PREFACE

In year 1, year 2 and year 3, we studied the theories of Data Communication and Computer Networking. I hope that now you have thorough knowledge on this subject. Next step should be, apply your knowledge to industry available network devices. In this year you mainly study on this area. I have put more attention to teach you that how to configure the network devices to obtain your expected outcome. Also I have added some theory on Network Security.

Tilak De Silva
(B.Sc.Eng., C.Eng. , FIESL , FIEE , MBCS)

CONTENTS

1 - Overview of Configuration of Network Devices

1.1 Introduction

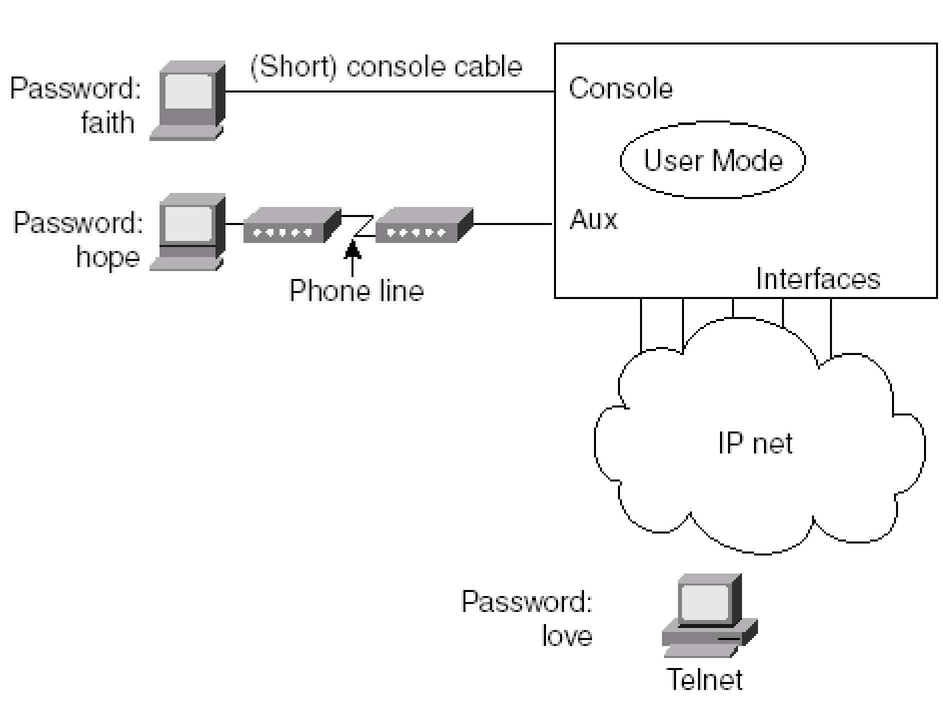
A network device is same as a small computer. It has a processor, RAM, ROM and Operating System (OS). In this book we mainly considered about CISCO devices. The other products also use the similar concepts with little deviations. The CISCO OS is called as Internetworking Operating System (IOS). For all examples throughout the book is used CISCO devices.

1.2 Command Line Interface (CLI)

Although computer has the monitor (terminal) for input commands and display results, network devices do not have a monitor. Therefore they use a computer monitor for this purpose.

CISCO uses the acronym CLI to refer to the terminal user command-line interfaces to the IOS. The term CLI implies that the user is typing command at a terminal, terminal emulator or a Telnet connection.

To access CLI it uses one of three methods as illustrated in figure.



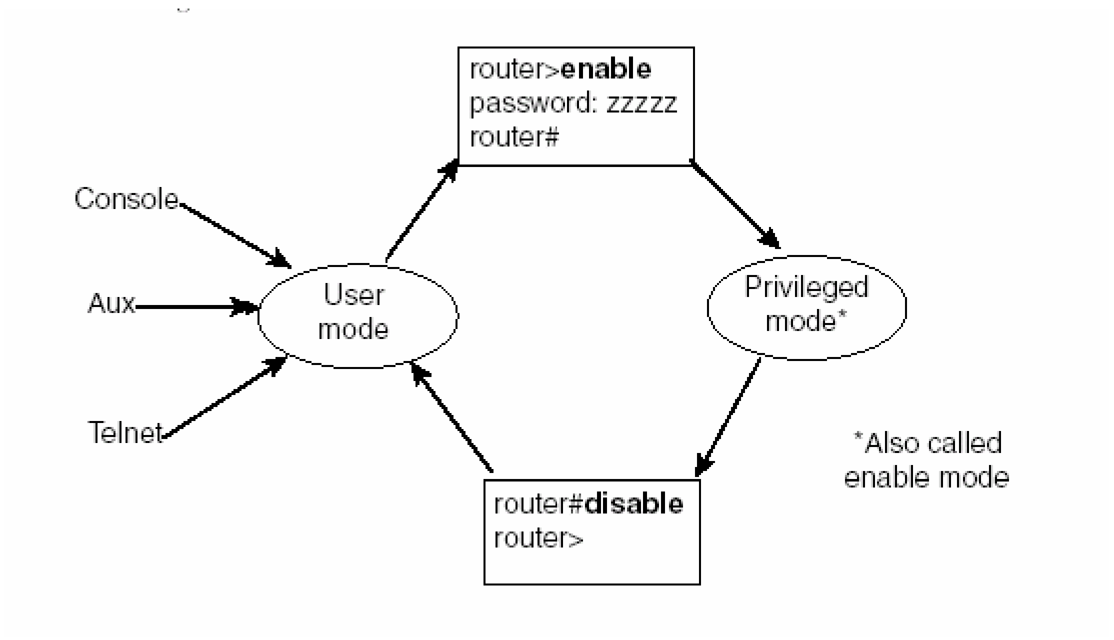
You access the router through the console, through a dialup device through a modem attached to the auxiliary port, or by using Telnet. The console, auxiliary, and Telnet passwords are all set separately. Some example passwords are shown in figure.

The commands used for each mode is shown in the table.

Access from . . .	**Password Type**	**Configuration**
Console	Console password	**line console** *0* **login** **password** *faith*
Auxiliary	Auxiliary password	**line aux** *0* **login** **password** *hope*
Telnet	vty password	**line vty** *0 4* **login** **password** *love*

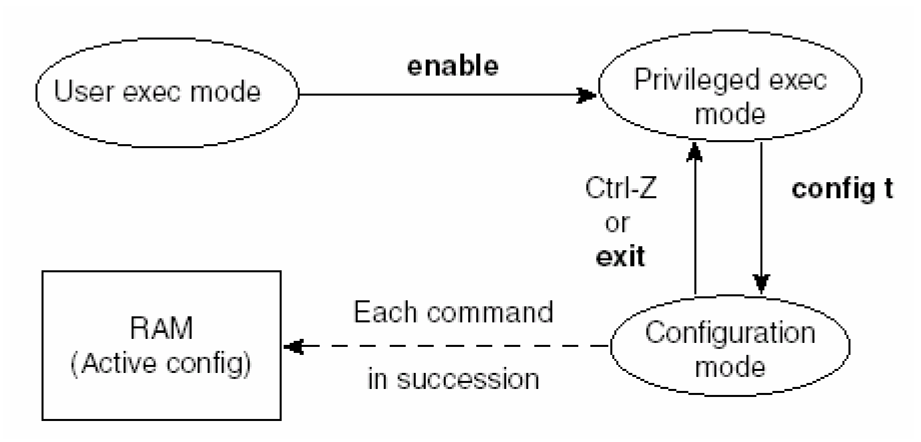
1.3 User configuration modes

There are two modes use in configuration.
User Mode
Privileged Mode (Enable Mode)



User mode is used to monitor the status of the device. In this mode you cannot change the existing configuration.
In order to change the configuration you should access through “privilege mode”. It is also called as “enable mode”.

C*onfiguration mode* is another mode used for the Cisco CLI, similar to user mode and privileged mode. User mode allows commands that are not disruptive to be issued, with some information being displayed to the user. Privileged mode supports a superset of commands compared to user mode, including commands that might harm the device. However, none of the commands in user or privileged mode changes the configuration of the device. Configuration mode is another mode in which configuration commands are typed. The figure illustrates the relationships among configuration mode, user exec mode, and privileged exec mode.



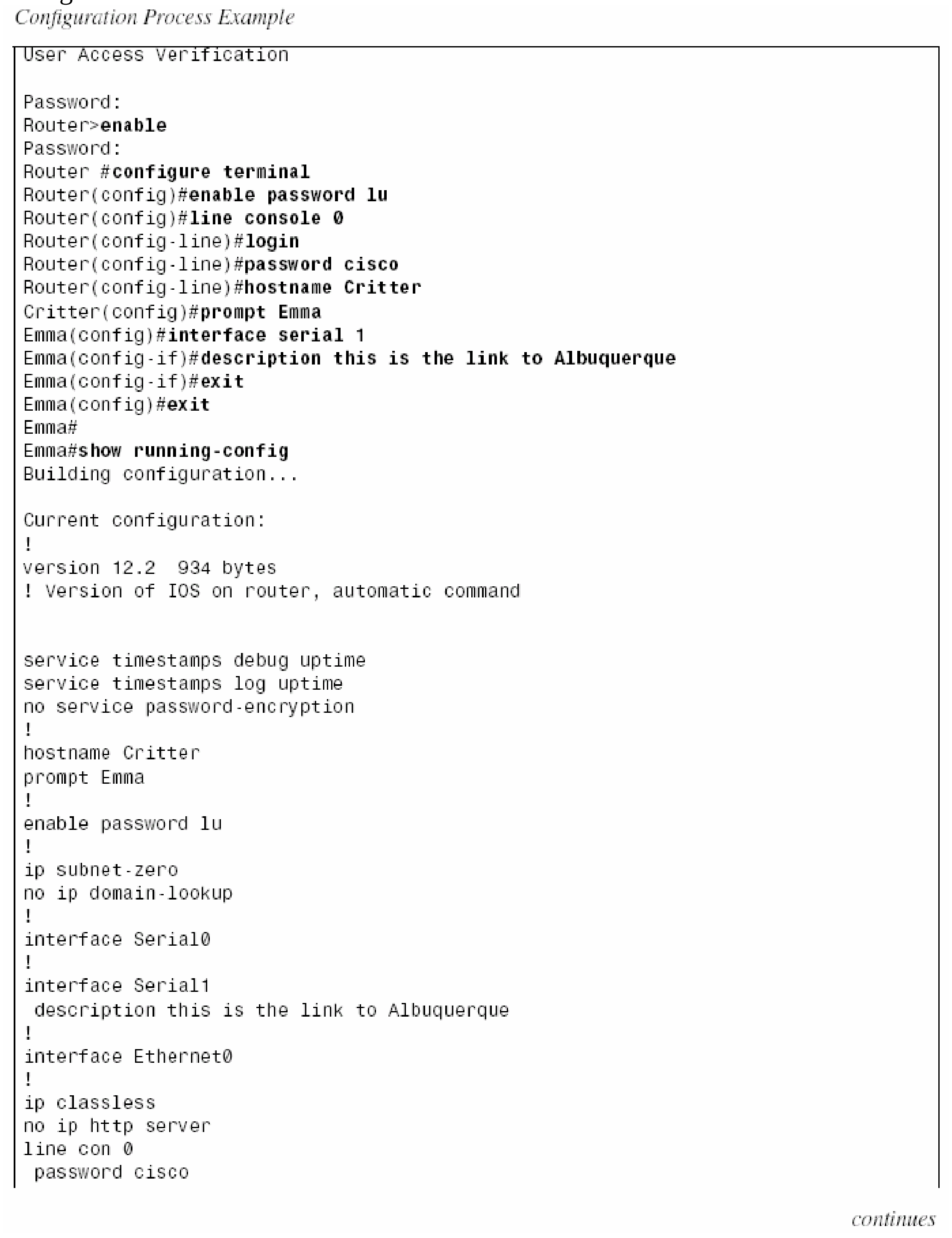
Commands typed in configuration mode update the active configuration file. *These changes to the configuration occur immediately each time you press the Enter key at the end of a command.*

There are two types of commands in the configuration mode. The commands common to the device is called "Global commands". For example the "hostname" is the command use to give a name to a router. It is a global command. There are parameters related to interfaces. For parameter changes in interfaces it should change to interface in the configuration mode. The relevant command is interface. For example, to change to Ethernet 0, the command *interface Ethernet 0* should be used. Then an ip address can be assigned to ethernet interface.

Use Ctrl-z from any part of configuration mode (or use the **exit** command from global configuration mode) to exit configuration mode and return to privileged exec mode. The configuration mode **end** command also exits from any point in the configuration mode back to privileged exec mode. The **exit** commands from submodes of configuration mode back up one level toward global configuration mode.

1.3.1 Example Configuration Process

Example illustrates how the console password is defined; provides banner, host name, prompt, and interface descriptions; and shows the finished configuration. The lines beginning with "!" are comment lines that highlight significant processes or command lines within the example. The **show running-config** command output also includes comment lines with just a "!" to make the output more readable—many comment lines in the examples in this book were added to explain the meaning of the configuration.



Configuration Process Example

```
User Access Verification

Password:
Router>enable
Password:
Router #configure terminal
Router(config)#enable password lu
Router(config)#line console 0
Router(config-line)#login
Router(config-line)#password cisco
Router(config-line)#hostname Critter
Critter(config)#prompt Emma
Emma(config)#interface serial 1
Emma(config-if)#description this is the link to Albuquerque
Emma(config-if)#exit
Emma(config)#exit
Emma#
Emma#show running-config
Building configuration...

Current configuration:
!
version 12.2  934 bytes
! Version of IOS on router, automatic command


service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Critter
prompt Emma
!
enable password lu
!
ip subnet-zero
no ip domain-lookup
!
interface Serial0
!
interface Serial1
 description this is the link to Albuquerque
!
interface Ethernet0
!
ip classless
no ip http server
line con 0
 password cisco
```

continues

Configuration Process Example (Continued)

```
 login
!
line aux 0
line vty 0 4
!
end
```

Several differences exist between user and privileged mode, compared to configuration mode. The **configure terminal** command is used to move into configuration mode. The command prompt changes based on the configuration subcommand mode that you are in. Plus, typing a **?** in configuration mode gives you help just on configuration commands. When you change from one configuration mode to another, the prompt changes. The same example repeats as shown below , but with more explanation.

