CHAPTER 3

Configuring TCP/IP

The Internet Protocol (IP) is a packet-based protocol used to exchange data over computer networks. IP handles addressing, fragmentation, reassembly, and protocol demultiplexing. It is the foundation on which all other Internet protocols, collectively referred to as the IP suite, are built. IP is a network-layer protocol that contains addressing information and some control information that allows data packets to be routed.

The Transmission Control Protocol (TCP) is built upon the IP layer. TCP is a connection-oriented protocol that specifies the format of data and acknowledgments used in the transfer of data. TCP also specifies the procedures that the computers use to ensure that the data arrives correctly. TCP allows multiple applications on a system to communicate concurrently because it handles all demultiplexing of the incoming traffic among the application programs.

This chapter describes how to configure the IP protocol, and, specifically, how Cisco Systems implements this protocol on its protocol translators. For a complete description of the commands in this chapter, refer to the "TCP/IP Configuration Commands" chapter of the *Protocol Translator Command Reference*. For historical background and a technical overview of IP and other protocols, see the *Internetworking Technology Overview.* For information about establishing Telnet and rlogin connections, refer to the *Communication Server and Protocol Translator Connection Guide.*

Cisco's Implementation of TCP/IP

Cisco's implementation of TCP/IP provides most of the major services defined in the various protocol specifications. The protocol translators also provide the TCP and UDP (User Datagram Protocol) services called Echo and Discard, which are described in RFC 862 and RFC 863, respectively.

Cisco supports both TCP and UDP at the transport layer, for maximum flexibility in services. Cisco supports all standards for IP broadcasts and also provides mechanisms for filtering IP packet traffic on the network.

Cisco's IP implementation for the protocol translator also provides several router discovery mechanisms that allow the protocol translator to know the most efficient data path and what action to take if it encounters a packet for which a route is unknown.

The Cisco implementation of TCP generally ensures good server performance on slow-speed links as well as high-speed LAN links. Cisco's TCP software includes Telnet, a simple remote terminal protocol that is part of the TCP/IP protocol suite. The software allows you to turn on or off TCP services such as stream processing and debugging modes for monitoring the connection. Additionally, the software supports rlogin, the BSD UNIX remote login service. For information about making Telnet and rlogin connections, see the *Communication Server and Protocol Translator Connection Guide*.

TCP/IP Configuration Task List

To configure TCP/IP, you must assign an IP address to the network interface. Doing so enables the interface and allows communication with hosts on that interface using TCP/IP. Associated with this task are decisions about subnetting and masking the IP address. You can optionally customize and optimize IP transmissions and the lines for Telnet connections to hosts.

The following list summarizes the tasks for configuring TCP/IP. At a minimum, you must assign IP addresses to network interfaces.

- Assign an IP address to a network interface (page 3-2)
- Configure IP addressing options (page 3-3)
- Assign an IP address to a TCP service (page 3-7)
- Configure routing assistance when IP routing is disabled (page 3-7)
- Configure broadcast packet handling (page 3-10)
- Configure IP services (page 3-11)
- Control access to IP networks (page 3-14)
- Configure IP security options (page 3-15)
- Optimize lines for Telnet (page 3-17)
- Monitor and maintain the TCP/IP network (page 3-19)

At the end of this chapter are examples illustrating how you might configure your network using TCP/IP. For detailed information about syntax and usage of the commands described in the tasks, refer to the *Protocol Translator Command Reference* publication.

Assign an IP Address to a Network Interface

IP addresses identify locations to which IP datagrams can be sent. To assign an IP address to a network interface on the protocol translator, perform the following task in interface configuration mode:

Task	Command
Set an IP address for an interface.	**ip address** *IP-address mask*

A mask identifies the bits in an IP address that denote the network number. When you use this feature to subnet a network, the mask is then referred to as a *subnet mask*. Subnets are described in the *Internetworking Technology Overview* publication.

Caution We support only network masks that use the bits that are contiguous with the network field.

Some IP addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. Table 1-1 lists ranges of IP addresses and shows which addresses are reserved and which are available for use. In addition to the addresses listed as “reserved” in Table 1-1, do not use IP addresses with all zeros or all ones in the host portion for your protocol translator address.

Table 1-1 Reserved and Available IP 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

The official description of IP 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, the NIC assigns a network number or range of addresses appropriate to the number of hosts on the network.

Configure IP Addressing Options

Our IP implementation allows you to control interface-specific handling of IP addresses by facilitating address resolution, name services, and other functions. You do this by performing the following tasks:

- Establish address resolution.
- Map logical names to IP addresses.

These tasks are discussed in the sections that follow.

