list is not set
   Output LAT service deny group code list is not set
   Output LAT service permit group code list is not set
   Access list for input filtering on type is 201; for LSAP is not set
   Access list for input address filter is not set
   Access list for output filtering on type is not set; for LSAP is not
set
   Access list for output address filter is not set
 --More--
Port 6 (Serial4) of bridge group 1 is forwarding. Path cost 64, priority
0
   Designated root has priority 127, address 0000.0c00.4369
   Designated bridge has priority 127, address 0000.0c00.4369
   Designated port is 6, path cost 0
   Timers: message age 0, forward delay 0, hold 1
   LAT compression is set
   Input LAT service deny group code list is not set
   Input LAT service permit group code list is not set
   Output LAT service deny group code list is not set
   Output LAT service permit group code list is not set
   Access list for input filtering on type is 202; for LSAP is not set
   Access list for input address filter is not set
   Access list for output filtering on type is 202; for LSAP is not set
   Access list for output address filter is not set
 --More--
```

```
Port 7 (Ethernet1) of bridge group 1 is down. Path cost 10, priority 0
   Designated root has priority 127, address 0000.0c00.4369
   Designated bridge has priority 127, address 0000.0c00.4369
   Designated port is 7, path cost 0
   Timers: message age 0, forward delay 0, hold 1
   LAT compression is not set
   Input LAT service deny group code list is not set
   Input LAT service permit group code list is not set
   Output LAT service deny group code list is not set
   Output LAT service permit group code list is not set
   Access list for input filtering on type is 201; for LSAP is not set
   Access list for input address filter is not set
   Access list for output filtering on type is not set; for LSAP is not
set
   Access list for output address filter is not set
```

Debugging the Transparent Bridge

This section describes the EXEC debugging commands you use to debug the transparent bridge. For each **debug** command, there is a corresponding **undebug** command to disable the reports.

debug span

Use the **debug span** command to track changes in the spanning-tree topology. This command is useful for verifying correct operation of the spanning-tree protocol.

debug lat

Use the **debug lat** command on a bridge to show group code filtering actions.

debug lat-packet

Use the **debug lat-packet** command to list all LAT service advertisements which were forwarded.

Transparent Bridging Global Configuration Command Summary

This section provides a summary of the transparent bridging-specific global configuration commands.

[no] access-list *list* {**permit** | **deny**} *type-code wild-mask*

Prepares access control information for filtering of frames by protocol type. The argument *list* is a user-selectable number between 200 and 299 that identifies the list. The keyword **permit** permits the frame; the keyword **deny** denies the frame. The argument *type-code* is a 16-bit hexadecimal number written with a leading "0x." The argument *wild-mask* is another 16-bit hexadecimal number whose ones bits correspond to bits in the *type-code* argument that should be ignored when making a comparison.

[no] access-list *list* {**permit** | **deny**} *address-mask*

Prepares access control information for filtering of frames by Ethernet and IEEE 802.3 address. The argument *list* is an integer from 700 to 799 selected for the list. The argument *address* and *mask* are 48-bit Ethernet addresses written in dotted triplet form. The ones bits in the *mask* argument are the bits to be ignored in *address*.

[no] bridge *group* **acquire**

The negative form of this command disables the dynamic learning process and is the default. The argument *group* is the spanning-tree group number.

[no] bridge *group* **address** *ethernet-address* [**forward** | **discard**] [*interface*]

Adds or removes an address from the forwarding database. The argument *group* is the spanning-tree group number. The argument *ethernet-address is* a 48-bit dotted triplet hardware address such as those displayed by the EXEC **show arp** command. The argument *ethernet-address* is either a station address, the broadcast address, or a multicast destination address. The optional **forward** keyword enables forwarding of a frame sent from or destined to the specified address. The optional **discard** keyword causes frames sent from or destined to the specified address to be discarded without further processing. The optional argument *interface* specifies an interface after the **discard** or **forward** keyword to indicate the interface on which that address can be reached. Use the **no** for of the command followed by the Ethernet address to remove an address from the forwarding database.

[no] bridge *group* **domain** *domain-number*

Enables/disables multiple domain spanning trees. Any number of router/bridges can be placed within the domain. The devices in the domain, and only those devices, will then share spanning-tree information. The argument *group* is the bridge group number, and must be a number between 0 and 10, as specified in the **bridge** *group* **protocol ieee** command. The keyword **domain** is required; the argument *domain-number* is a domain number you choose. The default domain number is zero, and this is the required domain number when communicating to IEEE bridges that do not support this domain extension.

Note: This command works only when the bridge group is running the IEEE spanning-tree protocol. Non-Cisco bridges may not work correctly on networks containing Cisco bridges with nonzero domain numbers.

[no] bridge *group* **forward-time** *seconds*

Sets or returns to the default the forward delay interval, or the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is any value between ten and 200 seconds. The default value is 30 seconds.

[no] bridge *group* **hello-time** *seconds*

Specifies or returns to the default the interval between HELLO Bridge Protocol Data Units (BPDUs). The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is any value between one and ten seconds. The default value is one second.

[no] bridge *group* **lat-service-filtering**

Enables or disables LAT service filtering. The argument *group* specifies the bridge group. The default is **no** LAT service filtering.

[no] bridge *group* **max-age** *seconds*

Specifies or removes the interval in which the spanning-tree topology is recomputed when a bridge does not hear BPDUs from the root bridge. The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is the interval the bridge will wait to hear BPDUs from the root bridge. The default interval is 15 seconds.

[no] bridge *group* **multicast-source**

Allows or disallows the forwarding, but not the learning, of multicast source addresses. The argument *group* is the group number assigned to the spanning tree.

[no] bridge *group* **priority** *number*

Sets the priority of an individual bridge for selection as the root bridge. The argument *group* is the group number assigned to the spanning tree. The argument *number* can range from 1 to 65,000. The default priority value is 128. A lower number increases the likelihood for selection as the root bridge.

[no] bridge *group* **protocol** {**dec**|**ieee**}

Defines or removes a spanning-tree protocol and spanning-tree group. The argument *group* is a number between one and nine that refers to a particular spanning tree. The keyword **protocol** specifies the protocol to use, either **ieee** or **dec.**

Transparent Bridging Interface Subcommand Summary

This section provides an alphabetical summary of the bridging-specific interface subcommands.

[no] bridge-group *group*

Assigns or removes the network interface to or from the spanning-tree group. The argument *group* is the group number assigned to the spanning tree.

[no] bridge-group *group* **circuit** *number*

Establishes or removes load balancing. The command marks a serial interface as belonging to circuit group number. The argument *group* is the spanning-tree group number.

The argument *number* defines the circuit group number and is a small integer less than ten. Parallel serial interfaces on both bridges must all be flagged as being members of the same circuit group.

[no] bridge-group *group* **input-address-list** *list*

Assigns or removes an access list to a particular interface for filtering the Ethernet or IEEE 802.3 source addresses. The argument *group* is the spanning-tree group number. The argument *list* is an access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[no] bridge-group *group* **output-address-list** *list*

Assigns or removes an access list to a particular interface for filtering the Ethernet or IEEE 802.3 destination addresses. The argument *group* is the spanning-tree group number. The argument *list* is an access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[no] bridge-group *number* **input-lat-service-deny** *grouplist*

When enabled, causes the system to not bridge any LAT service advertisement which match the group list specified on input. The argument *grouplist* lists the LAT groups. The argument *number* is the previously chosen spanning-tree group number. Default is no filtering.

[no] bridge-group *number* **input-lat-service-permit** *grouplist*

When enabled, causes the system to bridge only those LAT service advertisements which match the group list specified on input. The argument *grouplist* lists the LAT group codes. The argument *number* is the previously chosen spanning-tree group number. Default is no filtering.

[no] bridge-group *number* **output-lat-service-deny** *grouplist*

When enabled, causes the system to not bridge onto this output interface any LAT service advertisements that match any group in the argument *grouplist.* The argument *number* is the previously chosen spanning-tree group number. Default is no filtering.

[no] bridge-group *number* **output-lat-service-permit** *grouplist*

When enabled, causes the system to bridge onto this output interface only those service advertisements that match any group in the argument *grouplist*. The argument *number* is the previously chosen spanning-tree group number. If a message matches both a deny and a permit, the message will not be bridged. The EXEC **show span** command reports the group code filtering in effect. Default is no filtering.

[no] bridge-group *group* **input-lsap-list** *list*

Adds or removes a filter for IEEE 802.3-encapsulated packets on input. This access list is applied to all IEEE 802.3 frames received on that interface prior to the bridge-learning process. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[no] bridge-group *group* **output-lsap-list** *list*

Adds or removes a filter for IEEE 802.3-encapsulated packets on output. This access list is applied just before sending out a frame to an interface. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 200, which you assigned with the bridge **access-list** command.

[no] bridge-group *group* **input-type-list** *list*

Adds or removes a filter for Ethernet- and SNAP-encapsulated packets on input. The access list is then applied to all Ethernet frames received on that interface prior to the bridge learning process. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[no] bridge-group *group* **output-type-list** *list*

Adds or removes a filter for Ethernet- and SNAP-encapsulated packets on output. This access list is then applied just before sending out a frame to an interface. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[no] bridge-group *group* **lat-compression**

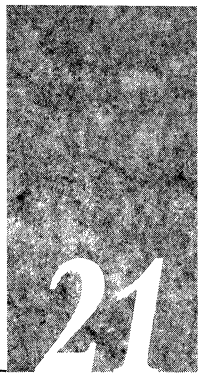
Reduces the amount of bandwidth LAT traffic consumes on serial interfaces. The argument *group* is the spanning-tree group number. Compression is applied to LAT frames being sent out the router/bridge through the interface in question. LAT compression may be specified only for serial interfaces. For the most common LAT operations (user keystrokes and acknowledgment packets), LAT compression reduces LAT's bandwidth requirements by nearly a factor of two.

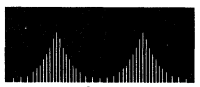
[no] bridge-group *group* **path-cost** *cost*

Sets or removes a different path cost. The path cost can range from 0 to 65,535, with higher values indicating higher costs. The argument *group* is the spanning-tree group number. The argument *cost* is the path cost. The default path cost is 100.

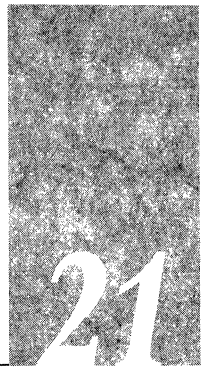
Assigns a priority to an interface. This priority is used in tie-breaking when computing a network topology. The argument *group* is the spanning-tree group number. The argument *number* can range from 0 to 255. The default value is zero, and the lower the number, the more likely it is that the bridge on this interface will be chosen as the root.

Chapter 21
Configuring Source-Route Bridging





Chapter 21
Configuring Source-Route Bridging



This chapter describes routing and bridging in Token Ring environments. These topics are included in this chapter:

- The fundamental concepts of source routing and how source-route bridges interact with Level 3 routers.
- The IEEE 802.5 Token Ring frame format and the routing information field (RIF).
- Enabling use of the RIF for source-route bridging.
- Filtering datagrams by protocol type, vendor code, and configuring NETBIOS access control filters.

Source-Route Bridging Overview

The Cisco bridging software includes source-route bridging capability. This ability allows the Cisco router/bridge to simultaneously act as a Level 3 router and a Level 2 source-route bridge. This allows protocols such as Novell or XNS to be routed on Token Rings, while other protocols such as SNA or NETBIOS are source-route bridged.

Source-route bridging technology is a combination of bridging and routing functions. A source-route bridge is allowed to make routing decisions based upon the contents of the Medium Access Control (MAC) frame header. Keeping the routing function at the MAC or Level 2 layer allows the higher layer protocols to execute their tasks more efficiently, and also allows the LAN to be expanded without the knowledge of the higher layer protocols.

As designed by IBM and the IEEE 802.5 committee, source-route bridges connect extended Token Ring LANs. A source-route bridge uses the Routing Information Field (RIF) in the IEEE 802.5 MAC header of a datagram (see Figure 21-1) to determine which rings, or Token Ring network segments, the packet must transit. The RIF is inserted into the MAC header immediately following the source address field in every frame by the source station, giving this style of bridging its name. The destination station reverses the routing field to reach the originating station.

The information in a RIF is derived from explorer packets generated by the source node. These explorer packets traverse the entire source-route bridge network, gathering information on the possible paths the source node might use.

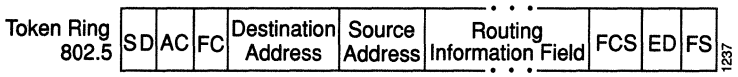


Figure 21-1 IEEE 802.5 Token Ring Frame Format

Unlike transparent spanning-tree bridging that requires time to recompute topology in the event of failures, source-route bridging allows multiple, active paths through the network, which provides for more timely switches to alternate routes in the event of failure. Most importantly, source-route bridging places the burden of transmitting frames with the end stations by allowing them to determine the routes the frames take.

Cisco's Implementation of Source-Route Bridging

Cisco's source-route bridging software implementation provides these features:

- Provides configurable fast switching software for source-route bridging.
- Provides for a local source-route bridge that connects two or more Token Ring networks.
- Provides *ring groups* to configure a source-route bridge with more than two network interfaces. A ring group is a collection of Token Ring interfaces in one or more Cisco routers that are collectively treated as a virtual ring.
- Provides explorer packets to collect RIF information. An *all rings* explorer packet follows all possible paths to a destination ring. Spanning explorer packets follow a statically configured spanning tree when looking for paths.
- Provides for remote source-route bridges that involve multiple router/bridges separated by non-Token Ring segments by either encapsulating the Token Ring traffic inside IP datagrams passed over a TCP connection between two Cisco router/bridges, or by using HDLC over a serial line between two routers attached to Token Ring networks.
- Provides for configurable limits to the size of the TCP backup queue.
- Provides a dynamically determined RIF cache based on the protocol. It also allows you to manually add entries to the RIF cache.
- Provides for filtering of NETBIOS frames. The frames can be filtered by station name or by a packet byte offset.
- Provides for filtering by MAC address, LSAP header, and protocol type.

Configuring Source-Route Bridging

To configure source-route bridging on your Cisco router/bridge, follow these steps:

Step 1: Enable use of the Routing Information Field (RIF) for routed protocols with the **multiring** command.

Step 2: Configure the Token Ring interface for source-route bridging. The Cisco router/bridge supports both local and remote source-route bridging.

To determine if your Token Ring interface has the proper hardware and firmware support for source-route bridging, examine the output of the EXEC **show interface** command for that interface. If the output has a line that says "Source Route Bridge capable," then you may proceed with configuration. Otherwise, you need to contact Cisco Systems for a hardware and firmware field upgrade.

Step 3: Define the types of explorer packets to use: either spanning tree or the default all rings explorer packets.

The source-route bridging software supports filtering of frames. Filtering can be done by protocol type or by vendor code. It is also possible to configure access control filters for packets transmitted across a Token Ring bridge using the NETBIOS interface. Additionally, EXEC-level commands for monitoring and debugging the bridge are also available. These tasks and commands are described in the following sections.

Enabling and Disabling the Source-Route Fast-Switching Cache

By default, fast switching software is enabled in the source-route bridging software. This feature allows for faster implementations of local source route bridging between 4/16 megabit Token Ring cards (CSC-R16) in the same Cisco router/bridge. To disable this feature, use the **source-bridge route-cache** interface subcommand. The full syntax of this command follows:

source-bridge route-cache

no source-bridge route-cache

Example:

These commands disable use of fast source-route bridging between two Token Ring units.

```
interface token 0
source-bridge 1 1 2
no source-bridge route-cache
!
interface token 1
source-bridge 2 1 1
no source-bridge route-cache
```

Configuring the Source-Route Information Field

Figure 21-2 illustrates the basic format for the Routing Information Field.

A *ring* is a single Token Ring network segment. Each ring in the extended Token Ring network is designated by a unique 12-bit ring number. Each bridge between two token rings is designated by a unique 4-bit bridge number. Bridge numbers must be unique only between bridges that connect the same two Token Rings. A RIF is built up of ring and bridge numbers.

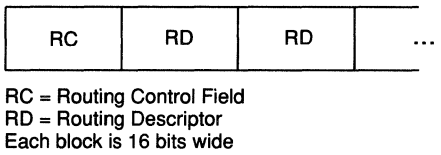


Figure 21-2 Basic RIF Format

Figure 21-3 illustrates the routing control format for the RIF. Descriptions of each field follow.

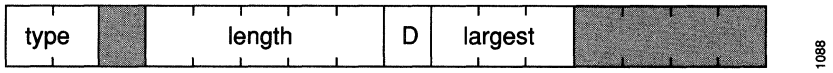


Figure 21-3 RIF Routing Control Format

- Shaded fields are reserved.
- type—RIF type, as follows:
  - — 00: Specific route
  - — 10: All rings, all routes
  - — 11: All rings, spanning routes (limited broadcast)
- length—Total length in bytes of the RIF.
- D—Direction, indicated as follows:
  - — 0: Interpret route left to right (forward)
  - — 1: Interpret route right to left (reverse)
- largest—Largest frame that can be handled by this route, as follows:
  - — 000: 516 bytes (DDN 1822)
  - — 001: 1500 bytes (Ethernet)
  - — 010: 2052 bytes

— 011: 4472 bytes (Token Ring, and Cisco maximum)

— 100: 8144 bytes (Token Bus)

— 101: 11407 bytes

— 110: 17800 bytes

— 111: 65535 (initial values)

Figure 21-4 describes the routing descriptor format of the RIF string.

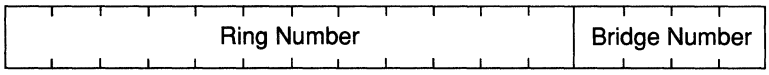


Figure 21-4 Routing Descriptor Format

- Ring Number—Unique decimal ring number within the bridged network.
- Bridge Number—Unique decimal bridge number between any bridges connecting the same two rings.

The following sections describe how to enable and configure static RIF entries.

Enabling Use of the RIF

Level 3 routers that use protocol-specific information (for example, Novell IPX or XNS headers), rather than MAC information to route datagrams, must also be able to collect and use RIF information to ensure that they can transmit datagrams across a source-route bridge. The Cisco software default is to *not* collect and use RIF information for routed protocols. This allows operation with software that does not understand or properly use RIF information, such as versions of Novell Netware prior to version 2.15c.

To enable collection and use of RIF information, use the **multiring** interface subcommand. The full command syntax follows:

multiring {*protocol-keyword* | **all** | **other**}

no multiring {*protocol-keyword* | **all** | **other**}

The **multiring** command was extended in software release 8.3 to allow for per-protocol specification of the interface's ability to append RIFs to routed protocols. When it is enabled for a protocol, the router will source packets that include information used by source-route bridges. This allows a Cisco router with Token Ring interfaces, for the protocol or protocols specified, to connect to a source-bridged Token Ring network. If a protocol is not specified for multiring, the Cisco router can only route packets to nodes directly connected to its local Token Ring.

Note: Previous to software release 8.3, the **multiring** command enabled multiring protocols, in particular, the use of explorers and RIFs, for *all* routable protocols. This sometimes caused problems when multiring-capable devices speaking one particular protocol were attached to the same ring as a non-multiring-capable device speaking a different network protocol. If the earlier **multiring** command (pre-8.3 release) was not not specified, nodes speaking one particular protocol would be able to communicate through the Cisco router, but nodes speaking the different protocol could not. The reverse was true when the multiring capability was specified on the interface.

The current software allows you to specify a protocol. This is specified by the argument *protocol-keyword*. The protocols supported and the keyword you enter follow:

- **apollo**—Apollo Domain
- **appletalk**—AppleTalk phase 1 and 2
- **clns**—ISO CLNS
- **decnet**—DECnet Phase IV
- **ip**—IP
- **novell**—Novell IPX
- **vines**—Banyan VINES
- **xns**—XNS

There are also two special keywords with the **multiring** command. The keyword **all** enables the multiring for *all* frames. The keyword **other** enables the multiring for *any* routed frame not included in the previous list of supported protocols.

Note: In 8.3 or later releases of the software, the command **multiring all** is equivalent to the previous **multiring** command.

The **no multiring** subcommand with the appropriate keyword disables the use of RIF information for the protocol specified.

Example:

These commands enable a Token Ring interface for the IP and Novell IPX protocols. RIFs will be generated for IP frames, but not for the Novell IPX frames.

```
!
interface tokenring 0
multiring ip
ip address 131.108.183.37  255.255.255.0
novell network 33
!
```

Determining the RIF Time-Out Interval

RIF information is maintained in a cache whose entries are aged. The global configuration command **rif timeout** determines the number of minutes an inactive RIF entry is kept. The full command syntax follows:

rif timeout *minutes*

no rif timeout

The default interval is 15 minutes. Assign a new interval value using the *minutes* argument.

The **no rif timeout** command restores the default.

The EXEC command **show rif** displays the contents of the RIF cache. The EXEC command **clear rif-cache** clears the contents of RIF cache. See the sections "Maintaining the Source-Route Bridge" and "Monitoring the Source-Route Bridge" later in this chapter for more information about these commands.

Example:

This command changes the time-out period to five minutes.

```
!
rif timeout 5
!
```

Configuring a Static RIF Entry

If a Token Ring host does not support the use of IEEE 802.2 TEST or XID datagrams as explorer packets, it may be necessary to add static information to the RIF cache of the router/bridge.

To enter static source route information into the RIF cache, use this following variation of the **rif** global configuration command (negative form of the command included):

rif *MAC-address* [*RIF-string*] [*interface-name* | **ring-group** *ring*]

no rif *MAC-address* [*RIF-string*] [*interface-name* | **ring-group** *ring*]

The argument *MAC-address* is a 12-digit hexadecimal string written as a dotted triple, for example 0010.0a00.20a6.

The command **rif** *MAC-address* (without any of the optional arguments), puts an entry into the RIF cache indicating that packets for this MAC address should *not* have RIF information.

The optional argument *RIF-string* is a series of 4-digit hexadecimal numbers separated by a dot (.). This RIF string is inserted into the packets sent to the specified MAC address.

An interface name (for example, tokenring0), can be specified with the optional *interface-name* argument, to indicate the origin of the RIF.

A ring group number (specified with the **source-bridge ring-group** global configuration command) may also be specified with the **ring-group** keyword and *ring* argument, to indicate the origin of the RIF. Ring groups are explained in the section "Configuring Ring Groups and Multiport Source-Bridges."

Do not configure a static RIF with any of the *all rings* type codes. Doing so causes traffic for the configured host to appear on more than one ring and leads to unnecessary congestion. The format of a RIF string is illustrated in Figure 21-2, Figure 21-3, and Figure 21-4.

Note: Input to the **source-bridge** configuration subcommand is in decimal format. RIF displays and input are in hexadecimal format, and IBM source-route bridges use hexadecimal for input. It is essential that bridge and ring numbers are consistent for proper network operation. This means you must explicitly declare the numbers to be hexadecimal by preceding the number with 0x, or you must convert IBM hexadecimal numbers to a decimal equivalent when entering these numbers. As an example, IBM hexadecimal bridge number 10 would be entered as hexadecimal number 0x10 or decimal number 16 in the Cisco configuration commands. In the displays, these commands will always be in decimal.

The command **no rif** with the MAC address argument removes an entry from the cache.

Example:

In this example configuration the path between rings 8 and 9 connected via source-route bridge *1* is described by the route descriptor *0081.0090.* A full RIF, including the route control field, would be *0630.0081.0090.* The static RIF entry would be submitted to the leftmost router as follows.

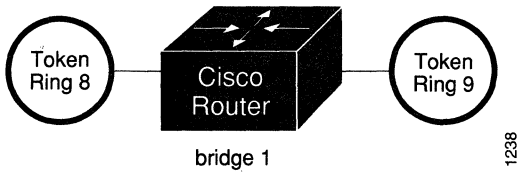


Figure 21-5 Assigning a RIF to a Source-Route Bridge

```
!
rif 1000.5A12.3456 0630.0081.0090
!
```

As another example, assume a datagram was sent from a Cisco router/bridge on ring 21 (15 hexadecimal), across bridge 5 to ring 256 (100 hexadecimal), and then across bridge 10 (A hexadecimal) to ring 1365 (555 hexadecimal) for delivery to a destination host on that ring.

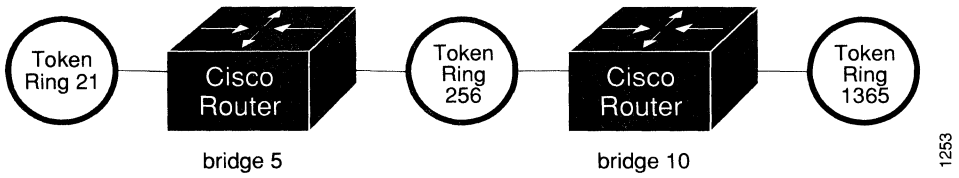


Figure 21-6 Assigning a RIF to a Two-Hop Path

The RIF in the leftmost router describing this two-hop path is *0830.0155.100a.5550*, and is entered as follows:

```
!
rif 1000.5A01.0203 0830.0155.100a.5550
!
```

Configuring Local Source-Route Bridging

This section describes how to configure the Cisco router/bridge as a local source-route bridge. A local source-route bridge directly connects two or more Token Ring networks. Bridged traffic does not pass across nonToken Ring media.

When acting as a source-route bridge, only those protocols that are not being routed are source-route bridged. For example, if Novell routing is enabled on the Cisco router/bridge, Novell datagrams will not be source-bridged. Datagrams for other non-routed protocols will be source-bridged, however.

Enabling Local Source-Route Bridging

To configure an interface for source-route bridging, use the **source-bridge** interface subcommand as follows:

source-bridge *local-ring bridge-number target-ring*

The argument *local-ring* is the ring number for this interface's Token Ring. A ring number is a decimal number between 1 and 4095 that uniquely identifies a network segment or ring within the bridged Token Ring network.

The argument *bridge-number* is a decimal number between 1 and 15 that uniquely identifies a bridge connecting the two rings.

The argument *target-ring* is the decimal ring number of the destination ring on this router/bridge. It must also be unique within the bridged Token Ring network.

The **no source-bridge** command disables source bridging on a particular interface.

Example:

The following example configures this simple two-port bridge.

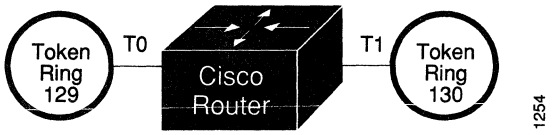


Figure 21-7 Dual Port Source-Route Bridge Configuration

```
!
interface token ring 0
source-bridge 129 1 130
!
interface token ring 1
source-bridge 130 1 129
!
```

Token Rings 129 and 130 are connected via the Cisco router/bridge.

Configuring Explorer Packets

There are two types of explorer packets used to collect RIF information:

- All rings, all routes explorer packets follow all possible paths to a destination ring. In a worst case scenario, the number of all-rings explorers generated may be exponentially large.
- Spanning or limited route explorer packets follow a spanning tree when looking for paths, greatly reducing the number of explorer packets required. There is currently no dynamic spanning tree algorithm to establish that spanning tree; it must be manually configured.

Enabling Spanning Explorers

Use the **source-bridge spanning** interface subcommand to enable use of spanning explorers. The full command syntax follows:

source-bridge spanning

no source-bridge spanning

The command puts the interface into a forwarding or active state with respect to the spanning tree.

The **no source-bridge spanning** command disables use of spanning explorers. Only spanning explorers will be blocked; everything else will be forwarded. Use of the **source-bridge spanning** command is recommended.

Limiting the Number of Source-Route Bridge Hops

If you wish to limit the maximum number of source-route bridge hops of your network, use the **source-bridge max-rd** interface subcommand. The full command syntax follows:

source-bridge max-rd *count*

no source-bridge max-rd

The argument *count* determines the number of route descriptors that may appear in the RIF. It is one more than the number of bridges an explorer packet may traverse. The typical maximum for interoperability with IBM equipment is 7 (eight rings and seven bridges).

The command **no source-bridge max-rd** resets the count back to the maximum value.

Example:

The following example builds on the dual-port, source-route bridge configuration seen in Figure 21-7. The example routes IP and source-route bridges all other protocols. Spanning explorers are used.

```
!
interface tokenring 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface tokenring 1
ip address 131.108.130.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
multiring all
!
```

The **multiring** subcommand causes the IP routing software to use RIFs, as necessary.

Configuring Ring Groups and Multiport Source-Bridges

To configure a source-route bridge with more than two network interfaces, Cisco uses the concept of a *ring group*. A ring group is a collection of Token Ring interfaces in one or more Cisco routers that are collectively treated as a virtual ring. The ring group is denoted by a ring number that must be unique for the network. The ring group's number is used just like a physical ring number, showing up in any route descriptors contained in packets being bridged.

A ring group is defined or removed with the **source-bridge ring-group** global command. The full command syntax follows:

source-bridge ring-group *ring-group-number*

no source-bridge ring-group *ring-group-number*

To configure a specific interface as part of a ring group, its target ring number parameter is set to the ring group number.

Example:

Following is an example configuration of a four-port Token Ring source route bridge.

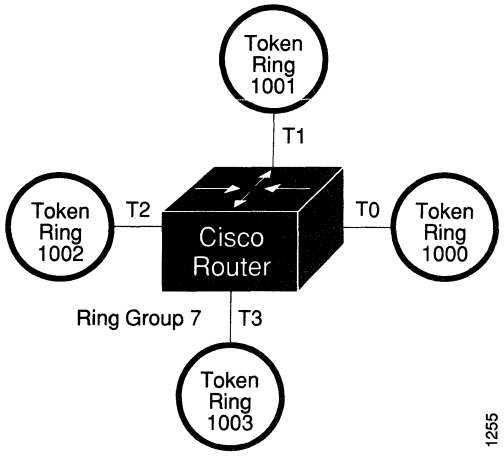


Figure 21-8 Four Port Source-Route Bridge

Rings 1000, 1001, 1002, and 1003 are all source-route bridged to each other across ring group 7.