Configuration Process with Annotations

```
User Access Verification

Password:
Router>enable
!In user mode, then you type the enable command
Password:
Router #configure terminal
!In privileged mode, using the configure terminal command to enter global
Router(config)#enable password lu
!The enable password command is a global command - so the prompt stays as a global
  command prompt
Router(config)#line console 0
!line console changes the context to console line configuration mode
Router(config-line)#login
!login is a console subcommand, so the prompt remains the same
Router(config-line)#password cisco
!password is also a console sub-command
Router(config-line)#hostname Critter
!hostname is a global command, so it is used, and the mode changes back to global
  config mode
Critter(config)#prompt Emma
!prompt is a global command, so the prompt stays as a global command mode prompt
Emma(config)#interface serial 1
!interface changes contexts to interface subcommand mode
Emma(config-if)#description link to Albuquerque
!description is a sub-command in interface config mode, so prompt stays the same
Emma(config-if)#exit
!exit backs up one mode towards global
Emma(config)#exit
!exit in global mode exits back to privileged mode
```

1.4 Memory

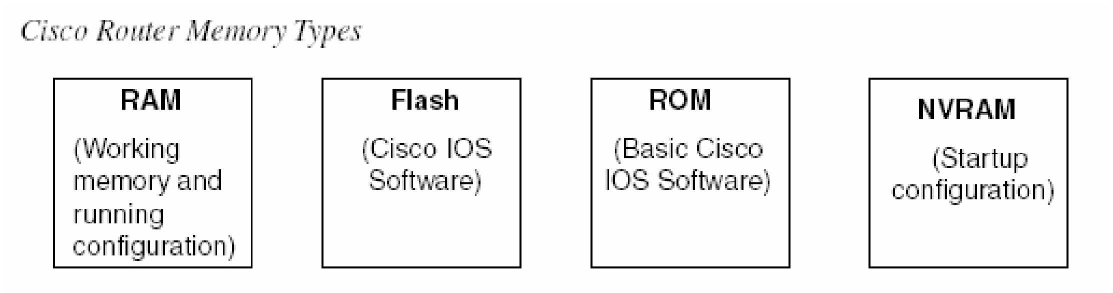
The configuration file contains the configuration commands that you have typed, as well as some configuration commands entered by default by the router. The onfiguration file can be stored in a variety of places, including two inside a router. The router has a couple of other types of memory as well:

• **RAM—**Sometimes called DRAM for dynamic random-access memory, RAM is used by the router just as it is used by any other computer: for working storage. The *running* or *active configuration* file is stored here.

• **ROM—**This type of memory (read-only memory) stores a bootable IOS image, which is not typically used for normal operation. ROM contains the code that is used to boot the router until the router knows where to get the full IOS image, or as a backup bootable image in case there are problems.

• **Flash memory—**Either an EEPROM or a PCMCIA card, Flash memory stores fully functional IOS images and is the default location where the router gets its IOS at boot time. Flash memory also can be used to store configuration files on some Cisco routers.

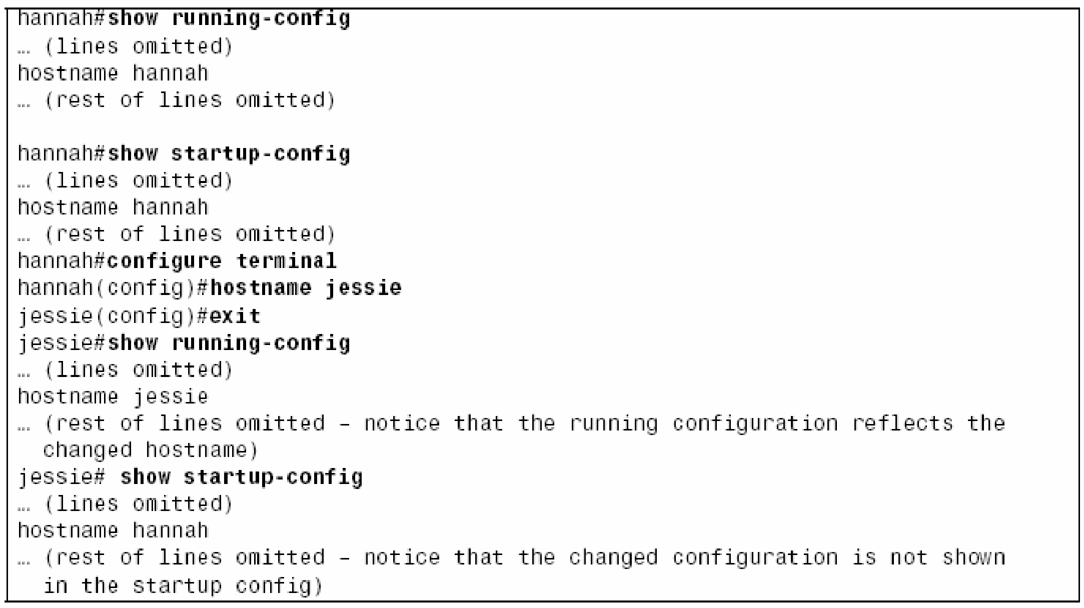
• **NVRAM—**Nonvolatile RAM stores the initial or *startup* configuration file.
All these types of memory, except RAM, are permanent memory. No hard disk or diskette storage exists on Cisco routers. The figure summarizes the use of memory in Cisco routers.


Cisco Router Memory Types
RAM
(Working memory and running configuration)
Flash
(Cisco IOS Software)
ROM
(Basic Cisco IOS Software)
NVRAM
(Startup configuration)

1.5 Managing Configuration files

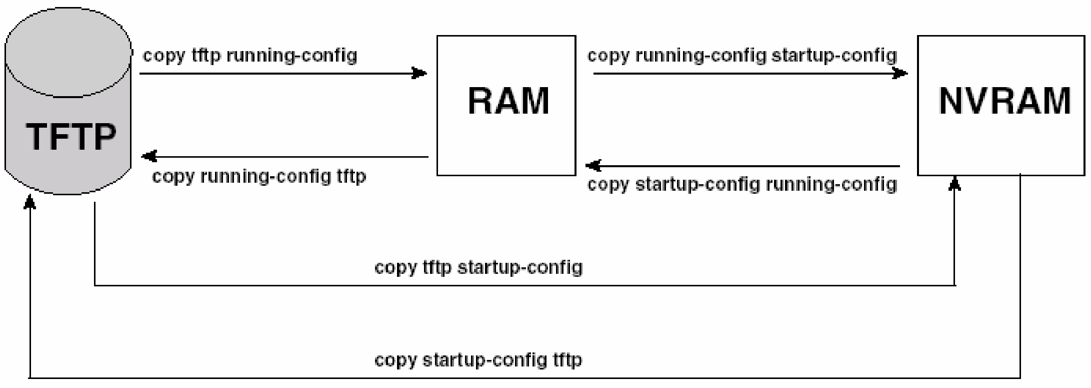
The startup configuration file is in NVRAM; the other file, which is in RAM, is the one that the router uses during operation. When the router first comes up, the router copies the stored configuration file from NVRAM into RAM, so the running and startup configuration files are identical at that point. Also, exterior to the router, configuration files can be stored as ASCII text files anywhere using TFTP or FTP.
Following example demonstrates the basic interaction between the two files. In this example, the **show running-config** and **show startup-config** commands are used. These commands display the currently used, active, *running* configuration, and the stored, *startup* configuration used when the router boots, respectively. The full command output is not shown.

Configuration Process Example



```
hannah#show running-config
… (lines omitted)
hostname hannah
… (rest of lines omitted)

hannah#show startup-config
… (lines omitted)
hostname hannah
… (rest of lines omitted)
hannah#configure terminal
hannah(config)#hostname jessie
jessie(config)#exit
jessie#show running-config
… (lines omitted)
hostname jessie
… (rest of lines omitted - notice that the running configuration reflects the
  changed hostname)
jessie# show startup-config
… (lines omitted)
hostname hannah
… (rest of lines omitted - notice that the changed configuration is not shown
  in the startup config)
```

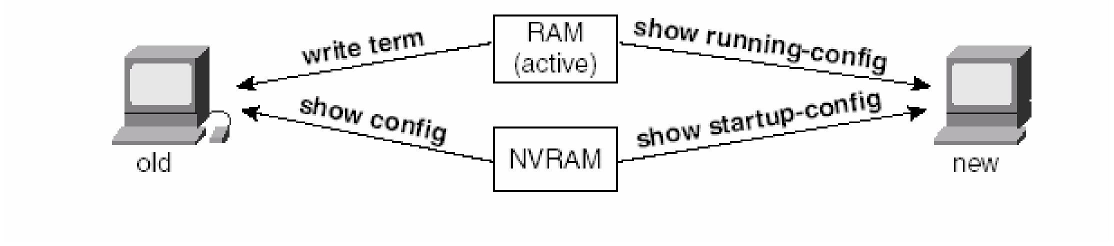
If you reload the router now, the host name would revert back to hannah. However, if you want to keep the changed host name of jessie, you would use the command **copy running-config startup-config**, which overwrites the current startup-config file with what is currently in the running configuration file. The **copy** command can be used to copy files in a router, most typically a configuration file, or a new version of the IOS Software. The most basic method for moving configuration files in and out of a router is by using a TFTP server. The **copy** command is used to move configuration files among RAM, NVRAM, and a TFTP server. The files can be copied between any pair, as shown in the following figure.



Two key commands can be used to erase the contents of NVRAM. The **write erase** command is the older command, and the **erase startup-config** command is the newer command. Both simply erase the contents of the NVRAM configuration file. Of course, if the router is reloaded at this point, there will be no initial configuration.

The following figure shows both the old and the new commands used to view configurations.

Configuration **show** *Commands*

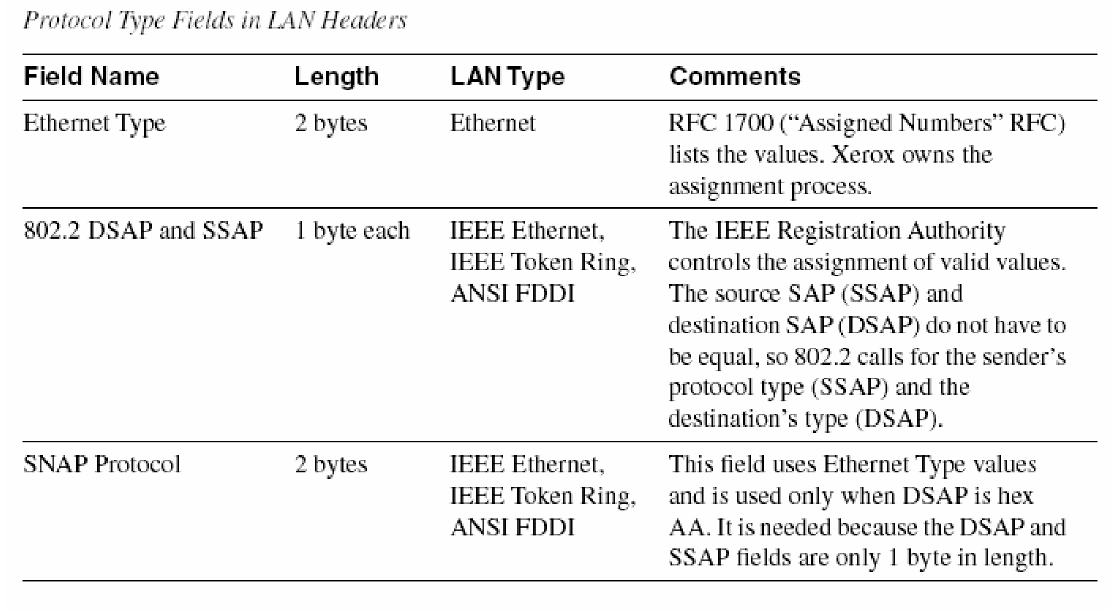


2 - Local Area Networks

2.1 LAN Frames

Different types of LAN frames are shown in the figure.

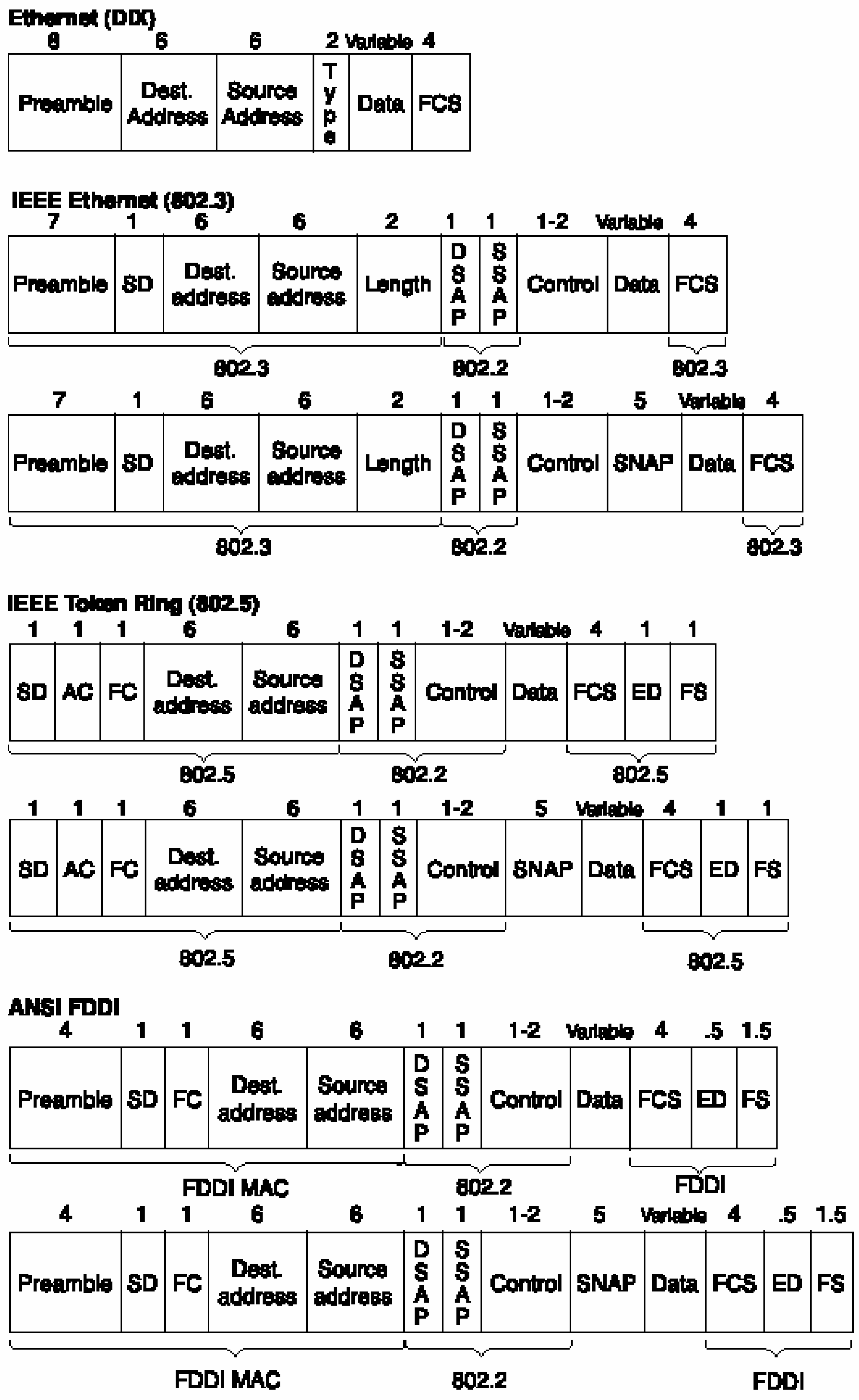
The LAN headers—namely, the Type, DSAP, and SNAP fields—is important. The 802.3 specification limits the data portion of the 802.3 frame to a maximum of 1500 bytes. The data was designed to hold Layer 3 packets; the term *maximum transmission unit (MTU)* defines the maximum Layer 3 packet that can be sent over a medium. Because the Layer 3packet rests inside the data portion of an Ethernet frame, 1,500 is the largest MTU allowed over an Ethernet.