Establish Address Resolution

A device in the Internet can 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). Local addresses are also referred to 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, for example, the protocol translator first must determine the 48-bit MAC, or local data link, address of that device. The process of determining the local data link address from an IP address is called *address resolution.* The process of determining the IP address from a local data link address is called *reverse address resolution.* The protocol translator uses Address Resolution Protocol (ARP) for address resolution. The protocol translator also uses the Reverse Address Resolution Protocol (RARP). ARP and RARP are defined in RFCs 826 and 903, respectively.

ARP is used to associate IP addresses with media or MAC addresses. Taking an IP address as input, ARP determines the associated media address. Once a media or MAC address is determined, the IP address/media address association is stored in an ARP cache for rapid retrieval. Then the IP

datagram is encapsulated in a link-layer frame and sent over the network. Encapsulation of IP datagrams and ARP requests and replies on IEEE 802 networks is specified by the Subnetwork Access Protocol (SNAP).

RARP works 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 protocol translator interface. RARP is often used by diskless nodes that do not know their IP address when they boot. The protocol translator uses RARP if it does not know the IP address of an interface at startup.

To define address resolution on the protocol translator, perform the following tasks:

- Define an ARP cache.
- Set ARP encapsulations.

The procedures for performing these tasks are discussed in the following sections.

Define an ARP Cache

ARP and other address resolution protocols provide a dynamic mapping between IP addresses and media addresses. Because most hosts support dynamic address resolution, you generally do not need to specify static ARP cache entries. If you do need to define them, you can do so globally. Specifying static ARP cache entries installs a permanent entry in the ARP cache. The protocol translator uses these entries to translate 32-bit IP addresses into 48-bit hardware (MAC-level) addresses.

Optionally, you can specify that the protocol translator respond to ARP requests as if it were the owner of the specified IP address, and you also have the option of specifying an ARP entry timeout period when you define ARP entries.

To provide dynamic mapping between IP addresses and media address, perform one or more of the following tasks in global configuration mode:

Task	Command
Associate an IP address with a media (hardware) address in the ARP cache.	**arp** *internet-address hardware-address type*
Specify that the protocol translator respond to ARP requests as if it were the owner of the specified IP address.	**arp** *internet-address hardware-address type* [**alias**]

Perform the following task in interface configuration mode:

Task	Command
Set the length of time for an interface that an ARP cache entry will stay in the cache.	**arp timeout** *seconds*

To display the type of ARP being used on a particular interface and the ARP timeout value, use the **show interfaces** EXEC command. To examine the contents of the ARP cache, use the **show arp** EXEC command. To show IP entries, use the **show ip arp** EXEC command shows. To remove all nonstatic entries from the ARP cache, use the privileged EXEC command **clear arp-cache**.

Set ARP Encapsulations

By default, standard Ethernet-style ARP encapsulation (represented by the **arpa** keyword) is enabled on the IP interface. You can change this encapsulation method to SNAP, as required by your network, to control the interface-specific handling of IP address resolution into 48-bit Ethernet hardware addresses.

To specify the ARP encapsulation type, perform the following task in interface configuration mode:

Task	Command
Specify one of three ARP encapsulation methods for a specified interface.	**arp** {**arpa** | **snap**}

Map Logical Names to IP Addresses

Each unique IP address assigned by the DDN NIC can have a logical name associated with it. The logical name can be used to define a location descriptor of sorts. The protocol translator maintains a cache of host name-to-address mappings for use by the EXEC **connect**, **telnet**, **ping** and related Telnet support operations. This cache speeds the process of converting names to addresses.

IP defines a naming scheme that allows a device to be identified by its location in the Internet. This is a hierarchical naming scheme that provides for *domains* in the sequence local–group–site. Domain names are pieced together by a period (.) as a delimiting character. For example, Cisco Systems is a commercial organization that the Internet identifies by a *com* domain name, so that our group and site domain name is *cisco.com*. A specific device in this domain, the FTP system for example, is identified as *ftp.cisco.com*.

To keep track of domain names, IP has defined the concept of a *name server* whose job it is to hold a cache, or database, of names mapped to IP addresses. To map domain names to IP addresses, you must first identify the logical names, then specify a name server, and enable the domain name system (DNS). To do these, perform the following tasks:

- Map IP addresses to a host name.
- Specify the domain name.
- Specify a name server.
- Enable the Domain Name System (DNS).

The following sections describe these tasks.