```
!
source-bridge ring-group 7
!
interface tokenring 0
source-bridge 1000 1 7
source-bridge spanning
!
interface tokenring 1
source-bridge 1001 1 7
source-bridge spanning
!
interface tokenring 2
source-bridge 1002 1 7
source-bridge spanning
!
interface tokenring 3
source-bridge 1003 1 7
source-bridge spanning
!
```

Configuring Remote Source-Route Bridging

The previous sections have discussed local source-route bridging, bridging between token rings connected by the same router/bridge. This section describes how to configure remote source-route bridges involving multiple router/bridges separated by nonToken Ring network segments. The following sections assume you are familiar with configuring a Cisco local source-route bridge.

There are two ways to set up remote source-route bridging: One is to encapsulate the source-route bridged traffic inside IP datagrams passed over a TCP connection between two router/bridges. TCP is used to ensure the reliable and ordered delivery of source-route-bridged traffic. TCP has the following advantages:

- Token Ring networks may be connected across arbitrary media including Ethernets, FDDI, serial interfaces, X.25 networks, and so forth.
- A multiprotocol backbone network may be used. There is no need to dedicate a special wide area network to carrying Token Ring traffic.
- If the IP network is engineered properly, the source-route traffic can take advantage of multiple redundant paths. Cisco multiprotocol routers can load share over the redundant paths. Also, if a path fails, there is no need for hosts to retransmit explorer packets. The IP routing handles the network reconfiguration transparently to the Token Ring hosts.

The second method for setting up a remote source-route bridge is to use a dedicated serial line between two routers attached to Token Rings. This method is recommended when you are running source-route bridge traffic over a slow serial line (56 kilobits per second or less). You do not have the flexibility of the TCP approach, but you do have better performance since there is less of the overhead associated with TCP.

Configuring Remote Source-routing over TCP

To configure a remote source-route bridge to use TCP, follow these steps:

- Define a ring group. Every Cisco router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. These other router/bridges are referred to as "peers."
- List your peers with multiple uses of the **source-bridge remote-peer** command. You must include one of your own IP addresses in that list. Each peer should appear only once in this list, not one time for each Token Ring present. All peers should have the same list of peers.
- Configure the Token Ring interfaces for source route bridging. The value of the target ring parameter for the **source-bridge** command should be the ring group number.

Listing the Peer Bridges

The **source-bridge remote-peer** global configuration command has the following syntax when using TCP (the negative form of the command is also listed):

source-bridge remote-peer *ring-group* **tcp** *ip-address* [**lf** *size*]

no source-bridge remote-peer *ring-group* **tcp** *ip-address*

This command is used to identify the IP address of a peer in our ring group with which to exchange source-bridge traffic using TCP.

The keyword **lf** specifies the maximum size frame to be sent to this remote peer. The router negotiates all transit routes down to this size or lower. This argument is useful in preventing time-outs in end hosts, by reducing the amount of data they have to transmit in a fixed interval. For example, in some networks containing slow links, it would be impossible to transmit an 8K frame and receive a response within a few seconds. These are fairly standard defaults for an application on a 16 megabit Token Ring. If the frame size is lowered to 516 bytes, then only 516 bytes must be transmitted and a response received in 2 seconds. This is a much easier accomplishment in a network with slow links. The legal values for this argument are 516, 1470, 2052, 4472, 8144, 11454, and 17,800 bytes.

Example:

The following example illustrates a configuration of two router/bridges configured for remote source-route bridging using TCP as a transport. Each router has two Token Rings. They are connected together by an Ethernet segment over which the source-route bridged traffic will pass. The first router configuration is a source-route bridge at address *131.108.2.29*.

```
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 131.108.2.29
source-bridge remote-peer 5 tcp 131.108.1.27
!
interface ethernet 0
ip address 131.108.4.4 255.255.255.0
!
interface tokenring 0
ip address 131.108.2.29 255.255.255.0
source-bridge 1000 1 5
source-bridge spanning
!
interface tokenring 1
ip address 131.108.128.1 255.255.255.0
source-bridge 1001 1 5
source-bridge spanning
!
```

The configuration of the source-route bridge at *131.108.1.27* is:

```
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 131.108.2.29
source-bridge remote-peer 5 tcp 131.108.1.27
!
interface ethernet 0
ip address 131.108.4.5 255.255.255.0
!
interface tokenring 0
ip address 131.108.1.27 255.255.255.0
source-bridge 10 1 5
source-bridge spanning
!
interface tokenring 1
ip address 131.108.131.1 255.255.255.0
source-bridge 11 1 5
source-bridge spanning
!
```

Limiting the Size of the Backup Queue

You can limit the size of the backup queue for remote source-route bridging to control the number of packets that can wait for transmission to a remote ring before packets start being thrown away. You use the **source-bridge tcp-queue-max** command to do this. The full syntax for this command follows.

source-bridge tcp-queue-max *number*

no source-bridge tcp-queue-max

The argument *number* is the number of packets to hold in any single outgoing TCP queue to a remote Cisco router. The default value is 100. Enter the **no source-bridge tcp-queue-max** command to defeat this limit.

Example:

If, for example, your network experiences temporary bursts of traffic using the default packet queue length, the following command raises the limit from 100 to 150 packets.

```
!
source-bridge tcp-queue-max 150
!
```

Configuring Remote Source-Routing over Point-to-Point Serial

To configure a remote source-route bridge to use a point-to-point serial line, follow these steps:

Step 1: Define a ring group. Every Cisco router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. These other router/bridges are referred to as peers.

Step 2: List the interfaces over which you will be sending source-route bridged traffic with the **source-bridge remote-peer** command. The interfaces must be serial, and must use the HDLC encapsulation.

Step 3: Configure the Token Ring interfaces for source-route bridging. The value of the target ring parameter for the **source-bridge** commands should be the ring group number.

The **source-bridge remote-peer** global configuration command has the following syntax when used to specify a point-to-point serial connection (the negative form of the command is also included):

source-bridge remote-peer *ring-group* **interface** *interface-name* [**lf** *size*]

no source-bridge remote-peer *ring-group* **interface** *interface-name*

This command is used to identify the interface over which to send source-route bridged traffic to another Cisco router/bridge in our ring group. The interface must be serial, and must use the HDLC encapsulation.

The keyword **lf** specifies the maximum size frame to be sent to this remote peer. The router negotiates all transit routes down to this size or lower. This argument is useful in preventing time-outs in end hosts, by reducing the amount of data they have to transmit in a fixed interval. For example, in some networks containing slow links, it would be impossible to transmit an 8K frame and receive a response within a few seconds. These are fairly standard defaults for an application on a 16 megabit Token Ring. If the frame size is lowered to 516 bytes, then only 516 bytes must be transmitted and a response received in 2 seconds. This is a much easier accomplishment in a network with slow links. The legal values for this argument are 516, 1470, 2052, 4472, 8144, 11454, and 17,800 bytes.

Note: It is possible to mix both serial and TCP transport methods within the same ring-group.

Example:

The following is an example configuration of a router/bridge configured for remote source-route bridging using both the serial and TCP transport methods.

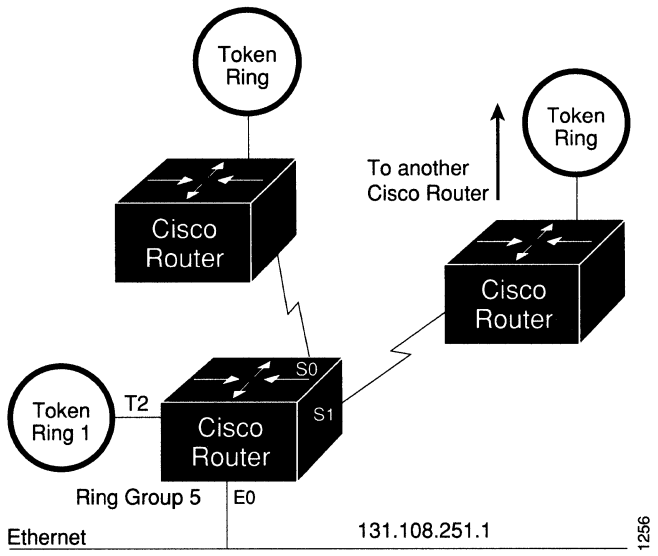


Figure 21-9 Remote Source-Route Bridge Using Both Serial and TCP Transport Methods

```
!
source-bridge ring-group 5
source-bridge remote-peer 5 interface serial0
source-bridge remote-peer 5 tcp 131.108.254.6
source-bridge remote-peer 5 tcp 131.108.251.1
!
interface tokenring 0
source-bridge 1 1 5
source-bridge spanning
!
interface ethernet 0
ip address 131.108.251.1 255.255.255.0
!
```

Note: The two peers using the serial transport method will only function correctly if there are Cisco router/bridges at the other end of serials 0 and 1 and they have been configured to use the serial transport. The peers must also belong to the same ring group.

Configuring the Largest Sized Frame

Use the **source-bridge largest-frame** global configuration command to configure the largest frame size that is used to communicate with any peers in this ring group. The full syntax of the command follows:

source-bridge largest-frame *ring-group size*

source-bridge largest-frame *ring-group*

The argument *ring-group* is the ring group number; the argument *size* is the maximum frame size.

The router negotiates all transit routes down to this size or lower. This argument is useful in preventing time-outs in end hosts, by reducing the amount of data they have to transmit in a fixed interval. For example, in some networks containing slow links, it would be impossible to transmit an 8K frame and receive a response within a few seconds. These are fairly standard defaults for an application on a 16 megabit Token Ring. If the frame size is lowered to 516 bytes, then only 516 bytes must be transmitted and a response received in 2 seconds. This is a much easier accomplishment in a network with slow links. The legal values for this argument are 516, 1470, 2052, 4472, 8144, 11454, and 17,800 bytes.

Configuring a Proxy Explorer

Cisco has implemented the *proxy explorer* function to allow the Cisco source-route bridge to respond to a particular destination node. Proxy explorers can be used to limit the amount of explorer traffic propagating through the source-bridge network, especially across low bandwidth serial lines. The use of proxy explorer is most useful for multiple connections to a single mode.

The following conditions must be met in order for a proxy response to occur:

- The destination node must be in the RIF cache.
- The destination node must not be on the same ring as the explorer.
- The explorer packet must be an IEEE 802.2 XID or TEST packet.
- The packet cannot be from the IBM Token Ring LAN Network Manager source SAP.

If all of the above conditions are met, the Cisco source-route bridge will turn the packet around, append the appropriate RIF, and reply to the source node.

Use the **source-bridge proxy-explorer** interface subcommand to configure the interface to respond to any explorer packets that meet the conditions described above. The command has this syntax:

source-bridge proxy-explorer

no source-bridge proxy-explorer

The default is to *not* respond with proxy explorer packets.

Example:

These commands configure the Cisco router/bridge to use proxy explorers on interface Token Ring 0.

```
!
interface tokenring 0
source-bridge proxy-explorer
!
```

Configuring Administrative Filtering

Source routing bridges normally filter datagrams according to the routing information contained in the datagram. That is, a bridge will not forward a datagram back to its originating network segment or any other network segment that the datagram has already traversed. Further types of filtering (administrative filtering) can be specified by the network manager.

Administrative filtering can be done by:

- Token Ring address
- Protocol type—IEEE 802.5 or SNAP
- Token Ring Vendor code

Filtering by Token Ring address or vendor code will cause no significant performance penalty. However, performance will be significantly affected when filtering by protocol type. A list of SNAP (Ethernet) type codes is provided in Appendix C.

Administrative Filtering by Protocol Type

The access list mechanism permits filtering by protocol type. Use the bridge **access-list** command to specify an element in an access list. The order in which **access-list** commands are entered into the system affects the order in which the access conditions are checked. Each condition is tested in succession. A matching condition is then used to execute a permit or deny decision. If no conditions match, then a deny decision is reached.

Note: If a *single condition* is to be denied, then there must be an **access-list** command that permits *everything* as well, or *all* access will be denied.

Configuring Protocol Type Access Lists

Use the **bridge access-list** global configuration command to configure the access list mechanism for filtering frames by protocol type. The command has this syntax:

access-list *list* {**permit** | **deny**} *type-code wild-mask*

The argument *list* is a user-selectable number in the range 200 – 299, inclusive, that identifies the list.

The keyword **permit** permits the frame; the keyword **deny** denies the frame.

The argument *type-code* is a 16-bit hexadecimal number written with a leading 0x, for example, 0x6000. Specify either a Link Service Access Point (LSAP) type code for 802.5-encapsulated packets, or a SNAP type code for SNAP-encapsulated packets. (LSAP, sometimes called SAP, refers to the type codes found in the DSAP and SSAP fields of the 802.2 header.)

The argument *wild-mask* is another 16-bit hexadecimal number whose ones bits correspond to bits in the type-code argument that should be ignored when making a comparison. (A mask for a DSAP/SSAP pair should always be 0x0101. This is because these two bits are used for purposes other than identifying the SAP code.

Example:

In this example, the access list permits only Novell frames (LSAP 0xE0E0) and filters out all other frame types. This set of access lists would be applied to an interface via the **source-bridge input-lsap list** or **source-bridge input-lsap list** command (described in following sections).

```
!
access-list 201 permit 0xE0E0 0x0101
access-list 201 deny  0x0000 0xFFFF
!
```

Combine the DSAP/LSAP fields into one number to do LSAP filtering: for example, 0x E0E0 — not 0xE0. (Note that the deny condition specified in the above example is not required; access lists have an implicit deny as the last statement. Adding this statement can serve as a useful reminder, however.)

The following access list filters out only SNAP type codes assigned to DEC (0x6000 through 0x6007) and lets all other types pass. This set of access lists would be applied to an interface via the **source-bridge input-type list** or **source-bridge output-type-list** command (described in a following section).

```
!
access-list 202 deny     0x6000 0x0007
access-list 202 permit   0x0000 0xFFFF
!
```

Note: Use the last item of an access list to specify a default action: for example, to permit everything else or to deny everything else. If nothing else in the access list matches, then the default action is to deny access—that is, filter out all other type codes.

Type code access lists will negatively affect system performance by greater than 30 percent. Therefore, Cisco Systems recommends keeping the lists as short as possible and using wild card bit masks whenever possible.

Filtering IEEE 802.5 Frames on Input

You can filter IEEE 802.5-encapsulated packets on input. The access list specifying the type codes to be filtered is given by this variation of the **source-bridge** interface subcommand:

source-bridge input-lsap-list *list*

The argument *list* is the access list number. This access list is applied to all IEEE 802.5 frames received on that interface prior to the source-routing process.

Example:

This command specifies access list 203.

```
!
source-bridge input-lsap-list 203
!
```

Filtering IEEE 802.5 Frames on Output

The software allows you to filter IEEE 802.5-encapsulated packets on output. The access list specifying the type codes to be filtered is given by this variation of the **source-bridge** interface subcommand:

source-bridge output-lsap-list *list*

The argument *list* is the access list number. This access list is applied just before sending out a frame to an interface.

Example:

This command specifies access list 251.

```
!
source-bridge output-lsap-list 251
!
```

Filtering SNAP Frames on Input

To filter SNAP-encapsulated packets on input, use the access list specifying the type codes to be filtered with this variation of the **source-bridge** interface subcommand:

source-bridge input-type-list *list*

The argument list is the access list number. This access list is then applied to all SNAP frames received on that interface prior to the source routing process.

Example:

This command specifies access list 202.

```
!
source-bridge input-type-list 202
!
```

Filtering SNAP Frames on Output

To filter SNAP-encapsulated on output, use the access list specifying the type codes to be filtered. This is entered with this variation of the **source-bridge** interface subcommand:

source-bridge output-type-list *list*

The argument *list* is the access list number. This access list is then applied just before sending out a frame to an interface.

Note: Input and output type code filtering on the same interface reduces performance and is not recommended.

Access lists for token ring- and IEEE 802.5-encapsulated packets affect only source-route bridging functions. Such access lists do *not* interfere with protocols that are being routed.

Example:

The following example allows only AppleTalk Phase 2 packets to be source-route bridged between token rings 0 and 1, and allows Novell packets only to be source-route bridged between token rings 2 and 3.

```
source-bridge ring-group 5
!
interface tokenring 0
ip address 131.108.1.1 255.255.255.0
source-bridge 1000 1 5
source-bridge spanning
source-bridge input-type-list 202
!
interface tokenring 1
ip address 131.108.11.1 255.255.255.0
source-bridge 1001 1 5
```

```
source-bridge spanning
source-bridge input-type-list 202
!
interface tokenring 2
ip address 131.108.101.1 255.255.255.0
source-bridge 1002 1 5
source-bridge spanning
source-bridge input-lsap-list 203
!
interface tokenring 3
ip address 131.108.111.1 255.255.255.0
source-bridge 1003 1 5
source-bridge spanning
source-bridge input-lsap-list 203
!
! SNAP type code filtering
! permit ATp2 data (0x809B)
! permit ATp2 AARP (0x80F3)
access-list 202 permit 0x809B 0x0000
access-list 202 permit 0x80F3 0x0000
access-list 202 deny 0x0000 0xFFFF
!
! LSAP filtering
! permit IPX (0xE0E0)
access-list 203 permit 0xE0E0 0x0101
access-list 203 deny 0x0000 0xFFFF
```

Note that it is **not** necessary to check for an LSAP of 0xAAAA when filtering SNAP encapsulated AppleTalk packets, because for source-route bridging, the use of type filters *implies* SNAP encapsulation.

Administrative Filtering by Vendor Code or Address

To configure administrative filtering by vendor code or address, define access lists which look for Token Ring addresses or for particular vendor codes for administrative filtering. No noticeable performance will be lost in using these access lists. The lists can be of indefinite length.

Configuring Vendor Code Access Lists

To configure a vendor code access list, use the global bridge **access-list** command for IEEE 802.5 address access lists. The command has the following form:

access-list *list* {**permit** | **deny**} *address mask*

no access-list *list* {**permit** | **deny**} *address mask*

The argument *list* is an integer from 700 to 799, inclusive, and *address* and *mask* are 48-bit token ring addresses written in dotted triplet form. The ones bits in *mask* are the bits to be ignored in *address*. See the section "Filtering Destination Addresses" for an example of command use.

Note that for source address filtering, the mask should always have the high order bit set. This is because the IEEE 802.5 standard uses this bit to indicate whether a RIF is present, and not as part of the source address.

Filtering Source Addresses

To configure filtering on IEEE 802.5 source addresses, assign an access list to a particular interface for filtering the token ring or IEEE 802.5 source addresses. Use this variation of the **source-bridge** interface subcommand to do this (following syntax includes negative form of the command):

source-bridge input-address-list *list*

no source-bridge input-address-list *list*

The argument *list* is the access list number. See the section "Filtering Destination Addresses" for an example of command use.

Filtering Destination Addresses

To configure filtering on IEEE 802.5 destination addresses, assign an access list to a particular interface for filtering the token ring or IEEE 802.5 destination addresses. Use this variation of the **source-bridge** interface subcommand to do this (negative form of the command included):

source-bridge output-address-list *list*

no source-bridge output-address-list *list*

The argument *list* is the access list number.

Example:

To disallow the bridging of token ring packets of all IBM workstations on token ring 1, use this sample configuration. Software assumes that all such hosts have token ring addresses with the vendor code 1000.5A00.0000. The first line of the access list denies access to all IBM workstations while the second line permits everything else. Then, the access list can be assigned to the input side of Token Ring 1.

```
access-list 700 deny 1000.5A00.0000   8000.00FF.FFFF
access-list 700 permit 0000.0000.0000   FFFF.FFFF.FFFF
interface token ring 1
source-bridge input-address-list 700
```

Configuring NETBIOS Access Control Filtering

NETBIOS is the interface used by the IBM Token Ring/PC Network Interconnect Program to transmit messages between stations (typically IBM PCs) on a Token Ring network. NETBIOS allows messages to be exchanged between the stations using a name rather than a station address. Each station knows its name and is responsible for knowing the names of other stations on the network.

The Cisco bridging software provides for configuring access control filters for packets transmitted across a Token Ring bridge using the NETBIOS interface. Two types of filters may be configured, one on source and destination station names, and one on arbitrary byte patterns in the packet itself.

Configuring Access Control Using Station Names

To configure access control using station names, follow these steps:

Step 1: Define the access list name and specify the access condition, either permit or deny.

Step 2: Specify the direction of the message to be filtered on the interface. The choices are incoming or outgoing messages.

Configuration Tips and Notes

Keep the following notes in mind as you configure NETBIOS access control:

- The access lists are scanned in the order they are entered.
- There is no way to put a new access list entry in the middle of an access list. All new additions to existing NETBIOS access lists are placed at the end of the existing list.
- The case of the letters you use to enter arguments is important. The software makes a literal translation, so that a lowercase "a" is different from an uppercase "A." (Most nodes are named in uppercase letters.)
- You may have both a host NETBIOS access list and byte NETBIOS access list with the same name. The two lists are identified as unique and bear no relationship to each other.
- The names in the access lists are compared with the source name field for NETBIOS commands 00 and 01 (ADD_GROUP_NAME_QUERY and ADD_NAME_QUERY) and destination name field for NETBIOS commands 08 and 0A (DATAGRAM and NAME_QUERY).

Note: The NETBIOS access filters are implemented to minimize their performance hit by not having them examine all packets. Rather, the filters examine a select few which are required to allow new NETBIOS client/server connections from forming and existing in the long term, thereby effectively stopping new access and load across the router. However, existing sessions will not be stopped immediately with the application of a new access filter. All new sessions will be filtered by the filter, but the existing sessions could stay for some time.

Assigning the Station Access List Name

The NETBIOS station access list contains the station name with which to match, along with a permit or deny condition. Use the **netbios access-list host** global configuration command to assign the name of the access list to a station or set of stations on the network. The full command syntax follows:

netbios access-list host *name* {**permit**|**deny**} *pattern*

no netbios access-list host *name* {**permit**|**deny**} *pattern*

The argument *name* is the name of the access list being defined.

The argument *pattern* is a set of characters. The characters can be the name of the station, or a combination of characters and pattern matching symbols that establish a pattern for a set of NETBIOS station names. This can be especially useful when stations have names with the same characters, such as a prefix. Table 21-1 explains the pattern matching symbols that can be used.

Table 21-1 Station Name Pattern Matching Characters

Character	Action
*	Used at the end of a string to match any character or string of characters.
?	Matches any single character.

The **no netbios access-list host** command removes an entire list, or just a single entry from a list, depending upon the argument given for *pattern*.

Examples:

This command specifies a full station name to match.

```
!
netbios access-list host marketing permit ABCD
!
```

This command specifies a prefix where the pattern matches any name beginning with the characters DEFG. Note that the string DEFG itself is included in this condition.

```
!
netbios access-list host marketing deny DEFG*
!
```

This command permits any station name with the letter W as the first character and the letter Y as the third character in the name. The second and fourth letters in the name can be any character. This example would allow stations named WXYZ and WAYB; however, stations named WY and WXY would not be included in this statement, as the ? must match some specific character in the name.

```
!
netbios access-list host marketing permit W?Y?
!
```

This example illustrates how to combine wildcard characters:

```
!
netbios access-list host marketing deny AC?*
!
```

The command specifies that the marketing list deny any name beginning with AC that is at least three characters in length (the ? would match any third character). The string ACBD and ACB would match, but the string AC would not.

This command removes the entire marketing NETBIOS access list.

```
!
no netbios access-list host marketing
!
```

To remove single entries from the list, use a command such as the following:

```
!
no netbios access-list host marketing deny AC?*
!
```

This example removes only the list that filters station names with the letters AC at the beginning of the name.

Keep in mind that the access lists are scanned in order. In this example the first list denies all entries beginning with the letters ABC, including one named ABCD. This voids the second command because the entry permitting a name with ABCD comes *after* the entry denying it.

```
!
netbios access-list host marketing deny ABC*
netbios access-list host marketing permit ABCD
!
```

Specifying a Station Access List Filter on Incoming Messages

To define an access list filter on incoming messages, use the **netbios input-access-filter host** interface subcommand. The full command syntax follows:

netbios input-access-filter host *name*

no netbios input-access-filter host *name*

The argument *name* is the name of a NETBIOS access filter previously defined with one or more of the **netbios access-list host** global configuration commands.

Use the **no netbios input-access-filter host** command with the appropriate argument to remove the entire access list.

Example:

These commands filter packets coming into Token Ring unit 1 using the NETBIOS access list named *marketing*.

```
interface token 1
netbios input-access-filter host marketing
```

Specifying a Station Access List Filter on Outgoing Messages

To define an access list filter on outgoing messages, use the **netbios output-access-filter host** interface subcommand. The full command syntax follows.

netbios output-access-filter host *name*

no netbios output-access-filter host *name*

The argument *name* is the name of a netbios access filter previously defined with one or more of the **netbios access-list** global configuration commands.

Use the **no netbios output-access-filter host** command to remove the entire access list.

Example:

These commands filter packets leaving Token Ring unit 1 using the NETBIOS access list named `engineering`.

```
!
interface token 1
netbios output-access-filter host engineering
!
```

Configuring Access Control Using a Byte Offset

To configure access control using a byte offset, follow these steps:

Step 1: Define the access list name and specify the access condition, either permit or deny.

Step 2: Specify the direction of the message to be filtered on the interface. The choices are incoming or outgoing messages.

Configuration Tips and Notes

Keep the following notes in mind while configuring access control using a byte offset:

- The access lists are scanned in the order they are entered.
- There is no way to put a new access list entry in the middle of an access list. All new additions to existing NETBIOS access lists are placed at the end of the existing list.
- When an access list entry has an offset plus the length of the pattern that is larger then the packet's length, the entry will not make a match for that packet.
- You may have both a host NETBIOS access list and byte NETBIOS access list with the same name. The two lists are identified as unique and bear no relationship to each other.
- As bytes access lists allow arbitrary byte offsets into packets, these access filters can have a significant impact on the amount of packets per second transiting across the bridge. They should be used only when situations absolutely dictate their use.

Assigning the Bytes Access List Name

The NETBIOS byte offset access list contains a series of offsets and hexadecimal patterns with which to match byte offsets in NETBIOS packets. Use the **netbios access-list bytes** global configuration command to define the offset and patterns. The full command syntax follows:

netbios access-list bytes *name* {**permit** | **deny**} *offset pattern*

no netbios access-list bytes *name* {**permit** | **deny**} *offset pattern*

The argument *name* is the name of the access list being defined.

The argument *offset* is a decimal number indicating the number of bytes into the packet where the byte comparison should begin. An offset of zero points to the beginning of the NETBIOS delimiter string (0xffef) at the start of each NETBIOS packet.

The argument *pattern* is a hexadecimal string of digits representing a byte pattern. The argument *pattern* must conform to certain conventions. These conventions follow.

Byte Offset Pattern Matching

- The byte pattern must be an even number of hex digits in length.

  The byte pattern in the following example is legal:

  ```
  netbios access-list bytes marketing permit 3 0xabcd
  ```

  But the byte pattern in this example would not be accepted:

  ```
  netbios access-list bytes marketing permit 3 0xabc
  ```

- The byte pattern must be no more than 16 bytes (32 hexadecimal digits) in length.

  The byte pattern in this example would *not* be permitted:

  ```
  netbios access-list bytes marketing permit 3 00112233445566778899aabbccddeeff00
  ```

- You can specify a wild card character in the byte string indicating that the value of that byte does not matter in the comparison. This is done by specifying two asterisks (**) in place of digits for that byte.

  For example, the following command would match 0xabaacd, 0xab00cd, and so on.

  ```
  netbios access-list bytes marketing permit 3 0xab**cd
  ```

Examples:

This command deletes the entire marketing NETBIOS access list named marketing.

```
!
no netbios access-list bytes marketing
!
```

This command removes a single entry from the list:

```
!
no netbios access-list bytes marketing deny 3 0xab**cd
!
```

Remember that, as with all Cisco access lists, the NETBIOS access lists are scanned in order.

In the following example:

```
!
netbios access-list bytes marketing deny 3 0xab
netbios access-list bytes marketing permit 3 0xabcd
!
```

The first line serves to deny all packets with a byte pattern starting in offset 3 of `0xab`. However, this denial would also include the pattern `0xabcd` because the entry permitting the pattern `0xabcd` comes *after* the first entry.

Specifying a Bytes Access List Filter on Incoming Messages

To define an access list filter on incoming messages, use the **netbios input-access-filter bytes** interface subcommand. The full command syntax follows:

netbios input-access-filter bytes *name*

no netbios input-access-filter bytes *name*

The argument *name* is the name of a NETBIOS access filter previously defined with one or more of the **netbios access-list bytes** global configuration commands.

Use the **no netbios input-access-filter bytes** command with the appropriate name to remove the entire access list.

Example:

These commands illustrate how to specify a filter on packets coming into Token Ring unit 1 of the NETBIOS access list named *marketing*.

```
!
interface token 1
netbios input-access-filter bytes marketing
!
```

Specifying a Bytes Access List Filter on Outgoing Messages

To define an access list filter on outgoing messages, use the **netbios output-access-filter bytes** interface subcommand. The full command syntax follows:

netbios output-access-filter bytes *name*

no netbios output-access-filter bytes *name*

The argument *name* is the name of a NETBIOS access filter previously defined with one or more of the **netbios access-list bytes** global configuration commands.

Use the **no netbios output-access-filter bytes** command to remove the entire access list.

Example:

These commands filter packets leaving Token Ring unit 1 using the NETBIOS access list named *engineering*.