Protocol Type Fields in LAN Headers

Field Name	Length	LAN Type	Comments
Ethernet Type	2 bytes	Ethernet	RFC 1700 (“Assigned Numbers” RFC) lists the values. Xerox owns the assignment process.
802.2 DSAP and SSAP	1 byte each	IEEE Ethernet, IEEE Token Ring, ANSI FDDI	The IEEE Registration Authority controls the assignment of valid values. The source SAP (SSAP) and destination SAP (DSAP) do not have to be equal, so 802.2 calls for the sender’s protocol type (SSAP) and the destination’s type (DSAP).
SNAP Protocol	2 bytes	IEEE Ethernet, IEEE Token Ring, ANSI FDDI	This field uses Ethernet Type values and is used only when DSAP is hex AA. It is needed because the DSAP and SSAP fields are only 1 byte in length.

LAN Header Formats


Ethernet (DIX)
8 6 6 2 Variable 4
Preamble | Dest. Address | Source Address | Type | Data | FCS
IEEE Ethernet (802.3)
7 1 6 6 2 1 1 1-2 Variable 4
Preamble | SD | Dest. address | Source address | Length | DSAP | SSAP | Control | Data | FCS
802.3 802.2 802.3
7 1 6 6 2 1 1 1-2 5 Variable 4
Preamble | SD | Dest. address | Source address | Length | DSAP | SSAP | Control | SNAP | Data | FCS
802.3 802.2 802.3
IEEE Token Ring (802.5)
1 1 1 6 6 1 1 1-2 Variable 4 1 1
SD | AC | FC | Dest. address | Source address | DSAP | SSAP | Control | Data | FCS | ED | FS
802.5 802.2 802.5
1 1 1 6 6 1 1 1-2 5 Variable 4 1 1
SD | AC | FC | Dest. address | Source address | DSAP | SSAP | Control | SNAP | Data | FCS | ED | FS
802.5 802.2 802.5
ANSI FDDI
4 1 1 6 6 1 1 1-2 Variable 4 .5 1.5
Preamble | SD | FC | Dest. address | Source address | DSAP | SSAP | Control | Data | FCS | ED | FS
FDDI MAC 802.2 FDDI
4 1 1 6 6 1 1 1-2 5 Variable 4 .5 1.5
Preamble | SD | FC | Dest. address | Source address | DSAP | SSAP | Control | SNAP | Data | FCS | ED | FS
FDDI MAC 802.2 FDDI

2.2 Fast Ethernet and Gigabit Ethernet

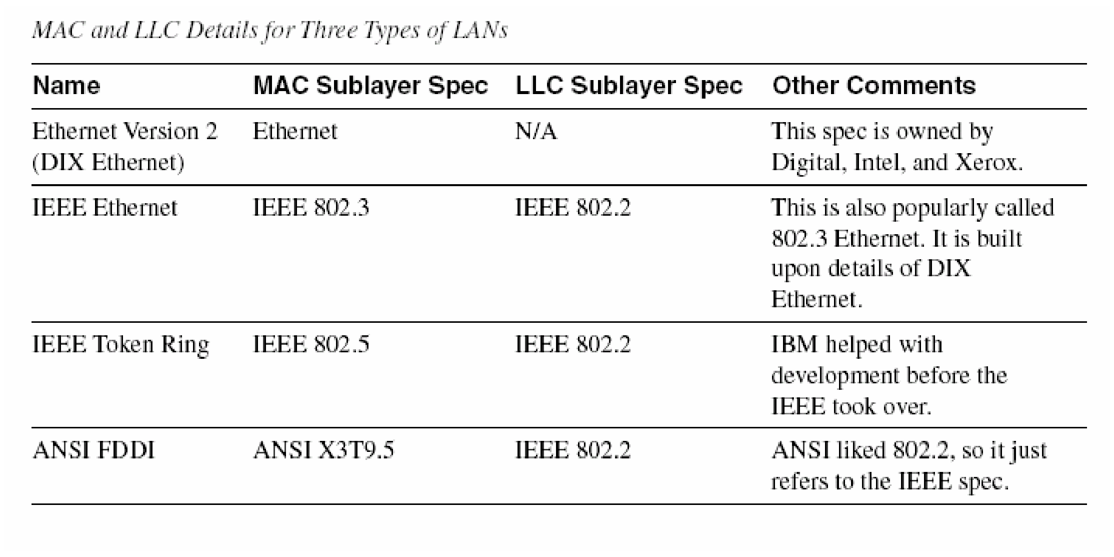
Fast Ethernet and Gigabit Ethernet provide faster Ethernet options. Both have gained widespread acceptance in networks today, with Fast Ethernet most likely being used to the desktop and Gigabit Ethernet being used between networking devices or to servers. Fast Ethernet retains many familiar features of 10-Mbps Ethernet variants. The age-old CSMA/CD logic still exists, but it can be disabled for full-duplex point-to-point topologies in which no collisions can occur. A variety of cabling options is allowed—unshielded and shielded copper cabling as well as multimode and single-mode fiber. Both Fast Ethernet shared hubs and switches can be deployed. However, because Fast Ethernet gained market acceptance around the same time that LAN switching became popular, most Fast Ethernet cards either are connected to a switch or are directly cabled to another device. The two key additional features of Fast Ethernet, as compared to 10-Mbps Ethernet, are higher bandwidth and autonegotiation. Fast Ethernet operates at 100 Mbps—enough said. The other key difference, autonegotiation, allows an Ethernet card or switch to operate at 10 or 100 Mbps. Also, support for half-duplex or full-duplex operation is negotiated. If the other device, such as a 10BaseT NIC, does not support autonegotiation, autonegotiation settles for half-duplex operation at 10 Mbps. The autonegotiation process has been known to fail. Cisco recommends that, for devices that seldom move, you should configure the LAN switch and the device to use the identical desired setting rather than depend on autonegotiation. Generally, for ports that are used for end-user devices, autonegotiation is enabled because these devices are moved frequently relative to servers or other network devices, such as routers. Gigabit Ethernet also retains many familiar features of slower Ethernet variants. CSMA/CD is still used and can be disabled for full-duplex support. Although gigabit hubs are allowed, it is more likely that Gigabit Ethernet switch ports will be the most popular use for Gigabit Ethernet, along with use as a trunk between high-throughput switches and routers.

Gigabit Ethernet is similar to its slower cousins in several ways. The most important similarity is that the same Ethernet headers and trailers are used, no matter whether it's 10 Mbps, 100 Mbps, or 1000 Mbps. If you understand how Ethernet works for 10 and 100 Mbps, then you know most of what you need to know about Gigabit Ethernet. Gigabit Ethernet differs from the slow specifications in how it encodes the signals onto the cable. Gigabit Ethernet is obviously faster, at 1000 Mbps, or 1 Gbps.

Both Fast Ethernet (FE) and Gigabit Ethernet (GE) relieve congestion in some fairly obvious ways. Collisions and wait time are decreased when compared to 10-Mbps Ethernet simply because it takes 90 percent (FE) or 99 percent (GE) less time to transmit the same frame on the faster LANs. Capacity is greatly increased as well: If all frames were 1250 bytes long, a theoretical maximum of 10,000 frames per second could be reached on Fast Ethernet, and a theoretical maximum of 100,000 frames per second could be reached on Gigabit Ethernet.

2.3 LAN Standards

The IEEE defines most of the standards for Ethernet and Token Ring, with ANSI defining standards for FDDI. These specifications match OSI Layer 2 and typically are divided into two parts: the Media Access Control (MAC) and Logical Link Control (LLC) sublayers. The MAC sublayer is very specific to that particular type of LAN, whereas the LLC sublayer performs functions that are relevant to several types of LANs. For instance, all LANs need some form of protocol type field; the LLC specification, rather than the MAC, defines the use of the DSAP field for that purpose. Table 4-4 lists the various protocol specifications.


MAC and LLC Details for Three Types of LANs

Name	MAC Sublayer Spec	LLC Sublayer Spec	Other Comments
Ethernet Version 2 (DIX Ethernet)	Ethernet	N/A	This spec is owned by Digital, Intel, and Xerox.
IEEE Ethernet	IEEE 802.3	IEEE 802.2	This is also popularly called 802.3 Ethernet. It is built upon details of DIX Ethernet.
IEEE Token Ring	IEEE 802.5	IEEE 802.2	IBM helped with development before the IEEE took over.
ANSI FDDI	ANSI X3T9.5	IEEE 802.2	ANSI liked 802.2, so it just refers to the IEEE spec.

Different Ethernet standards are given in the following table.

For more information on Fast Ethernet and information on Gigabit Ethernet, try the following
Web pages:

- www.host.ots.utexas.edu/ethernet/ethernet-home.html

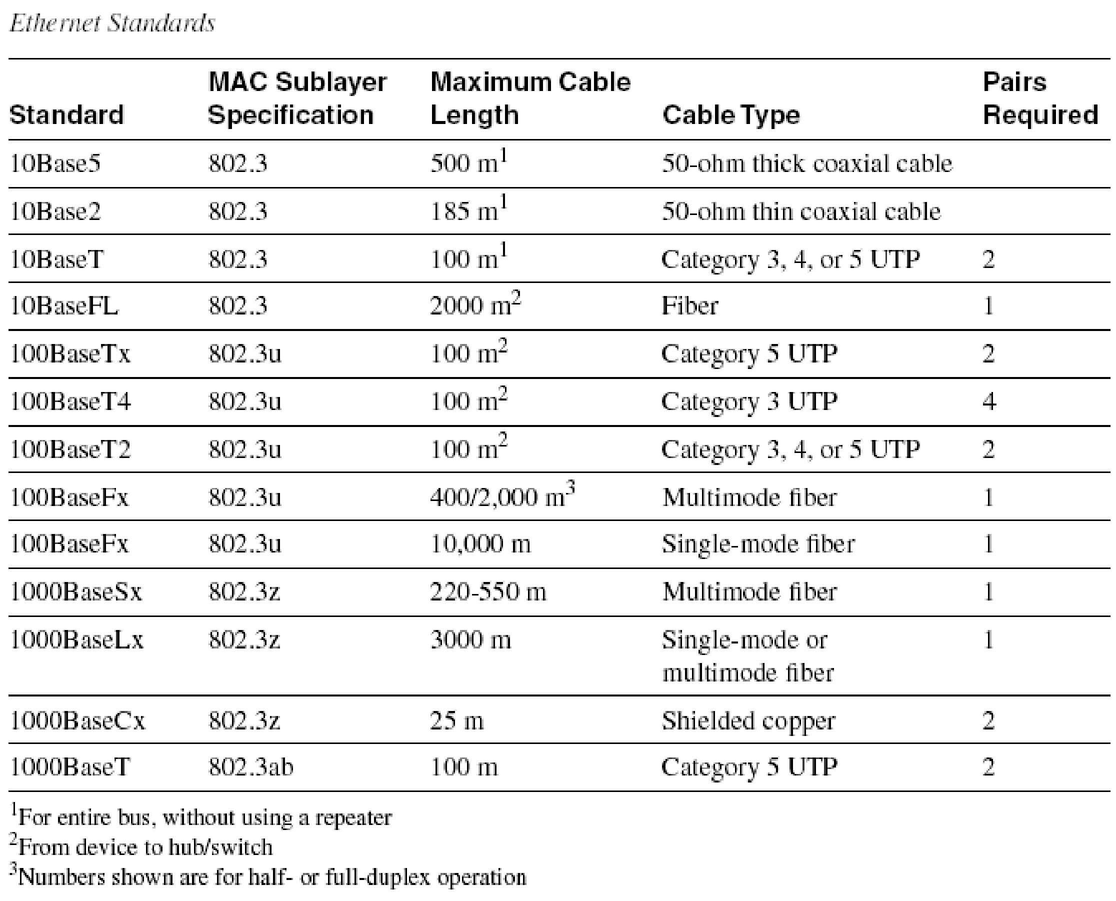
- www.ots.utexas.edu/ethernet/descript-100quickref.html

- www.iol.unh.edu/training

- www.cisco.com/warp/customer/cc/so/neso/lnso/lnmnso/feth_tc.htm

- www.cisco.com/warp/customer/cc/techno/media/lan/gig/tech/index.shtml

- www.10gea.org



Ethernet Standards

Standard	MAC Sublayer Specification	Maximum Cable Length	Cable Type	Pairs Required
10Base5	802.3	500 m[1]	50-ohm thick coaxial cable	
10Base2	802.3	185 m[1]	50-ohm thin coaxial cable	
10BaseT	802.3	100 m[1]	Category 3, 4, or 5 UTP	2
10BaseFL	802.3	2000 m[2]	Fiber	1
100BaseTx	802.3u	100 m[2]	Category 5 UTP	2
100BaseT4	802.3u	100 m[2]	Category 3 UTP	4
100BaseT2	802.3u	100 m[2]	Category 3, 4, or 5 UTP	2
100BaseFx	802.3u	400/2,000 m[3]	Multimode fiber	1
100BaseFx	802.3u	10,000 m	Single-mode fiber	1
1000BaseSx	802.3z	220-550 m	Multimode fiber	1
1000BaseLx	802.3z	3000 m	Single-mode or multimode fiber	1
1000BaseCx	802.3z	25 m	Shielded copper	2
1000BaseT	802.3ab	100 m	Category 5 UTP	2

[1]For entire bus, without using a repeater
[2]From device to hub/switch
[3]Numbers shown are for half- or full-duplex operation