Map IP Addresses to a Host Name

The protocol translator maintains a table of logical names and their corresponding addressing, also called a *host name-to-address mapping*. Higher-layer protocols such as Telnet use host names to identify network devices (hosts). The protocol translator and other network devices must be able to associate host names with IP addresses to communicate with other IP devices. Host names and IP addresses can be associated with one another through static or dynamic means.

Manually assigning host names to addresses is useful when you want to force a local address association and you are reasonably certain this address association will not conflict with other associations elsewhere in the internetwork.

To map IP addresses to a host name, perform the following global configuration task:

Task	Command
Statically associate a host names with IP addresses.	**ip host** *name* [*TCP-port-number*] *address1* [*address2...address8*]

Specify the Domain Name

You can specify a default domain name that the protocol translator software will use to complete domain name requests. You can specify a single domain name or a list of domain names. Any IP host name that does not contain a domain name will have the domain name appended to it before being added to the host table.

To specify a domain name, perform one of the following global configuration tasks:

Task	Command
Define a default domain name that the protocol translator will use to complete unqualified host names.	**ip domain-name** *name*
Define a list of default domain names to complete unqualified host names.	**ip domain-list** *name*

Specify a Name Server

To specify one or more hosts (up to six) that can function as a name server to supply name information for the DNS, perform the following global configuration task:

Task	Command
Specify one or more hosts that supply name information.	**ip name-server** *server-address1* [*server-address2...server-address6*]

Enable the Domain Name System (DNS)

If your network devices require connectivity with devices in networks for which you do not control name assignment, you can assign device names that uniquely identify your devices within the entire internetwork. The Internet uses a global naming scheme called the Domain Name System (DNS) that accomplishes this task. DNS is enabled by default.

To enable or disable the DNS, perform one of the following global configuration tasks, as appropriate:

Task	Command
Enable DNS-based host name-to-address translation.	**ip domain-lookup**
Disable DNS-based host name-to-address translation.	**no ip domain-lookup**

Assign an IP Address to a TCP Service

You can assign an IP address to the service provided on a TCP port. Additionally, multiple IP addresses can be assigned to provide connection to the first free line in a rotary group that appears transparent to a user.

To assign an IP address to a TCP service, perform the following global configuration task:

Task	Command
Assign an IP address or addresses to a TCP service.	**ip alias** *IP-address TCP-port*

The IP addresses must be on the same network as the protocol translator's main IP address and must not be used by another host on that network or subnetwork.

Configure Routing Assistance When IP Routing is Disabled

When IP routing is disabled, you can configure the protocol translator to act as an IP host and provide routing assistance. There are three methods for configuring the protocol translator software to do this:

- Configure proxy ARP
- Configure a default router (also known as a default gateway)
- Configure the router discovery mechanism

When IP routing is disabled, the default gateway feature is enabled and proxy ARP is disabled. When IP routing is enabled, the default gateway feature is disabled and you can configure proxy ARP. The router discovery mechanism works only when IP routing is disabled.

Configure Proxy ARP

The most common method of learning about other routes is by using proxy ARP. Proxy ARP, which is defined in RFC 1027, enables an Ethernet host with no knowledge of routing to communicate with hosts on other networks or subnets. Such a host assumes that all hosts are on the same local Ethernet and that it can use ARP to determine their hardware addresses.

Under proxy ARP, if a protocol translator receives an ARP Request for a host that is not on the same network as the ARP Request sender, the protocol translator evaluates whether it has the best route to that host. If the protocol translator does have the best route, it sends an ARP Reply packet giving its own Ethernet hardware address. The host that sent the ARP Request then sends its packets to the protocol translator, which forwards them to the intended host. The software treats all networks as if they are local and performs ARP requests for every IP address.

Proxy ARP works as long as other protocol translators support it. While this feature is supported on our protocol translators, many other systems, especially host-based routing software, do not support it.

Proxy ARP is enabled by default.

Configure a Default Router

Another method for locating routes is to define a default router (also referred to as a gateway).The protocol translator sends all nonlocal packets to this router, which routes them appropriately. If another router has a better route to the requested destination, the default router sends an ICMP redirect message to the protocol translator. The ICMP redirect message indicates a better local route to use next time. The protocol translator caches these redirect messages, and routes each packet thereafter as efficiently as possible. The limitation to this method is that there is no means of detecting when the default router has crashed or is unavailable and no method of picking another gateway if one of these events occurs.

When you configure a default gateway, proxy ARP is disabled.

The protocol translator can use a default router to send packets to hosts that are not on the local subnet or network.

To set up a default gateway for the protocol translator, perform the following global configuration task:

Task	Command
Set up a default router (gateway).	**ip default-gateway** *address*

To display the address of the default router, use the **show ip redirects** EXEC command.