```
!
interface token 1
netbios output-access-filter bytes engineering
!
```

Interoperability Issues

This section describes known interoperability issues between Cisco router/bridges and specific Token Ring implementations.

PC/3270 Emulation Software

The IBM PC/3270 emulation program version 3.0 does not properly send packets over a Cisco source-route bridge.

Note: This problem exists only when using the older four-megabit (CSC-R) Token Ring card.

The Cisco implementation confuses the IBM implementation into not looking beyond the local ring for the remote host.

source-bridge old-sna

no source-bridge old-sna

The interface subcommand **source-bridge old-sna** rewrites the RIF headers of explorer packets send by the PC/3270 emulation program to go beyond the local ring. The **no source-bridge old-sna** command disables this compatibility mode.

Examples:

These commands enable RIF rewriting.

```
!
interface tokenring 0
source-bridge old-sna
!
```

These commands disable RIF rewriting.

```
!
interface tokenring 0
no source-bridge old-sna
!
```

TI MAC Firmware

There is a known defect in earlier versions of the Texas Instruments' (TI) Token Ring MAC firmware. This implementation is used by Proteon, Apollo, and IBM RTs. A host using a MAC address whose first two bytes are zeros (such as a Cisco router/bridge) will not properly communicate with hosts using that version of TI firmware.

Cisco provides two solutions. The first involves installing a static RIF entry for every faulty node with which the router communicates. If there are many such nodes on the ring this may not be practical. The second solution involves setting the MAC address of the Cisco Token Ring to a value that works around the problem.

The interface subcommand **mac-address** sets the MAC layer address, and has this syntax:

mac-address *ieee-address*

The argument ieee-address is a 48-bit IEEE MAC address written as a dotted triple of four digit hexadecimal numbers.

This command forces the use of a different MAC address on the specified interface, thereby avoiding the TI MAC firmware problem. It is up the network administrator to ensure that no other host on the network is using that MAC address.

Example:

This example command sets the MAC layer address where *xx.xxxx* is an appropriate second half of the MAC address to use.

```
!
interface tokenring 0
mac-address 5000.5axx.xxxx
!
```

Spurious Frame-Copied Errors

An IBM 3174 controller can be configured to report frame-copied errors to LANmanager software. These errors indicate that another host is responding to the MAC address of the 3174 controller. If a Cisco router/bridge is present on the same ring configured for source-route bridging, however, then the likely problem is the 3174 noticing that the Cisco router/bridge is setting the Address Recognized and Frame Copied bits. There is not a problem, merely a warning. No data is being lost.

Both the 3174 and the LANmanager software can be configure to ignore frame-copied errors.

Note: This problem exists only when using the older 4 megabit (CSC-R) Token Ring card.

Source-Route Bridging Configuration Examples

The Token Ring system software is written such that a minimum of configuration is the normal case. A Token Ring equipped router is by default a single-ring host. Source bridging is off by default. Following are configuration examples that you may use to make your own configuration files. Refer to previous illustrations in this chapter for a visual orientation of the networks illustrated here.

Basic Token Ring Configuration

This example configures the Cisco router/bridge for IP and Novell IPX routing.

Example:

```
!
novell routing
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
novell network 32
multiring all
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
novell network 461
multiring all
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
novell network 95
!
```

Basic Token Ring Source Bridge Configuration

In this partial configuration, the source bridge is turned on, IP is routed, and all other protocols are bridged.

Example:

```
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
!
```

Source Bridge Only Configuration

In this partial configuration, all protocols are bridged including IP. Because IP is being bridged, the system has only one IP address.

Example:

```
!
no ip routing
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
!
interface TokenRing 1
ip address 131.108.129.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
!
interface Ethernet 0
ip address 131.108.129.2 255.255.255.0
!
```

Other Protocols Routed at the Same Time

In this configuration IP, XNS, and Novell are all being routed while all others will be bridged between rings. While not strictly necessary, the Novell and XNS network numbers are set consistently with the IP subnetwork numbers. This makes the network easier to maintain.

Example:

```
!
xns routing 0000.0C00.02C3
!
novell routing 0000.0C00.02C3
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
xns network 129
novell network 129
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
xns network 130
novell network 130
source-bridge 130 1 129
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
xns network 2
novell network 2
!
```

Remote Source-Route Bridge with Simple Reliability

In the following sample, the basic two-port remote source-route bridge configuration is extended to include both reliability and load sharing. The two routers are connected by two serial lines. When both serial lines are up, traffic is split between them, effectively combining the bandwidth of the connections. If either one of the serial lines goes down, all traffic is routed to the remaining line with no disruption. This happens transparently with respect to the end connections, unlike other source-route bridges which would abort those connections. This configuration is shown in Figure 21-10.

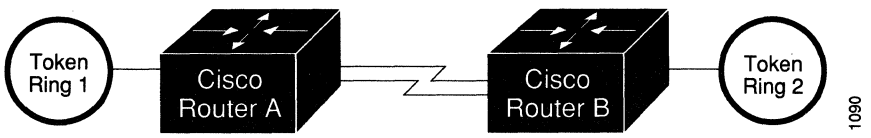


Figure 21-10 Remote Source-Route Bridge—Simple Reliability

Configuration for Router/Bridge A:

```
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 204.31.7.1
source-bridge remote-peer 5 tcp 204.31.8.1
!
interface TokenRing 0
ip address 204.31.7.1 255.255.255.0
source-bridge 1 1 5
source-bridge spanning
multiring all
!
interface Serial 0
ip address 204.31.9.1 255.255.255.0
!
interface Serial 1
ip address 204.31.10.1 255.255.255.0
!
router igrp 109
network 204.31.7.0
network 204.31.9.0
network 204.31.10.0
!
hostname RouterA
!
```

Configuration for Router/Bridge B:

```
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 204.31.7.1
source-bridge remote-peer 5 tcp 204.31.8.1
!
interface TokenRing 0
ip address 204.31.8.1 255.255.255.0
source-bridge 2 1 5
source-bridge spanning
multiring all
!
interface Serial 0
ip address 204.31.9.2 255.255.255.0
!
interface Serial 1
ip address 204.31.10.2 255.255.255.0
!
router igrp 109
network 204.31.8.0
network 204.31.9.0
network 204.31.10.0
!
hostname RouterB
!
```

Remote Source-Route Bridge—Complex Configuration

In the following example, a triangular configuration is used to provide the maximum reliability with minimal cost. In addition, one of the links is doubled to gain better bandwidth. In addition to IP and source-route traffic, AppleTalk is also being routed between all the sites.

This configuration is shown in Figure 21-11.

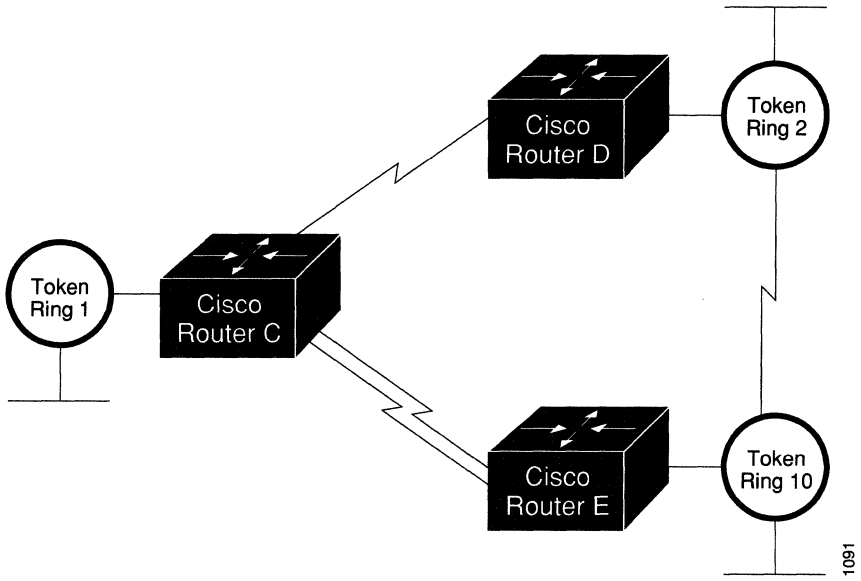


Figure 21-11 Remote Source-Route Bridge—Complex Configuration

Configuration for Router/Bridge C:

```
!
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
source-bridge remote-peer 5 tcp 132.21.2.6
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.1.1 255.255.255.0
source-bridge 1 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.4.25 255.255.255.0
appletalk address 4.25
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.16.1 255.255.255.0
appletalk address 16.1
appletalk zone Twilight
```

```
!
interface Serial 1
ip address 131.21.17.1 255.255.255.0
appletalk address 17.1
appletalk zone Twilight
!
interface Serial 2
ip address 131.21.18.1 255.255.255.0
appletalk address 18.1
appletalk zone Twilight
!
router igrp 109
network 131.21.0.0
!
hostname RouterC
!
```

Configuration for Router/Bridge D:

```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
source-bridge remote-peer 5 tcp 132.21.2.6
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.2.6 255.255.255.0
source-bridge 2 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.5.1 255.255.255.0
appletalk address 5.1
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.16.2 255.255.255.0
appletalk address 16.2
appletalk zone Twilight
!
interface Serial 1
ip address 131.21.19.1 255.255.255.0
appletalk address 19.1
appletalk zone Twilight
!
router igrp 109
network 131.21.0.0
!
hostname RouterD
!
```

Configuration for Router/Bridge E:

```
!
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
source-bridge remote-peer 5 tcp 132.21.2.6
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.10.200 255.255.255.0
source-bridge 10 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.7.1 255.255.255.0
appletalk address 7.1
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.19.2 255.255.255.0
appletalk address 19.2
appletalk zone Twilight
!
interface Serial 1
ip address 131.21.17.2 255.255.255.0
appletalk address 17.2
appletalk zone Twilight
!
interface Serial 2
ip address 131.21.18.2 255.255.255.0
appletalk address 18.2
appletalk zone Twilight
!
router igrp 109
network 131.21.0.0
!
hostname RouterE
!
```

Maintaining the Source-Route Bridge

Use this EXEC command to maintain the source-route bridge cache.

clear rif-cache

The **clear rif-cache** command clears the entire RIF cache.

Monitoring the Source-Route Bridge

Use the EXEC commands described in this section to obtain displays of activity on the source-route bridge.

Displaying the RIF Cache

The **show rif** EXEC command displays the current contents of the RIF cache. Enter this command at the EXEC prompt:

show rif

The following is a sample display of **show rif**:

```
Codes: * interface, - static, + remote
Hardware Addr  How   Idle (min)  Routing Information Field
5C02.0001.4322 rg5           -   0630.0053.00B0
5A00.0000.2333 TR0           3   08B0.0101.2201.0FF0
5B01.0000.4444 -             -   -
0000.1403.4800 TR1           0   -
0000.2805.4C00 TR0           *   -
0000.2807.4C00 TR1           *   -
0000.28A8.4800 TR0           0   -
0077.2201.0001 rg5          10   0830.0052.2201.0FF0
```

Table 21-2 RIF Cache Display Field Description

Field	Description
Hardware Addr	Lists the MAC-level addresses.
How	Describes how the RIF has been learned. Possible values include a ring group (rg), or interface (TR).
Idle	Indicates how long, in minutes, since the last response was received directly from this node.
Routing Information Field	Lists the RIF.

Entries marked with an asterisk (*) are the router/bridge's interface addresses. Entries marked with a dash (–) are static entries. Entries with a number denote cached entries. If the RIF timeout is set to something other than the default of 15 minutes, the timeout is displayed at the top of the display.

Displaying the Current Bridge Configuration

The **show source-bridge** EXEC command displays the current source bridge configuration and miscellaneous statistics. Enter this command at the EXEC prompt:

show source-bridge

The following is sample output:

```
Local Interfaces:          max       receive            transmit
       srn bn  trn rg px sp rd      cnt:bytes          cnt:bytes       drops
 TR0    1  1    5  *  *  *  8        5:233              0:0               0
 TR1    2  1    5  *  *  *  8        4:224              0:0               0
Ring Group 5:
This peer: TCP 131.108.2.29.
Peers:               state    pkts_rx    pkts_tx    drops     TCP q len
 TCP 131.108.2.68    open           0          0        0         0
 TCP 131.108.2.29    -              0          0        0         0
 TCP 131.108.161.2   open           0          0        0         0
Rings:
    bridge 1  ring 1    local  TokenRing0          forwards: 0
    bridge 1  ring 2    local  TokenRing1          forwards: 0
    bridge 2  ring 102  remote TCP 131.108.161.2   forwards: 0
    bridge 2  ring 101  remote TCP 131.108.161.2   forwards: 0
    bridge 1  ring 6    remote TCP 131.108.2.68    forwards: 0
    bridge 1  ring 7    remote TCP 131.108.2.68    forwards: 0
```

Table 21-3 Current Bridge Configuration Field Descriptions

Field	Description
Local Interfaces:	Description of local interfaces.
max	Maximum routing descriptor length.
receive	Packets:bytes received on interface for source bridging.
transmit	Packets:bytes transmitted on interface for source bridging.
srn	Ring number of this Token Ring.
bn	Bridge number of this router, for this ring.
trn	Indicates the group in which the interface is configured.
rg	Indicates a ring group, noted by an asterisk (*).
px	Indicates an interface that can respond with proxy explorers, noted by an asterisk (*).
sp	Indicates a spanning explorer enabled on the interface, noted by an asterisk (*).
Ring Group:	Describes the ring group.
This peer:	Lists the address and address type of this peer.
Peers:	Lists the addresses and address types of the ring group peers.
state	Lists the current state of the peer, open or closed. A hyphen indicates this router.
pkts_rx	Lists the number of packets received.
pkts_tx	Lists the number of packets transmitted.
drops	Lists the number of dropped packets.
TCP q len	Lists the current TCP backup queue length.
Rings:	Describes the ring groups. Information displayed includes the bridge groups, ring groups, whether the group is local or remote, the address or interface type, and the number of packets forwarded.

Displaying Information about the Token Ring Interface

The EXEC command **show controllers token** displays internal state information about the token ring interfaces in the system. Enter this command at the EXEC prompt:

show controllers token

The command displays the versions number of the Token Ring firmware, and the source-bridge capability of the interface. These statistics are most useful to Cisco personnel for system troubleshooting.

Displaying Token Ring Interface Statistics

The **show interface** EXEC command display for Token Rings provides high-level statistics about the state of source bridging for a particular interface. Enter this command at the EXEC prompt:

show interface

Refer to the section "Token Ring Interface Support" in Chapter 6 for a description of the information this command displays.

Debugging the Source-Route Bridge

Use the privileged-level EXEC **debug** commands described in this section to track activity in the source-route bridge network. Generally, these commands will be executed when working with Cisco Customer Engineering, to track system problems. For each **debug** command there is a corresponding **debug** command that stops the output.

The **debug rif** command provides informational displays for entries entering and leaving the RIF cache. Enter this command at the EXEC prompt.

debug rif

Descriptions of the messages with that can be generated with **debug rif** follow.

```
RIF: L Sending XID for <address>
```

The router/bridge wanted to send a packet to `<address>` but did not find it in the RIF cache. It sent an XID explorer packet to determine which RIF it should use. The attempted packet is dropped.

```
RIF: L No buffer for XID to <address>
```

Similar to the previous display, however a buffer in which to build the XID packet could not be obtained.

```
RIF: U chk <address>[<rif>]
```

A packet is being checked to see if its MAC address is already in the RIF cache. `<interface>` is either the physical interface the packet arrived on, or the `static/remote` interface. `<ring group>`, a number, is only valid if this RIF check is for a remote packet. `<code>` denotes the kind of RIF entry being checked; this is an internal code and is not documented.

```
RIF: U remote rif too small [<rif>]
```

A packet's RIF was too short to be valid.

```
RIF: U rej <address>too big [<rif>]
```

A packet's RIF exceeded the maximum size allowed and was rejected. The maximum size is 18 bytes.

```
RIF: U upd interface <address>
```

The RIF entry for this router/bridge's interface has been updated.

```
RIF: U ign <address>interface update
```

A RIF entry that would have updated an interface corresponding to one of this routers interfaces.

```
RIF: U upd <address>[<rif>]
```

The RIF entry for <address> has been found and updated.

```
RIF: U add <address>[<rif>]
```

The RIF entry for <address> has been added to the RIF cache.

```
RIF: U no memory to add rif for <address>
```

No memory to add a RIF entry for <address>.

```
RIF: removing rif entry for <address>, type <code>
```

The RIF entry for <address> has been forcibly removed.

```
RIF: flushed <address>
```

The RIF entry for <address> has been removed because of a RIF cache flush.

```
RIF: expired <address>
```

The RIF entry for <address> has been aged out of the RIF cache.

```
RIF: rcvd XID response from <address>
```

An XID response from <address> was inserted into the RIF cache.

```
RIF: rcvd TEST response from <address>
```

A TEST response from <address> was inserted into the RIF cache.

The **debug source-bridge** command provides informational displays of source bridging activity. Enter this command at the EXEC prompt:

debug source-bridge

Samples of messages displayed include the following:

In the following sample display, SRBn or RSRBn denotes a message associated with interface Token Ring *n*. An *n* of 99 denotes the remote side of the network.

```
SRBn: no path, s: <src MAC addr>d: <dst MAC addr>rif: <rif>
```

A bridgeable packet came in on interface Token Ring n but there was no where to send it. This is most likely a configuration error. For example, an interface has source bridging turned on but it is not connected to another source bridging interface or a ring group.

```
SRBn: direct forward (srn <ring>bn <bridge>trn <ring>)
```

A bridgeable packet has been forwarded from Token Ring n to the target ring. The two interfaces are directly linked.

```
SRBn: br dropped proxy XID,  <address>for <address>, wrong vring (rem)
SRBn: br dropped proxy TEST, <address>for <address>, wrong vring (rem)
SRBn: br dropped proxy XID,  <address>for <address>, wrong vring (local)
SRBn: br dropped proxy TEST, <address>for <address>, wrong vring (local)
SRBn: br dropped proxy XID,  <address>for <address>, no path
SRBn: br dropped proxy TEST, <address>for <address>, no path
```

A proxy explorer reply was not generated because there was no way to get there from this interface. The packet came from the node with the first <address>.

```
SRBn: br sent proxy XID,  <address>for <address>[<rif>]
SRBn: br sent proxy TEST, <address>for <address>[<rif>]
```

An appropriate proxy explorer reply was generated on behalf of the second <address>. It is sent to first <address>.

```
RSRB: sent RingXreq to <ring group>/<ip addr>
```

A Ring eXchange request was sent to the indicated peer. This tells the remote side which rings this node has and requests a reply indicating which rings that side has.

```
RSRB: <label>: sent <op>to <ring group>/<ip addr>
```

A message has been sent to the indicated remote peer. Where <label> may be AHDR (active header), PHDR (passive header), HDR (normal header), or DATA (data exchange). Where <op> may be Forward, Explorer, Ring Xchg, Req, Ring Xchg, Rep, Unknown Ring Group, Unknown Peer, and Unknown Target Ring.

```
RSRB: removing bn <bridge>rn <ring>from <ring group>/<ip addr>
RSRB: added bridge <bridge>, ring <ring>for <ring group>/<ip addr>
```

The remote bridge and ring pair have been removed from or added to the local ring group table because the remote peer has changed.

```
RSRB: peer <ring group>/<ip addr>closed [last state n
RSRB: passive open <ip addr>(remote port) -><local port>
RSRB: CONN: opening peer <ring group>/<ip addr>, attempt n
RSRB: CONN: Remote closed <ring group>/<ip addr>on open
RSRB: CONN: peer <ring group>/<ip addr>open failed, <reason>[code]
```

Miscellaneous remote peer connection establishment messages.

```
RSRBn: sent local explorer, bridge <bridge>trn <ring>, [rif]
```

An explorer packet was propagated onto the local ring from the remote ring group.

```
RSRBn: ring group <ring group>not found
RSRBn: explorer rif [rif] not long enough
```

The remote source-route bridging code found the packet to be in error.

The **debug source-event** command enables an interesting subset of the source-bridge debugging messages, including bridged packets rejected and remote peer connection activities. Enter this command at the EXEC prompt:

debug source-event

The **debug token-event** command provides informational displays about significant Token Ring hardware events. Enter this command at the EXEC prompt:

debug token-event

The **debug token-ring** command invokes verbose Token Ring hardware debugging. This includes detailed displays as traffic arrives and departs the unit. Cisco Systems recommends using this feature only on router/bridges with light loads. Enter this command at the EXEC prompt:

debug token-ring

Source-Route Bridge Global Configuration Command Summary

Following is an alphabetical summary of the global configuration commands for source-route bridging.

[**no**] **rif** *mac-address* [*rif-string*][**interface** | **ring-group** *ring*]

Inserts or removes an entry into the RIF cache. If *rif-string* is present, it is checked for validity and used. Otherwise, the entry is flagged as having no RIF. An **interface** or appropriate ring-group **ring** may also be specified to indicate the direction from which this RIF entry would have arrived.

[**no**] **rif timeout** *minutes*

Determines the period of inactivity allowed before unused RIF cache entries are removed. The **no** form of the command resets the RIF timeout period to its default of 15 minutes.

[**no**] **source-bridge remote-peer** *ring-group* **interface** *interface-name* [**lf** *size*]

Defines or removes a serial interface over which to run bridged Token Ring traffic. The keyword **lf** and the *size* argument define the largest frame size to communicate with all peers in the ring group.

[**no**] **source-bridge largest-frame** *ring-group size*

Defines the largest frame size to communicate with all peers in the ring group.

[**no**] **source-bridge remote-peer** *ring-group* **tcp** *ip-address* [**lf** *size*]

Defines or removes a remote peer for the specified ring group. We would make a TCP connection to carry bridged Token Ring traffic. The keyword **lf** and the *size* argument define the largest frame size to communicate with all peers in the ring group.

[no] source-bridge ring-group *ring-number*

Establishes or removes a ring group.

Source-Route Bridge Interface Subcommand Summary

Following is an alphabetical summary of the interface subcommands for source-route bridging.

[no] multiring {*protocol-keyword* | **all** | **other**}

Enables or disables the specified interface's ability to collect and use source-route (RIF) information for routable protocols. The argument *protocol-keyword* is one of: **apollo**, **appletalk**, **clns**, **decnet**, **ip**, **vines**, or **xns**. The **all** keyword enables the multiring for all frames; the **other** keyword enables the multiring for any frame not included in the previous list.

[no] source-bridge *local-ring bridge-number target-ring*

Enables and disables source bridging on a specific interface.

[no] source-bridge max-rd *count*

Allows the system administrator to limit the maximum RIF size that this bridge will deal with. *The argument count* determines the number of route descriptors that may appear in any explorer RIF. The command **no source-bridge max-rd** resets the count back to the maximum value.

[no] source-bridge old-sna

Enables or disables a workaround some source-route bridging behavior exhibited by older SNA nodes.

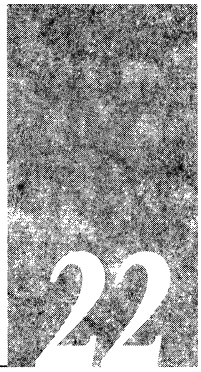
[no] source-bridge proxy-explorer

Enables and disables the proxy explorer function. The default is disabled.

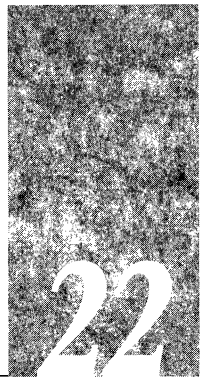
[no] source-bridge spanning

Manually changes the forwarding state of spanning explorer packets; the **no** form disables forwarding.

Chapter 22
Configuring Serial Tunneling in SDLC and HDLC Environments



Chapter 22
Configuring Serial Tunneling in SDLC and HDLC Environments



This chapter describes Cisco's Serial Tunneling (STUN) implementation. The following topics are included in this chapter:

- Description of the SDLC transport function and procedures for configuration.
- Configuring the Cisco router/bridge to exchange data over HDLC-compliant links.
- Configuring a proxy polling feature that allows Cisco routers to act as proxies for IBM devices, and thereby reduce traffic on the intermediate network links.
- Creating a custom serial transport protocol.

The Cisco Serial Tunnel (STUN) Function

The Cisco Serial Tunnel (STUN) function allows two devices using SDLC- or HDLC-compliant protocols that are normally connected by a direct serial link, to be connected through one or more Cisco routers. The serial frames can then be propagated over arbitrary media and topologies to another Cisco router with a STUN link to an appropriate end point. The intervening network is not restricted to STUN traffic, but rather, is multiprotocol. Instead of running parallel backbones for DECnet and SNA/SDLC traffic, for example, this traffic can now be integrated into an enterprise backbone network.

As another example, using the SDLC protocol, STUN allows networks with IBM mainframes and communications controllers to share data using Cisco routers and existing network links. As an SDLC transport function, STUN fully supports the IBM Systems Network Architecture (SNA), and allows IBM Synchronous Data Link Control (SDLC) frames to be transmitted across the network media and and/or shared serial links. Figure 22-1 illustrates a typical network configuration with and without the Cisco SDLC transport function.

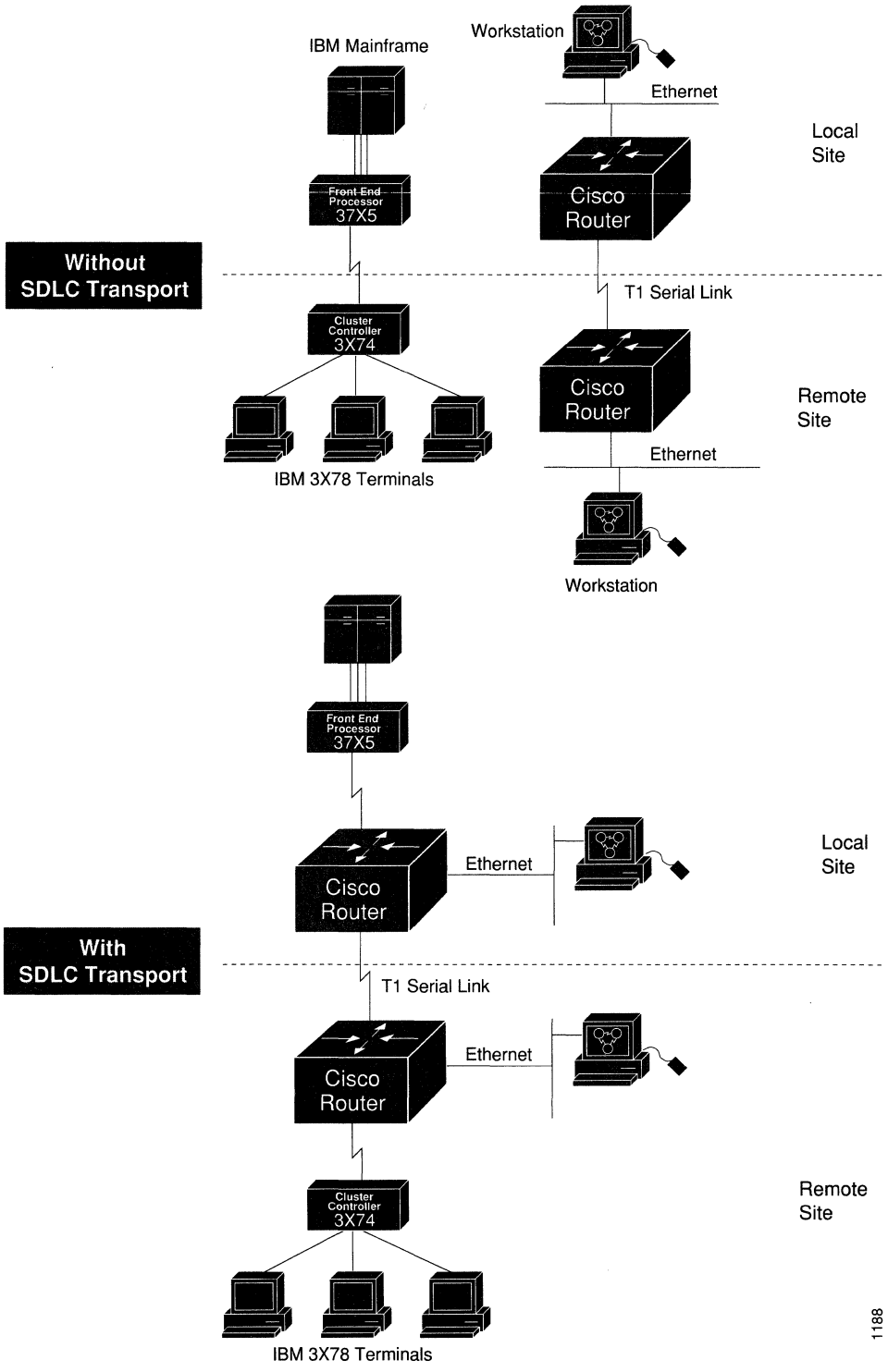


Figure 22-1 IBM Network Configuration With and Without SDLC Transport

The software encapsulates SDLC frame traffic into IP packets and routes them over any of the IP-supported network media—serial, FDDI, Ethernet, and Token Ring, X.25, SMDS, and T1/T3—using the TCP transport mechanism. Because the TCP transport mechanism is used, you may use the Cisco IGRP routing protocol to route the packets. Use of IGRP allows load sharing of SDLC data for faster throughput.

As an SDLC transport, STUN copies frames to destinations based on address, but does not modify the frames in any way, or participate in SDLC windowing or retransmission; these functions are left to the communicating hosts. The STUN SDLC transport function can be treated as similar to a multidrop serial line.

The STUN function also provides for configuration of redundant links to provide transport paths in the event part of the network goes down.

Another important part of STUN's SDLC transport function is the proxy polling feature. An SDLC link is described by the type of stations connected to it, and how they are configured. The two types are point-to-point and multidrop.

In a typical configuration, a primary station, (a communications controller), manages a link that has several other stations on it called secondary stations. The stations transmit data to each other using a mode of transmission called *normal response mode,* whereby a secondary station can only transmit after the primary node has polled it. In networks where many secondary nodes (terminals) must be polled, this can result in traffic overloads and network bottlenecks.

The STUN proxy poll function provides a more efficient way for stations to transmit data by allowing Cisco routers to act as proxies for SDLC terminals and controllers, that is, to allow the routers to poll the secondary nodes, and then to collect and pass only significant data.

Configuration Overview

The Cisco STUN software provides several methods by which devices using HDLC-compliant protocols may exchange data via links connected to one or more Cisco routers.

One way is to use the SDLC transport feature, which behaves like a multidrop or point-to-point serial line and passes frames across arbitrary, intermediate network media unchanged, thereby expecting the host to take care of SDLC windowing and retransmission functions.

Another way is to use the serial tunneling (STUN) method, which allows you to configure arbitrary HDLC, ISO 3309-compliant frames using the TCP transport mechanism over any IP network media. Or you may define a serial transport method that propagates the tunneled frames over a serial line using a simpler, shorter encapsulation.

The software also supports configuration of redundant links, and provides commands to monitor and debug the STUN.

The following sections outline the steps you must take to configure your Cisco router for these features. These are followed by sections that describe the tasks and provide configuration examples, and the commands to maintain the links. An alphabetically arranged summary of the configuration commands is also provided at the end of the chapter.

Configuring the SDLC Transport

Follow these steps to configure the SDLC transport function:

Step 1: Enable STUN and define the transport peers using the **stun peer-name** command.

Step 2: Specify the SDLC transport protocol using the **stun protocol-group** command and the **sdlc** keyword.

The SDLC transport uses the TCP and simple serial transport mechanisms. You assign transport peers using IP addresses or serial interface names, and transport groups by assigning group numbers and listing the protocol. Continue with these steps to configure STUN's SDLC transport on the interface:

Step 3: Configure STUN encapsulation on the interface using the **encapsulation stun** command.

Step 4: Specify the group in which the interface will participate using the **stun group** command. This step and step 3 together assign the STUN protocol to be used.

Step 5: Define how the frames will be forwarded using the **stun route** command.

Step 6: Configure transport-specific features such as proxy polling, if needed.

Configuring Non-SLDC Serial Tunneling

Follow these steps to configure serial tunneling:

Step 1: Enable STUN and define the transport peers using the **stun peer-name** command.

Step 2: Define the protocol to be used. You can choose from predefined protocols, or define your own protocol. (If you define your own protocol, this must be done before step 1 using the **stun schema** command.)

Step 3: Assign the predefined or new protocols to a group using the **stun protocol-group** command.

The STUN uses the TCP and simple serial transport mechanisms. You assign transport peers using IP addresses or serial interface names, and transport groups by assigning group numbers and listing the protocol. Continue with these steps to configure STUN on the interface:

Step 4: Configure STUN encapsulation on the interface using the **encapsulation stun** command.

Step 5: Specify the group in which the interface will participate using the **stun group** command. This step and step 4 together assign the STUN protocol to be used.

Step 6: Define how the frames will be forwarded using the **stun route** command.

Step 7: Configure transport-specific features such as proxy polling, if needed.

Notes and Tips About Configuring STUN

Keep the following caveats in mind when configuring STUN:

- The STUN function will only work on full-duplex, NRZ-encoded lines.
- If you are using the SDLC transport function of STUN in a multipoint (multidrop) configuration, you may need to ensure (with cabling) that the Carrier Detect (Receive Line Signal Detect, or RLSD) pin leading into your Primary SNA device is held low.

Enabling Serial Tunneling

Use the **stun peer-name** global configuration command to enable the STUN function. The command has this syntax:

stun peer-name *ip-address*

no stun peer-name *ip-address*

Enter the IP address by which this STUN peer is known to other STUN peers that are using the TCP transport for the argument *ip-address*. (Even if you do not use the TCP transport, you must issue this command to define a peer name.)

Use the **no stun peer-name** command with the appropriate IP address to disable the STUN function.

Example:

This command assigns IP address *131.108.254.6* as the STUN peer:

```
!
stun peer-name 131.108.254.6
!
```

Defining the STUN Protocol

Each STUN interface is placed in a group which defines the ISO 3309-compliant framed protocol running on that link. Use the **stun protocol-group** global configuration command to define the group number and protocol. The command has this syntax:

stun protocol-group *group-number protocol-keyword*

no stun protocol-group *group-number protocol-keyword*

The **stun protocol-group** command associates group numbers with protocol names. The *group-number* argument can be any number you select between 1 and 255. There are two predefined STUN protocols, **basic** and SDLC. These are specified by supplying the keyword **basic** or **sdlc** for the **protocol-keyword** argument. You can also define your own STUN protocol. See the section "Defining Your Own STUN Protocols" later in this chapter.

Use the **no stun protocol-group** command with the appropriate group number and protocol to remove an interface from the group.

Note: If you are defining a custom protocol, you must do so before doing this step; see the section "Defining Your Own STUN Protocols" later in this chapter for the procedure.

Choosing the SDLC Transport

The STUN SDLC transport protocol is used for placing Cisco routers in the midst of either point-to-point or multipoint (multidrop) SDLC links. At the current time, only full-duplex, NRZ-encoded links are supported. An example of how to set up the SDLC transport follows:

Example:

This example command specifies that group 7 use the SDLC STUN protocol:

```
!
stun protocol-group 7 sdlc
!
```

Note: Selecting this predefined protocol allows use of the optional proxy polling feature.

Choosing the Basic STUN Protocol

The basic STUN protocol is unconcerned with details of serial protocol addressing and is used when addressing is unimportant. Use this when your goal with the STUN is to replace one or more sets of point-to-point (not multidrop) serial links by using a protocol other than SDLC. An example of how to set up the basic STUN protocol follows.

Example:

This command specifies that group 5 use the basic protocol:

```
!
stun protocol-group 5 basic
!
```

Configuring STUN on the Interface

To enable and configure the STUN function on a particular serial interface, use the interface subcommand. The command has this syntax:

encapsulation stun

This command must be specified. It is not possible to further configure the interface without first specifying this command.

Example:

These commands enable interface serial 0 for STUN:

```
!
interface serial0
encapsulation stun
!
```

Placing the Interface in a STUN Group

Each STUN-enabled interface on a Cisco router must be placed in a previously defined STUN group. Packets will only travel between STUN-enabled interfaces that are in the same group. Use this interface subcommand to do this:

stun group *group-number*

no stun group *group-number*

The argument *group-number* is a number you assign and which must be a decimal integer between 1 and 255 (inclusive).

Example:

These commands place serial interface 2 in STUN group 1, which is defined to run the SDLC transport:

```
!
stun protocol-group 1 sdlc
!
interface serial 2
encapsulation stun
stun group 1
!
```

Note: Once a given serial link is configured for the STUN function, it is no longer a shared multiprotocol link. All traffic that arrives on the link will be transported to the corresponding peer as determined by the current STUN configuration.

Defining How Frames Will Be Forwarded

To define how frames will be forwarded on the interface, use one of the following variations of the **stun route** interface subcommand:

stun route all tcp *ip-address*
no stun route all tcp *ip-address*
stun route all interface serial *interface-number*
no stun route all interface serial *interface-number*

stun route all interface serial *interface-number* **direct**
no stun route all interface serial *interface-number* **direct**

stun route address *address-number* **tcp** *ip-address*
no stun route address *address-number* **tcp** *ip-address*

stun route address *address-number* **interface serial** *interface-number*
no stun route address *address-number* **interface serial** *interface-number*

stun route address *address-number* **interface serial** *interface-number* **direct**
no stun route address *address-number* **interface serial** *interface-number* **direct**

Use the command forms with the **all** keyword when *all* STUN traffic received on the input interface will be propagated regardless of what address is contained in the serial frame. (These are the only **stun route** command forms allowed with the basic STUN transport protocol.)

The **tcp** keyword causes the TCP transport mechanism to be used to propagate frames that match the entry. The TCP transport allows movement of serial frames across arbitrary media types and topologies. This is particularly useful for building shared, multiprotocol enterprise network backbones. Enter the address that identifies the remote STUN peer that is connected to the far serial link for the *ip-address* argument.

The **interface serial** keywords cause the Serial Transport method of the STUN function to be used to propagate the serial frame. There must be an appropriately configured Cisco router on the other end of the designated serial line. The outgoing serial link can still be used for other kinds of traffic (the frame is not encapsulated TCP). This mode is primarily used when the complexity of the TCP transport is unjustifiable, or when higher performance is needed. Enter the serial line number connected to the Cisco router for the *interface-number* argument.

Use the command forms with the **address** keyword to specify how a serial frame that contains a particular address is to be propagated. The address you enter for the *address-number* argument varies with the protocol type. The argument will accept octal, decimal, or hexadecimal addresses, as appropriate, in the range allowed by the protocol. As an example, SDLC uses hexadecimal digits and has a one byte address field. The allowable addresses for this protocol must be between 00 and FF, inclusive.

It is possible to use both the **all** and **address** keywords for the same input serial interface. When this is done, the address specifications take effect first. If none of these match, the **all** keyword will be used to propagate the frame.

Use the command form with the **direct** keyword at the end of the command to indicate that the specified interface is also a direct STUN link, rather than a serial connection to another peer.

STUN Configuration Examples

This section contains configuration examples illustrating use of the commands described in this chapter.

Expanding the IBM Network Capability Using Cisco Routers

Figure 22-2 depicts a typical IBM network configuration.

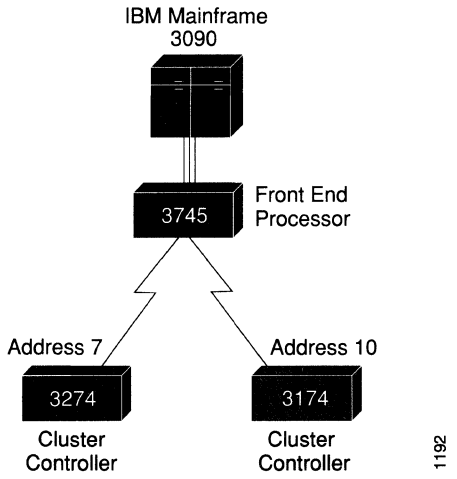


Figure 22-2 IBM Network Without Cisco Router

The configuration has a 3090 mainframe channel-attached to a 3745 controller, and these are connected by SDLC link to 3274 and 3174 cluster controllers.

Figure 22-3 modifies the configuration by placing Cisco routers between the devices connected via SDLC link, thus expanding the capabilities of the network.

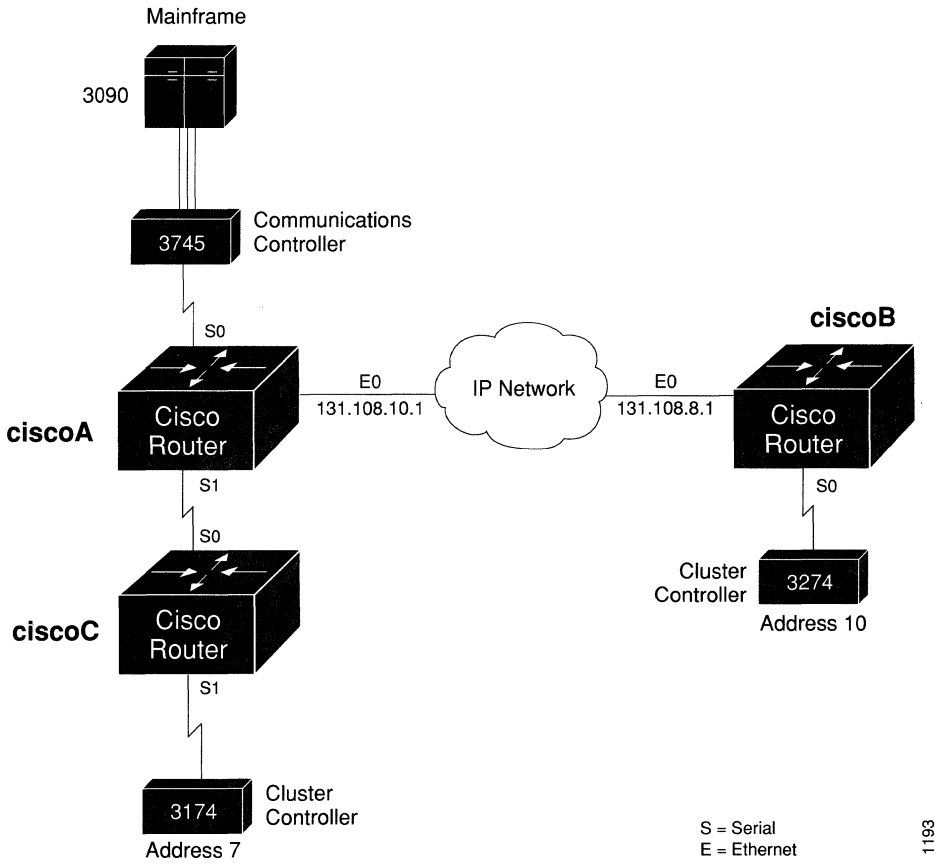


Figure 22-3 IBM Network with Cisco Routers and SDLC Links

The configuration files for connecting the Cisco routers to the IBM network follow.

Configuration for Router ciscoA

```
!
stun peer-name 131.108.10.1
stun protocol-group 1 sdlc
!
interface serial 0
encapsulation stun
stun group 1
stun route address 7 interface serial 1
stun route address 10 tcp 131.108.8.1
!
interface serial 1
ip address 131.108.62.1 255.255.255.0
!
interface ethernet 0
ip address 131.108.10.1 255.255.255.0
!
hostname ciscoA
router igrp 161
network 131.108.0.0
```

Configuration for Router ciscoB

```
!
stun peer-name 131.108.8.1
stun protocol-group 1 sdlc
!
interface serial 0
encapsulation stun
stun group 1
stun route address 10 tcp 131.108.10.1
!
interface ethernet 0
ip address 131.108.8.1 255.255.255.0
!
hostname ciscoB
router igrp 161
network 131.108.0.0
```

Configuration for Router ciscoC

```
!
stun peer-name 131.108.62.2
stun protocol-group 1 sdlc
!
interface serial 0
ip address 131.108.62.2 255.255.255.0
!
interface serial 1
encapsulation stun
stun group 1
stun route address 7 interface serial 0
!
hostname ciscoC
router igrp 161
network 131.108.0.0
```

Extended IBM Network

As another example, the previous configuration can be extended by connecting a remote 3745 to another serial interface connected to ciscoB. All other traffic from the primary node (the 3745 connected to the 3090), other than addresses 7 and 10, should go to this remote 3745. Figure 22-4 illustrates this configuration.

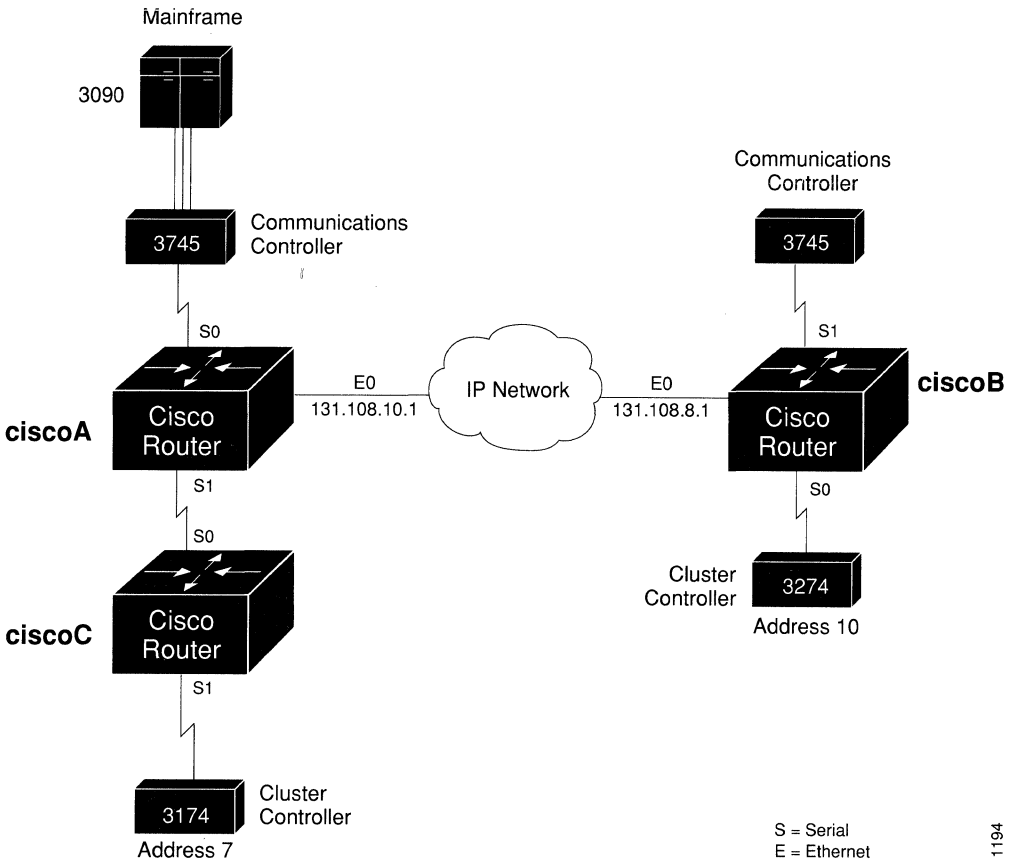


Figure 22-4 Extended IBM Network with Cisco Routers and SDLC Links

The following configuration changes would need to be made to the serial interface in the router labeled *ciscoB*.

```
!
interface serial 1
encapsulation stun
stun group 1
stun route all tcp 131.108.10.1
!
```

Serial 0 on *ciscoA* now looks like this:

```
!
interface serial 0
encapsulation stun
stun group 1
stun route address 7 interface serial 1
stun route address 10 tcp 131.108.8.1
stun route all tcp 131.108.8.1
!
```

Notice in the above examples that the same STUN group number is used in all cases. If these numbers differ between the three routers, attempts at communication will be unsuccessful. There are times, however, when having multiple groups can be useful. Such an example is illustrated, next.

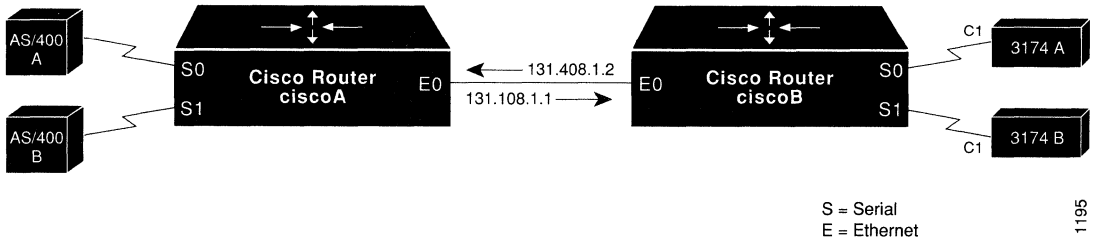


Figure 22-5 IBM Network with Multiple Groups of Controllers

In the following configuration, the AS/400 device labeled *A* wants to communicate with the 3174 device labeled *A*, as do the AS/400 device labeled *B* and 3174 device labeled *B*. Notice that both of the 3174 devices are at SDLC address C1. A first attempt at the configuration files for the routers labeled *ciscoA* and *ciscoB* could be as follows.

Configuration for Router ciscoA

```
!
stun peer-name 131.108.1.1
stun protocol-group 1 sdlc
!
interface ethernet 0
ip address 131.108.1.1 255.255.0.0
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.2
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.2
!
```

Configuration for Router ciscoB

```
!
stun peer-name 131.108.1.2
stun protocol-group 1 sdlc

!
interface ethernet 0
ip address 131.108.1.2 255.255.0.0
!
interface serial 0
encapsulation stun
stun group 1
stun route address C1 tcp 131.108.1.1
!
interface serial 1
encapsulation stun
stun group 1
stun route address C1 tcp 131.108.1.1
!
```

A problem occurs when the router labeled *ciscoA* transfers an SDLC frame with address C1 from the AS/400 A device to the router labeled *ciscoB*. There would be no way to determine that it came from the AS/400 device *A* and was therefore destined to the 3174 device *A*, or that it came from the AS/400 device *B* and was therefore destined to the 3174 device *B*.

The use of distinct group numbers solves this problem. The configuration shown below will ensure correct communication, as frames that come in on group 1 links will only go out of group 1 links. The same holds true for frames coming in on group 2, or any other numbered group link. The new configuration follows.

Configuration for Router ciscoA

```
!
stun peer-name 131.108.1.1
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface ethernet 0
ip address 131.108.1.1 255.255.0.0
!
interface serial 0
encapsulation stun
stun group 1
stun route address C1 tcp 131.108.1.2
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.2
!
```

Configuration for Router ciscoB

```
!
stun peer-name 131.108.1.2
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface ethernet 0
ip address 131.108.1.2 255.255.0.0
!
interface serial 0
encapsulation stun
stun group 1
stun route address C1 tcp 131.108.1.1
!
interface serial 1
encapsulation stun
stun group 2
stun route address C1 tcp 131.108.1.1
!
```

Prioritizing STUN Traffic

At times, you may wish to prioritize STUN traffic in your network over that of other protocols. The use of priority output queuing, described in Chapter 7, "Adjusting Interface Characteristics," enables this functionality. STUN uses a special serial line protocol called STUN for the simple serial encapsulation, and TCP port 1992 for the TCP encapsulation. Therefore, to fully specify STUN traffic on priority list 4 for high priority, the following configuration is used.

Example:

```
priority-list 4 stun high
priority-list 4 ip high tcp 1992
```

Configuring Redundant Links

In the following sample network, there are two redundant links between the Cisco routers.

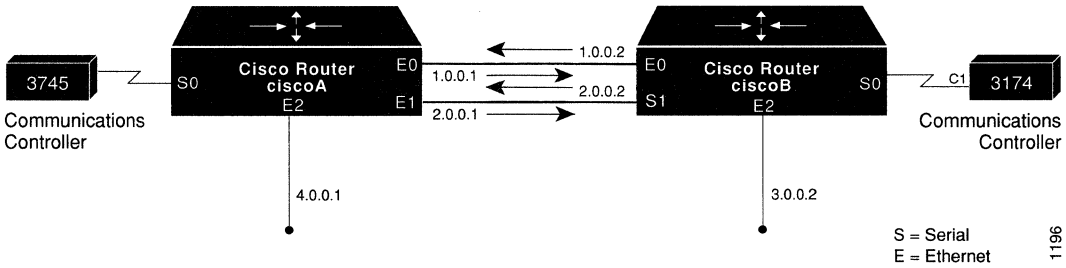


Figure 22-6 Redundant Links in an IBM Network

However, if you were to specify remote peers on network 1 for the two routers, the redundancy would be in effect only as long as network 1 were available. The following example shows the network configuration.

Example:

The configuration for the router labeled *ciscoA* is as follows:

```
!
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
interface ethernet 0
ip address 1.0.0.1 255.0.0.0
interface ethernet 1
ip address 2.0.0.1 255.0.0.0
interface ethernet 2
ip address 4.0.0.1 255.0.0.0
interface serial 0
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.2
!
```

The configuration for the router labeled *ciscoB* is as follows:

```
!
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
interface ethernet 0
ip address 1.0.0.2 255.0.0.0
interface ethernet 1
ip address 2.0.0.2 255.0.0.0
interface ethernet 2
ip address 3.0.0.2 255.0.0.0
interface serial 0
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.1
!
```

In the event that network 1 went down, then the access to network 1.0.0.1 from the router labeled *ciscoB* (and to network 1.0.0.2 from *ciscoA*) would be lost. Therefore, connectivity would be lost for the 3745 and the 3174, even though a second path exists.

When you configure redundant links, keep the following rule in mind:

- Specify a peer name and network address for remote routers that are not interface on the normal path between the two routers.

Example:

In the above case, specifying the following configuration would allow connectivity between the 3745 and 3174 in all cases, when either or both network 1.0.0.0 or network 2.0.0.0 were working:

The configuration for the router labeled *ciscoA* is as follows:

```
!
stun peer-name 4.0.0.1
stun protocol-group 1 sdlc
interface ethernet 0
ip address 1.0.0.1 255.0.0.0
interface ethernet 1
ip address 2.0.0.1 255.0.0.0
interface ethernet 2
ip address 4.0.0.1 255.0.0.0
interface serial 0
encapsulation stun
stun group 1
stun route address C1 tcp 3.0.0.2
!
```

The configuration for the router labeled *ciscoB* is as follows:

```
!
stun peer-name 3.0.0.2
stun protocol-group 1 sdlc
interface ethernet 0
ip address 1.0.0.2 255.0.0.0
interface ethernet 1
ip address 2.0.0.2 255.0.0.0
interface ethernet 2
ip address 3.0.0.2 255.0.0.0
interface serial 0
encapsulation stun
stun group 1
stun route address C1 tcp 4.0.0.1
!
```

There may still be cases where such remote networks do not exist. Consider the following network.

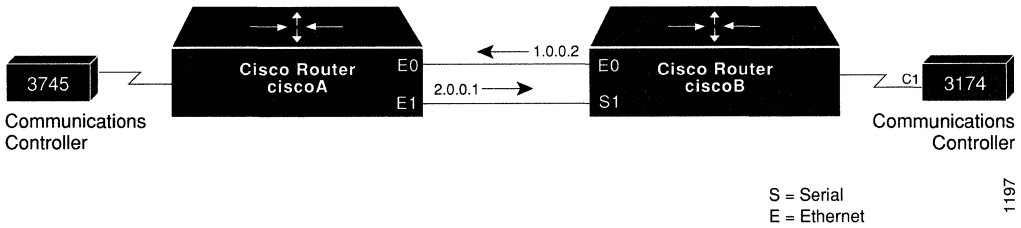


Figure 22-7 Redundant Links in an IBM Network Using IP Addresses for Reference

In this situation, keep this rule in mind:

- IP addresses may be specified for serial links which have STUN encapsulation enabled. This remote reference allows connections from remote devices into the router onto this IP address when this SDLC serial interface is up.

Both networks 1.0.0.0 and 2.0.0.0 in the above example would be available when the following configuration is used.

Example configuration for the router labeled *ciscoA*:

```
!
stun peer-name 4.0.0.1
stun protocol-group 1 sdlc
interface ethernet 0
ip address 1.0.0.1 255.0.0.0
interface ethernet 1
ip address 2.0.0.1 255.0.0.0
interface serial 0
encapsulation stun
stun group 1
ip address 4.0.0.1 255.0.0.0
stun route address C1 tcp 3.0.0.2
!
```

Example configuration for the router labeled *ciscoB*:

```
!
stun peer-name 3.0.0.2
stun protocol-group 1 sdlc
interface ethernet 0
ip address 1.0.0.2 255.0.0.0
interface ethernet 1
ip address 2.0.0.2 255.0.0.0
interface serial 0
encapsulation stun
stun group 1
ip address 3.0.0.2 255.0.0.0
stun route address C1 tcp 4.0.0.1
!
```

Note: The router cannot talk to itself on this interface. As an example, while other devices could talk to the *ciscoB* device by specifying address 3.0.0.2, *ciscoB* could not talk to itself by specifying the same address.

Using STUN in SDLC Environments

In normal communication between an SDLC primary node and its secondary node, the secondary node is only allowed to send data to the primary node in response to a poll from the primary node. In the following example, an AS/400 host is attached to a 3174 controller that handles transfers from several attached 3270 terminals:

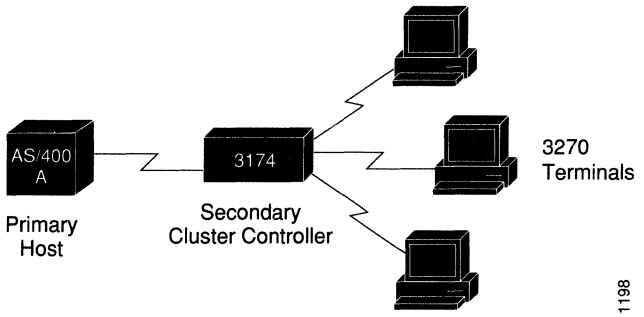


Figure 22-8 Typical IBM SDLC Primary and Secondary Node Configuration

If one of the 3270-style terminals attached to the 3174 controller wants to initiate a data transfer, (usually the result of a user at the terminal pressing the Enter key), the 3174 device would not be able to immediately transfer the data back to the host AS/400 since the 3174 is the secondary node on the link. Instead, it must hold the data until a poll is received from the AS/400, which is the primary node in this example. Once the poll is received, the data can then be transmitted.

The primary host ensures a reasonable response time for its secondary nodes by sending out polls at a rate that's often more than 20 times per second. With two devices sharing a single, dedicated serial line, as in the example above, this poses no problem, since the link would be idle without the polls.

In the following example, the frequent polls and their replies would constantly travel between the two Cisco routers across the shared Ethernet.

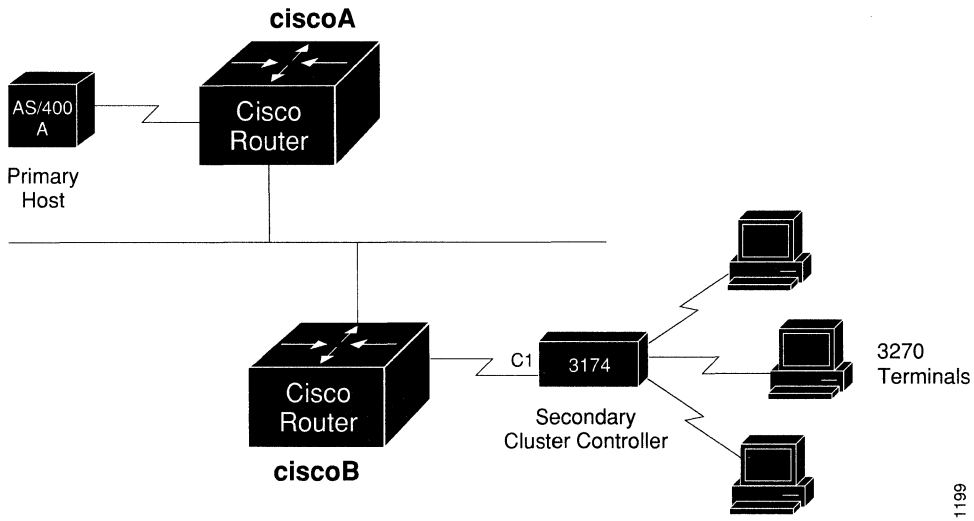


Figure 22-9 IBM SDLC Primary and Secondary Nodes with Cisco Proxy Polling Feature

Such constant traffic can create bottlenecks and loads where they cannot be appropriately handled.

Configuring Proxy Polling

The proxy polling feature alleviates the load across the network by allowing the Cisco routers to act as proxies for the primary and secondary nodes, thus keeping polling traffic off of the shared links.

With proxy polling enabled, the router labeled *ciscoA* in Figure 22-9 would reply to the AS/400 poll requests, as a proxy for the secondary node, thereby keeping the polls and requests off of the shared Ethernet. Similarly, the router labeled *ciscoB* would act as a proxy for the primary node and periodically send polls to the secondary 3174 device, and thereby keep its replies off of the shared cable. Only significant information is passed across the shared Ethernet.

Enabling Proxy Polling

Use these interface subcommands to enable proxy polling:

stun proxy-poll address *address* **modulus** *modulus* {**primary** | **secondary**}

no stun proxy-poll address *address* **modulus** *modulus* {**primary** | **secondary**}

stun proxy-poll address *address* **discovery**

no stun proxy-poll address *address* **discovery**

Enter the address of the device on which to enable proxy polling with the *address* argument.

Enter the modulus of the link as defined by the MODULUS parameter specified in the line descriptions on your SDLC host with the *modulus* argument. (This most often will be eight.)

Use the **primary** or **secondary** keyword to indicate which role the SDLC device is playing.

Use the command form with the **discovery** keyword when you do not want to specify the primary and secondary ends and the modulus used on an SDLC link, or when such connections are negotiable.

Use of the **discovery** keyword is not recommended except where you cannot avoid end hosts that negotiate the status, since proxying will be disabled on the link until session start-up, when the appropriate primary and secondary status, as well as modulus on the link, can be discovered. Until this time, all polls travel through the network.

By default, proxy polling is disabled. Once enabled, use the **no stun proxy-poll** command with the appropriate arguments to return to the default state.

Example:

This command enables proxy polling for a secondary device at address C1 on interface serial 2 running with modulus 8:

```
!
interface serial 2
stun proxy-poll address C1 modulus 8 secondary
!
```

Configuring a Proxy Poll Interval

You may change the number of milliseconds between each sequence of proxy polls generated on the secondary side of a connection. Use the global configuration command **stun poll-interval** to do so. The command has this syntax:

stun poll-interval *milliseconds*

no stun poll-interval

The command applies to all secondary proxy sessions in the router.

Enter the number of milliseconds desired with the *milliseconds* argument. The default and minimum value that can be specified is 20, or 1/50th of a second.

The **no stun poll-interval** command returns the default state.

Example:

This example sets the poll interval to 100 milliseconds, or 10 times per second:

```
!
stun poll-interval 100
!
```

Configuring a Primary Side Pass-Through Interval

Periodically, even when proxy polling is enabled, the router on the primary side of an SDLC connection will pass through a poll from the primary SDLC device through the network to the secondary SDLC device. This action causes the secondary device's reply to also traverse the entire network. This periodic pass-through provides an insurance mechanism that makes sure the primary SDLC device maintains an accurate notion of the secondary SDLC device status.

You can change the number of seconds between each pass through of polls between the primary and secondary SDLC devices. Use the global configuration command **stun primary-pass-through** to do so. The full command syntax follows.

stun primary-pass-through *seconds*

no stun primary-pass-through

The **stun primary-pass-through** command applies to all primary proxy sessions in the router. Use the **no stun primary-pass-through** command to return to the default state.

Example:

This example sets the primary pass-through interval to 20 seconds.