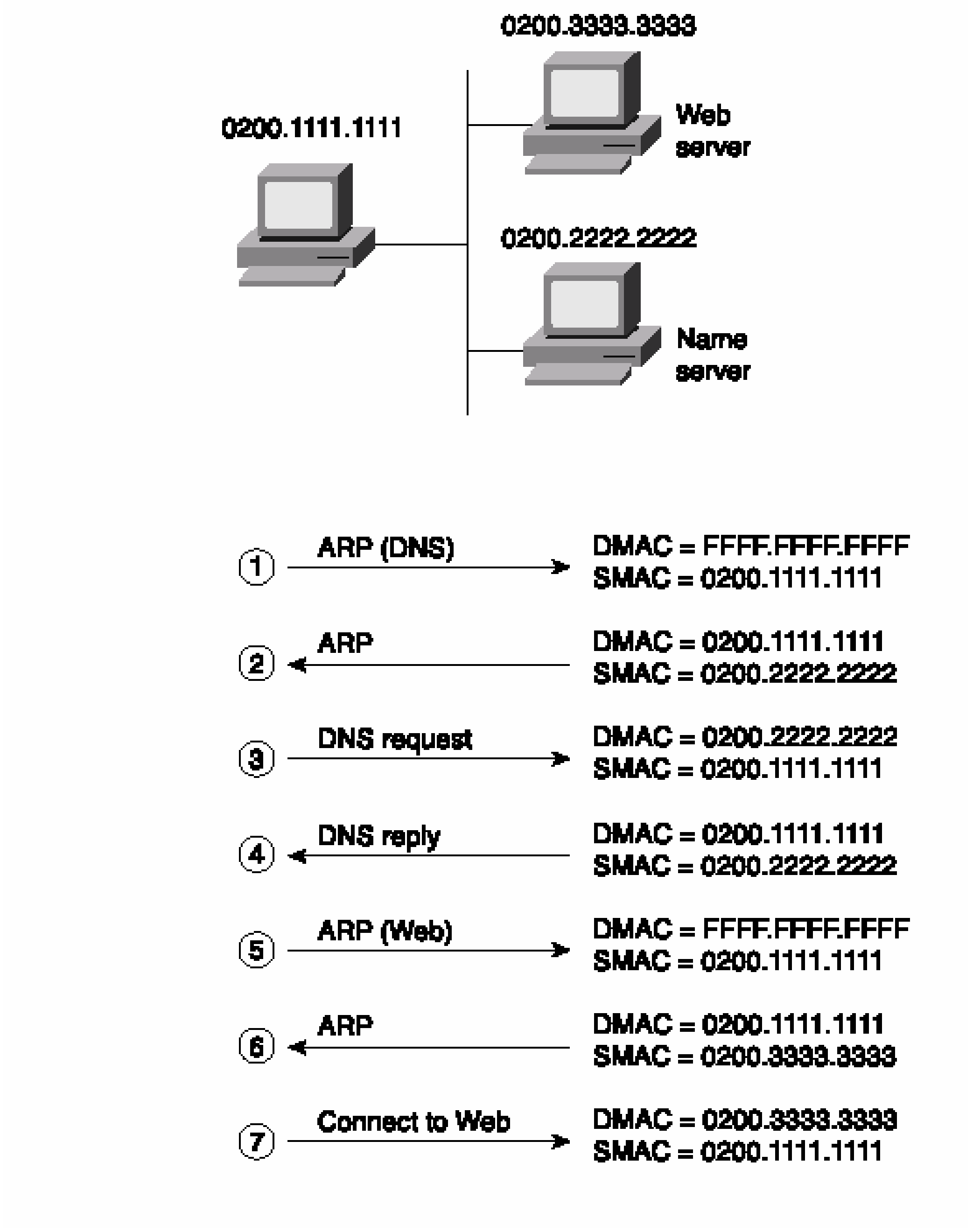
2.4 Transparent Bridges

Transparent bridges were created to alleviate congestion problems on a single ethernet segment and to extend allowed cabling distance. Transparent bridging is called "transparent" because the endpoint devices do not need to know that the bridge(s) exist(s). In other words, the computers attached to the LAN do not behave any
differently in the presence or absence of transparent bridges. Transparent bridges forward frames when necessary and do not forward when there is no need to do so, thus reducing overhead. To accomplish this, transparent bridges perform three actions:

• Learning MAC addresses by examining the source MAC address of each frame received by the bridge
• Deciding when to forward a frame or when to filter (not forward) a frame, based on the destination MAC address
• Creating a loop-free environment with other bridges by using the Spanning Tree Protocol

To fully understand transparent bridging logic, consider the following figure.A client first asks for DNS name resolution and then connects to a Web server. All three devices are on the same LAN segment. The LAN segment is drawn as if it is a 10Base2 or 10Base5 network, but it could be 10BaseT using a shared hub. Regardless, focus on the Ethernet addresses and the bridges actions in the figure.

Example Protocol Flows—Single Ethernet Segment



0200.3333.3333
Web server
0200.1111.1111
0200.2222.2222
Name server
1 ARP (DNS) DMAC = FFFF.FFFF.FFFF SMAC = 0200.1111.1111
2 ARP DMAC = 0200.1111.1111 SMAC = 0200.2222.2222
3 DNS request DMAC = 0200.2222.2222 SMAC = 0200.1111.1111
4 DNS reply DMAC = 0200.1111.1111 SMAC = 0200.2222.2222
5 ARP (Web) DMAC = FFFF.FFFF.FFFF SMAC = 0200.1111.1111
6 ARP DMAC = 0200.1111.1111 SMAC = 0200.3333.3333
7 Connect to Web DMAC = 0200.3333.3333 SMAC = 0200.1111.1111

In this example, the devices send messages just like, the user,the Web server, and the DNS were on the same segment. The destination and source MAC addresses are listed in the figure. The following list provides the process of finding out the MAC address of Web server.

1.The PC is pre-configured with the IP address of the DNS; it must use ARP to find the DNS's MAC address.

2.The DNS replies to the ARP request with its MAC address, 0200.2222.2222

3.The PC requests name resolution by the DNS for the Web server's name.

4.The DNS returns the IP address of the Web server to the PC.

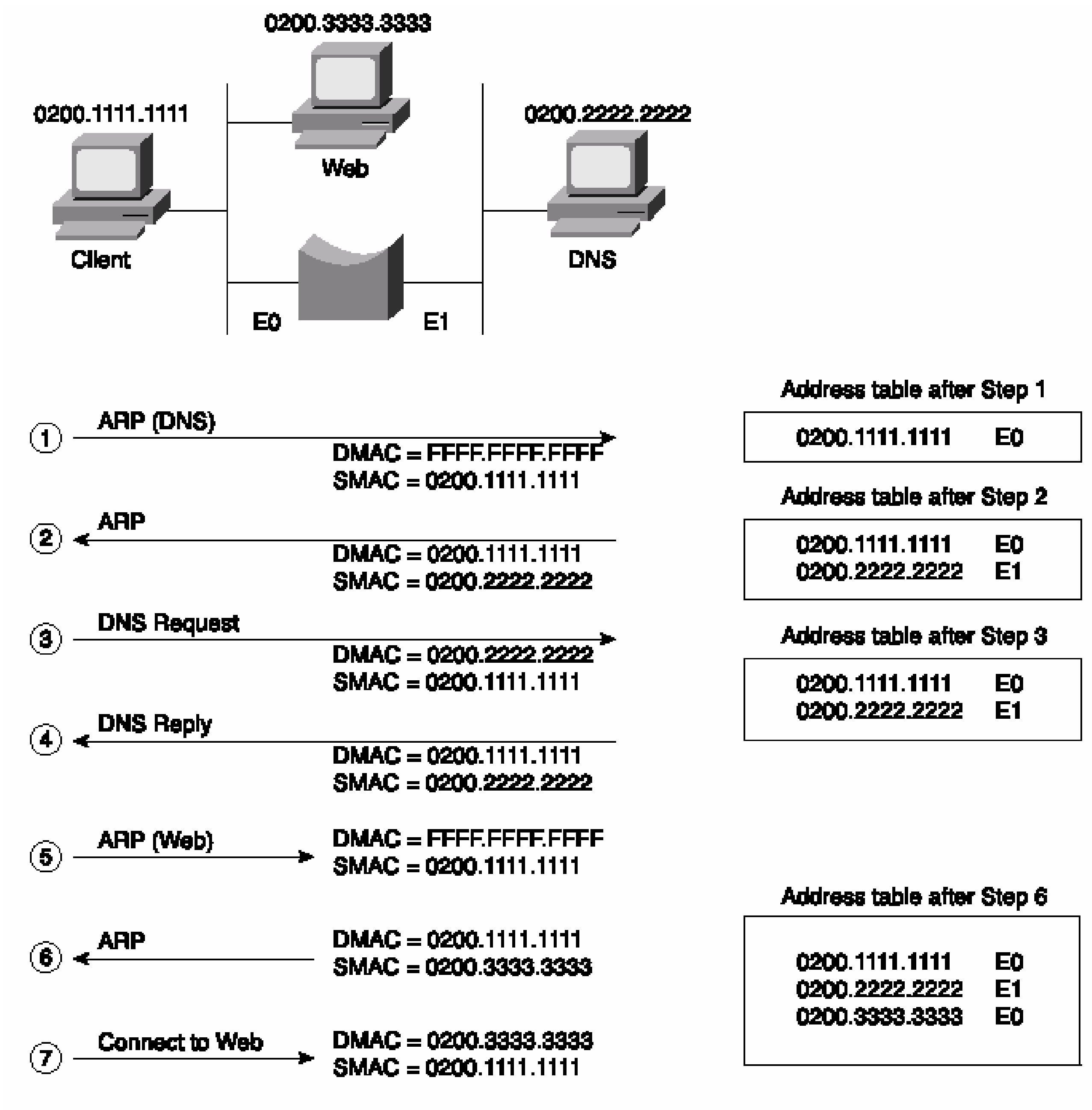
5.The PC does not know the Web server's MAC address, but it does know its IP address, so the PC sends an ARP broadcast to learn the MAC address of the Web server.

6. The Web server replies to the ARP, stating that its MAC address is 0200.3333.3333.

7. The PC now can send frames directly to the Web server.

In this case, the single LAN segment is a bus, or a 10BaseT hub. All frames are received by all devices. Ethernet is considered a *broadcast medium* because when one device sends, all the rest receive the electrical signal. Each device then must decide whether to process the frame.

A transparent bridge would not always forward a copy of each frame on each segment. Now consider the same protocol flow, but with the DNS on a separate segment and a transparent bridge separating the segments, as shown in figure below. The computers act no differently, sending the same frames and packets.



The following list outlines the logic used at each step in the process.

1. The first frame is a broadcast, so the bridge forwards the frame. The source MAC address, 0200.1111.1111, is added to the address table.

2.The ARP reply is a unicast destined to 0200.1111.1111, so the bridge knows to forward it out its E0 port, according to the address table. The source MAC of the frame, 0200.2222.2222, is added to the address table.

3. The DNS request is a unicast frame, and the bridge knows where the destination, 0200.2222.222, is. The bridge forwards the frame. The bridge checks the source address 0200.1111.1111 and notices that it is already in the table.

4. The DNS reply is a unicast, with a known destination 0200.1111.1111 and a known source (0200.2222.2222). The bridge forwards the frame.

5. The ARP broadcast is destined to MAC address FFFF.FFFF.FFFF, so it is forwarded by the bridge, in spite of the fact that the ARP broadcast will reach the Web server without the bridge forwarding the frame.

6. The ARP reply from the Web server is a unicast to 0200.1111.1111, and the bridge has that MAC in its address table. The bridge does not forward the frame because it came in its E0 interface and it is destined out that same interface. The source MAC address, 0200.3333.3333, is added to the address table.

7. The last frame is a unicast whose destination MAC address 0200.3333.3333 is in the address table, and the bridge should not forward it.

Networks using bridges have the following general characteristics:

• Broadcasts and multicast frames are forwarded by a bridge.

• Transparent bridges perform switching of frames using Layer 2 headers and Layer 2 logic and are Layer 3 protocol-independent. This means that installation is simple because no Layer 3 address group planning or address changes are necessary. For example, because the bridge retains a single broadcast domain, all devices on all segments attached to the bridge can look like a single subnet.

• Store-and-forward operation is typical in transparent bridging devices. Because an entire frame is received before being forwarded, additional latency is introduced (as compared to a single LAN segment).

• The transparent bridge must perform processing on the frame, which also can increase latency (as compared to a single LAN segment).

2.5 Switches

An Ethernet switch uses the same logic as a transparent bridge. However, switches perform more functions, have more features, and have more physical ports. Switches use hardware to learn addresses and to make forwarding and filtering decisions. Bridges use software running on general-purpose processors, so they tend to run much more slowly than switches. The reason behind this difference is simply the time frame in which each technology was developed. Switches came later and took advantage of newer hardware capabilities, such as application specific integrated circuits (ASICs). But if you look at the basic forward/filter logic, just as with a transparent bridge, the basic logic of a LAN switch is as follows:

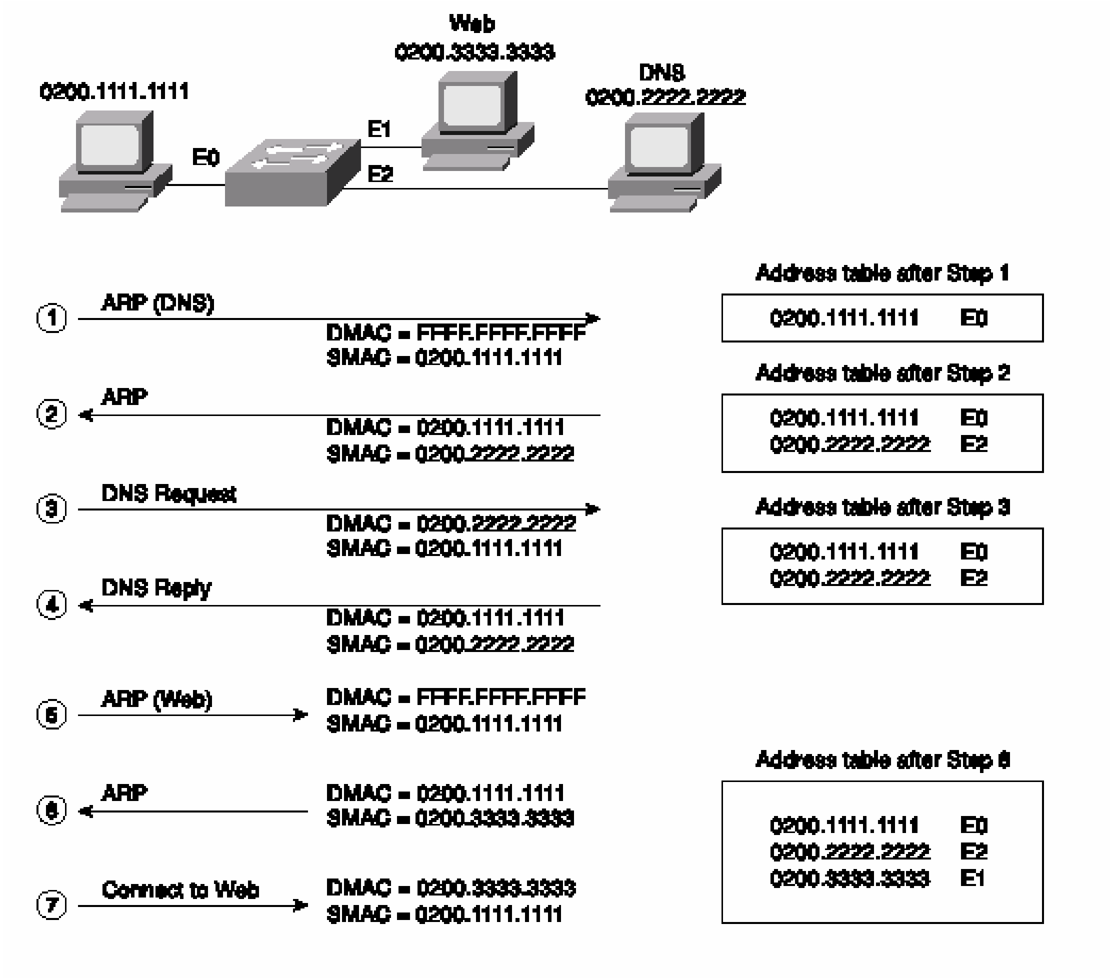
1. A frame is received.