Configure the Router Discovery Mechanism

The router discovery mechanism allows the protocol translator to learn the routes to other networks that use routing protocols. This mechanism uses the Cisco-defined Gateway Discovery Protocol (GDP) for detecting routers or the ICMP Router Discovery Protocol (IRDP). The protocol translator software also can wiretap RIP and IGRP routing updates and infer from these updates the location of routers. The server-client implementation of router discovery does not actually examine or store the full routing tables sent by routers, it merely keeps track of which systems are sending such data.

You can use any combination of the following routing protocols to learn routes to other networks:

- Gateway Discovery Protocol (GDP)
- ICMP Router Discovery Protocol (IRDP)
- Routing Information Protocol (RIP)
- Interior Gateway Routing Protocol (IGRP)

You can configure these protocols in any combination. When possible, use IRDP as the routing discovery mechanism, because it allows each protocol translator to specify both a priority and an amount of time after which a protocol translator should be assumed down if no further packets are received. Routers discovered using IGRP are assigned an arbitrary priority of 60. Routers discovered through RIP are assigned a priority of 50. For IGRP and RIP, the software attempts to measure the time between updates, and will assume that the router is down if no updates are received for 2.5 times that interval.

Each router discovered by the protocol translator becomes a candidate for the default router. The list of candidates is scanned and a new highest-priority router is selected when any of the following events occur:

- When a higher-priority router is discovered. The list of routers is polled at 5-minute intervals.
- When the current default router is declared down.

- When a TCP connection is about to time out due to excessive retransmissions. In this case, the server flushes the ARP cache and the ICMP redirect cache and picks a new default router in an attempt to find a successful route to the destination.

Detect Routers Using GDP

To configure the router discovery feature using the Cisco GDP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the GDP protocol to configure router discovery.	**ip gdp gdp**

Use the **no ip gdp gdp** command to turn the router discovery mechanism off.

Detect Routers Using IRDP

To configure the router discovery mechanism using the IRDP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the IRDP protocol to configure router discovery.	**ip gdp irdp**

Use the **no ip gdp irdp** command to turn the router discovery mechanism off.

Detect Routers Using RIP

To configure the router discovery feature using the RIP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the UNIX RIP protocol to configure router discovery.	**ip gdp rip**

Use the **no ip gdp rip** command to turn the router discovery mechanism off.

Detect Routers Using the IGRP

To configure the router discovery feature using the IGRP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the IGRP protocol to configure router discovery.	**ip gdp igrp**

Use the **no ip gdp igrp** command to turn the router discovery mechanism off.

Configure Broadcast Packet Handling

A broadcast packet is a data packet destined for all hosts on a particular physical network. Network hosts recognize broadcasts by special addresses. Broadcasts are heavily utilized by some protocols, including several important Internet protocols. Control of broadcast messages is an essential part of the IP network administrator's job.

Our protocol translators 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.

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.

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

For detailed discussions of broadcast issues, see RFC 919, "Broadcasting Internet Datagrams," and RFC 922, "Broadcasting Internet Datagrams in the Presence of Subnets." The protocol translator support for Internet broadcasts generally complies with RFC 919 and RFC 922; however, the protocol translator does not support multisubnet broadcasts as defined in RFC 922.

There are several standard ways of indicating an IP broadcast address. Currently, the most popular way is an address consisting of all ones (255.255.255.255). However, our protocol translators can be configured to generate as well as receive and understand any form of IP broadcast address.

To change the protocol translator's broadcast address, perform the following task in interface configuration mode:

Task	Command
Establish a broadcast address.	**ip broadcast-address** [*address*]

If the protocol translator does not have nonvolatile memory and you want to specify the broadcast address to use before the protocol translator has been configured, you can change the IP broadcast address by setting jumpers in the processor configuration register. Setting bit 10 causes the protocol translator 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 protocol translator to include the network and subnet portions of its address in the broadcast address. Table 1-2 shows the combined effect of setting bits 10 and 14.

Table 1-2 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, consult the hardware installation and maintenance manual for your system.

Configure IP Services

The Internet protocol suite offers a number of services that control and manage IP connections. Many of these services are provided by the Internet Control Message Protocol (ICMP). ICMP messages are sent by protocol translators to hosts or other protocol translators when a problem is discovered with the IP header. For detailed information about ICMP, see RFC 792.

This section describes the following tasks for configuring IP services:

- Disable generation of ICMP Protocol Unreachable messages.
- Disable generation of ICMP Redirect messages.
- Set the IP MTU size.
- Enable ICMP Mask Reply messages.
- Configure IP source-route header options.