```
!
stun primary-pass-through 20
!
```

Defining Your Own Protocols

Cisco's STUN implementation allows you to define your own STUN protocols. This step must be done before defining the protocol group using the **stun protocol-group** command.

This feature allows you to transport any serial protocol that meets the following criteria across a Cisco router internetwork. The following conditions must be obeyed by your serial protocol to have it transferred in this manner:

- The protocol uses full-duplex conventions (RTS/CTS always high)
- The protocol uses NRZ encoding
- The protocol uses standard HDLC checksums and framing, (beginning/end of frames, data between frames).

In addition, if you want to use anything but the predefined, basic protocol, which does not allow the specification and routing by an address to achieve a virtual multidrop result, your protocol must also meet the following constraints:

- Addresses are contained in a constant location (offset) within the frame.
- Addresses are found on a byte boundary.

As an overview, SDLC frames have two formats, basic and extended. In the basic SDLC frame shown in Figure 22-10, the Address and Control fields are both 8 bits (also one byte or octet) wide. In the Extended Control format, the Control field is 16 bits wide.

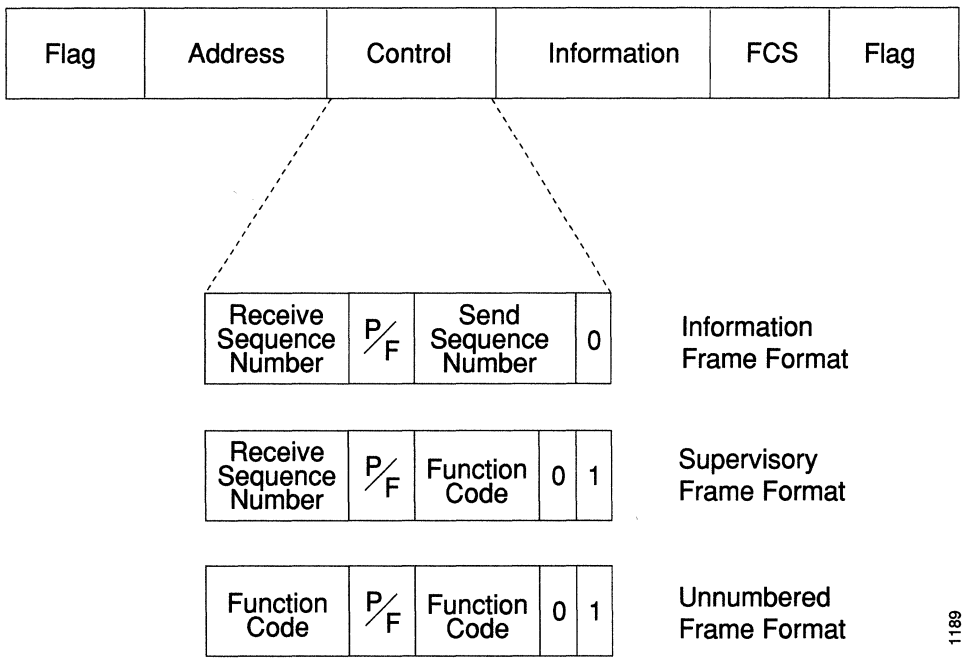


Figure 22-10 SDLC Frame Format

The Address field contains the address of a sending or receiving station, depending upon the procedure. The Control field contains information that determines the type of SDLC frame being transmitted, either:

- Information frames used for data transfer.
- Supervisory frames that control the flow of data.
- Unnumbered frames that control the link.

The Frame Check Sequence (FCS) field is always 16 bits long. A Flag is a special bit sequence that defines the beginning and end of a frame. The bytes in the SDLC frame, except for the FCS, are always transmitted low-order bit first. The FCS bits are transmitted high-order bit first.

Use the **stun schema** global configuration command to specify the format, or schema, for your protocol. The command syntax follows:

stun schema *name* **offset** *constant-offset* **length** *address-length* **format** *format-keyword*

no stun schema

The argument *name* is a name that can be up to 20 characters in length that defines your protocol.

The argument *constant-offset* specifies the constant offset, in bytes, for the address to be found in the frame. The argument *address-length* specifies the length of that address. The length is limited and these limits are described in Table 22-1.

The argument *format-keyword* specifies the format to be used to specify and display addresses for routes on interfaces that use this STUN protocol. The allowable formats are listed in Table 22-1:

Table 22-1 Allowable Schema Formats

Format	Allowable Base
decimal	base 10 addresses (0-9)
hexadecimal	base 16 addresses (0-f)
octal	base 8 addresses (0-7)

The *address-length* argument, in bytes, is limited to the following:

If *format-keyword* is:	the *address-length* limit is:
decimal	4
hexadecimal	8
octal	4

The command **no stun schema** removes the schema.

Example:

This command defines a format of SDLC as a new STUN protocol. (This definition of SDLC would not support the proxy polling option available with the predefined protocol definition.)

```
!
stun schema new-sdlc offset 0 length 1 format hexadecimal
!
```

Once defined, new protocols may be used in the **stun protocol-group** interface subcommands to tie STUN groups to the new protocols.

Monitoring STUN

This section describes the EXEC commands you use to monitor the state of the STUN.

Displaying the Current Status of STUN

Use the **show stun** command to examine the current status of the STUN. The command has this format:

show stun

Sample output is shown below:

```
ciscoA#show stun
This peer: 131.108.10.1
Serial0 -- 3174 Controller for test lab (group 1 [sdlc])
state  rx_pkts  tx_pkts  drops  poll
  7[ 1] IF Serial1          open     20334    86440      5  8P
 10[ 1] TCP 131.108.8.1     open      6771     7331      0
all[ 1] TCP 131.108.8.1     open    612301  2338550   1005
```

The first entry reports proxy polling enabled for address 7 and that Serial 0 is running with modulus 8 on the primary side of the link. The link has received 20,334 packets, transmitted 86,440 packets, and dropped 5 packets.

Table 22-2 STUN Status Display Field Descriptions

Field	Descriptions
This peer	Lists the peer-name or address. The interface name (as defined by the interface **description** subcommand), its STUN group number, and the protocol associated with the group are shown on the header line.
STUN address	Address or the word "all" if the default forwarding entry is specified, followed by a repeat of the group number given for the interface.
Type of link	Description of link, either a serial interface using Serial Transport ("IF" followed by interface name), or a TCP connection to a remote router ("TCP" followed by IP address).
state	State of the link: open is the normal, working state; direct indicates a direct link to another line, as specified with the **direct** keyword on the **stun route** interface subcommand.
rx_pkts	Number of received packets.
tx_pkts	Number of transmitted packets.
drops	Number of packets that for whatever reason had to be dropped.
poll	Report of the proxy poll parameters, if any. A "P" indicates a primary and an "S" indicates a secondary node. The number before the letter is the modulus of the link.

Displaying the Proxy States

Use the **show stun sdlc** command to examine the proxy state of various interfaces on an address-by-address basis. The command has this format:

show stun sdlc

Sample output is shown below:

```
ciscoA#show stun sdlc
Serial 1 -- 3174 controller for test lab
B3: s2 C1: p4 DE: d6
```

This example output shows us that Serial 1 has three addresses for which proxy polling is enabled. These are `B3`, for which this end is a secondary link in state 2 and `C1` for which this end is a primary link and in state 4. Finally, DE is in the primary/secondary/modulus discovery state 6.

The display reports the status of the interfaces using SDLC encapsulation and whether proxy polling is enabled for that interface. Interfaces with proxy polling enabled are noted with the address followed by an indication of node type, either "s" for secondary, "p" for primary or "d" for discovery, and the state of the node. Possible node states are defined in Table 22-3.

Table 22-3 Node States

State	Description
0	Significant data travelling between the primary and secondary node. No proxy polling occurring.
1-3	Proxy polling in process of being initiated.
4	Proxy polling is activated for the link at this time. If this in the primary node, the Cisco router responds to the primary's poll. If this is the secondary node, the Cisco router will periodically generate polls for potential response by the secondary.
5	The primary Cisco has data from the secondary to send to the primary SDLC machine, but is waiting for a poll from that machine before transmitting the data.
6-7	This Cisco router is still trying to determine the primary/secondary character and modulus of the SDLC hosts attached to it. No proxying is done in these states.

The above sample reports that serial 0, while using SDLC encapsulation, is not using the proxy polling feature. Serial 0 has three address, B3, C1, and DE on which proxy polling is enabled. The address B3 is a secondary link in state 2; the address C1 is the primary link, and is in state 4; the address DE is in discovery state 6.

Debugging STUN

This section describes the privileged EXEC debugging commands you use to debug operation of the STUN. Generally, you will enter these commands during troubleshooting sessions with Cisco Customer Engineers. For each **debug** command, there is a corresponding **undebug** command to disable the reports.

debug stun-packet [*group*] [*address*]

The debug stun-packet command enables debugging of packets traveling through the STUN links. When enabled, it will produce numerous messages showing every packet travelling through the links. The optional argument *group* is the decimal integer assigned to a group. When the *group* parameter is given, output will be limited to only packets associated with STUN group specified. If the optional *address* argument is also specified, output will be further limited to only those packets with the STUN address specified. The *address* argument is in the format appropriate for the STUN protocol running for the group for which it is specified.

debug stun

The **debug stun** command enables debugging of STUN connections and status. When enabled, it will cause messages showing connection establishment and other overall status message to be displayed.

STUN Global Configuration Command Summary

Following are the global configuration commands used to configure the STUN function. These commands may appear anywhere in the configuration file.

[no] stun peer-name *ip-address*

Enables or disables STUN. The argument *ip-address* is the IP address by which this STUN peer is known to other STUN peers that are using the TCP transport.

[no] stun poll-interval *milliseconds*

Changes the interval between each sequence of proxy polls generated on the secondary side of the connection. The argument *milliseconds* is the interval desired, in milliseconds.

Default and minimum value is 20, or 1/50th of a second, which is returned by use of the **no** keyword.

[no] stun primary-pass-through *seconds*

Changes the number of seconds between each pass through of polls between the primary and secondary SDLC devices. The argument *seconds* defines the interval.

[no] stun protocol-group *group-number protocol*

Associates or removes group numbers with protocol names. The *group-number* argument can be any number you select between 1 and 255. The *protocol* argument is any predefined protocol, or protocol defined by you.

[no] stun schema *name* **offset** *constant-offset* **length** *address-length* **format** *format-keyword*

Specifies or removes a format, or schema, for a user-defined protocol.

The argument *name* defines the protocol. The argument *constant-offset* specifies the constant offset, in bytes, for the address to be found in the frame. The argument *address-length* specifies the length of that offset. The argument *format-keyword* specifies the format to be used to specify and display addresses for routes on interfaces that use this STUN protocol. The allowable formats and their maximum lengths are as follows:

Formats	Base	Length
decimal	base 10 addresses (0-9)	4
hexadecimal	base 16 addresses (0-f)	8
octal	base 8 addresses (0-7)	4

STUN Interface Subcommand Summary

Following are the interface subcommands used to configure STUN. These commands follow an **interface** command.

encapsulation stun

Enables the STUN function. This command must be specified in order to use STUN.

[no] stun group *group-number*

Places STUN-enabled interface in a previously defined group.

The argument *group-number* is user-defined and must be between 1 and 255 (decimal).

[no] stun proxy-poll address *address* **modulus** *modulus* {**primary** | **secondary**}

[no] stun proxy-poll address *address* **discovery**

Enable or disable proxy polling.

The *address* argument is the address of the device on which to enable proxy polling.

The *modulus* argument is the modulus of the link as defined by the MODULUS parameter specified in the line descriptions on the SDLC host.

The **primary** or **secondary** keyword indicate which role the SDLC device is playing.

The **discovery** keyword is used when primary and secondary ends are not specified and connections are negotiated.

Default is proxy polling disabled.

[no] stun route all tcp *ip-address*

[no] stun route all interface serial *interface-number*

[no] stun route all interface serial *interface-number* **direct**

[no] stun route address *address-number* **tcp** *ip-address*

[no] stun route address *address-number* **interface serial** *interface-number*

[no] stun route address *address-number* **interface serial** *interface-number* **direct**

Enable or disable forwarding of frames on the interface.

The **all** keyword is used when all STUN traffic received on the input interface will be propagated regardless of what address is contained in the SDLC frame.

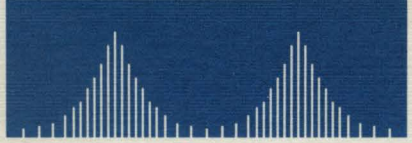
The **tcp** keyword causes the TCP transport mechanism to be used to propagate frames that match the entry. Enter the address that identifies the remote STUN peer that is connected to the far SDLC link for the *ip-address* argument.

The **interface serial** keywords cause the Serial Transport method of the STUN function to be used to propagate the SDLC frame. There must be an appropriately configured Cisco router on the other end of the designated serial line. Enter the serial line number connected to the Cisco router for the *interface-number* argument.

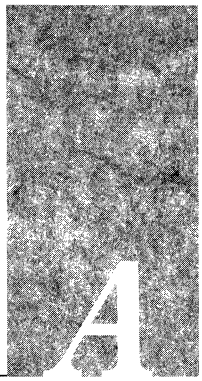
The **address** keyword specifies how an SDLC frame that contains a particular address is to be propagated. The *address-number* argument varies with the protocol type. The argument will accept any octal, decimal, or hexadecimal address in the range allowed by the protocol.

The **all** and **address** keywords are used for the same input serial interface. When this is done, the address specifications take effect first. If none of these match, the **all** keyword will be used to propagate the frame.

The **direct** keyword is used to indicate that the specified interface is also a direct STUN link, rather than a serial connection to another peer.



Appendices



Appendix A
System Error Messages

This appendix documents all system error messages, as of software release 8.3, for the router, terminal server, and protocol translator products. These are the error messages which the system software sends to the console (and, optionally, to a logging server on another system) during operation.

Not all of these messages indicate problems with your system. Some are purely informational, while others may help to diagnose problems with communications lines, internal hardware, or the system software.

How to Read the Error Messages

All messages begin with a percent sign, and are displayed in the following format:

```
%FACILITY-SEVERITY-MNEMONIC: Message-text
```

`FACILITY` is a code, consisting of two to five uppercase letters, indicating the facility to which the message refers. A facility may be a hardware device, a protocol, or a module of the system software. Table A-1 lists the codes for all of the system facilities.

`SEVERITY` is a single-digit code from 0 to 7 which reflects the severity of the condition. The lower the number, the more serious the situation. Table A-2 lists the severity levels.

`MNEMONIC` is a code, consisting of uppercase letters, that uniquely identifies the message.

`Message-text` is a text string describing the condition. This portion of the message sometimes contains detailed information about the event being reported, including terminal port numbers, network addresses, or addresses that correspond to locations in the system's memory address space. Because the information in these variable fields changes from message to message, it is represented here by short strings enclosed in square brackets (`[ ]`). For example, a decimal number is represented as `[dec]`. A complete list of the kinds of variable fields, and the information contained in them, appears in Table A-3.

If one or more error messages recur after you take the recommended action, contact Cisco or contact your local field service organization.

Note: Each section of this chapter describes the error messages produced by a different system facility. Messages are listed alphabetically by mnemonic. If several error messages share the same explanation and recommended action, the messages are presented as a group followed by the common explanation and recommended action. A quick index is also provided at the end of this appendix to help you find the messages quickly.

Use the following tables to interpret the facility codes that begin each message, the severity level numbers associated with each message, and the variables fields contained in some of the messages.

Table A-1 Facility Codes

Code	Facility
AT	AppleTalk
BGP	Border Gateway Protocol
CBUS	The cBus controller
CHAOS	CHAOSnet
CLNS	OSI Connectionless Network Services
CSC2	CSC2/CSC3 CPU card
DNET	DECnet
EGP	Exterior Gateway Protocol
HELLO	HELLO Protocol
HSSI	High Speed Serial Interface Card
IGRP	Interior Gateway Routing Protocol
ILAN	Interlan Ethernet controller
IMP	Interface Message Processor
IP	Internet Protocol
IPRT	IP Routing
LANCE	STS-10x or IGS Ethernet Interface
LAPB	X.25 Link Access Protocol – Binary
LAT	DEC LAT (Local Area Transport)
LINK	Data Link
MAILBOX	Chipcom Mailbox Support
MCI	Multiport Communications Interface
MK5	MK5025 serial controller
PAD	X.25 Packet Assembler/Disassembler
PPP	Point-to-Point Protocol
PR	Parallel Printer devices
PUP	Xerox PARC Universal Packet protocol
RIP	BSD IP Routing Information Protocol

RSRB	Remote Source Routing Bridge
SBE	SBE serial interface
SEC	IP Security
SLIP	Serial Link IP
SRB	Source Routing Bridge
STUN	Serial Tunneling
SYS	Operating system
TAC	Terminal Access Control protocol
TCP	Transmission Control Protocol
TMQ	Inbound Terminal Port Queuing
TN	Telnet
TR	Token Ring
ULTRA	Ultranet card
VINES	Banyan VINES
X25	X.25
XNS	Xerox Network Services

Table A-2 Severity Levels

Number	Explanation
0	System is unusable.
1	Action must be taken immediately.
2	Critical condition.
3	Functionality may be affected.
4	Warning condition.
5	Normal but significant condition.
6	Informational message.
7	Appears during debugging only.

Table A-3 Representation ofVariable Fields in Error Messages

Representation	Type of Information
[dec]	Decimal number
[hex]	Hexadecimal number
[char]	Single character
[chars]	Character string
[enet]	Ethernet address (for example, 0000.DEAD.00C0)
[inet]	Internet address (for example, 12.128.2.16)
[t-line]	terminal line number in octal (or decimal if the decimal-tty service is enabled)

Error Message Traceback Reports

Some error messages which describe internal errors contain traceback information. This information is very important and should be included when you report a problem to Cisco Systems.

The following sample error message includes traceback information.

```
-Process= "Exec", level= 0, pid= 17
-Traceback= 1A82 1AB4 6378 A072 1054 1860
```

AppleTalk Error Messages

Error Message:

```
%AT-5-ADDRINUSE [chars]: AppleTalk address [network].[node] already in use
```

Explanation:

The initial hint address was in use. A search will be made for a valid address.

Recommended Action:

No action required.

Error Message:

```
%AT-6-ADDRUSED [chars]: using AppleTalk address [network].[node]
```

Explanation:

No hint address or a bad hint address was specified. This message indicates the AppleTalk address that will be used.

Recommended Action:

No action required.

Error Message:

```
%AT-6-AQUIREMODE [chars]: AppleTalk querying network for configuration
```

Explanation:

This is an advisory message only.

Recommended Action:

No action is required.

Error Message:

```
%AT-5-BADNEIGHBOR: [chars]:AppleTalk neighbor (advertising [chars]) is miscon-
figured, ignored
```

Explanation:

A neighboring router's AppleTalk configuration does not agree with this router's AppleTalk configuration. This can be due to the neighbor's network range not matching this router's network range.

Recommended Action:

Modify this router or the neighboring router's configuration so that the network ranges agree.

Error Message:

```
%AT-6-BADROUTE AppleTalk route to net [chars] has been marked bad
```

Explanation:

The system has not heard about a route within the required time-out period. The route has therefore been marked as bad. This may mean that a network has become unreachable for some reason—perhaps a broken connection. This message does not necessarily reflect an error condition.

Recommended Action:

Advisory message only. No action required.

Error Message:

```
%AT-5-COMPATERR1: [chars]: Neighbor at [chars] only supports AppleTalk
internets in compatibility mode
```

Explanation:

A neighboring router only supports extended AppleTalk networks with a cable range of 1, example: 25-25.

Recommended Action:

Configure the interface via which the Cisco router communicates with the neighboring router to have a cable range of one.

Error Message:

```
%AT-5-COMPATERR2: Zones for net [chars] are incompatible with some AppleTalk
routers in internet
```

Explanation:

One or more seed routers in the AppleTalk network have zone lists which are missing some zone names.

Recommended Action:

Be sure that all seed routers in an AppleTalk network are configured with identical zone lists.

Error Message:

```
%AT-5-COMPATERR3: Cable range [chars] is incompatible with some AppleTalk
routers in internet
```

Explanation:

One or more routers in the AppleTalk network has a cable range configuration which is not the same as the cable range configuration on this router.

Recommended Action:

It will be necessary to properly configure all routers to have identical cable ranges for a common cable.

Error Message:

```
%AT-6-CONFIGOK [chars]: AppleTalk configuration verified by [dec].[dec], port
activated
```

Explanation:

The AppleTalk configuration was verified by consulting the indicated router.

Recommended Action:

No action required.

Error Message:

```
%AT-5-DUPADDR: [chars]: Our AppleTalk node address used by [enet], restarting
line protocol
```

Explanation:

Another AppleTalk node on a common network interface has claimed the same AppleTalk address which this router was already using.

Recommended Action:

None. This Cisco router will restart AppleTalk processing on the common network interface to resolve the address conflict.

Error Message:

```
%AT-3-IFCONFLICT: [chars]: Configuration conflicts with port [chars]
(interface [chars])
```

Explanation:

An attempt was made to configure an interface to have the same AppleTalk address or cable-range as another interface on the same router.

Recommended Action:

Verify that you are not specifying an AppleTalk address or cable range used previously on this router and reconfigure the interface.

Error Message:

```
%AT-5-INTCLEARED: [chars]: Interface cleared, AppleTalk port reinitializing
```

Explanation:

This message is generated by performing the EXEC command **clear interface** with the interface type on an interface currently routing AppleTalk.

Recommended Action:

None. This message is for informational purpose.

Error Message:

```
%AT-5-INTDOWN [chars]: AppleTalk port disabled, Line protocol went down
```

Explanation:

An AppleTalk hardware interface has been disabled due to a bad serial line, a configuration command, or a bad interface.

Recommended Action:

If the port has not been intentionally disabled, this message may reflect a hardware problem. If so, service the hardware.

Error Message:

```
%AT-6-INTUP [chars]: AppleTalk port reinitializing, Line protocol came up
```

Explanation:

An AppleTalk port that was previously shut down has been restarted.

Recommended Action:

Advisory message only. No action required.

Error Message:

```
%AT-6-LOSTNEIGHBOR AppleTalk neighbor at [network].[node]has been deleted
```

Explanation:

A peer router has become unreachable.

Recommended Action:

Advisory message only. No action required.

Error Message:

```
%AT-3-MCMISMATCH [chars]: Netinfo zone multicast address disagrees with us
(theirs=[enet], ours=[enet])
```

Explanation:

A computed multicast address disagrees with that provided by another AppleTalk router. The other AppleTalk router may be misconfigured or faulty.

Recommended Action:

Correct the problem at the other router.

Error Message:

```
%AT-6-NEIGHBORUP: [chars]: AppleTalk neighbor at [chars] has restarted
```

Explanation:

A neighboring router was to which this router previously lost connectivity, has reappeared on the network.

Recommended Action:

None. This is an informational message.

Error Message:

```
%AT-3-NETDISAGREES [chars]: Net range does not match other routers, port
disabled
```

Explanation:

The configurations of one or more other AppleTalk routers are inconsistent with the configuration of this router.

Recommended Action:

Reconfigure one or more of the routers.

Error Message:

```
%AT-4-NETINVALID [chars]: Net [chars] is not valid for current AppleTalk
internet, port disabled
```

Explanation:

The network range overlaps with that of other networks in the internet.

Recommended Action:

Reconfigure the router.

Error Message:

```
%AT-6-NEWNEIGHBOR AppleTalk neighbor at [network].[node] has been added
```

Explanation:

The router discovered a new peer AppleTalk router.

Recommended Action:

Advisory message only. No action required.

Error Message:

```
%AT-6-NEWROUTE AppleTalk net [chars] has been added, [dec]hops, reported by
[network].[node]
```

Explanation:

A new network has come up.

Recommended Action:

Advisory message only. No action required.

Error Message:

```
%AT-3-NOADDRSAVAIL [chars]: Couldn't assign any AppleTalk address
```

Explanation:

No free node could be found on the interface.

Recommended Action:

If there are less than 250 nodes on your AppleTalk network, contact Cisco. Otherwise, you must split your AppleTalk network into smaller ones.

Error Message:

```
%AT-6-NODEWRONG: AppleTalk node [chars] is not valid for [chars], sent correct
configuration
```

Explanation:

An AppleTalk node has sent a GetNet Info request to this router, specifying an invalid network number for the source of the GetNet Info request.

Recommended Action:

None. This is an information message.

Error Message:

```
%AT-3-NOSRCADDR: [chars]: No AppleTalk node address available to use as packet
source
```

Explanation:

The interface out which a newly created AppleTalk packet is destined to be sent has no AppleTalk address.

Recommended Action:

Verify that the interface specified in the error message is properly configured.

Error Message:

```
%AT-3-NOTRUNNING AppleTalk not running
```

Explanation:

You have tried to show or change the AppleTalk configuration, but AppleTalk routing was not turned on.

Recommended Action:

Change the configuration to activate AppleTalk routing.

Error Message:

```
%AT-5-NOTSUPPORTED: [chars]: line protocol does not support this AppleTalk port
configuration
```

Explanation:

The interface specified has an encapsulation method which does not support fast switching.

Recommended Action:

This is an informational message. No action required.

Error Message:

`%AT-5-OLDMCI: [chars]: AppleTalk route cache disabled, MCI firmware is obsolete`

Explanation:

The firmware on the MCI controller card does not support fast switching of AppleTalk.

Recommended Action:

Verify that the MCI controller firmware for the interface specified in the error message is at level 1.7 or higher.

Error Message:

`%AT-6-ONLYROUTER [chars]: No other AppleTalk router found, port activated`

Explanation:

No other AppleTalk routers were found on the network attached to the port.

Recommended Action:

Advisory message only. No action required.

Error Message:

`%AT-6-ROUTENOTIFY AppleTalk route to net [chars] reported bad by [dec].[dec]`

Explanation:

A router has gone down somewhere on the AppleTalk network. The specified peer notified this router of the change.

Recommended Action:

Advisory message only. No action required.

Error Message:

`%AT-6-ROUTEOK: AppleTalk route to net [chars] has been restored`

Explanation:

A routing update has been received for a previously suspect route.

Recommended Action:

This message is for informational purposes only.

Error Message:

```
%AT-3-ZONEDISAGREES [chars]: Zone does not match other routers, port disabled
```

Explanation:

The configurations of one or more other routers are inconsistent with the configuration of this router.

Recommended Action:

Reconfigure one of the routers.

Error Message:

```
%AT-6-ZONEGC: AppleTalk zone, [chars], has been discarded
```

Explanation:

The router has purged an unused zone from the zone table.

Recommended Action:

This message is for informational purposes only.

BGP Error Messages

Error Message:

```
%BGP-3-NOMEMORY No memory for [chars] entry, resetting
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%CBUS-3-BIGBUF Controller [dec], Error ([hex]), Big Buffers [dec]
```

Explanation:

A hardware device did not respond appropriately to a request.

Recommended Action:

Make sure the device is functioning and is configured correctly.

Error Message:

```
%CBUS-3-BUFFER: Controller [dec], Error ([hex]), Buffersize = [dec], Bufferpool
= [dec], number [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

Contact Cisco Systems for assistance.