2. If the destination is a broadcast or multicast, forward on all ports.

3. If the destination is a unicast and the address is not in the address table, forward on all ports.

4. If the destination is a unicast and the address is in the address table, and if the associated interface is not the interface in which the frame arrived, forward the frame.

The following figure explains process of finding out the MAC address of the Web server.



It appears that exactly the same thing happened as in bridged network explained before. The basic logic is similar, but there are some differences. The key physical difference is that each device is attached to a separate port on the switch, whereas, in bridge , the client and the Web server share the same port on the bridge.

The complete process is explained below.

1. The PC is preconfigured with the IP address of the DNS. The PC notices that the DNS IP address is in the same subnet as its own IP address; therefore, the PC sends an ARP broadcast hoping to learn the DNS's MAC address. *This frame is forwarded out all other ports by the switch*, just like the transparent bridge did.

2.The DNS replies to the ARP request with its MAC address, 0200.2222.2222. The destination MAC is 0200.1111.1111, which is already in the address table, *so the switch forwards this frame only out port E0*.

3. The PC requests name resolution for the Web server by sending a packet with the destination IP address of the DNS. The destination MAC is 0200.2222.2222, which is already in the address table, *so the switch forwards this frame only out port E2*.

4. The DNS returns the IP address of the Web server to the PC in the DNS reply. The destination MAC is 0200.1111.1111, which is already in the address table, *so the switch forwards this frame only out port E0*.

5. The PC does not know the Web server's MAC address, so it sends an ARP broadcast to learn the MAC address. Because it is a MAC broadcast, *the switch forwards the frame on all ports*.

6.The Web server replies to the ARP, stating that its MAC address is 0200.3333.3333. This frame has a destination MAC of 0200.1111.1111, which is already in the address table, *so the switch forwards this frame only out port E0*.

7. The PC now can connect to the Web server. This frame has a destination MAC of 0200.3333.3333, which is already in the address table, *so the switch forwards this frame only out port E1*. The switch behaves exactly like a transparent bridge would if the bridge used three ports instead of two. The switch learned the MAC addresses by examining the source MAC addresses and made forwarding decisions based on the contents of the address table.

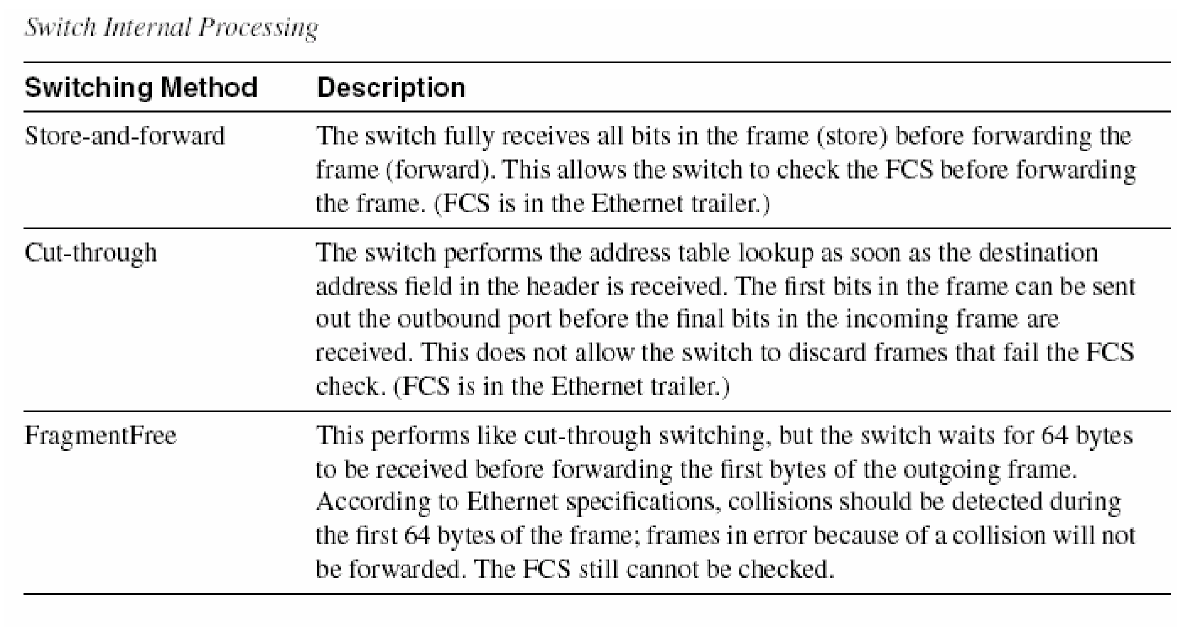
The switch reduces the possibility of collisions. For example, imagine a fourth device, on port E3. If that new device sent a frame to the DNS's address of 0200.2222.2222 at the same time that the client sent a frame to the Web server (0200.3333.3333), the switch's basic logic would prevent a collision. For different reasons, the switch prevents a broadcast sent by this new device from colliding with the frame that the client sent to the server. The switch buffers one of the frames until the first frame has been sent and then forwards the second frame.

2.6 Internal Switching Paths

The internal processing on a switch can decrease latency for frames. Transparent bridges use store-and-forward processing, which means that the entire frame is received before the first bit of the frame is forwarded. Switches can use store-and-forward processing as well as *cutthrough* processing logic. With cut-through processing, the first bits of the frame are sent out the outbound port before the last bit of the incoming frame is received instead of waiting for the entire frame to be received. In other words, as soon as the incoming switch port receives enough of the frame to see the destination MAC address, the frame is transmitted out the appropriate outgoing port to the destination device.

Cut-through processing can help, and it can hurt. Because the frame check sequence (FCS) is in the Ethernet trailer, a cut-through forwarded frame might have bit errors that the switch will not notice before sending most of the frame. And, of course, if the outbound port is busy, the switch stores the frame until the output port is available, gaining no advantage of store-and forward switching.

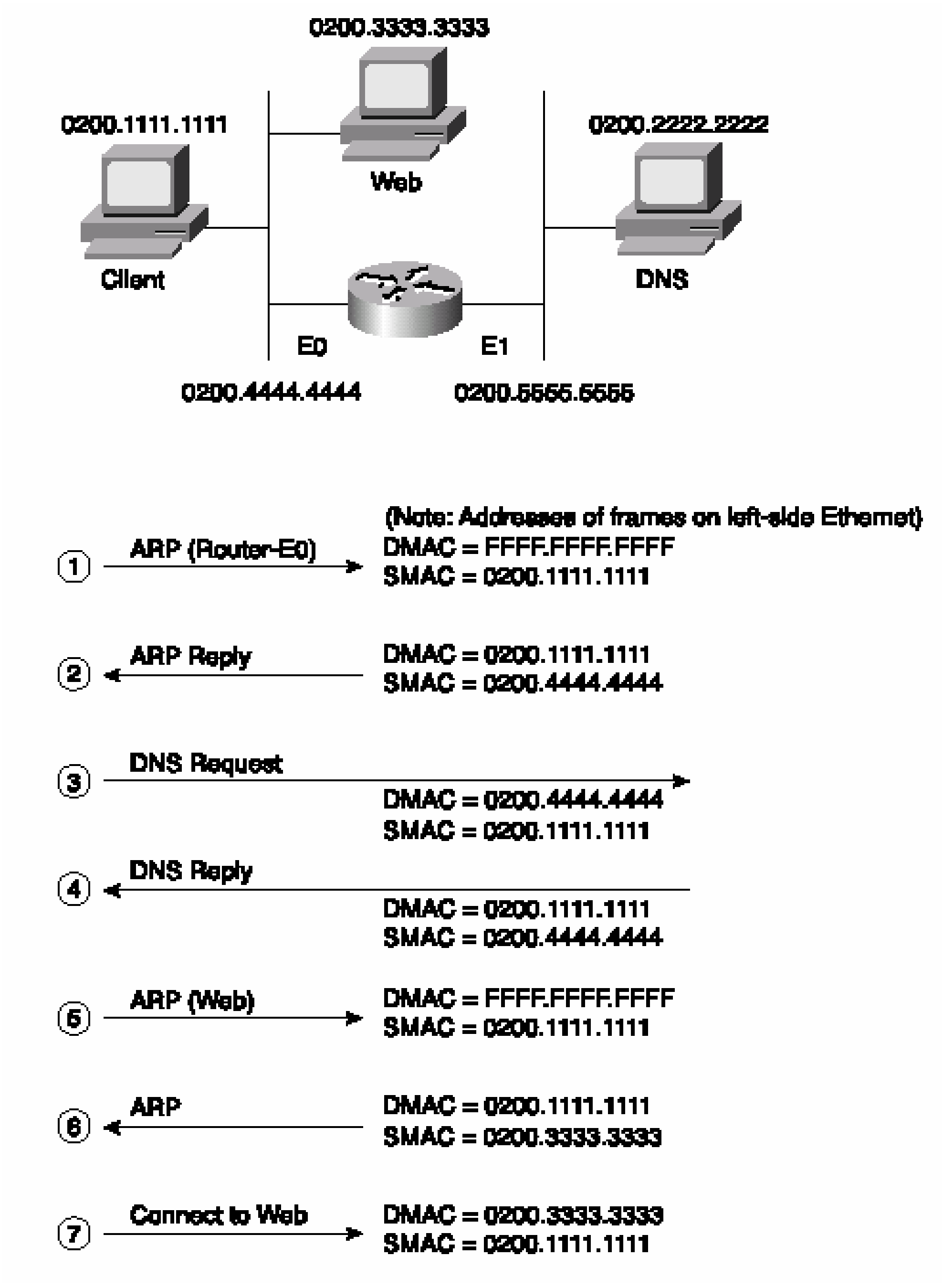
The internal processing algorithms used by switches vary among models and vendors; regardless, the internal processing can be categorized as one of the methods listed in following table.



Switch Internal Processing

Switching Method	Description
Store-and-forward	The switch fully receives all bits in the frame (store) before forwarding the frame (forward). This allows the switch to check the FCS before forwarding the frame. (FCS is in the Ethernet trailer.)
Cut-through	The switch performs the address table lookup as soon as the destination address field in the header is received. The first bits in the frame can be sent out the outbound port before the final bits in the incoming frame are received. This does not allow the switch to discard frames that fail the FCS check. (FCS is in the Ethernet trailer.)
FragmentFree	This performs like cut-through switching, but the switch waits for 64 bytes to be received before forwarding the first bytes of the outgoing frame. According to Ethernet specifications, collisions should be detected during the first 64 bytes of the frame; frames in error because of a collision will not be forwarded. The FCS still cannot be checked.

Finding out the Web server MAC address including a router in the network is explained below.



The process is as follows.

1.The PC is preconfigured with the IP address of the DNS. The PC notices that the IP address of the DNS is on a different subnet, so the PC wants to forward the packet to its default router. However, the PC does not know its default router’s MAC address yet, so *it must use ARP to find that router’s MAC address. The ARP broadcast is not forwarded by the router.*

2.The router replies to the ARP request with its MAC address, 0200.4444.4444.

3.The PC requests name resolution for the Web server by sending a packet with the destination IP address of the DNS. The destination MAC address in the frame sent by the PC is the router’s E0 MAC address. The router receives the frame, extracts the packet, and forwards it.

4.The DNS returns the IP address of the Web server to the PC in the DNS reply.

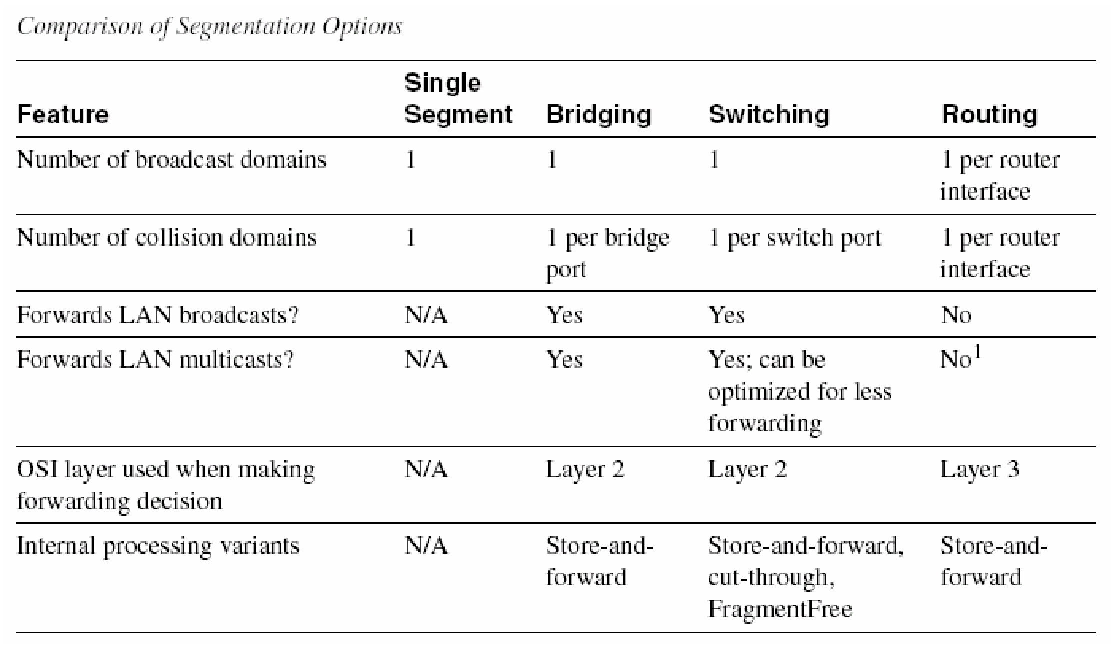
5.The PC does not know the Web server’s MAC address, so it sends an ARP broadcast to learn the MAC address. The router has no need to forward the ARP broadcast.