Disable Generation of ICMP Protocol Unreachable Messages

If the protocol translator receives a nonbroadcast packet destined for itself that uses an unknown protocol, it sends an ICMP Protocol Unreachable message back to the source. Similarly, if the protocol translator receives a packet that it is unable to deliver to the ultimate destination because it knows of no route to the destination address, it sends an ICMP Host Unreachable message to the source.

This feature is enabled by default. To disable the sending of ICMP protocol unreachable messages, perform the following task in interface configuration mode:

Task	Command
Disable the sending of ICMP Protocol Unreachable and Host Unreachable messages.	**no ip unreachables**

Disable Generation of ICMP Redirect Messages

Routes sometimes can become less than optimal. For example, it is possible for the protocol translator to be forced to resend a packet through the same interface on which it was received. If this happens, the protocol translator sends an ICMP Redirect message to the packet's originator that it is on a subnet directly connected to the protocol translator when the protocol translator would otherwise forward the packet to another system on the same subnet. It does this because the

originating host presumably could have sent that packet to the next hop without involving the protocol translator at all. The Redirect message instructs the sender to remove the protocol translator from the route and substitute a specified device representing a more direct path.

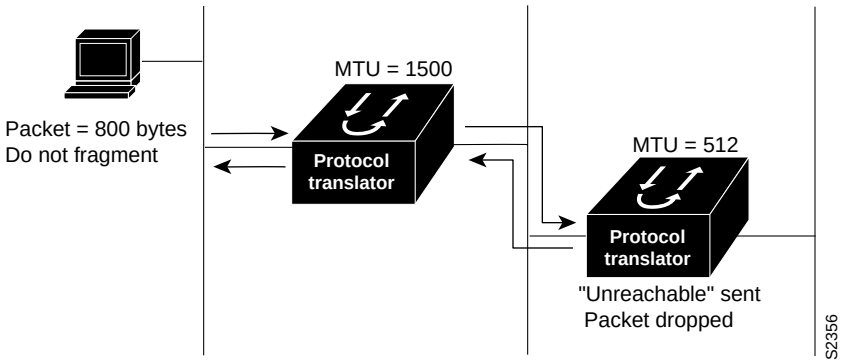
This feature is enabled by default.To disable the sending of ICMP Redirect messages, perform the following task in interface configuration mode:

Task	Command
Disable the sending of ICMP Redirect messages to learn routes.	**no ip redirects**

Set the IP MTU Size

Our protocol translators support the IP Path Maximum Transmission Unit (MTU) Discovery mechanism, as defined in RFC 1191. 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 protocol translator is unable to forward a datagram because it requires fragmentation (the packet is larger than the MTU you set for the interface with the **ip mtu** interface configuration command), but the "Don't fragment" (DF) bit is set. The protocol translator sends a message to the sending host, alerting it to the problem. The host will have to fragment packets destined for the receiving interface so that they fit the smallest packet size of all the links along the path. This technique is shown in Figure 1-1.

Figure 1-1 Path MTU Discovery



Path MTU Discovery is useful when a link in a network goes down, forcing use of another, different MTU-sized link (and different protocol translators). As shown in Figure 1-1, suppose one host is trying to send IP packets over a network where the MTU in the first protocol translator is set to 1500 bytes, but then reaches a protocol translator 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-byte protocol translator is unable to forward it. All packets larger than 512 bytes will be dropped in this case. The second protocol translator 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 also would include the MTU of the next-hop network link in the low-order bits of an unused header field.

Path MTU Discovery also is 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.

Note MTU Discovery is a process initiated by end hosts. If an end host does not support MTU Discovery, a protocol translator will have no mechanism available to avoid fragmenting datagrams generated by the end host.

All interfaces have a default MTU size. You can adjust the IP MTU size so that if an IP packet exceeds the MTU set for a protocol translator's interface, the protocol translator will fragment it.

Changing the MTU value (with the **mtu** interface configuration command) can affect the IP MTU value. If the current IP MTU value is the same as the MTU value, and you change the MTU value, the IP MTU value will be modified automatically to match the new MTU. However, the reverse is not true: changing the IP MTU value has no effect on the value for the **mtu** interface subcommand.

Also, all devices on a physical medium must have the same protocol MTU size in order to operate.