Error Message:

```
%CBUS-3-CORRUPT Controller [dec], wrote 0x[hex], read 0x[hex], loc0x[hex] -
dci_memtest()

%CBUS-3-DAUGHTER Unit [dec], daughter controller [dec] failed[chars] test -
interface disabled
```

Explanation:

A hardware component failed an internal diagnostic test.

Recommended Action:

Replace the malfunctioning device.

Error Message:

```
%CBUS-3-FDDIRSET Interface [chars], Error ([hex]) [chars] -fddi_reset()
%CBUS-3-FDDIRSETU Unit [dec], Error ([hex]) [chars] - fddi_reset()
%CBUS-3-INITERR Interface [dec], Error ([hex]) [chars] - cbus_init()
%CBUS-4-INTR Interface [dec], [chars] - cbus_interrupt()
```

Explanation:

A hardware device did not respond appropriately to a request.

Recommended Action:

Make sure the device is functioning and is configured correctly.

Error Message:

```
%CBUS-3-HSSIRSET: Interface [chars], Error ([hex]) [chars] - hssi_reset()
```

Explanation:

A hardware component did not respond to a reset command.

Recommended Action:

Contact Cisco Systems for assistance.

Error Message:

```
%CBUS-3-HSSIRSETU: Unit [dec], Error ([hex]) [chars] - hssi_reset()
```

Explanation:

A hardware component did not respond to a reset command.

Recommended Action:

Contact Cisco Systems for assistance.

Error Message:

```
%CBUS-3-NOMEMORY No memory for [chars]
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%CBUS-4-OUTHUNG Interface [chars] output hung, restarting cBus[dec] controller
- mci_output()
```

Explanation:

A cBus controller took too long to respond to a command, and was restarted by software.

Recommended Action:

If this message recurs, service the hardware.

Error Message:

```
%CBUS-3-TESTFAIL Unit [dec], failed [chars] test - interface disabled
```

Explanation:

A hardware component failed an internal diagnostic test.

Recommended Action:

Replace the malfunctioning device.

Error Message:

```
%CBUS-3-TXALLOC Error ([hex]) tx_allocate - cbus_init()
```

Explanation:

A hardware device did not respond appropriately to a request.

Recommended Action:

Make sure the device is functioning and is configured correctly.

Error Message:

```
%CBUS-3-ULTRARSET: Interface [chars], Error ([hex]) [chars] - ultra_reset()
```

Explanation:

A hardware component did not respond to a reset command.

Recommended Action:

Contact Cisco Systems for assistance.

Error Message:

```
%CBUS-3-ULTRARSETU: Unit [dec], Error ([hex]) [chars] - ultra_reset()
```

Explanation:

A hardware component did not respond to a reset command.

Recommended Action:

Contact Cisco Systems for assistance.

CHAOSnet Error Messages

Error Message:

```
%CHAOS-3-BADMASK Illegal host mask specified: [inet]
%CHAOS-3-SENDSELF Gateway tried forwarding to self
%CHAOS-3-ZEROSUBMASK Zero subnet mask
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%CLNS-4-EDATFAIL Encapsulation failed, dst= [chars]

%CLNS-4-EESHFAIL Interface [chars], encapsulation of ESH failed, HLEN [dec]
```

Explanation:

This message may occur when an interface is down and there is a static neighbor entry in the system's CLNS routing table.If this is not the case, the message indicates an internal software error.

Recommended Action:

Check the interface. If the interface is not down or there is no static neighbor entry for that interface, contact Cisco Systems.

Error Message:

```
%CLNS-3-NSAPES Invalid NSAP type in ES table: [hex] for [chars]

%CLNS-4-NSAPIS Invalid NSAP type in IS table: [hex] [dec]

%CLNS-4-REDIRECT Redirect found for non-route entry, dst=[chars], next-hop=
[chars]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

CSC2 Error Messages

Error Message:

```
%CSC2-4-BREAK AUX port hung because 'Transmit Break' would not clear
```

Explanation:

The UART (Universal Asynchronous Receiver/Transmitter) in the console serial interface is malfunctioning.

Recommended Action:

Service the console serial port hardware.

DECnet Error Messages

Error Message:

```
%DNET-3-HEARSELF Hello type [hex] for my address from [dec].[dec]via [chars]
```

Explanation:

The system is receiving its own DECnet packets. Possible causes:

1) A serial line is looped back

2) Another host with the same DECnet address is already present on the Ethernet

Recommended Action:

Check the serial lines (if present) and the DECnet configuration.

Error Message:

```
%DNET-4-MAPCON Map entry [dec].[dec] conflicts with adjacency to[dec].[dec]
```

Explanation:

Your DECnet configuration is incorrect. A host which is specified as nonlocal is present on your local network.

Recommended Action:

Correct the configuration.

Error Message:

```
%DNET-3-NOMEMORY No memory available for [chars]
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

EGP Error Messages

Error Message:

```
%EGP-3-NOPDB No pdb for [inet]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%EGP-3-TOOBIG Insufficient ([dec]) buffering for update message
```

Explanation:

An EGP update message was too large to fit into a single buffer.

Recommended Action:

Contact Cisco Systems for assistance.

HELLO Error Messages

Error Message:

```
%HELLO-2-NORDB: Redistributed IGRP without rdb
```

Explanation:

An internal software error has occurred.

Recommended Action:

Contact Cisco Systems for assistance.

IGRP Error Messages

Error Message:

```
%IGRP-3-NOSOCKET Unable to open socket for AS [dec]
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

ILAN Error Messages

Error Message:

```
%ILAN-4-RSETFAIL Interface [chars], unable to reset because [chars]
```

Explanation:

The InterLAN Ethernet controller was unable to reset.

Recommended Action:

Repair or replace the controller.

Error Message:

```
%IMP-4-DATERR Interface [chars], PSN data error
```

Explanation:

The Internet Message Processor has received corrupted data. This may be due to cable problems, a hardware problem in the IMP, or a malfunctioning IMP interface.

Recommended Action:

If this message recurs, service the hardware.

Error Message:

```
%IMP-6-GDOWN Interface [chars], PSN going down [chars]
```

Explanation:

The Internet Message Processor has received corrupted data. This may be due to cable problems, a hardware problem in the IMP, or a malfunctioning IMP interface.

Recommended Action:

If this message recurs, service the hardware.

Error Message:

```
%IMP-3-HDHBADMSG Interface [chars], HDH bad message mode
```

Explanation:

The system received bad data from the IMP.

Recommended Action:

If this message recurs, service the IMP and interface hardware.

Error Message:

```
%IMP-3-HDHHIOFF Interface [chars], HDH H/I bit off
```

Explanation:

The IMP origin bit was not set when it should have been.

Recommended Action:

If this message recurs, service the IMP and interface hardware.

Error Message:

`%IMP-3-LEADERR Interface [chars], PSN leader error, subtype [dec]`

`%IMP-3-LEADFMT Interface [chars], PSN unknown leader format`

`%IMP-3-LEADTYPE Interface [chars], PSN unknown leader type [dec] imperror`

Explanation:

The Internet Message Processor has received corrupted data. this may be due to cable problems, a hardware problem in the IMP, or a malfunctioning IMP interface.

Recommended Action:

If this message recurs, service the hardware.

Error Message:

`%IMP-5-PSNSTATE: Interface [chars], PSN is [chars]`

Explanation:

The connection has gone up or down, depending upon the message in `[chars]`

Recommended Action:

Determine why the PSN has changed state and take appropriate action.

Error Message:

`%IMP-6-RESET Interface [chars], PSN interface reset`

Explanation:

This is an advisory message only.

Recommended Action:

No action is required.

Error Message:

```
%IP-4-CLASS Bad IP address [inet] and mask [inet]in class_resolve()
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%IP-3-DESTHOST src=[inet], dst=[inet], NULL desthost
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%IP-4-DUPADDR Duplicate address [inet] on [chars], sourced by [enet]
```

Explanation:

Another system is using your IP address.

Recommended Action:

Change the IP address of one of the two systems.

IPRT Error Messages

Error Message:

```
%IPRT-2-COMPRESS Bad route_compress() call, sdb= [hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%IPRT-3-NOMEMORY No memory available for [chars]
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%IPRT-4-SAMENET [chars] and [chars] are on the same network or subnet
```

Explanation:

The system configuration is inconsistent with the structure of the internet.

Recommended Action:

Change the system configuration to reflect the actual network topology.

Error Message:

`%LANCE-4-BABBLE Unit [dec], babble error, csr0 = 0x[hex]`

Explanation:

An Ethernet interface is malfunctioning.

Recommended Action:

Service the hardware.

Error Message:

`%LANCE-3-BADENCAP Unit [dec], bad encapsulation in idb->enctype= 0x[hex]`

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%LANCE-3-BADUNIT Bad unit number [dec]`

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%LANCE-5-COLL Unit [dec], excessive collisions. TDR=[dec]`

Explanation:

An Ethernet cable is broken or unterminated, or the transceiver is unplugged.

Recommended Action:

Service the hardware.

Error Message:

```
%LANCE-0-INITFAIL Unit [dec], initialization timeout failure,csr[dec]=0x[hex]
```

Explanation:

The hardware failed to initialize correctly.

Recommended Action:

Service the hardware.

Error Message:

```
%LANCE-5-LATECOLL Unit [dec], late collision error
```

Explanation:

An Ethernet transceiver is malfunctioning, or the Ethernet is overloaded, or the Ethernet cable is too long.

Recommended Action:

Service the hardware.

Error Message:

```
%LANCE-5-LOSTCARR Unit [dec], lost carrier. Transceiver problem?
```

Explanation:

An Ethernet transceiver is unplugged or faulty.

Recommended Action:

Service the hardware.

Error Message:

```
%LANCE-0-MEMERR Unit [dec], memory error, csr0 = 0x[hex]
```

Explanation:

An Ethernet interface detected a hardware problem.

Recommended Action:

Service the hardware.

Error Message:

`%LANCE-0-NOMEMORY Unit [dec], no memory for [chars]`

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

`%LANCE-3-OWNERR Unit [dec], buffer ownership error`

Explanation:

An Ethernet interface is malfunctioning or an internal software error has occurred.

Recommended Action:

Service the hardware or contact Cisco Systems.

Error Message:

`%LANCE-3-SPURIDON Unit [dec], spurious IDON interrupt`

Explanation:

An Ethernet interface generated a spurious Initialization Done interrupt.

Recommended Action:

Service the hardware.

Error Message:

`%LANCE-3-SPURINT Unit [dec], spurious interrupt, csr0 = 0x[hex]`

Explanation:

An Ethernet interface detected a spurious interrupt.

Recommended Action:

Service the hardware.

Error Message:

```
%LANCE-3-UNDERFLO Unit [dec], underflow error
```

Explanation:

The Ethernet hardware is requesting data faster than the system can supply it. This should never happen unless a serious malfunction has occurred.

Recommended Action:

Contact Cisco Systems.

LAPB Error Messages

Error Message:

```
%LAPB-4-CTRLBAD Interface [chars], Invalid control field
```

Explanation:

A bad X.25 packet was received.

Recommended Action:

If this message recurs, check the X.25 serial line and the devices attached to that line.

Error Message:

```
%LAPB-4-FRAMEERR Interface [chars], Frame error: CF 0x[hex], VS[dec] [char] VR
[dec], Reason 0x[hex]
```

Explanation:

A FRMR packet was received. This may be due to a noisy serial line, an overloaded X.25 packet switch, or corrupted data.

Recommended Action:

If this message recurs, service the serial line and other devices attached to that line.

Error Message:

```
%LAPB-4-INFOBAD Interface [chars], Info field not permitted
```

```
%LAPB-4-INVNR Interface [chars], Invalid NR value
```

```
%LAPB-4-N1TOOBIG Interface [chars], N1 too largebadx25pkt
```

Explanation:

A bad X.25 packet was received.

Recommended Action:

If this message recurs, check the X.25 serial line and the devices attached to that line.

Error Message:

```
%LAPB-3-NOINPIDB Input idb not set
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%LAPB-3-NULLPAK Interface [chars], NULL packet ptr, rvr [dec],vs [dec], vr
[dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%LAPB-6-OUTPUT Interface [chars], LAPB O [chars] [chars]
```

Explanation:

An FRMR packet was discarded due to congestion.

Recommended Action:

If this message recurs, contact Cisco Systems.

LAT Error Messages

Error Message:

```
%LAT-3-BADDATA Tty[t-line], Data pointer does not correspond to current packet
```

```
%LAT-3-BUFFULL Tty[t-line], data buffer full with count [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%LAT-3-ETHERNET No Ethernet to run LAT over
```

Explanation:

The LAT facility cannot be used unless the system is connected to and active on an Ethernet.

Recommended Action:

Make sure your system is configured correctly, and that the Ethernet interface, if present, is operational.

Error Message:

```
%LAT-3-ETHERNET Tty[t-line], Output data ptrs out of sync with byte count
```

```
%LAT-3-NULLIDB: Null IDB pointer with destination [enet]
```

```
%LAT-3-QBSPACED Queue block at [hex] not found for HI connection
```

```
%LAT-3-REUSE Tty[t-line], Attempt to re-use slot array, empty =[dec], fill =
[dec]
```

```
%LAT-3-SIGERR Tty[t-line], Unknown signal [dec] in [chars]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%LINK-3-BADETHER Interface [chars], Bad ethernet encap code
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%LINK-5-CHANGED Interface [chars], changed state to [chars]
```

Explanation:

This is an advisory message only.

Recommended Action:

No action is required.

Error Message:

```
%LINK-2-NOSOURCE Source idb not set
%LINK-3-TOOBIG Interface [chars], Output packet size of [dec] bytes too big
%LINK-2-UENCAP Unknown line encapsulation code [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

MAILBOX Error Messages

Error Message:

`%MAILBOX-3-BADDATA: Bad data in mailbox test, got 0x[hex], expected 0x[hex]`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%MAILBOX-3-FLAGSTAT: Timeout while waiting for FLAGSTAT, status=0x[hex]`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%MAILBOX-3-INITPC2: Timeout waiting for PC2 during initialization, status=0x[-hex]`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%MAILBOX-3-INTERR: Bad mailbox interrupt = 0x[hex]`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%MAILBOX-3-MAIL020: Timeout while waiting for MAIL020, status=0x[hex]`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%MAILBOX-3-MAILFAIL: Mailbox failed to initialize`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%MAILBOX-3-PC2: Timeout while waiting for PC2, status=0x[hex]`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%MAILBOX-3-SPURINT: Spurious mailbox interrupt = 0x[hex], MAIL020 not asserted`

Explanation:

A hardware problem was detected in a Chipcom interface board.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

MCI Error Messages

Error Message:

```
%MCI-5-INPUTERR Interface [chars] excessive input error rate
```

Explanation:

The input error rate was so high that the interface was temporarily disabled. The interface will be automatically re-enabled in 30 seconds.

Recommended Action:

If this message recurs, check the communications lines.

Error Message:

```
%MCI-4-NOKEEPALIVE Interface [chars] keepalive not sent
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%MCI-5-OBSOLETE Obsolete MCI firmware: can't route [chars] and bridge simulta-
neously
```

Explanation:

The firmware on your MCI controller card is out of date.

Recommended Action:

Upgrade your MCI firmware.

Error Message:

```
%MCI-4-RSETFAIL Interface [chars] failed to reset properly in[chars], code
0x[hex]

%MCI-3-RXINDEX Unit [dec], invalid RX index [dec]

%MCI-3-SETUPERR Unit [dec], Error ([hex]) on setup, index [hex],restarting con-
troller - mci_interrupt()
```

Explanation:

A hardware device did not respond appropriately to a request.

Recommended Action:

Make sure the device is functioning and is configured correctly.

Error Message:

```
%MCI-4-TESTFAIL Unit [dec] failed [chars] test, skipping
```

Explanation:

A hardware component failed an internal diagnostic test.

Recommended Action:

Replace the malfunctioning device.

MK5 Error Messages

Error Message:

```
%MK5-3-BADENCAP Unit [dec], bad encapsulation inidb->enctype = 0x[hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%MK5-3-BADUNIT Bad unit number [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%MK5-0-INITFAIL Unit [dec], initialization timeout failure,csr[dec]=0x[hex]

%MK5-0-INITNOPPRIM Unit [dec], initialization failure - NoCSR1_PPRIM_INIT_CONF,
csr1 = 0x[hex]

%MK5-0-INITUERR Unit [dec], initialization CSR1_UERR failure,csr1=0x[hex]
```

Explanation:

The hardware failed to initialize correctly.

Recommended Action:

Service the hardware.

Error Message:

```
%MK5-0-MEMERR Unit [dec], memory error, csr0 = 0x[hex]
```

Explanation:

A network serial interface detected a hardware problem.

Recommended Action:

Service the hardware.

Error Message:

```
%MK5-0-NOMEMORY Unit [dec], no memory for [chars]
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%MK5-3-ODDSTART Interface [chars], Odd datagram start = 0x[hex], pak =0x[hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%MK5-3-OUTENCAP Unit [dec], bad output packet encapsulation: 0x[hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%MK5-3-PLOSTERR Unit [dec], provider primitive lost,csr0 = 0x[hex], csr1 =
0x[hex]

%MK5-3-PPRIMERR Unit [dec], unexpected provider primitive,csr0 = 0x[hex], csr1
= 0x[hex] csr0 = 0x[hex], csr1 = 0x[hex]
```

```
%MK5-3-SPURPPRIMERR Unit [dec], spurious provider primitive,csr0 = 0x[hex],
csr1 = 0x[hex]

%MK5-3-UPRIMERR Unit [dec], user primitive error,csr0 = 0x[hex], csr1 = 0x[hex]
```

Explanation:

A network serial interface detected a hardware problem.

Recommended Action:

Service the hardware.

PAD Error Messages

Error Message:

```
%PAD-3-GETLINE Tty[t-line], bad return code [dec] fromx3_getline()

%PAD-2-GETSTRING: Pad[t-line], Invalid lci idb pointer [hex], pcb: [hex], lci
[hex]

%PAD-2-INTR [chars] called at interrupt level [hex]

%PAD-2-PUTSETUP Tty[t-line], buffer already setup
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

PPP Error Messages

Error Message:

```
%PPP-4-CONFNAK fsm_rconfnak([hex]) - possible CONFNAK loop
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%PPP-6-LOOPED The line appears to be looped back
```

Explanation:

The communications line appears to be echoing the characters that are sent to it.

Recommended Action:

Check your data communications equipment to make sure it is configured correctly.

PR Error Messages

Error Message:

```
%PR-3-DAEMON: lpt_daemon assigned to a non-lpt
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%PR-3-WRONGWATCH: Tty [t-line], lptwatch called for non-lpt
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

PUP Error Messages

Error Message:

```
%PUP-2-ZEROSUB: Zero subnet mask
```

Explanation:

The IP network used in the **pup map** command is not subnetted.

Recommended Action:

Enter a subnetted IP address as required by the **pup map** command.

RIP Error Messages

Error Message:

```
%RIP-3-NOSOCKET Unable to open socket
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%RSRB-4-BADLEN Peer [dec]/[inet], [chars], bad length [dec], trn [dec]
```

```
%RSRB-4-BADLENIP Peer [dec]/[inet], [chars], bad length [dec],trn [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%RSRB-3-BADVERSIONIF IFin: [chars]: version mismatch, mine [dec], theirs [dec]
```

Explanation:

The remote end of a direct serial peer is running the wrong version of the system software. Either the local end, the remote end, or both are not up to date.

Recommended Action:

Call Cisco Systems for an update.

Error Message:

```
%RSRB-3-BADVERSIONTCP [chars]: [dec]/[inet]: version mismatch, mine [dec],
theirs [dec]
```

Explanation:

The remote end of a TCP remote peer is running the wrong version of the system software. Either the local end, the remote end, or both are not up to date.

Recommended Action:

Call Cisco Systems for an update.

Error Message:

```
%RSRB-4-BADVRE Bad vre type

%RSRB-4-CONIPST Peer [dec]/[inet], CONN, illegal state [dec]

%RSRB-4-CONNILLSTATE Peer [dec]/[inet], CONN, illegal state [dec]

%RSRB-4-CONNSTAT Peer [dec]/[interface], IFin, bad connection state [dec]

%RSRB-3-HDRNOVRP Peer [inet], HDR, no vrp

%RSRB-4-HDRRECV Peer [dec]/[inet], HDR, recv state invalid, not empty [dec]

%RSRB-3-HDRVRP Peer [dec]/[inet], HDR, vrp state wrong, [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%RSRB-3-IFERR [chars]: [chars]: [chars], op [hex], len [dec], trn [dec]
```

Explanation:

The remote end of a direct serial RSRB connection has detected a configuration problem or traffic that is not recognized by the configuration.

Recommended Action:

Examine the configuration on both sides of the serial connection for possible problems. Examine the traffic being offered for propagation with respect to the configuration. The destination target ring is denoted by the value of *trn*.

Error Message:

```
%RSRB-4-ILLPEER Peer [chars] [[hex]], illegal state [dec]

%RSRB-4-LOCAL Unit [dec], local/vring set simultaneously,vrn [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%RSRB-3-NOMEMORY Unit [dec], no memory for [chars]
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%RSRB-3-NOTREM Null idb and not remote

%RSRB-4-PEERSTAT Peer [chars], wrong state [dec]

%RSRB-4-OPTNULL: Remopened and t NULL
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%RSRB-3-SENDPUNTIF [chars]: sent [chars] to [dec]/[chars]
```

Explanation:

The local end of a direct serial RSRB connection has detected a configuration problem or traffic that is not recognized by the configuration.

Recommended Action:

Examine the configuration on both sides of the serial connection for possible problems. Examine the traffic being offered for propagation with respect to the configuration.

Error Message:

```
%SBE-5-BADBREAK Questionable break? buffer [dec];[hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%SBE-5-HUNGUP Interface [chars], hung up - resetting
```

```
%SBE-4-RSETFAIL Failed to reset interface Serial[dec]
```

Explanation:

The SBE hardware is malfunctioning, or the firmware is improperly inserted.

Recommended Action:

Service the hardware.

IP Security Error Messages

Error Message:

```
%SEC-2-NOOPT Box secured, no option on internal packet

%SEC-2-NOTSEC First opt in tcb not BASIC security

%SEC-2-SECINS Security opt in tcb not SECINSERT
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%SEC-4-TOOMANY Box secured, too many options on internal packet
```

Explanation:

No room for all desired IP header options. Packet discarded.

Recommended Action:

Configure for fewer IP header options.

SLIP Error Messages

Error Message:

```
%SLIP-2-BADQUOTE Impossible quoted character [hex]

%SLIP-2-BADSTATE Impossible input state [hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

STUN Error Messages

Error Message:

`%STUN-3-BADCONN: CONN: bad connection ([dec]), peer: [chars]`

`%STUN-3-BADLENOP: [chars]: bad len or unknown op, op [dec], len [dec]`

`%STUN-3-BADMAGIC: [chars]: wrong magic, mine [hex], theirs [hex] ([dec])`

`%STUN-3-BADMAGICTCP: [chars]: peer [chars], wrong magic, min [hex], theirs [hex]`

`%STUN-3-BADPASSIVEOPEN: passive open from [inet]([dec]) -> [dec] failed`

`%STUN-3-CONNILLSTATE: CONN: Peer [chars], illegal state [dec]`

Explanation:

An internal software error has occurred.

Recommended Action:

If any of these messages recur, contact Cisco Systems for assistance.

Error Message:

`%STUN-6-CONNOPENFAIL: CONN: peer [chars] open failed, [chars] [[dec]]`

Explanation:

An attempt to connect to a remote TCP STUN peer has failed.

Recommended Action:

Verify that the remote peer is accessible from this router, that it is running software capable of supporting the Serial Tunnel, and that it is configured correctly.

Error Message:

`%STUN-6-ERR: [chars]: [chars]: [chars], op [hex], len [dec]`

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%STUN-3-NOPROTMEM: No memory for STUN protocol definition of [chars]`

Explanation:

The system does not have sufficient memory to support some or all of the Serial Tunnel configuration.

Recommended Action:

Reduce system configuration and take other actions as appropriate to increase memory available for Serial Tunnel operations.

Error Message:

`%STUN-6-OPENED: [chars]: peer [chars] opened, [previous state [chars]]`

`%STUN-6-OPENING: CONN: opening peer [chars], [dec]`

`%STUN-6-PASSIVEOPEN: passive open [inet]([dec]) -> [dec]`

Explanation:

A connection attempt to a remote peer has completed (`OPENED`, `PASSIVEOPEN`) successfully or is in the process of being opened (`OPENING`).

Recommended Action:

This is good, expected behavior. Nothing need be done.

Error Message:

`%STUN-6-PEERSHUTDOWN: shutting down peer [chars] on [chars]`

Explanation:

A connection to a remote peer is being shut down. This is typically the result of user intervention in Serial Tunnel reconfiguration or disabling.

Recommended Action:

This is good, expected behavior. Nothing need be done.

Error Message:

`%STUN-4-PEERSTATE: Peer [chars], wrong state [dec] ([dec])`

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

`%STUN-6-RECONNECT: PHDR: reconnect from peer [chars]`

Explanation:

A remote peer has reestablished a connection to this router.

Recommended Action:

This is good, expected behavior. Nothing needs be done.

Error Message:

`%STUN-3-SENDPUNT: [chars]: sent [chars] to [chars]`

`%STUN-3-SENDPUNTTCP: [chars]: sent [chars] to ([[dec]])[inet]`

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

`%STUN-6-TCPFINI: peer [chars] closed [previous state [chars]]`

Explanation:

A remote peer has closed a Serial Tunnel connection with this router.

Recommended Action:

Examine the other router to see why it closed this connection with this peer. (This can be caused by normal events, such as reconfiguration.)

Error Message:

`%STUN-6-TCPPEERSHUT: [chars] [chars], [inet]([dec])`

Explanation:

This route has closed a Serial Tunnel connection with a remote peer.

Recommended Action:

Examine this router to see why it closed this connection with this peer. (This can be caused by normal events, such as reconfiguration.)

Error Message:

%SYS-2-ALREADYFREE: Buffer [hex] already in free pool [chars]

%SYS-3-BADDISP Bad dispose code [hex] in [chars]

%SYS-2-BADPID Bad pid [dec] for tty [t-line]

%SYS-2-BADSHARE Bad sharecount in [chars], ptr=[hex],count=[hex]

%SYS-2-CFORKLEV cfork at level [dec]

%SYS-5-CONFIG: Configured from [chars] by [inet]

%SYS-5-CONFIG_I: Configured from [chars] by [chars]

%SYS-5-CONFIG_M: Configured via MOP from [chars] by [enet]

%SYS-2-FREEBAD Attempted to free memory at [hex], not part of buffer pool

%SYS-2-FREEFREE Attempted to free unassigned memory at[hex], deallocated at [hex]

%SYS-2-GETBUF Bad getbuffer, bytes= [dec]

%SYS-3-HARIKARI Process [chars] top-level routine exited

%SYS-2-HEADER Attempt to return buffer with invalid bufferheader r, ptr= [hex]

%SYS-2-INLIST Buffer in list, ptr= [hex]

%SYS-2-INLIST1: Buffer in list, ptr= [hex], caller= [hex]

%SYS-2-INPUTQ INPUTQ set, but no idb, ptr=[hex]

%SYS-2-INSCHED [chars] within scheduler

%SYS-2-INTSCHED [chars] at level [dec]

%SYS-2-INVALID Free([dec]): [hex] [hex] [hex] [hex] [hex][hex] [hex] [hex] [hex] [hex] -malloc([dec]): [hex] [hex] [hex] [hex] [hex] [hex] [hex] [hex] [hex] [hex]

%SYS-2-INVFREE Invalid free list

%SYS-2-INVRETURN Invalid returned memory 0x[hex]

%SYS-2-LINKED Bad enqueue of [hex] in queue [hex]

%SYS-2-LINKED Bad p_dequeue of [hex] in queue [hex]

%SYS-2-LINKED Bad p_requeue of [hex] in queue [hex]

%SYS-2-LINKED Bad requeue of [hex] in queue [hex]

%SYS-2-NOBLOCK [chars] with blocking disabled

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%SYS-6-NOBRIDGE Bridging software not present
```

Explanation:

Your system is not equipped to be a bridge.