6.The Web server replies to the ARP, stating that its MAC address is 0200.3333.3333.

7.The PC can now connect to the Web server.

The router does not forward ARP broadcasts. In fact, the logic in Step 1 begins with an ARP looking for the MAC address of the client’s default router—namely, the router’s E0 MAC address. This broadcast was not forwarded by the router, a fact that causes a router to be called a *broadcast firewall*. Comparing this to a transparent bridge or a LAN switch, this difference in broadcast treatment is the biggest advantage of routers.

The following table lists several features relating to segmenting LANs with bridges, switches, and routers. Essentially, this chart summarizes features that might differ among the three devices.

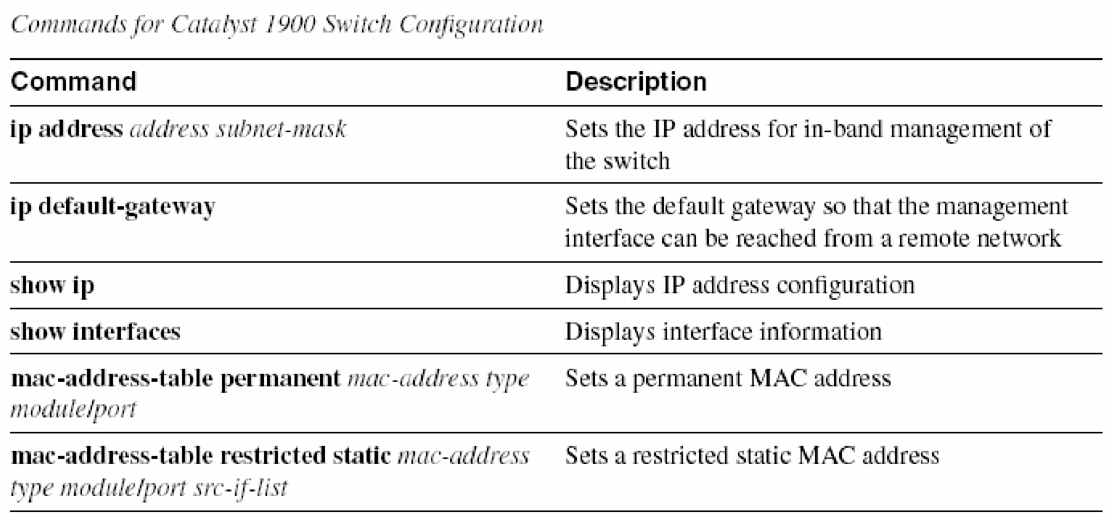


Comparison of Segmentation Options

Feature	Single Segment	Bridging	Switching	Routing
Number of broadcast domains	1	1	1	1 per router interface
Number of collision domains	1	1 per bridge port	1 per switch port	1 per router interface
Forwards LAN broadcasts?	N/A	Yes	Yes	No
Forwards LAN multicasts?	N/A	Yes	Yes; can be optimized for less forwarding	No[1]
OSI layer used when making forwarding decision	N/A	Layer 2	Layer 2	Layer 3
Internal processing variants	N/A	Store-and-forward	Store-and-forward, cut-through, FragmentFree	Store-and-forward

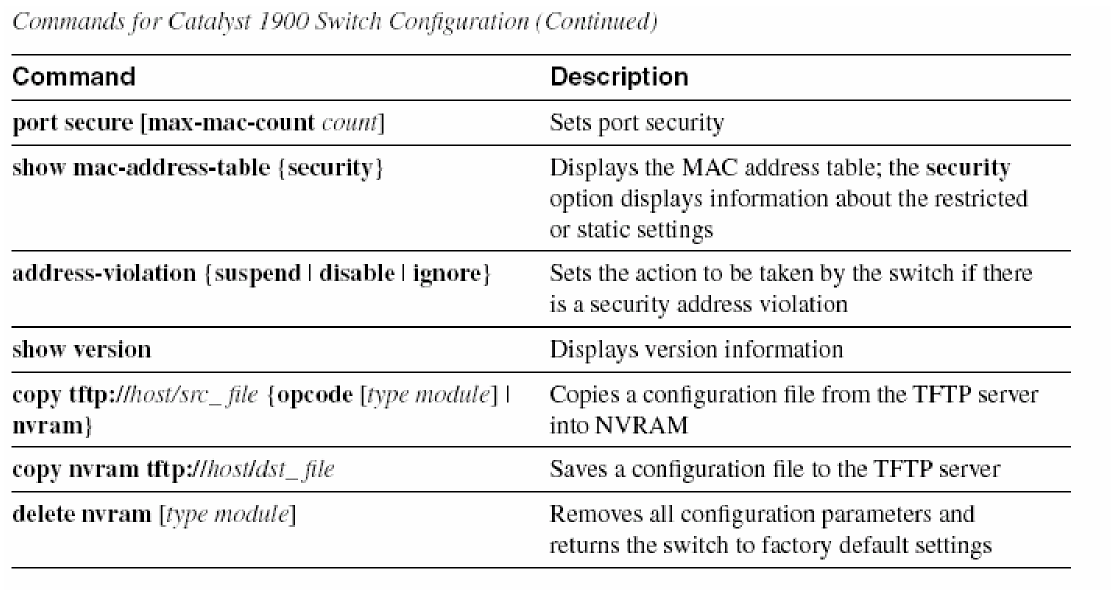
2.7 Configuration of Switches

There is a set of commands to configure switches. Here we consider the CISCI Catalyst 1900 switch configuration. Some of the commands are shown in the following table.



Commands for Catalyst 1900 Switch Configuration

Command	Description
ip address *address subnet-mask*	Sets the IP address for in-band management of the switch
ip default-gateway	Sets the default gateway so that the management interface can be reached from a remote network
show ip	Displays IP address configuration
show interfaces	Displays interface information
mac-address-table permanent *mac-address type module/port*	Sets a permanent MAC address
mac-address-table restricted static *mac-address type module/port src-if-list*	Sets a restricted static MAC address



Commands for Catalyst 1900 Switch Configuration (Continued)

Command	Description
port secure [max-mac-count *count*]	Sets port security
show **mac-address-table {security}**	Displays the MAC address table; the **security** option displays information about the restricted or static settings
address-violation {suspend | disable | ignore}	Sets the action to be taken by the switch if there is a security address violation
show version	Displays version information
copy tftp://*host/src_file* {**opcode** [*type module*] | **nvram**}	Copies a configuration file from the TFTP server into NVRAM
copy nvram tftp://*host/dst_file*	Saves a configuration file to the TFTP server
delete nvram [*type module*]	Removes all configuration parameters and returns the switch to factory default settings

2.7.1 Default settings of 1900 switch

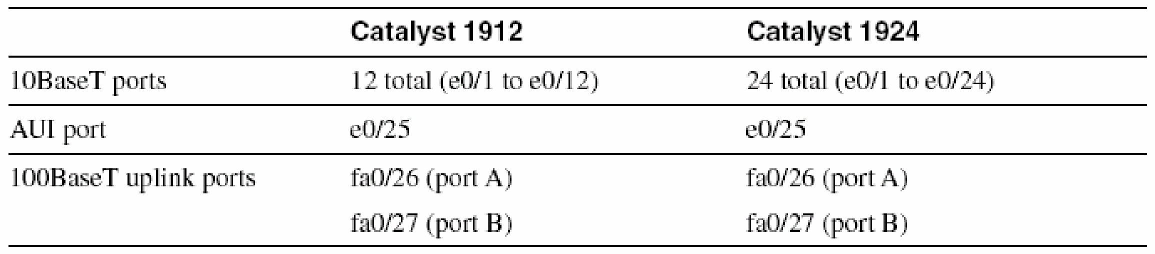
The default values vary depending on the features of the switch. The following list provides some of the default settings for the Catalyst 1900 switch. (Not all the defaults are shown in this example.)

• IP address: 0.0.0.0
• CDP: Enabled
• Switching mode: Fragment Free
• 100BaseT port: Autonegotiate duplex mode
• 10BaseT port: Half duplex
• Spanning Tree: Enabled
• Console password: None

2.7.2 Numbering Ports (Interfaces)

The terms *interface* and *port* both are used to describe the physical connectors on the switch hardware. For instance, the **show running-config** command uses the term *interface*; the **show spantree** command uses the term *port*. The numbering of the interfaces is relatively straightforward; the interface numbering convention for the 1912 and 1924 switches is shown in the table.

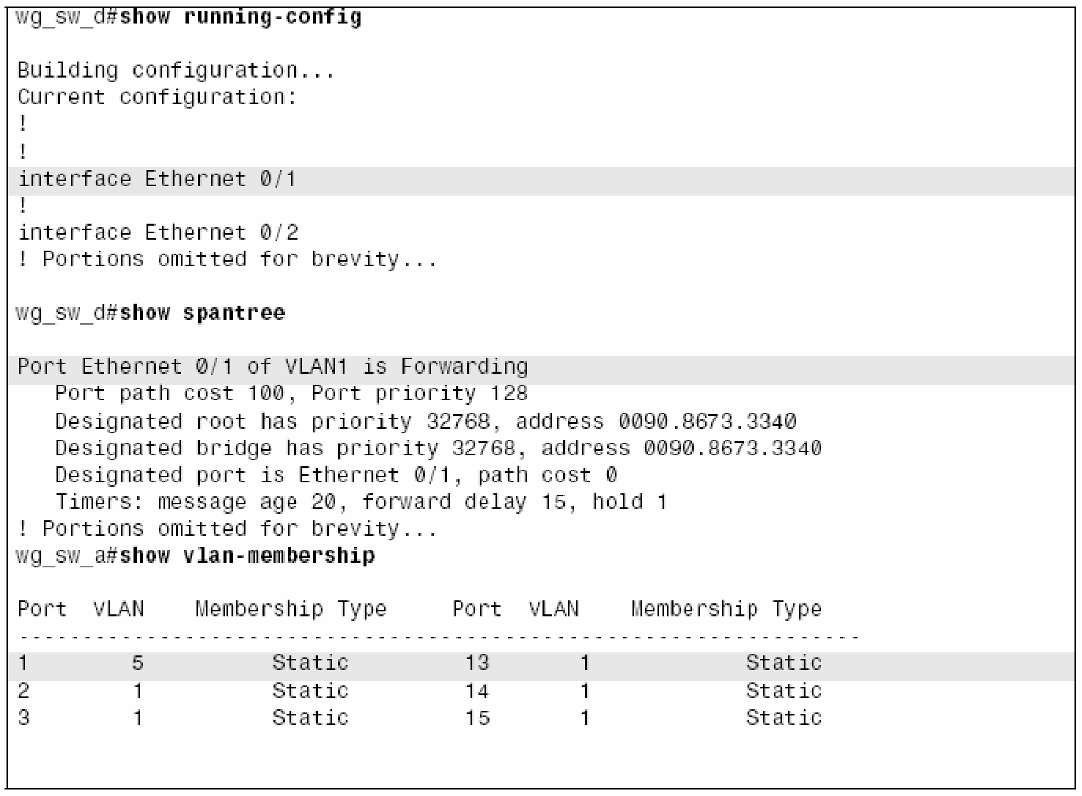
Catalyst 1912 and 1924 Interface/Port Numbering



	Catalyst 1912	Catalyst 1924
10BaseT ports	12 total (e0/1 to e0/12)	24 total (e0/1 to e0/24)
AUI port	e0/25	e0/25
100BaseT uplink ports	fa0/26 (port A) fa0/27 (port B)	fa0/26 (port A) fa0/27 (port B)

The following examples shows three exec commands and highlights the use of the terms *interface* and *port*.

show run *Output Refers to Port e0/1 as Interface Ethernet 0/1*



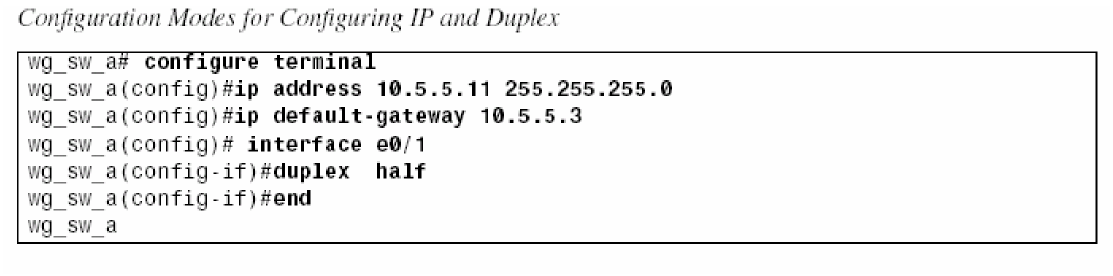
```
wg_sw_d#show running-config

Building configuration...
Current configuration:
!
!
interface Ethernet 0/1
!
interface Ethernet 0/2
! Portions omitted for brevity...

wg_sw_d#show spantree

Port Ethernet 0/1 of VLAN1 is Forwarding
   Port path cost 100, Port priority 128
   Designated root has priority 32768, address 0090.8673.3340
   Designated bridge has priority 32768, address 0090.8673.3340
   Designated port is Ethernet 0/1, path cost 0
   Timers: message age 20, forward delay 15, hold 1
! Portions omitted for brevity...
wg_sw_a#show vlan-membership

Port  VLAN    Membership Type     Port  VLAN    Membership Type
-----------------------------------------------------------------
1        5          Static         13       1            Static
2        1          Static         14       1            Static
3        1          Static         15       1            Static
```

2.7.3 Basic IP and Port Duplex Configuration

Two features commonly configured during switch installation are assign IP address for management of switch and the setting of duplex on key switch ports. Switches support IP, but in a different way than a router. The switch acts more like a normal IP host, with a single address/mask for the switch and a default router. Each ort/interface does not need an IP address because the switch is not performing Layer 3 routing. In fact, if there were no need to manage the switch, IP would not be needed on the switch at all.

The second feature typically configured at installation time is to pre-configure some ports to always use half- or full-duplex operation rather than allow negotiation. At times, auto-negotiation can produce unpredictable results. For example, if a device attached to the switch does not support auto-negotiation, the Catalyst switch sets the corresponding switch port to half-duplex mode by default. If the attached device is configured for full-duplex operation, a duplex mismatch occurs. To avoid this situation, manually set the duplex parameters of the switch to match the attached device. Similar to the router IOS, the Catalyst 1900 switch has various configuration modes.