To set the IP MTU size for a specified interface, perform the following task in interface configuration mode:

Task	Command
Set the IP MTU packet size for an interface.	**ip mtu** *bytes*

Enable ICMP Mask Reply Messages

Occasionally, network devices need to know the subnet mask for a particular subnetwork in the internetwork. To obtain this information, the devices can send ICMP Mask Request messages. Devices that have the requested information respond to these messages with ICMP Mask Reply messages. You can configure the protocol translator so that it sends ICMP Mask Reply messages by performing the following task in interface configuration mode:

Task	Command
Enable the sending of ICMP Mask Reply messages.	**ip mask-reply**

Configure IP Source-Route Header Options

The protocol translator examines IP header options on every packet. It supports the IP header options Strict Source Route, Loose Source Route, Record Route, and Time Stamp, which are defined in RFC 791. If the protocol translator finds a packet with one of these options enabled, it performs the appropriate action. If it finds a packet with an invalid option, it sends an ICMP Parameter Problem message to the source of the packet and discards the packet.

IP allows the source IP host to specify a route through the IP network. This is known as source routing. Source routing is specified as an option in the IP header. If source routing is specified, the protocol translator forwards the packet according to the specified source route. Use this feature to force a packet to take a certain route through the network.

By default, source routing is enabled.To disable source routing, perform the following global configuration task:

Task	Command
Discard any IP datagram containing a source-route option.	**no ip source-route**

Control Access to IP Networks

To control access to IP networks, you create access lists and then apply them with filters to individual interfaces. You can use access lists to control the transmission of packets on an interface and to control virtual terminal line access.

To control access to IP networks, perform the following tasks:

Step 1 Create an access list.

Step 2 Apply an access list to an interface.

These steps are described in the next sections.

Create an Access List

An access list is a sequential collection of permit and deny conditions that apply to IP addresses. The protocol translator tests addresses against the conditions in an access list one by one. The first match determines whether the protocol translator accepts or rejects the address. Because the protocol translator stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the protocol translator rejects the address.

The software supports two styles of IP access lists for IP:

- Standard IP access lists use source addresses for matching operations.
- Extended IP access lists use source and destination addresses for matching operations, as well as optional protocol type information.

Keep in mind when making the standard access list that, by default, the end of the access list contains an implicit deny statement for *everything* if it did not find a match before reaching the end. Further, if you omit the mask from an associated IP host address access list specification, 0.0.0.0 is assumed to be the mask. Be aware of these default conditions; plan your access conditions carefully.

There are *implicit* masks in IP access lists. Refer to the “IP Configuration Examples” section in this chapter for examples of implicit masks.

To create an access list, perform one of the following tasks in global configuration mode:

Task	Command
Define an IP access list number and the access conditions.	**access-list** *access-list-number* {**permit** | **deny**} *source source-mask*
Define an extended IP access list number and the access conditions.	**access-list** *access-list-number* {**permit** | **deny**} *protocol source source-mask destination destination-mask* [*operator operand*] [**established**]

After you initially create an access list, any subsequent additions (possibly entered from the terminal) are placed at the *end* of the list. In other words, you cannot selectively add or remove access list command lines from a specific access list.

Apply an Access List to an Interface

You can apply an access list to one or more interfaces by performing one or both of the following tasks in line configuration mode:

Task	Command
Restrict incoming and outgoing connections between a particular virtual terminal line (into a device) and the addresses in an access list.	**access-class** *access-list-number* {**in** | **out**}
Filter incoming and outgoing packets.	**ip access-group** *access-list-number* {**in** | **out**}

After receiving and routing a packet to a controlled interface, the protocol translator checks the source address of the packet against the access list. If the access list permits the address, the protocol translator transmits the packet. If the access list rejects the address, the protocol translator discards the packet and returns an ICMP Host Unreachable message.

You can apply access lists are either on *either* outbound or inbound interfaces.

Note Enabling input access lists on any interface disables fast switching of IP for the *entire protocol translator.*

When you apply to an interface an access list that has not yet been defined, the protocol translator acts as if the access list has not been applied to the interface and it will accept all packets. Consider this behavior if you use undefined access lists as a means of security in your network.

Note Set identical restrictions on all the virtual terminal lines, because a user can connect to any of them.

Configure IP Security Options

Our IP Security Option (IPSO) support addresses both the basic and extended security options as described in RFC 1108. Our implementation is only minimally compliant with RFC 1108, because our protocol translator accepts and generates a four-byte IPSO only. Generally, IPSO is used to comply with U.S. Department of Defense security policy.

IPSO provides the following features:

- Defines security levels on a per-interface basis
- Defines single-level or multilevel interfaces
- Provides a label for incoming packets

- Strips labels on a per-interface basis
- Reorders options to put any basic security options first
- Accepts or rejects messages with extended security options

To configure IPSO-level security, perform the following steps:

Step 1 Enable IPSO and set the security classifications.

Step 2 Select the security levels.

These tasks are described in the following sections.