Recommended Action:

Install the bridging software option.

Error Message:

```
%SYS-3-NOPROC Process table full

%SYS-2-NOTDEAD Process self-destruction failed

%SYS-2-NOTQ p_unqueue didn't find [hex] in queue [hex]

%SYS-2-NOTQ unqueue didn't find [hex] in queue [hex]

%SYS-3-NULLIDB Null idb in [chars]

%SYS-2-QCOUNT Bad dequeue [hex] count [dec]

%SYS-2-QCOUNT Bad p_dequeue [hex] count [dec]

%SYS-2-QCOUNT Bad p_unqueue [hex] count [dec]

%SYS-2-QCOUNT Bad unqueue [hex] count [dec]

%SYS-4-REGEXP [chars]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%SYS-5-RELOAD Reload requested

%SYS-5-RESTART System restarted
```

Explanation:

This is an advisory message only.

Recommended Action:

No action is required.

Error Message:

```
%SYS-2-RETBUF Bad retbuffer, ptr= [hex]

%SYS-2-RETBUF1: Bad retbuffer, ptr= [hex], caller= [hex]

%SYS-2-SELF s_setblock() on own process

%SYS-2-SELFLINKED: Buffer [hex] linked to itself in free pool [chars]

%SYS-2-SHARED Attempt to return buffer with sharecount [dec],ptr= [hex]

%SYS-2-SHARED1: Attempt to return buffer with sharecount [dec], ptr= [hex],
caller= [hex]

%SYS-2-SMASHED Smashed block at [hex], next [hex], prev [hex],size [dec]

%SYS-2-SMASHEDINFO: Smashed block last freed [hex], allocated by [hex], name
[chars]

%SYS-3-SOCKUNKN Unknown socket protocol [dec]

%SYS-6-STACKLOW Stack for [chars] [chars] running low, [dec]/[dec]

%SYS-2-SPEC Trying to set unknown special character [dec] to [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%SYS-3-TTYMEM Tty[t-line], no memory to save tty default parameters
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

TAC Error Messages

Error Message:

```
%TAC-6-SENDTMO Send type [dec] to [inet] timed out
```

Explanation:

A background TACACS notification (enabled with the command **tacacs notify**) was not acknowledged by the TACACS server processor within the timeout period (five minutes). The information contained in that notification has been lost. This may interfere with accounting or auditing on the server.

This condition arises when the TACACS server is misconfigured, has crashed, or has become unreachable via the network.

Recommended Action:

Check the TACACS server and the network attached to it.

Error Message:

```
%TAC-4-UNEXREP Reply for non-existent request, [dec] on queue
```

Explanation:

The TACACS facility received a message it was not expecting. This may occur when a TACACS server sends duplicate responses, or when it responds to a request which has already timed out. It may also be due to an internal software problem.

Recommended Action:

If this message recurs, contact Cisco Systems.

TCP Error Messages

Error Message:

```
%TCP-2-BUFFER Tty[t-line], buffering bug
```

```
%TCP-2-PUTBYTE Tty[t-line], tcp_putbyte() with blocking disabled
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

TMQ Error Messages

Error Message:

```
%TMQ-3-NOTFOUND: TMQ, Attempt to delete entry not in queue
```

Explanation:

An attempt was made to delete an entry not in the queue.

Recommended Action:

Informational message only.

Telnet Error Messages

Error Message:

```
%TN-2-BADLOGIN Bad login string pointer 0x[hex]

%TN-3-BADSTATE Illegal state [dec]

%TN-3-READLINE Unknown return code [dec] from telnet_readline()
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Token Ring Error Messages

Error Message:

```
%TR-3-ASYNCUSE Unit [dec], [chars], async in use
```

Explanation:

Another internal process is communicating to the given Token Ring interface.

Recommended Action:

Wait one minute and try again. If the condition persists then it probably indicates an internal software error. Contact Cisco Systems.

Error Message:

```
%TR-4-BADADAPT Unit [dec], adapter not responding 0x[hex] 0x[hex]0x[hex]
0x[hex]
```

Explanation:

The Token Ring interface is not responding to initialization.

Recommended Action:

Check the Token Ring card's configuration and make sure it is in the correct slot. The Token Ring card (CSC-R) is a Multibus master; it must be part of a contiguous group of bus master cards that includes the CPU. In most configurations the Token Ring interface will be the card immediately adjacent to the CPU. If the interface is in the correct slot, this message indicates a probable hardware failure. Contact Cisco Systems.

Error Message:

```
%TR-3-BADMUL Unit [dec], Can't set TR address to hardware multicast address
[enet]
```

Explanation:

An attempt was made to set the Token Ring interface MAC address to a reserved multicast address.

Recommended Action:

Check your configuration. Make sure that your XNS and/or Novell Token Ring addresses have not inadvertently been set to reserved multicast addresses.

Error Message:

```
%TR-3-BADRNGNUM: Unit [dec], ring number ([dec]) doesn't match established
number ([dec])
```

Explanation:

The number you have configured for the local ring does not match the value currently in use on the ring.

Recommended Action:

Check the configuration to make sure you used the correct ring number. If it is correct, check the configuration of all other bridges on the ring to make sure they are using the same ring number.

Error Message:

```
%TR-3-BADSTART Unit [dec], Start completion and wrong idb state -state= [dec]

%TR-4-BKGND Unit [dec], bkgnd int: last packet out [dec]ms

%TR-4-BKGNDINT Unit [dec], bkgnd int: [hex][hex] [hex][hex][hex][hex]
[hex][hex]

%TR-3-COMPFAIL Unit [dec], Tbuf completion failure, result 0x[hex]

%TR-3-FILTFAIL Unit [dec], SRB filter set failed: 0x[hex] 0x[hex]0x[hex]
0x[hex] (blk 0x[hex]) downing

%TR-3-GETCONF Unit [dec], get_config failed, result 0x[hex] 0x[hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

Error Message:

```
%TR-3-INCOM Unit [dec], incompatible firmware: system 0x[hex],interface 0x[hex]
```

Explanation:

The Token Ring interface contains firmware that is incompatible with the system software.

Recommended Action:

Call Cisco Systems to upgrade the Token Ring Monitor firmware and/or the system software.

Error Message:

```
%TR-4-INTFAIL Unit [dec] interface failure: 0x[hex] 0x[hex] 0x[hex],idb state
[dec]
```

Explanation:

The Token Ring Monitor firmware has detected a fatal error due either to an internal software problem or a hardware failure.

Recommended Action:

The error message should be copied down exactly as it appears and conveyed to Cisco Systems.

Error Message:

```
%TR-3-MODFAIL Unit [dec], tra_modify failed, result 0x[hex] 0x[hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%TR-3-NOMEMORY Unit [dec], no memory for [chars]
```

Explanation:

The requested operation could not be accomplished because of a low memory condition.

Recommended Action:

Reduce other system activity to ease memory demands. If conditions warrant, upgrade to a larger memory configuration.

Error Message:

```
%TR-3-NOPAK Unit [dec], Tbuf completion and no pakout
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%TR-3-OPENFAIL Unit [dec], open failed: [chars], [chars] (0x[hex])
```

Explanation:

The Token Ring interface was unable to insert itself into the ring. This is an indication of a busy or broken ring. The first character string in this error message will indicate the stage of initialization at which error occurred, and the second will describe the error.

Recommended Action:

Try to open the interface again. This can generally be accomplished by issuing the **clear interface tokenring** command. If the error message recurs, contact Cisco Systems.

Error Message:

`%TR-3-OPENFAIL2: Unit [dec], open failed: check the lobe cable connection.`

Explanation:

The Token Ring interface was unable to insert itself into the ring, and the error code returned indicates a wiring problem.

Recommended Action:

Check the cable connecting the router to the Token Ring MAU, and try to open the interface again. This can generally be accomplished by issuing the **clear interface tokenring** command. If the error message recurs, contact Cisco Systems.

Error Message:

`%TR-3-PANIC Unit [dec], panic [hex] [hex] [hex]`

Explanation:

The Token Ring Monitor firmware has detected a fatal error which indicates an impending interface failure.

Recommended Action:

The error message should be copied down exactly as it appears and included together with the interface error message. All this information should be conveyed to Cisco Systems.

Error Message:

`%TR-3-PANICINF Unit [dec], PI [hex] [hex] [hex] [hex][hex] [hex]`

Explanation:

This message is similar to the `TR-3-PANIC` error message but indicates a nonfatal error. This message appears in very unusual situations that should not arise in normal operation.

Recommended Action:

The error message should be copied down exactly as it appears and conveyed to Cisco Systems.

Error Message:

`%TR-3-PANICTYPE: Unit [dec], [chars] error`

Explanation:

This message is similar to the `TR-3-PANIC` error message but indicates a nonfatal error. This message appears in very unusual situations that should not arise in normal operation.

Recommended Action:

The error message should be copied down exactly as it appears and conveyed to Cisco Systems.

Error Message:

```
%TR-3-RESETFAIL: Unit [dec], reset failed, error code [hex].
```

Explanation:

An internal software error occurred.

Recommended Action:

If message recurs, contact Cisco Systems.

Error Message:

```
%TR-4-SNKBRIDG Unit [dec], removing invalid bridge
```

Explanation:

The system could not start a Token Ring source-route bridge because the interface in question does not have the appropriate Token Ring Monitor firmware or hardware.

Recommended Action:

Call Cisco Systems.

Error Message:

```
%TR-3-SETFAIL Unit [dec], couldn't set interface to physical address [enet]
(0x[hex])
```

Explanation:

An illegal MAC address was specified for this interface.

Recommended Action:

Normally the Token Ring interface uses the burned-in address contained in a PROM on the board. However certain protocol stacks require the MAC address to be modified to an illegal value. Check your configuration for possible problems. If the problem persists call Cisco Systems.

Error Message:

```
%TR-3-SETFUNFAIL: Unit [dec], set functional address failed (code [hex])
```

Explanation:

An internal software error occurred.

Recommended Action:

If message recurs, contact Cisco Systems.

Error Message:

`%TR-3-SETGRPFAIL: Unit [dec], set group address failed (code [hex])`

Explanation:

An internal software error occurred.

Recommended Action:

If message recurs, contact Cisco Systems.

Error Message:

`%TR-3-SMTSTATFAIL: Unit [dec], get_smt: get smt stats failed - result [hex]`

Explanation:

An internal software error occurred.

Recommended Action:

If message recurs, contact Cisco Systems.

Error Message:

`%TR-3-STARTFAILED Unit [dec], start failed, result 0x[hex],reason 0x[hex]`

Explanation:

The Token Ring Monitor firmware attempted to insert the system into the ring and the chip set detected a non-standard fatal error.

Recommended Action:

This message indicates a problem with the Token Ring chip set hardware. Call Cisco Systems.

Error Message:

`%TR-6-STATE [chars]`

Explanation:

This message is displayed when the Token Ring's status has changed as determined by the chip set. This information is also used to automatically determine if the interface is still usable to propagate network traffic.

Recommended Action:

This is an advisory message only. No action is required.

Error Message:

```
%TR-3-STATFAIL Unit [dec], get_static: get stats failed -Async: [hex] [hex]
[hex] [hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%TR-6-STATRING [chars] ([hex]) LIV 0x[hex]
```

Explanation:

This message is displayed when the Token Ring's status has changed as determined by the chip set. This information is also used to automatically determine if the interface is still usable to propagate network traffic.

If the hexadecimal status code in this message is zero (0), it indicates that the ring has returned to a normal state. Otherwise, the first hexadecimal number is the sum of one or more status codes which indicate unusual conditions on the ring. The meaning of each status code is shown in Table A-4.

Table A-4 Token Ring Status Codes

Code	Explanation	Fatal?
0x8000	Signal loss	Yes
0x4000	Hard error	Yes
0x2000	Soft error	No
0x1000	Transmit beacon	Yes
0x0800	Lobe wire fault	Yes
0x0400	Auto removal error	Yes
0x0100	Remove request received	Yes
0x0080	Counter overflow	No
0x0040	Single station	No
0x0020	Ring recovery	No

Recommended Action:

Check the ring for the indicated condition.

Error Message:

```
%TR-3-TRACMD tra_cmd: bad state: cmd 0x[hex] 0x[hex],result 0x[hex] 0x[hex]
(blk 0x[hex], iob 0x[hex])

%TR-4-UNKNOWNCOMP Unit [dec], Unknown command completion:cmd 0x[hex], rslt
0x[hex]

%TR-3-UNKNOWNT2M Unit [dec], Unknown t2m command [hex] [hex]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

VINES Error Messages

Error Message:

```
%VINES-2-NEIGHBOR vines_update called with null neighbor
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

Error Message:

```
%VINES-2-ROUTEADD: Cannot add route for [hex]:[hex], neighbor doesn't exist.
```

Explanation:

An internal software error has occurred.

Recommended Action:

If this message recurs, contact Cisco Systems for assistance.

X.25 Error Messages

Error Message:

```
%X25-2-BADLIST Regular expression access check with bad list [dec]

%X25-2-ILLP4 Illegal state [chars] when P4

%X25-3-INTIMEQ Interface [chars], LCN [dec] already in timer queue, new time
[dec]

%X25-3-NOLCI Delete: lci [dec] not found in [chars] table

%X25-3-NOTFINDI Can't find address [inet] to delete

%X25-3-NOTFINDS Can't find address [chars] to delete

%X25-3-REWRITE Error rewriting called address in X25-Switch

%X25-3-SPURD1 Spurious D1 timer wakeup on LCI [dec]
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.

XNS Error Messages

Error Message:

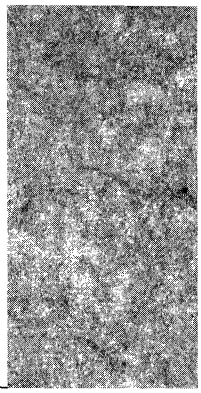
```
%XNS-3-BADPATHS Invalid number of paths ([dec]) for [dec]

%XNS-2-BADROUTE Error [chars] route - null table
```

Explanation:

An internal software error has occurred.

Recommended Action:

If any one of these messages recurs, contact Cisco Systems for assistance.



Quick Index of System Error Messages

The following quick index lists all the error messages and the pages on which they are documented. The index, like the text, is arranged alphabetically first by facility code and then by mnemonic.

Symbols

A

C

D

E

I

L

M

P

R

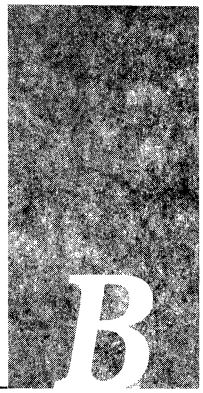
S

T

V

X

Appendix B
Access List Summary



This appendix summarizes the command syntax and number ranges or symbolic names used for the access lists supported by the Cisco software. The summaries are listed by protocol, in alphabetical order. The command to create the access list is given first, followed by the command you use to assign the access list number to an interface.

Access list ranges are included in the summary descriptions; however, in actual use, only one number is selected from the given range.

Table B-1 lists the access list number ranges in numerical order.

Apollo Domain Access List

apollo access-list *name* **permit | deny** [*firstnet-*]*lastnet.host* [*wildcard-mask*]

apollo access-group *name*

AppleTalk Access List

access-list *600-699* **permit | deny** *network*

appletalk access-group *600-699*

DECnet Access List

One of:

access-list *300-399* **permit | deny** *address mask*

access-list *300-399* **permit | deny** *source source-mask dest dest-mask*

and then:

decnet access-group *300-399*

Ethernet Address Access List

access-list *700-799* **permit | deny** *address mask*

bridge-group *1-9* **input-address-list | output-address-list** *700-799*

Ethernet Protocol Access List

access-list *200-299* **permit | deny** 0x*type-code* 0x*mask*

bridge-group *1-9* **input-type-list | output-type-list** *200-299*

IP Access List

access-list *1-99* **permit | deny** *address mask*

ip access-group *1-99*

access-class 1-99 out | in (for terminal line assignment)

Extended IP Access List

access-list *100-199* **permit | deny ip | tcp | udp | icmp** *source source-mask dest dest-mask* [**lt | gt | eq | neq** *dest-port*]

ip access-group *100-199*

Novell Access Lists

access-list *800-899* **permit | deny** *net*[*.source-address*][*source-mask*]*net*[*.dest-address*][*dest-mask*]

novell access-group *800-899*

Extended Novell Access Lists

access-list *900-999* **permit | deny** *xns-protocol net*[*.source-address*][*source-mask*]*source-socket net*[*.dest-address*][*dest-mask*]*dest-socket*

novell access-group *900-999*

Novell SAP Access List Filter

access-list *1000-1099* **permit | deny** *network.*[*address*][*service-type*]

Source-Route Bridge Protocol Type Access List

access-list *200-299* {**permit** | **deny**} *type-code wild-mask*

Transparent Bridge Access List

access-list *list* {**permit | deny**} *type-code wild-mask*

VINES Access List

One of:

vines access-list *1-100* **{permit | deny} IP** *source-address source-mask dest-address dest-mask*

vines access-list *1-100* **permit | deny** *protocol source-address source-mask source-port dest-address dest-mask dest-port*

and then:

vines access-group *list*

XNS Access Lists

access-list *400-499* **permit | deny** *net*[*.source-address*][*source-mask*]*net*[*.dest-address*][*dest-mask*]

xns access-group *400-499*

Extended XNS Access Lists

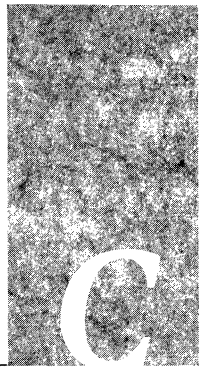
access-list *500-599* **permit | deny** *xns-protocol net*[*.source-address*] [*source-mask*]*source-socket net*[*.dest-address*][*dest-mask*]*dest-socket*

xns access-group *500-599*

Table B-1 Summary of Numerical Ranges

Protocol	Range
IP	1—99
Extended IP	100—199
Ethernetl type code	200—299
DECnet	300—399
XNS	400—499
Extended XNS	500—599
AppleTalk	600—699
Ethernet address	700—799
Novell	800—899
Extended Novell	900—999
Novell SAP	1000—1099

Appendix C
Ethernet Type Codes



This appendix lists known Ethernet type codes for use with type code filtering configuration commands.

Note: Not all codes are used on both Token Ring or Ethernet media.

Hexadecimal	Description (Notes)
0000-05DC	IEEE802.3 Length Field
0101-01FF	Experimental; for development (Conflicts with 802.3 length fields)
0200	Xerox PUP Conflicts with IEEE802.3 length fields
0201	Xerox PUP Address Translation (Conflicts with IEEE802.3 length fields)
0600	Xerox XNS IDP
0800	DOD Internet Protocol (IP) * #
0801	X.75 Internet
0802	NBS Internet
0803	ECMA Internet
0804	CHAOSnet
0805	X.25 Level 3
0806	Address Resolution Protocol (for IP and CHAOS)
0807	XNS Compatibility
081C	Symbolics Private
0888-088A	Xyplex
0900	Ungermann-Bass Network Debugger
0A00	Xerox IEEE802.3 PUP
0A01	Xerox IEEE802.3 PUP Address Translation
0BAD	Banyan VINES IP
0BAE	Banyan VINES Loopback
0BAF	Banyan VINES Echo
1000	Berkeley trailer negotiation

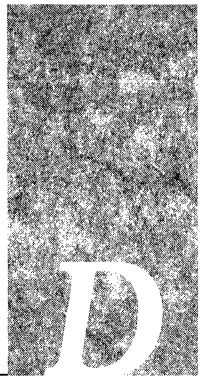
1001-100F	Berkeley trailer encapsulation for IP
1600	VALID system protocol
4242	PCS Basic Basic Block Protocol
5208	BBN Simnet Private
6000	DEC unassigned
6001	DEC Maintenance Operation Protocol (MOP) Dump/Load Assistance
6002	DEC MOP Remote Console
6003	DEC DECNet Phase IV
6004	DEC Local Area Transport (LAT)
6005	DEC DECNet Diagnostics
6006	DEC DECNet Customer Use
6007	DEC Local Area VAX Cluster (LAVC)
6008	DEC unassigned
6009	DEC unassigned
6010-6014	3Com Corporation
7000	Ungermann-Bass (UB) Download
7001	UB diagnostic/loopback
7002	UB diagnostic/loopback
7020-7029	LRT (England)
7030	Proteon
8003	Cronus VLN
8004	Cronus Direct
8005	HP Probe protocol
8006	Nestar
8008	AT&T
8010	Excelan
8013	Silicon Graphics diagnostic (obsolete)
8014	Silicon Graphics network games (obsolete)
8015	Silicon Graphics reserved type (obsolete)
8016	Silicon Graphics XNS NameServer, bounce server (obsolete)

8019	Apollo Domain
802E	Tymshare
802F	Tigan, Inc.
8035	Reverse Address Resolution Protocol (RARP)
8036	Aeonic Systems
8038	DEC LANBridge Management
8039	DEC unassigned
803A	DEC unassigned
803B	DEC unassigned
803C	DEC unassigned
803D	DEC Ethernet CSMA/CD Encryption Protocol
803E	DEC unassigned
803F	DEC LAN Traffic Monitor Protocol
8040	DEC unassigned
8041	DEC unassigned
8042	DEC unassigned
8044	Planning Research Corp.
8046-8047	AT&T
8049	ExperData (France)
805B	Versatile Message Translation Protocol RFC-1045 (Stanford)
805C	Stanford V Kernel, production
805D	Evans & Sutherland
8060	Little Machines
8062	Counterpoint Computers
8065-8066	University Of Mass. at Amherst
8067	Veeco Integrated Automation
8068	General Dynamics
8069	AT&T
806A	Autophon (Switzerland)
806C	ComDesign
806D	Compugraphic Corporation
806E-8077	Landmark Graphics Corporation
807A	Matra (France)
807C	Merit Internodal
807D-8080	Vitalink Communications
8080	Vitalink TransLAN III Management

8081-8083	Counterpoint Computers
8088-808A	Xyplex
809B	Kinetics EtherTalk (AppleTalk over Ethernet)
809C-809E	Datability
809F	Spider Systems Ltd.
80A3	Nixdorf Computers (West Germany)
80A4-80B3	Siemens Gammasonics Inc.
80C0-80C3	Digital Communications Assoc. Inc.
80C1	DCA Data Exchange Cluster
80C4	Banyan VINES IP
80C5	Banyan VINES Echo
80C6	Pacer Software
80C7	Applitek Corporation
80C8-80CC	Intergraph Corporation
80CD-80CE	Harris Corporation
80CF-80D2	Taylor Instrument
80D3-80D4	Rosemount Corporation
80D5	IBM SNA Services over Ethernet
80DD	Varian Associates
80DE	Integrated Solutions Transparent Remote File System (TRFS)
80DF	Integrated Solutions
80E0-80E3	Allen-Bradley
80E4-80F0	Datability
80F2	Retix
80F3	Kinetics AppleTalk Address Resolution Protocol (AARP)
80F4-80F5	Kinetics
80F7	Apollo Computer
80FF-8103	Wellfleet Communications
8069	AT&T
807B	Dansk Data Elektronik A/S
8107	Symbolics Private
8108	Symbolics Private
8109	Symbolics Private
8130	Waterloo Microsystems Inc.
8131	VG Laboratory Systems
8137	Novell NetWare IPX (old)

8137-8138	Novell, Inc.
8139-813D	KTI
9000	Loopback (Configuration Test Protocol)
9001	Bridge Communications XNS Systems Management
9002	Bridge Communications TCP/IP Systems Management
FF00	BBN VITAL LANBridge cache wakeups

Appendix D
Pattern Matching



This appendix describes the pattern matching scheme used by the X.25 switching feature.

Regular Expression Pattern Matching

The X.121 address and Call User Data are used to find a matching routing table entry. The list is scanned from the beginning to the end and each entry is pattern-matched with the incoming X.121 address and Call User Data to the X.121 and Call User Data in the routing table entry. If the pattern match for both entries succeeds, then that route is used. If the incoming call does not have any Call User Data, then only the X.121 address pattern match need succeed with an entry which only contains an X.121 pattern. If Call User Data is present, and while scanning a route is found which matches the X.121 address, but the route doesn't have a Call User Data pattern, then that route is used when an exact match cannot be found.

Regular expressions are used to allow pattern-matching operations on the X.121 addresses and Call User Data. The most common operation is to do prefix matching on the X.121 DNIC field and route accordingly. For example, the pattern ^3306 will match all X.121 addresses with a DNIC of 3306. The caret (^) is a special regular expression character that says to anchor the match at the beginning of the pattern.

If a matching route is found, the incoming call is forwarded to the *next hop* depending on the routing entry. If no match is found, the call is cleared. If the route specifies a serial interface running X.25, the call will attempt to be forwarded over that interface. If the interface is not operational or out of available virtual circuits, the call will be cleared. Otherwise, the expected Clear Request or Call Accepted message will be forwarded back towards the originator. The "null 0" interface can be used to early-terminate or refuse calls to specific locations.

If the matching route specifies an IP address, a TCP connection will be established to port 1998 at the specified IP address which must be another Cisco router. The Call Request packet will be forwarded to the remote router where it will be processed in a similar fashion. If a routing table entry isn't present or the serial interface is down or out of virtual circuits, a Clear Request will be sent back and the TCP connection will be closed. Otherwise, the call will be forwarded over the serial interface and the expected Clear Request or Call Accepted packet will be returned. Incoming calls received via TCP connections that match a routing entry specifying an IP address will be cleared. This restriction prevents Cisco

routers from establishing a TCP connection to another router which would establish yet another TCP connection. A router must always connect to the remote router with the destination DTE directly attached.

Regular expressions provide a way to specify wide ranges of X.121 addresses and Call User Data fields by using just a few keystrokes. If you are familiar with regular expressions from UNIX programs such as *regexp*, you are already familiar with much of Cisco System's regular expression implementation.

Writing regular expressions is simple once you see and try a few examples. A regular expression is a formula for generating a set of strings. If a particular string can be generated by a given regular expression, then that string and regular expression *match*. In many ways, a regular expression is a program, and the regular expression matches the strings it generates.

A regular expression is built up of different components, each of which is used to build the regular expression string-generating program. The simplest usable component is the *atom*, but first, you need to understand *ranges*, as atoms are built of these.

Ranges

A range is a sequence of characters contained within left and right square brackets ([ ]). A character matches a range if that character is contained within the range; for example, the following syntax forms the range consisting of the characters "a," "q," "c," "s," "b," "v," and "d." The order of characters is usually not important; however, there are exceptions and these will be noted.

[aqcsbvd]

You can specify an ASCII sequence of characters by specifying the first and last characters in that sequence, and separating them with a hyphen (-).

[a-dqsv]

The above example could also be written so as to specify right square brackets ( ]) as a character in a range. To do so, enter the bracket as the *first* character after the initial left square bracket that starts the range.

This example matches right bracket and the letter "d."

[ ]d]

To include a hyphen (-), enter it as either the first or the last character of the range.

You can reverse the matching of the range by including a wedge (caret) at the start of the range. This example matches any letter *except* the ones listed. When using the wedge with the special rules for including a bracket or hyphen, make the wedge the very first character.

[^a-dqsv]

This example matches anything except a right square bracket ( ])or the letter "d":

[^]d]

Atoms

Atoms are the most primitive usable part of regular expressions. An atom can be as simple as a single character. The letter "a" is an atom, for example. It is also a very simple regular expression, that is, a program that generates only one string, which is the single-letter string made up of the letter "a." While this may seem trivial, it is important to understand the set of strings that your regular expression program generates. As will be seen in upcoming explanations and examples, much larger sets of strings can be generated from more complex regular expressions.

Certain characters have a special meaning when used as atoms; refer to Table D-1.

Table D-1 Special Symbols Used as Atoms

.	Matches any single character
^	Matches the null string at the beginning of the input string
$	Matches the null string at the end of the input string
\ character	Matches *character*

1274

Note another use for the ^ symbol.

As an example, the regular expression matches "abcd" only if "abcd" *starts* the full string to be matched:

^abcd

Whereas the following expression is an atom that is a range that matches any single letter, as long as it is *not* the letters "a," "b," "c," or "d."

[^abcd]

It was previously stated that a single character string such as the letter "a" is an atom. To remove the special meaning of a character, precede it with a backslash (\).

```
$
```

Whereas this atom matches a dollar sign ($):

```
\$
```

Any character can be preceded with the backslash character with no adverse affect.

```
\a
```

Atoms are also full regular expressions surrounded by parentheses. For example, both "a" and "(a)" are atoms matching the letter "a." This will be important later, as we see patterns to manipulate entire regular expressions.

Pieces

A piece is an atom optionally followed by one of the symbols listed in Table D-2:

Table D-2 Special Symbols Used With Pieces

*	Matches 0 or more sequences of the atom
+	Matches 1 or more sequences of the atom
?	Matches the atom or the null string

1245

For example matches any number of occurrences of the letter "a," including *none.*

```
a*
```

This string requires there to be at least one letter "a" in the string to be matched:

```
a+
```

This string means that the letter "a" can be there *once*, but it does not have to be:

```
a?
```

This string matches any number of asterisks (*).