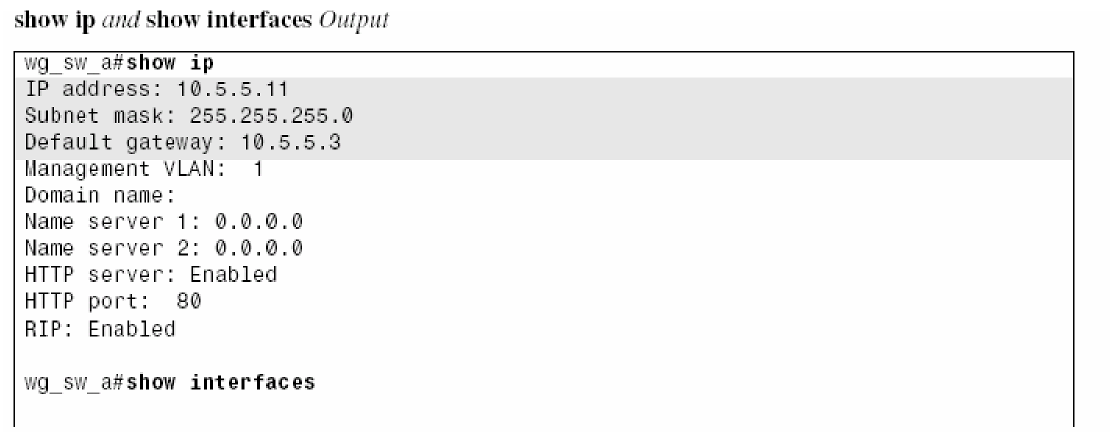
Following example shows the initial configuration of IP and duplex, with the actual prompts showing the familiar exec and configuration modes.


Configuration Modes for Configuring IP and Duplex

```
wg_sw_a# configure terminal
wg_sw_a(config)#ip address 10.5.5.11 255.255.255.0
wg_sw_a(config)#ip default-gateway 10.5.5.3
wg_sw_a(config)# interface e0/1
wg_sw_a(config-if)#duplex  half
wg_sw_a(config-if)#end
wg_sw_a
```

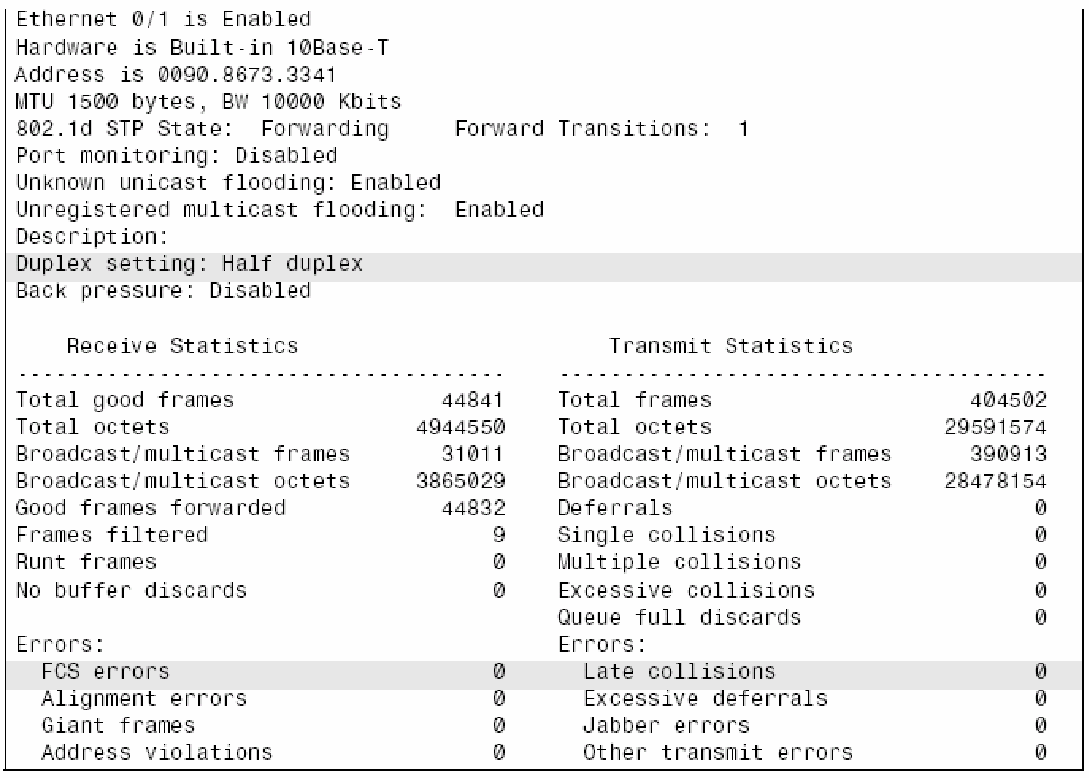
In the example, the duplex could have been set to one of the following modes:
• **auto**—Sets autonegotiation of duplex mode. This is the default option for 100-Mbps TX ports.
• **full**—Sets full-duplex mode.
• **full-flow-control**—Sets full-duplex mode with flow control.
• **half**—Sets half-duplex mode. This is the default option for 10-Mbps TX ports.

To verify the IP configuration and duplex settings on a given interface, use the **show ip** and **show interface** commands, as shown in the example.


show ip *and* **show interfaces** *Output*

```
wg_sw_a#show ip
IP address: 10.5.5.11
Subnet mask: 255.255.255.0
Default gateway: 10.5.5.3
Management VLAN:  1
Domain name:
Name server 1: 0.0.0.0
Name server 2: 0.0.0.0
HTTP server: Enabled
HTTP port:  80
RIP: Enabled

wg_sw_a#show interfaces
```

show ip *and* **show interfaces** *Output (Continued)*



```
Ethernet 0/1 is Enabled
Hardware is Built-in 10Base-T
Address is 0090.8673.3341
MTU 1500 bytes, BW 10000 Kbits
802.1d STP State:  Forwarding     Forward Transitions:  1
Port monitoring: Disabled
Unknown unicast flooding: Enabled
Unregistered multicast flooding:  Enabled
Description:
Duplex setting: Half duplex
Back pressure: Disabled

    Receive Statistics                       Transmit Statistics
-------------------------------------    -------------------------------------
Total good frames               44841    Total frames                   404502
Total octets                  4944550    Total octets                 29591574
Broadcast/multicast frames      31011    Broadcast/multicast frames     390913
Broadcast/multicast octets    3865029    Broadcast/multicast octets   28478154
Good frames forwarded           44832    Deferrals                           0
Frames filtered                     9    Single collisions                   0
Runt frames                         0    Multiple collisions                 0
No buffer discards                  0    Excessive collisions                0
                                         Queue full discards                 0
Errors:                                  Errors:
  FCS errors                        0      Late collisions                   0
  Alignment errors                  0      Excessive deferrals               0
  Giant frames                      0      Jabber errors                     0
  Address violations                0      Other transmit errors             0
```

No IP address is in the **show interface** output because the IP address is associated with the entire switch, not just a single interface. The spanning-tree state of the interface is shown, as is the duplex setting. If duplex was mismatched with the device on the other end, the late collisions counter most likely would increment rapidly.

2.7.4 Viewing and Configuring Entries in the MAC Address Table

The switching/bridging table concept discussed earlier in this chapter is called the *MAC address table* on the 1900 family of switches. The MAC address table contains dynamic entries, which are learned when the switch receives frames and examines the source MAC address. Two other variations of entries in the MAC address table are important to switch configuration and are outlined along with dynamic entries in the following list:

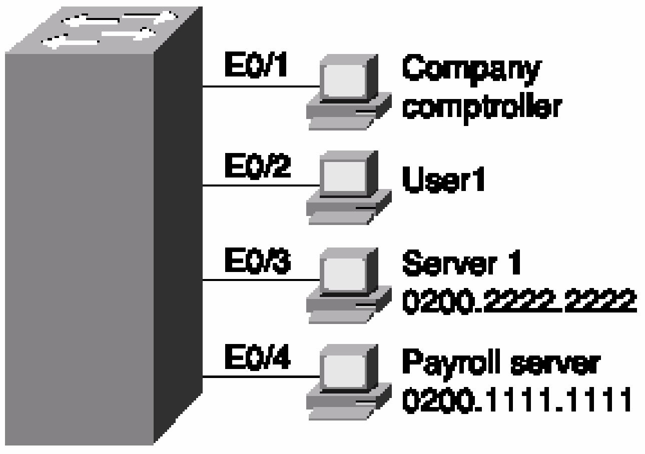
• **Dynamic addresses—**MAC addresses are added to the MAC address table through normal bridge/switch processing. In other words, when a frame is received, the source MAC of the frame is associated with the incoming port/interface. These entries in the table time out with disuse (default 300 seconds on a 1900 series switch) and are cleared whenever the entire table is cleared.

• **Permanent MAC addresses—**Through configuration, a MAC address is associated with a port, just as it would have been associated as a dynamic address. However, permanent entries in the table never time out.

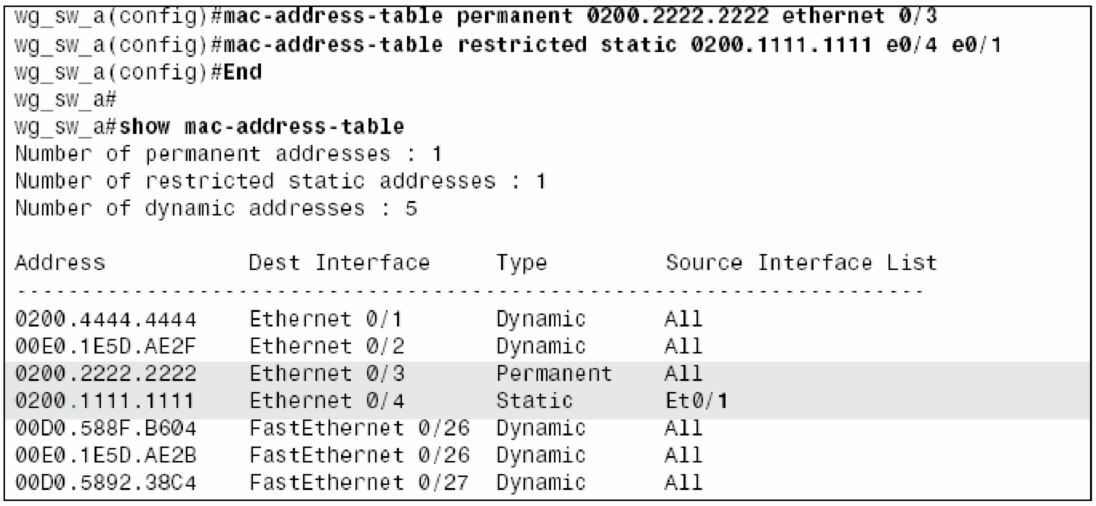
• **Restricted-static entries**—Through configuration, a MAC address is configured to be associated only with a particular port, with an additional restriction: Frames destined to that MAC address must have entered through a particular set of incoming ports.

Following figure provides a simple example to show the use of permanent and restricted-static addresses. A popular server (Server 1) is on port E0/3, and there is never a case when its MAC address should not be in the table. The payroll server is also on this switch, and only the company comptroller is allowed access. The configuration and resulting MAC address table are shown in the example , which follows the figure.

MAC Address Table Manipulation—Sample Network



The MAC Address Table, with Dynamic, Permanent, and Restricted Static Entries



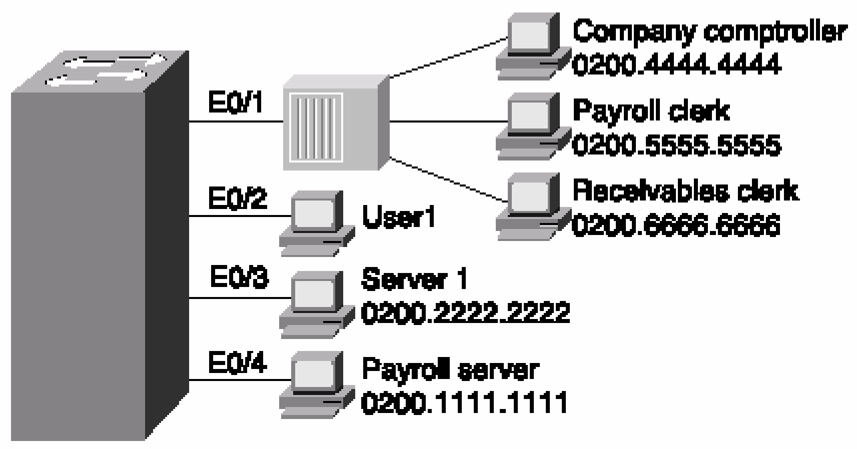
```
wg_sw_a(config)#mac-address-table permanent 0200.2222.2222 ethernet 0/3
wg_sw_a(config)#mac-address-table restricted static 0200.1111.1111 e0/4 e0/1
wg_sw_a(config)#End
wg_sw_a#
wg_sw_a#show mac-address-table
Number of permanent addresses : 1
Number of restricted static addresses : 1
Number of dynamic addresses : 5

Address          Dest Interface     Type        Source Interface List
----------------------------------------------------------------------
0200.4444.4444   Ethernet 0/1       Dynamic     All
00E0.1E5D.AE2F   Ethernet 0/2       Dynamic     All
0200.2222.2222   Ethernet 0/3       Permanent   All
0200.1111.1111   Ethernet 0/4       Static      Et0/1
00D0.588F.B604   FastEthernet 0/26  Dynamic     All
00E0.1E5D.AE2B   FastEthernet 0/26  Dynamic     All
00D0.5892.38C4   FastEthernet 0/27  Dynamic     All
```

In the example, Server 1 is always in the address table as the permanent entry. The payroll server is always in the table off port 0/4, and only devices on port 0/1 are allowed to send frames to it.

Another feature affecting the MAC address table is called *port security*. Port security is a feature that, when enabled, limits the number of MAC addresses associated with a port in the MAC address table. In other words, there is a preset limit to the number of sources that can forward frames into that switch port.

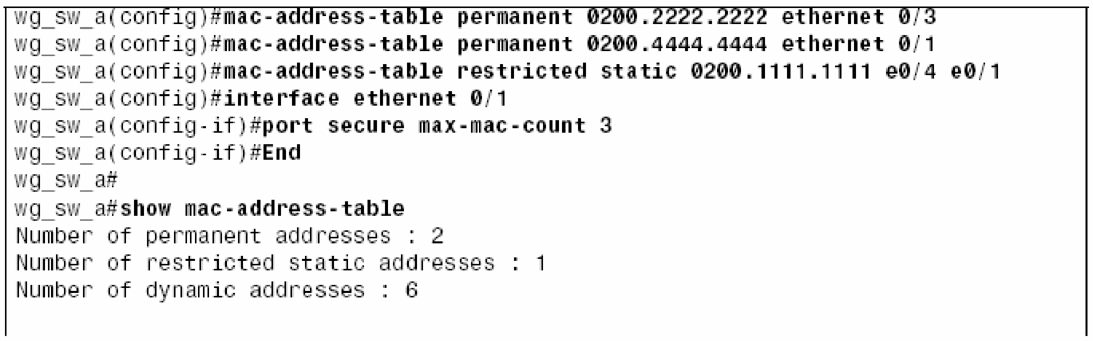
Consider the following figure , which shows a similar configuration to previous figure ,except that the finance department has increased to three employees. These three employees are on the same shared hub, which then is cabled to switch port 0/1.