Enable IPSO and Set the Security Classifications

To enable IPSO and set security classifications on an interface, perform one of the following tasks in interface configuration mode:

Task	Command
Set an interface to the requested IPSO classification and authorities.	**ip security dedicated** *level authority* [*authority* ...]
or	
Set an interface to the requested IPSO range of classifications and authorities.	**ip security multilevel** *level1* [*authority1* ...] **to** *level2 authority2* [*authority2* ...]

Select the Security Levels

To select IPSO security levels, perform one of the following interface configuration tasks:

Task	Command
Ensure that all outgoing packets contain a basic security option.	**ip security add**
Accept packets that have an extended security option.	**ip security extended-allowed**
Prioritize security options on a packet.	**ip security first**
Ignore the authorities field of all incoming packets.	**ip security ignore-authorities**
Treat as valid any packets that have Reserved1 to Reserved4 security levels.	**ip security reserved-allowed**
Classify packets that have no IPSO with an implicit security label.	**ip security implicit-labelling** [*level authority* [*authority* ...]]
Remove any basic security option that may be present on an outgoing packet.	**ip security strip**

Default Values for Minor Keywords

In order to fully comply with IPSO, the default values for various **ip security** commands have become complex. Default value usages include the following:

- The default for all of the **ip security** commands is off, with the exception of **ip security implicit-labelling** and **ip security add.**
- The default for **ip security implicit-labelling** is on if the interface is unclassified Genser; otherwise, it is off.
- The default for **ip security add** is on if the interface is not unclassified Genser; otherwise, it is off.

Table 1-3 provides a list of all default values.

Table 1-3 Default Security Keyword Values

Interface Type	Security Level	Authority	Implicit Labelling	Add IPSO
None	None	None	On	Off
Dedicated	Unclassified	Genser	On	Off
Dedicated	Any	Any	Off	On
Multilevel	Any	Any	Off	On

The default for any interface is “dedicated, unclassified Genser.” Note that this implies implicit labeling. This may seem unusual, but it makes the system entirely transparent to packets without options. This is the setting generated when you specify the **no ip security** interface configuration command.

Optimize Lines for Telnet

The IP suite includes the simple remote terminal protocol called Telnet. Telnet allows a user at one site to establish a TCP connection to a login server at another site, then passes the keystrokes from one system to the other. Telnet can accept either an Internet address or a domain name as the remote system address. Telnet offers three main services:

- Network virtual terminal connection
- Option negotiation
- Symmetric connection

Our implementation of Telnet supports the following Telnet options:

- Remote 3cho
- Binary transmission
- Suppress go ahead
- Timing mark
- Terminal type
- Send location
- Terminal speed
- Remote flow control
- X display location

For information about configuring lines to support Telnet connections, see the chapter entitled "Configuring Terminal Sessions and Modem Support" in the *Router Products Configuration Guide*. For information about making Telnet connections, see the *Communication Server and Protocol Translator Connection Guide*.

To optimize lines for Telnet, perform the following tasks:

- Optimize response to Telnet user interrupt characters.
- Set the TCP connection-attempt time.

Optimize Response to Telnet User Interrupt Characters

When used with a correctly operating host, protocol translators implement the Telnet Synchronize and Abort Output signals, which can stop output within one packet's worth of data from the time the user types the interrupt character. You can configure a faster response to user interrupt characters.

Changing the chunk size affects neither the size of the packet used nor the TCP window size, either of which would cause serious efficiency problems for the remote host as well as for the protocol translator. Instead, the Telnet status is checked after the number of characters specified, causing only a relatively minor performance loss.

To optimize response to Telnet user interrupt characters, complete the following global configuration task:

Task	Command
Optimize the line by setting the number of characters output before the interrupt executes.	**ip tcp chunk-size** *number*

Set the TCP Connection-Attempt Time

You can set the amount of time the protocol translator will wait to attempt to establish a TCP connection. In previous versions of protocol translator software, the system would wait a fixed 30 seconds when attempting to do so. This is not sufficient time in networks that have dialup asynchronous connections, such as a network consisting of dial-on-demand links that are implemented over modems, because it will affect your ability to Telnet over the link (from the protocol translator) if the link must be brought up.

Because this is a host parameter, it does not pertain to traffic going through the protocol translator, just to traffic originating at the protocol translator.

To set the TCP connection-attempt time, perform the following global configuration task:

Task	Command
Set the amount of time the protocol translator will wait to attempt to establish a TCP connection.	**ip tcp synwait-time** *seconds*

Monitor and Maintain the TCP/IP Network

To monitor and maintain the TCP/IP network, perform the following tasks:

- Display system and network statistics.
- Clear caches, tables, and databases.