```
\**
```

Here is an example using parentheses. This string matches any number of the two-atom string "ab."

(ab)*

As a more complex example, this string matches one or more instances of letter-digit pairs (but not none, that is, an *empty string* is not a match):

([A-Za-z][0-9])+

The order for matches using the optional *, +, or ? symbols is longest construct first. Nested constructs are matched from outside to inside. Concatenated constructs are matched beginning at the left side of the construct.

Branch

A branch is simply a set of zero or more concatenated pieces. The previous letter-digit example was an example of a branch as concatenated pieces. Branches are matched in the order normally read—from left to right. For example, in the previous example, the regular expression matches "A9b3," but not "9Ab3" because the alphabet is given first in the two-atom branch of [A-Za-z][0-9].

Regular Expressions

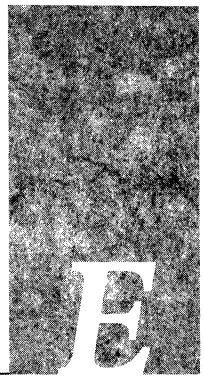
A regular expression is a branch, or any number of branches separated by a vertical bar (|). A string is said to match the regular expression if it is generated by the "program" specified in any of the branches. Of course, a string can be generated by more than one branch. For example, "abc" is generated by all branches in the regular expression

abc|a*(bc)+|(ab)?c|.*

Also remember that if a regular expression can match two different parts of an input string, it will match the earliest part first.

The regular expression support and the technical information for this portion of the documentation is based on Henry Spencer's public domain *rgrep(3)* library package.

Appendix E
ASCII Character Set



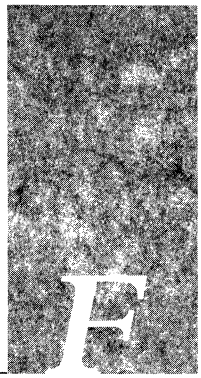
Many of the configuration commands described in this manual require the decimal representation of an ASCII character. Table E-1 provides translations from ASCII character to decimal number. To find the decimal equivalent of the ASCII character, add the row heading and column heading numbers together. To find the ASCII code for the percent (%) character, for example, add row 30 to column 7, so the ASCII code is 37.

Table E-1 ASCII-to-Decimal Translation Table

	0	1	2	3	4	5	6	7	8	9
0	^@	^A	^B	^C	^D	^E	^F	^G	^H	^I
10	^J	^K	^L	^M	^N	^O	^P	^Q	^R	^S
20	^T	^U	^V	^W	^X	^Y	^Z	ESC	FS	GS
30	RS	US	SP	!	”	#	$	%	&	’
40	(	)	*	+	,	–	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	‘	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	{	~	DEL		

1320

Appendix F
References and Recommended Reading



Presented here are lists of references used in, and suggested references for understanding concepts presented in this publication. Some tips for obtaining these references are also provided following the lists.

Commercially Available Publications

Comer, Douglas E. *Internetworking with TCP/IP: Principles, Protocols, and Architecture.* Vol. I, Prentice-Hall, 1988.

Davidson, John. *An Introduction to TCP/IP.* Springer-Verlag, 1988.

Martin James and the ARBEN Group. *Local Area Networks.* Prentice Hall, 1989.

McNamara, John E. *Local Area Networks.* Digital Press, Educational Services, Digital Equipment Corporation, 12 Crosby Drive, Bedford, Massachusetts 01730.

Meijer, Anton. *Systems Network Architecture: A Tutorial.* John Wiley & Sons, Inc., 1987.

Miller, Mark A. *LAN Protocol Handbook.* M&T Publishing, 1990.

Rose, Marshall T. *The Simple Book: An Introduction to Management of TCP/IP-based Internets.* Prentice Hall, 1991.

Schlar, Sherman K. *Inside X.25: A Manager's Guide.* McGraw-Hill, Inc., 1990.

Sherman, Ken. *Data Communications: A User's Guide,* 2nd edition, Reston Publishing Company, Inc., 1985.

Sidhu, Gursharan S., Richard F. Andrew, and Alan Oppenheimer. *Inside AppleTalk,* Second Edition, Addison-Wesley Publishing Company, 1990

Stallings, William. *Handbook of Computer Communications Standards,* Vols. 1–3, Macmillan Publishing Company, 1987.

Technical Publications and Standards

ANSI X3T9.5 Committee. *FDDI Station Management (SMT).* Rev. 6.1, March 15, 1990.

Apple Computers. *AppleTalk Network System Overview.* Addison-Wesley Publishing Company, Inc., 1989.

Bellcore. *Generic System Requirements in Support of a Switched Multi-Megabit Data Service.* Technical Advisory ,TA-TSY-000772, October, 1989.

——. *Local Access System Generic Requirements, Objectives, and Interface Support of Switched Multi-Megabit Data Service.* Technical Advisory TA-TSY-000773, Issue 1, December, 1985.

——. *Switched Multi-megabit Data Service (SMDS) Operations Technology Network Element Generic Requirements,* Technical Advisory TA-TSY-000774

CCITT. *CCITT Data Communication Networks—Interfaces, Recommendations X.20-X.23,* Volume VIII, 1984.

Cisco Systems. *HSSI Specifications.*

Digital Equipment Corporation. *DECNET/OSI Phase V: Making the Transition From Phase IV.* EK-PVTRN-BR, 1989.

——. *DIGITAL Network Architecture (Phase V),* EK-DNAPV-GD-001, September 1987.

——. *DECserver 200 Local Area Transport (LAT) Network Concepts.* AA-LD84A-TK, June 1988.

Hemrick, C. and L. Lang, "Introduction to Switched Multi- megabit Data Service (SMDS), an Early Broadband Service," publication pending in the Proceedings of the XIII International Switching Symposium (ISS 90), May 27, 1990–June 1, 1990.

Hewlett-Packard. *X.25: The PSN Connection; An Explanation of Recommendation X.25,* 5958-3402, October 1985.

IEEE Project 802 – *Local & Metropolitan Area Networks. Proposed Standard: Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN),* February 7, 1990.

National Security Agency. *Blacker Interface Control Document (ICD).*

StrataCom. *Frame Relay Interface Specification.* 040-207460, REv. 2.3, August 9, 1990.

XEROX. *Internet Transport Protocols.* XNSS 029101, January 1991.

Obtaining Technical Information

Many of the references cited in this list can be obtained from book stores or ordered directly from the publisher or company that produced the reference. In the world of data communications, however, there are numerous standards that are kept by various companies and agencies. This section provides some tips on obtaining this type information.

Obtaining RFCs

Information about the Internet suite of protocols are contained in documents called *Requests for Comments*, or RFCs. These documents are maintained by Government Systems, Inc. (GSI). You can obtain copies in either printed or electronic form. You may contact GSI in these ways:

Postal:

Government Systems, Incorporated
Attn: Network Information Center
14200 Park Meadow Drive, Suite 200
Chantilly, Virginia 22021

Telephone:

1-800-365-3642
1-703-802-4535
1-703-802-8376 (FAX)

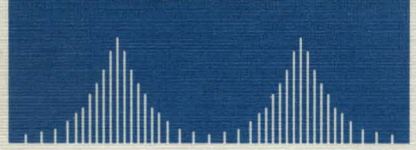
Electronic Mail:

NIC@NIC.DDN.MIL
Network address: 192.112.36.5
Root domain server: 192.112.36.4

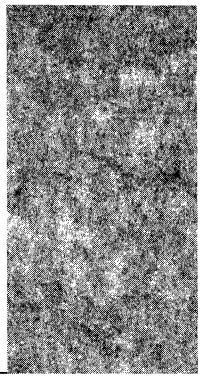
Obtaining Technical Standards

Following are some places from which you may obtain technical standards:

- Omnicom at 1-800-OMNICOM.
- Global Engineering Documents, 2805 McGraw Avenue, Irvine, California, 92714. Telephone: 1-800-854-7179.
- American National Standards Institute, 1430 Broadway, New York, New York, 10018. Telephone: 212-642-4932 and 212-302-1286.



Glossary



Glossary and Acronym List

The following is a glossary of terms and a list of acronyms used in literature from Cisco Systems.

access list A list kept by the network server to control access to or from the network server for a number of services (for example, to restrict packets with a certain IP address from leaving a particular interface on the network server).

address resolution A method for mapping Level 3 addresses onto media-specific addresses.

advertising A way by which routers maintain lists of usable routes by sending routing updates within specified rates of time.

AFP AppleTalk Filing Protocol.

agent Code that processes queries and returns replies on behalf of a client or server application.

applique A mounting plate containing the connector hardware for attachment to the network. Appliques translate and transpose the serial communications signals into the signals expected by the communication standard of choice, such as RS-232, V.35, and so on.

ANSI (American National Standards Institute) The coordinating body for voluntary standards groups with the United States.

area A concept in ISO CLNS-based networks to define a set of networks connected by routers.

AARP AppleTalk Address Resolution Protocol.

ARP (Address Resolution Protocol) The protocol used to bind an IP address to Ethernet/802.2 addresses.

ASCII (American Standard Code for Information Interchange) An eight-bit code for character representation, including seven bits plus parity.

AS (autonomous system) A collection of networks under a common administration sharing a common routing strategy. An autonomous system must be given a unique, 16-bit number which is assigned by the DDN Network Information Center.

ATG (Address Translation Gateway) Cisco's DECnet routing software function that allows a router to route multiple, independent DECnet networks, and to establish a user-specified address translation for selected nodes between networks.

AUI (Attachment Unit Interface) An Ethernet transceiver cable, or the backpanel connector to which such a cable might attach.

bandwidth The range of frequencies that can pass over a given circuit.

big-endian A method of storing or transmitting data where the most significant bit or byte comes first.

BDPDU Bridge Protocol Data Units; see **PDU**.

BootP A protocol which is used by a network node to determine the IP address of its Ethernet interfaces.

bridge A device that connects two network segments using the same medium and passes packets between them. Bridges operate at Level 2 of the ISO model (the data-link layer) and are protocol-insensitive.

broadband The IEEE specification 802.4 standard for connecting a set of network hosts using a single multichannel transmission medium. The broadband standard permits simultaneous multiple high-bandwidth channels, with each channel occupying a particular frequency range. Broadband systems are commonly used to transmit video, voice, and additional data channels.

broadcast A packet or frame whose destination address contains an address to which all entities on the network must listen. Typically, this address contains all ones.

CCITT French acronym for International Telegraph and Telephone Consultative Committee, an international organization that develops communications standards such as Recommendation X.25.

checksum A method for checking the integrity of transmitted data. A checksum is an integer value computed from a sequence of octets by treating them as integers and computing the sum. The value is recomputed at the receiving end and compared for verification.

CLNS (Connectionless Network Service) A network layer service that does not require a circuit to be established before data is transmitted.

connectionless A method of interconnection that does not require a circuit to be established before data is transmitted.

connection-oriented A method of interconnection that requires connection establishment, data transfer, and connection release.

CMT (Connection Management) An FDDI process that handles the transition of the ring through its various states — off, active, connect, and so on — as defined by the X3T9.5 specification.

CRC Cyclic redundancy checksum; see **checksum.**

cross talk Interfering energy transferred from one circuit to another.

CSMA/CD (Carrier-Sense Multiple Access with Collision Detection) The style of network access used by Ethernet and IEEE 802.3.

CSU/DSU (Customer Service Unit/Digital Service Unit) A device that converts V.35 or RS-449 signals to a properly coded T1 transmission signal.

DARPA (Defense Advanced Research Project) A government agency that funded research and experimentation with the DARPA Internet.

DCA Defense Communications Agency.

DDP Datagram Delivery Protocol.

DCE (Data Communications Equipment) The devices and connections of a communications network which connect the communication circuit with the end device (data terminal equipment). A modem can be considered DCE.

DDN (Defense Data Network) The MILNET and associated parts of the Internet that connect military installations. Used loosely, it refers to the MILNET, ARPANET, and the TCP/IP protocols they use.

DECnet Refers to a protocol suite developed and supported by Digital Equipment Corporation.

DNS (Domain Name System) A part of the Internet protocol that allows a router to automatically determine host-name-to-address mappings.

DoD Department of Defense.

domain names A directory service for matching host names with IP addresses.

DSP (Domain Specific Part) That part of the CLNS address that contains an area identifier, station identifier, and a selector byte.

DTE (Data Terminal Equipment) The part of a data station that serves as a data source, destination, or both, and that provides for the data communications control function according to protocols. DTE includes computers, protocol translators, and multiplexers.

dynamic resolution Use of an address resolution protocol to determine and store address information.

EGP (Exterior Gateway Protocol) A protocol for exchanging routing information with the DDN core network server system and similar large groups of networks. Refer to RFC 904.

encapsulation Refers to the wrapping of data in a certain protocol header. For example, on Ethernet, all data is encapsulated in either an Ethernet header or IEEE 802.2 header.

End System A non-routing host or node in an ISO CLNS network.

ERPDU Error Protocol Data Unit; see **PDU.**

ES-IS OSI End System to Intermediate System protocol whereby *End Systems* (hosts) announce themselves to *Intermediate Systems* (routers).

Ethernet A baseband local area network (LAN) specification invented by Xerox Corporation and developed jointly by Xerox, Intel, and Digital Equipment Corporation. Ethernet networks operate at 10 megabits per second using CSMA/CD to run over coaxial cable or shielded twisted pair wiring.

EXEC Refers to the interactive command processor of the Cisco software.

FDDI (Fiber Distributed Data Interface) An ANSI-defined standard for timed, token-passing over fiber optic cable.

flash update A routing update sent asynchronously in response to a change in the network topology. Normal updates are sent at fixed intervals.

FTAM (File Transfer, Access, and Management) The OSI standard developed by the ISO for network file exchange and management between network nodes.

FTP (File Transfer Protocol) The standard, high-level protocol for transferring files from one network node to another using TCP/IP.

gateway An older term that originally referred to an IP routing device. The term currently refers to a special purpose device that connects two or more networks, and routes packets from one to another using different protocols by converting one network's protocol to the format used by the other.

HDH (HDLC Distant Host protocol) A means of running the 1822protocol over synchronous serial links instead of over special purpose 1822 hardware. HDH is essentially 1822 leaders and data encapsulated in LAPB (X.25 Level 2) packets.

HDLC (High-Level Data Link Control) Specifies an encapsulation method of data on synchronous serial data links. The Cisco HDLC support performs only framing and checksum functions. No retransmission of windowing is done.

host The controlling computer in a communications network that primarily provides services and is the source or destination of messages.

ICMP (Internet Control Message Protocol) A protocol that provides message packets to report changes in packet processing. Refer to RFC 792.

IDP (Initial Domain Part) That part of a CLNS address containing an authority and format identifier and a domain identifier.

IEEE (Institute of Electrical and Electronic Engineers) Institute that, among other things, hosts committees that develop and propose standards for computers and networks, such as the 802-series of protocols.

IEN Internet Engineering Notes.

IETF Internet Engineering Task Force.

IGRP (Internal Gateway Routing Protocol) A protocol developed by Cisco Systems to address the problem of routing within a large network of general topology comprised of segments having different bandwidth and delay characteristics.

interface The physical connection between two systems or devices; the boundary between between adjacent layers in the OSI model.

Intermediate System Router in an ISO CLNS network.

Internet Large, national backbone of networks that includes regional, military, and university networks spanning the globe.

internet Collection of networks interconnected by routers creating a virtual network, or a network that functions as a single network.

Internet address A 32-bit address assigned to hosts using TCP/IP. The address is written as four octets separated with periods (dotted decimal format) that are made up of a network portion and a host portion to make routing of information using the address easier.

internetwork A network of networks; also called an *internet*. An internetwork is a group of LANs and WANs that are geographically or organizationally separate, but appear to users as one integrated network.

interoperability The ability of computing equipment manufactured by different vendors to communicate successfully over one integrated network.

IP (Internet Protocol) A Level 3 protocol which contains addressing information and some control information which allows packets to be routed. Refer to MIL-STD-1777.

IPSO (IP Security Option) That part of the Internet Protocol that defines security levels on a per-interface basis.

ISO (International Standards Organization) An organization which establishes international standards for computer network architecture. The ISO established the Open Systems Interconnection seven-layer model of network interconnection.

ISO layer Any of seven levels in a model proposed by the International Standards Organization (ISO) to describe the functions and relationships in computer networks. The lowest layers (1 and 2) specify media standards; upper layers specify functions more visible to users and programs using the network.

IS-IS OSI Intermediate System to Intermediate System protocol whereby Intermediate Systems (routers) exchange routing information.

LAN (Local Area Network) A LAN consists of local segments of Ethernet cable, broadband cable, Token Rings, or other similar media.

LAPB (link access procedure balanced) A modified form of High-Speed Data Link Control. X.25 represents Levels 2 and 3 of the OSI reference model; LAPB is the protocol that implements Level 2. This protocol provides a mechanism to exchange data (frames), detect out-of-sequence or missing frames, and provide for retransmission and acknowledgment.

LAT (Local Area Transport) A protocol developed by Digital Equipment Corporation.

MAC (Media Access Control) A part of the second layer of the OSI model. This is a method of access to the network media by which network stations can transmit information.

MAU (Medium Attachment Unit) Also known as an Ethernet transceiver, an MAU is a device that converts digital data from the Ethernet interface for connection to the appropriate medium. Token Ring MAUs also exist.

media The physical cabling plant, satellite, or microwave circuits over which network data passes. Common network media are coaxial and fiber optic cable, twisted-pair wiring, and telephone circuits.

MIB (Management Information Base) A collection of objects that may be accessed using the Simple Network Management Protocol.

MTU (Maximum Transmission Unit) Refers to the maximum packet size, in bytes, that a particular interface will handle.

multicast A broadcast packet that is delivered only to a defined subset of all possible destinations.

name resolution A method of mapping domain names into the corresponding address. See **domain names**.

name server A server provided on the network which responds to domain name requests. Refer to RFC 882.

NBP Name Binding Protocol.

NET (Network Entity Title) Network addresses, as defined by ISO network architecture, and as used in ISO CLNS-based networks.

NetBIOS (Network Basic Input/Output System) A session layer interface for PC networks from IBM and Microsoft.

network Refers to a collection of computers and other devices that are able to communicate with each other over distances.

NSAP (Network Service Access Points) ISO network addresses, as specified by ISO 8348/Ad2. The point at which the OSI Network Service is made available to an entity at Level 4 of the ISO OSI seven level reference model.

octet A byte which explicitly contains eight bits.

OSI (Open System Interconnection) The International Standards Organization's model for standards-based networking.

packet A collection of bits that constitutes one network transmission. Packets must include relevant network address and accounting information, as well as user data.

packet-switched Type of network on which each packet contends with others for data transmission. The channel is occupied only for the duration of the packet. Routers are called packet switches when they move packets along a route to its destination. In contrast, a circuit-switched network system dedicates one circuit at a time to data transmission.

PAD (Packet Assembler/Disassembler) The device that buffers data sent between hosts and terminals across an X.25 network, as defined by CCITT Recommendation X.3, X.28 and X.29.

PAP Print Access Protocol.

PDU (Protocol Data Unit) Another word for "packet" as defined by the OSI. A PDU is exchanged between devices within a specific level of the ISO OSI reference model.

physical layer The first layer in the Open Systems Interconnect model.

PHY FDDI physical sublayer; designation for FDDI fiber optic cables.

ping (Packet internet groper) Refers to the ICMP echo message and its reply. Refer to RFC 792.

poison reverse A routing update that specifically indicates that a network or subnet is unreachable, rather than omit mentioning it in updates.

PPP Point-to-Point Protocol.

Probe An address resolution protocol developed by Hewlett-Packard.

protocol A formal description of a set of rules and conventions that govern how devices on a network exchange information in an orderly and meaningful way.

Proxy ARP The function of a router sending an Address Resolution Protocol (ARP) response to a host which does not know how to use a router, and that pretends to be a remote target host.

PSN (Packet Switch Node) Refers to the central switching node in the X.25 architecture. Usually the PSN is Data Communication Equipment (DCE) and allows for connection to Data Terminal Equipment (DTE). See **X.25**.

public standard protocol A network specification available to the public. Anyone may write a program that implements a public standard protocol and thus enables network hosts to communicate in this language.

PUP (PARC Universal Protocol) A protocol developed at Xerox Palo Alto Research Center that is similar to IP.

QOS Quality of Service.

RARP (Reverse Address Resolution Protocol) The logical reverse of ARP that provides a method for finding IP addresses based on Ethernet/802.2 addresses. Refer to RFC 903.

RDPDU Redirect Protocol Data Unit; see **PDU**.

redirect A part of the ICMP protocol which allows a network server to tell a host to use another network server.

RFC (Request for Comments) Documents specifying some particular functionality for a data communications protocol available from the DDN Network Information Center.

redistribute A router subcommand which allows routing information discovered through one protocol to be distributed with another protocol.

RIF (Routing Information Field) That portion of the IEEE 802.5MAC header of a datagram used by a bridge to determine to which Token Ring network segments a packet must transit. A RIF is made up of ring and bridge numbers.

RIP (Routing Information Protocol) An interior routing protocol supplied with Berkeley UNIX systems.

ring group A collection of Token Ring interfaces on one or more Cisco routers that is part of one bridged Token Ring network.

rlogin A TCP connection protocol that allows connection to a UNIX host.

router A device that can decide which of several paths network traffic will follow based on the fastest or cheapest route. Also called a network server, it forwards packets of data from one network to another, based on network-level (ISO model Level 3) information.

routing The process of finding a path to the destination host. Routing is very complex in large networks, because of the many potential intermediate destinations a packet might traverse before reaching its destination host.

routing domains A concept in ISO CLNS-based networks to describe areas that are connected to other areas. See **area**.

RTMP Routing Table Maintenance Protocol.

SAP (Service Access Point) A point on the interface between two Open System Interconnection layers. *Also* Service Advertisement Protocol.

SMT (Station Management) That part of the FDDI that manages stations on the ring, as defined in the X3T9.5 specification.

SMTP (Simple Mail Transfer Protocol) A protocol that provides electronic mail over TCP/IP.

SNA (Systems Network Architecture) A network architecture developed by IBM.

SNMP (Simple Network Management Protocol) A protocol that provides a means to access and set configuration and runtime parameters of the router and terminal servers. Refer to RFC 1155, RFC 1156, RFC 1157, RFC 1212, and RFC1213.

static route A route that is manually entered into the routing table.

STUN Cisco's Serial Tunneling mechanism.

subnet A subnetwork address.

subnet mask Subnetting allocates a portion of the host part of a Class A, B or C Internet address for use as a subnet. A subnet mask is a 32-bit value used to distinguish the combined network and subnet parts of the Internet address from the remaining host part. Bits in a subnet mask set to 1 correspond to the bits in the network portion of the Internet address. Bits in a subnet mask set to 0 correspond to the bits in the host portion of the Internet address. A subnet mask may be specified when a server is initially started up.

T1 Refers to a telephone signaling standard used for transmission of data through the telephone hierarchy. The rate of transmission is 1.544 megabits per second in the USA and 2.048 Mbps in Europe.

T1 converter Device that converts the HDLC synchronous serial data stream of the server into a T1 data stream with the correct framing and ones density.

TACACS (Terminal Access Controller Access System) A system developed by the Defense Data Network to control access to its TAC terminal servers.

TCP/IP (Transmission Control Protocol/Internet Protocol) A protocol corresponding to levels three and four (network and transport) in the ISO OSI model. It provides for the reliable transmission of data through retransmission. TCP/IP was developed by the U.S. Department of Defense to support the construction of world-wide internetworks. This protocol is the most commonly used public standard protocol available today. Most computer systems can support TCP/IP.

Telnet (Telecommunications Network Protocol) A protocol used for remote terminal access used within the TCP/IP protocol.

terminal server A communications processor that connects asynchronous devices to any local or wide area network that uses the TCP/IP or X.25 protocol suites.

TFTP (Trivial File Transfer Protocol) A simplified method of transfer of logical files on an IP network; refer to RFC 783.

third-party mechanism A means by which a router can tell a peer router that another router on the shared network is the appropriate router for some set of definitions.

THT (token holding timer) Timer defined by the FDDIX3T9.5 specification that determines the amount of bandwidth available for transmissions.

token A packet of control information.

Token Ring A token access method that involves the use of sequential or ring network topology. Each computer knows the address of the computer that should receive the token next. When the token is not for a given computer, it passes the token to the next computer in line. Refer to IEEE 802.5.

TOS Type of service.

transit bridging A bridging function that allows Ethernet datagrams across an FDDI ring. The term *transit* refers to the fact that the source and destination of the datagram cannot be on the FDDI ring itself.

transparent bridge Level 2 bridges used with Ethernet networks. They are *transparent* in that they need not know any addressing information in order to pass messages through the bridge.

TRT (token rotation timer) Timer defined by the FDDIX3T9.5 specification to control ring scheduling, and to detect and recover from ring errors.

topology The physical arrangement of network nodes and connections.

TVX (transmission valid timer) Timer defined by the FDDIX3T9.5 specification to recover from a transient ring error.

UDP (User Datagram Protocol) A transaction-oriented transport layer protocol paralleling TCP; however, unlike TCP, it is stateless. Refer to RFC 768.

WAN (Wide Area Network) A computer network over a wide geographic area.

wildcard mask A wildcard mask is a 32-bit quantity used in conjunction with an Internet address to determine which bits in an Internet address should be ignored when comparing that address with another Internet address. A wildcard mask is specified when setting up access lists.

X.21 A CCITT recommendation that defines a protocol for communication between a circuit-switched network and user devices.

X.25 A CCITT standard that defines the packet format for data transfers in a public data network. Many establishments have X.25 networks in place that provide remote terminal access. These networks can be used for other types of data, including the Internet Protocol, DECnet and XNS.

X.28 A CCITT recommendation that defines the terminal-PAD interface.

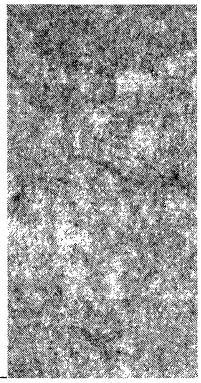
X.29 A CCITT recommendation that defines the PAD-computer interface.

X.3 A CCITT recommendation that defines the PAD parameters.

X3T9.5 The number assigned to the Task Group of Accredited Standards Committee for their internal, working document describing the Fiber Distributed Data interface.

ZIP Zone Information Protocol.

Index



Index

Symbols

A

B

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

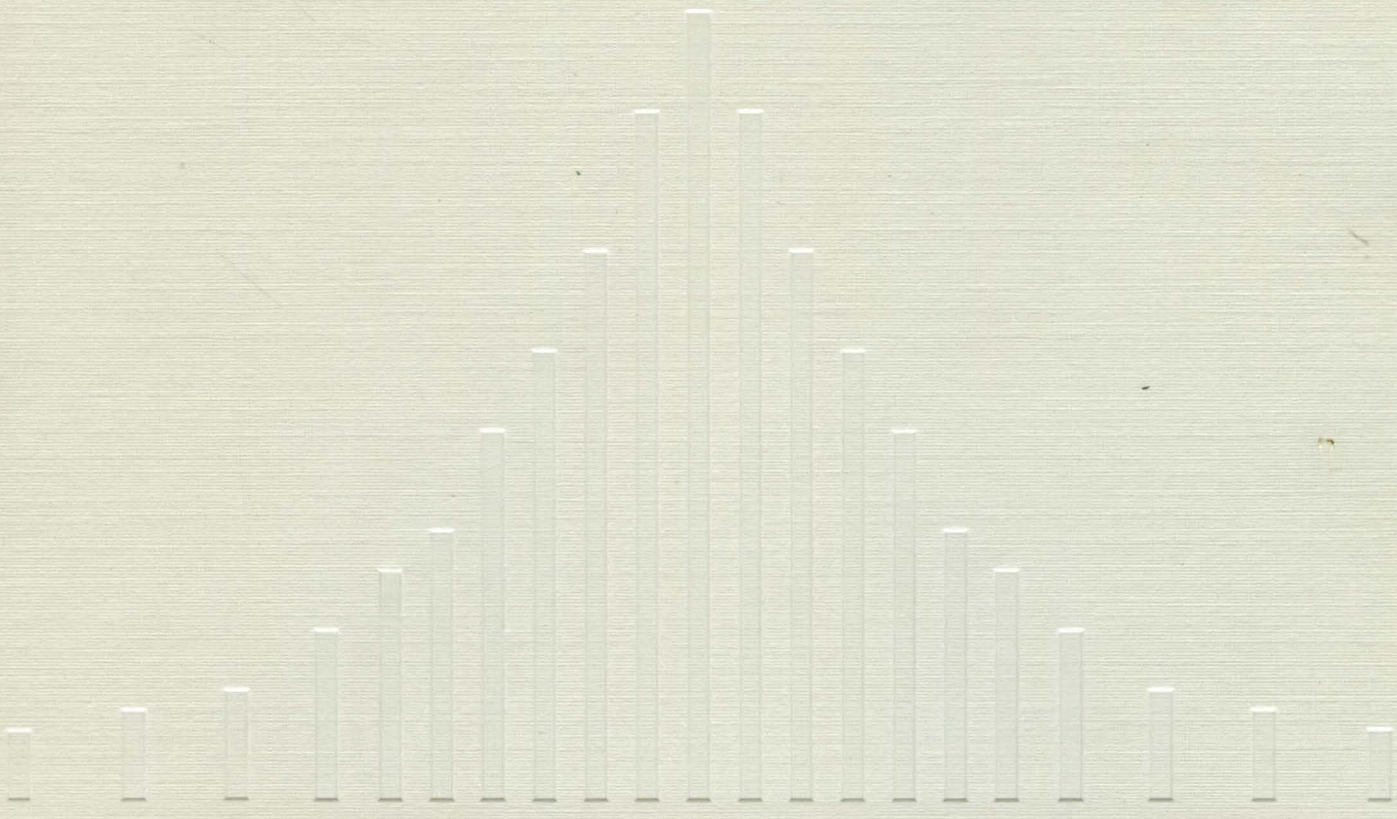
U

V

W

X

Z



Corporate Headquarters:
Cisco Systems, Inc.

1525 O'Brien Drive
Menlo Park, CA 94025
Tel: 1-415-326-1941
Fax: 1-415-326-1989
1-800-553-NETS