These tasks are described in the following sections.

Display System and Network Statistics

You can display specific protocol translator statistics such as the contents of IP routing tables, caches, and databases. This information can be used to determine resource utilization and solve network problems. You can also display information about node reachability and discover the routing path your protocol translator's packets are taking through the network.

To display system and network statistics, perform one or more of the following tasks in EXEC mode:

Task	Command
Display the contents of all current access lists.	**show access-lists**
Display the default domain name, style of lookup service, the name server hosts, and the cached list of host names and addresses.	**show hosts**
Display Internet addresses mapped to TCP ports (aliases).	**show ip aliases**
Display the IP ARP cache.	**show ip arp**
Display the usability status of interfaces.	**show ip interface** [*interface unit*]
Display the address of a default gateway.	**show ip redirects**
Display IP protocol statistics.	**show ip traffic**
Test network node reachability.	**ping**
Test network node reachability using a simple ping facility.	**ping** *protocol* {*host* | *address*}
Trace packet routes through the network.	**trace** [*destination*]

Clear Caches, Tables, and Databases

You can remove all contents of a particular cache, table, or database. Clearing a cache, table, or database can become necessary when the contents of the particular structure have become or are suspected to be invalid.

To clear caches, tables, and databases, perform one or both of the following tasks in EXEC mode:

Task	Command
Clear the IP ARP cache, the fast-switching cache, and the IP route cache.	**clear arp-cache**
Remove one or all entries from the hostname and address cache.	**clear host** {*name* | *****}

IP Configuration Examples

This section shows complete configuration examples for the following common configuration situations:

IP Domain Example

The following example establishes a domain list with several alternate domain names.

```
ip domain-list cisco.com
ip domain-list telecomprog.edu
ip domain-list merit.edu
```

Dynamic Lookup Example

A cache of host name-to-address mappings is used by **connect**, **telnet**, **ping**, **trace**, **write network**, and **configure network** EXEC commands to speed the process of converting names to addresses. The commands used in the following example specify the form of dynamic name lookup to be used. Static name lookup can also be configured.

The following example configures the host name-to-address mapping process for the protocol translator. IP DNS-based translation is specified, the addresses of the name servers are specified, and the default domain name is given.

```
! IP Domain Name System (DNS)-based host name-to-address resolution is enabled
ip domain-lookup
! 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
! Defines cisco.com as the default domain name the protocol translator uses to complete
! unqualified host names
ip domain-name cisco.com
```

Access List Examples

In the following 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 protocol translator would accept one address on subnet 48 and reject all others on that subnet. The protocol translator would accept addresses on all other network 36.0.0.0 subnets; that is the purpose of the last line of the list.

```
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
ip access-group 2
```

Implicit Masks in Access Lists

There are *implicit* masks in IP access lists. For instance, if you omit the mask from an associated IP host address access list specification, 0.0.0.0 is assumed to be the mask. Consider the following example configuration:

```
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
```

In the following example, the following masks are implied in the first two lines:

```
access-list 1 permit 0.0.0.0 0.0.0.0
access-list 1 permit 131.108.0.0 0.0.0.0
```

The last line in the configuration (using the deny keyword) can be left off, because IP access lists implicitly *deny* all other access. This is equivalent to finishing the access list with the following command statement:

```
access-list 1 deny 0.0.0.0 255.255.255.255
```

The following access list only allows access for 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. Any hosts with a source address that does not match the access list statements will be rejected.

```
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
! (Note: all other access implicitly denied)
```

To specify a large number of individual addresses more easily, you can omit the address mask that is all zeros from the **access-list** global 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

In the following example, 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.

```
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
ip access-group 102
```

For another example of using an extended access list, suppose you have a network connected to the Internet, and you want any host on an 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 to hosts on the Ethernet except to the mail (SMTP) port of a dedicated mail host.

SMTP uses TCP port 25 on one end of the connection and a random port number on the other end. The same two port numbers are used throughout the life of the connection. Mail packets coming in from the Internet will have a destination port of 25. Outbound packets will have the port numbers reversed. The fact that the secure system behind the protocol translator always will be accepting mail connections on port 25 is what makes it possible to separately control incoming and outgoing services.The access list can be configured on either the outbound or inbound interface.

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. The keyword **established** is used only for the TCP protocol to indicate an established connection. A match occurs if the TCP datagram has the ACK or RST bits set, which indicate that the packet belongs to an existing connection.

```
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255 established
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 0.0.0.0 eq 25
interface ethernet 0
ip access-group 102
```