Sample Network with Port Security



Port security can be used to restrict port 0/1 so that only three MAC addresses can source frames that enter port 0/1; this is because only the finance department is expected to use the shared hub. Any permanent or restricted-static MAC addresses count against this total of three. Following example shows a sample configuration, with **show** commands.

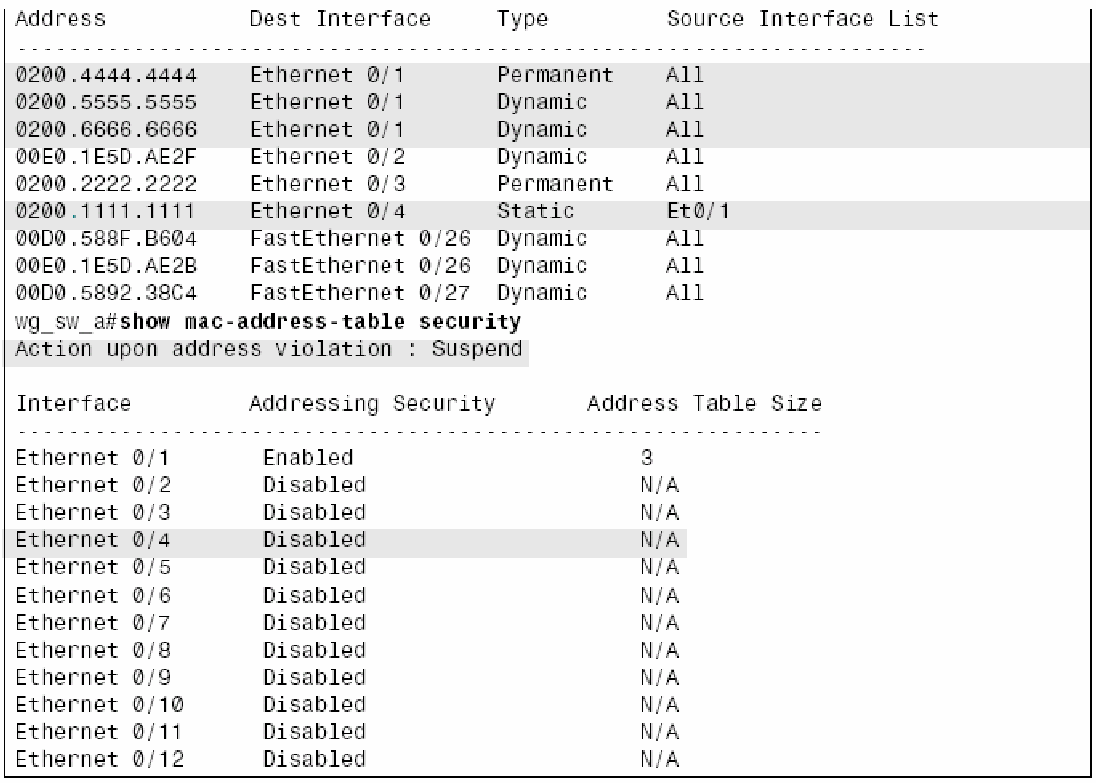
Port Security Example



```
wg_sw_a(config)#mac-address-table permanent 0200.2222.2222 ethernet 0/3
wg_sw_a(config)#mac-address-table permanent 0200.4444.4444 ethernet 0/1
wg_sw_a(config)#mac-address-table restricted static 0200.1111.1111 e0/4 e0/1
wg_sw_a(config)#interface ethernet 0/1
wg_sw_a(config-if)#port secure max-mac-count 3
wg_sw_a(config-if)#End
wg_sw_a#
wg_sw_a#show mac-address-table
Number of permanent addresses : 2
Number of restricted static addresses : 1
Number of dynamic addresses : 6
```

continues

Port Security Example (Continued)



```
Address           Dest Interface     Type         Source Interface List
---------------------------------------------------------------------
0200.4444.4444    Ethernet 0/1       Permanent    All
0200.5555.5555    Ethernet 0/1       Dynamic      All
0200.6666.6666    Ethernet 0/1       Dynamic      All
00E0.1E5D.AE2F    Ethernet 0/2       Dynamic      All
0200.2222.2222    Ethernet 0/3       Permanent    All
0200.1111.1111    Ethernet 0/4       Static       Et0/1
00D0.588F.B604    FastEthernet 0/26  Dynamic      All
00E0.1E5D.AE2B    FastEthernet 0/26  Dynamic      All
00D0.5892.38C4    FastEthernet 0/27  Dynamic      All
wg_sw_a#show mac-address-table security
Action upon address violation : Suspend

Interface         Addressing Security       Address Table Size
------------------------------------------------------------
Ethernet 0/1       Enabled                      3
Ethernet 0/2       Disabled                     N/A
Ethernet 0/3       Disabled                     N/A
Ethernet 0/4       Disabled                     N/A
Ethernet 0/5       Disabled                     N/A
Ethernet 0/6       Disabled                     N/A
Ethernet 0/7       Disabled                     N/A
Ethernet 0/8       Disabled                     N/A
Ethernet 0/9       Disabled                     N/A
Ethernet 0/10      Disabled                     N/A
Ethernet 0/11      Disabled                     N/A
Ethernet 0/12      Disabled                     N/A
```

In this example, the permanently defined MAC address of 0200.4444.444, the comptroller’s MAC address, is always associated with port e0/1. Notice that the two new employees’ MAC addresses are also in the MAC address table. The **port secure max-mac-count 3** command means that a total of three addresses can be learned on this port.

What should the switch do when a fourth MAC address sources a frame that enters E0/1? An address violation occurs when a secured port receives a frame from a new source address that, if added to the MAC table, would cause the switch to exceed its address table size limit for that port. When a port security address violation occurs, the options for action to be taken on a port include suspending, ignoring, or disabling the port. When a port is suspended, it is re-enabled when a frame containing a valid address is received. When a port is disabled, it must be manually re-enabled. If the action is ignored, the switch ignores the security violation and keeps the port enabled.

Use the **address-violation** global configuration command to specify the action for a port address violation. The syntax for this command is as follows:

address-violation {suspend **|** *disable* **|** *ignore}*

Use the **no address-violation** command to set the switch to its default value, which is **suspend**.

3 - Spanning Tree, VLAN and Trunking

3.1 Spanning Tree Protocol

In the absence of STP, frames would loop for an indefinite period of time in networks with physically redundant links. STP blocks some ports so that only one active path exists between any pair of LAN segments (collision domains). The result of STP is both good and bad: Frames do not loop infinitely, which makes the LAN usable, which is good. However, the network does not actively take advantage of some of the redundant links because they are blocked to prevent frames from looping. Some users' traffic travels a seemingly longer path through the network because a shorter physical path is blocked, which is bad. If frames looped indefinitely, the LAN would be unusable, anyway. So, STP has some minor unfortunate side effects compared to the major benefit of letting us build redundant LANs. To avoid loops, all bridging devices, including switches, use STP. STP causes each interface on a bridging device to settle into a *blocking* state or a *forwarding* state. Blocking means that the interface cannot forward or receive data frames, but it can send and receive bridge protocol data units (BPDUs); forwarding means that the interface can both send and receive data frames as well as BPDUs. By having a correct subset of the interfaces blocked, a single currently active logical path will exist between each pair of LANs. STP behaves identically for a transparent bridge and a switch. So, in this section, the terms *bridge*, *switch* , and *bridging device* refer to a device that can run STP. A simple example makes the need for STP more obvious. Remember, frames destined for unknown MAC addresses, or broadcasts, will be forwarded out all interfaces. Figure shows a single frame, sent by Larry, looping forever because the network has redundancy but STP is not enabled.

Larry sends a single unicast frame to Bob's MAC address, but Bob is powered off and none of the switches has learned Bob's MAC address yet. Frames addressed to Bob's MAC address will loop forever—or at least until time is no more! Because the switches never learn Bob's MAC address, they keep forwarding the frame out all ports, and copies of the frame go around and around. Ethernet does not define a mechanism to mark a frame so that it can be thrown away by a bridge if it the frame loops. (IP does have that feature, using the Time To Live field.) The frame would loop until one of the links that create the redundancy fails. Similarly, bridges and switches forward broadcasts on all interfaces, so if any of the PCs sent a broadcast, the broadcast would loop indefinitely as well.

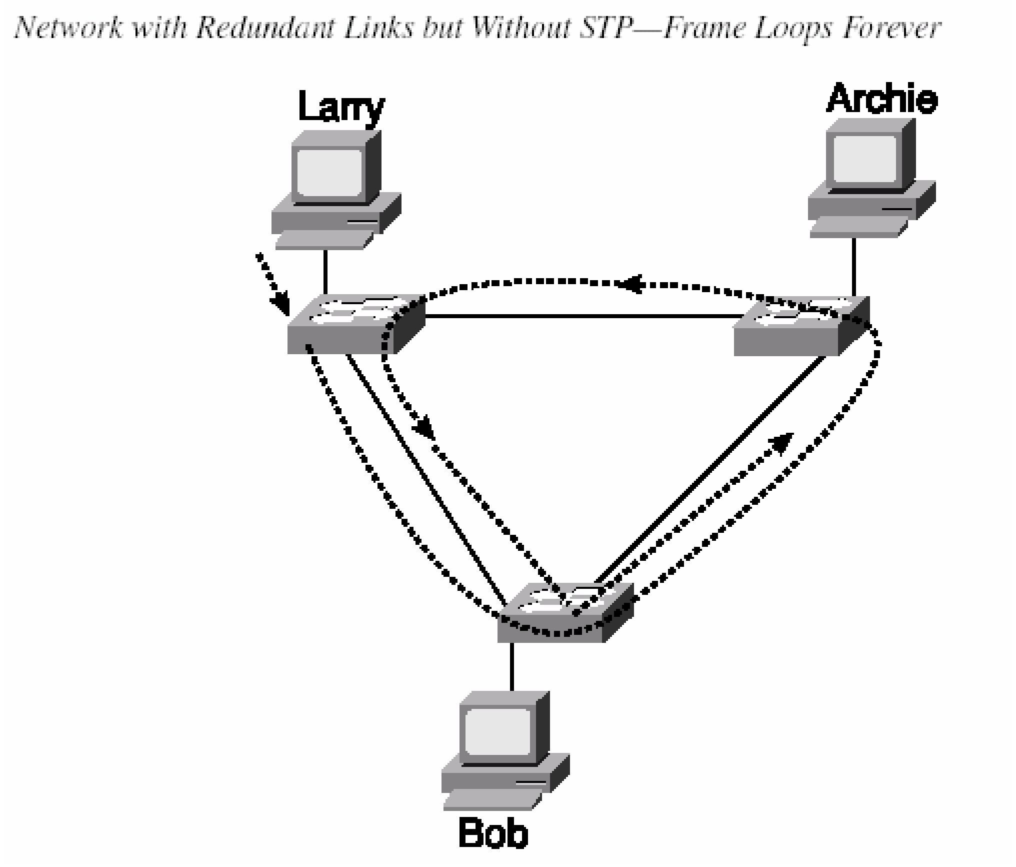

Network with Redundant Links but Without STP—Frame Loops Forever
Larry
Archie
Bob

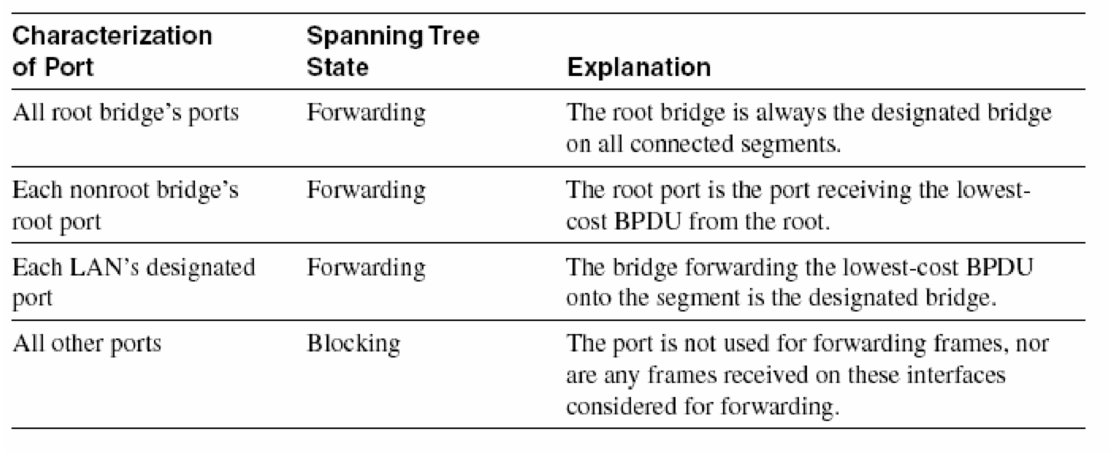
fig 3.1

3.1.1 3.1.1How Spanning Tree works?

The STP algorithm creates a spanning tree of interfaces that either forward or block. STP actually places interfaces into forwarding state; by default, if an interface has no reason to be in forwarding state, it is placed into a blocking state. In other words, STP simply picks which interfaces should forward. So, how does STP choose whether to put an interface into forwarding state? Well, it uses three criteria:
•STP elects a root bridge. All interfaces on the root bridge are in forwarding state.
•Each nonroot bridge considers one of its ports to have the least administrative cost between itself and the root bridge. STP places this least-root-cost interface, called that bridge's *root port* , into the forwarding state.
•Many bridges can attach to the same segment. These bridges advertise BPDUs declaring their administrative cost to the root bridge. The bridge with the lowest such cost of all bridges on that segment is called the *designated bridge*. The interface on the *designated bridge* that sends this lowest-cost BPDU is the *designated port* on that LAN segment, and that port is placed in a forwarding state.

All other interfaces are placed in a blocking state. The following table summarizes the reasons why spanning tree places a port in forwarding or blocking state.

Spanning Tree: Reasons for Forwarding or Blocking



Characterization of Port	Spanning Tree State	Explanation
All root bridge's ports	Forwarding	The root bridge is always the designated bridge on all connected segments.
Each nonroot bridge's root port	Forwarding	The root port is the port receiving the lowest-cost BPDU from the root.
Each LAN's designated port	Forwarding	The bridge forwarding the lowest-cost BPDU onto the segment is the designated bridge.
All other ports	Blocking	The port is not used for forwarding frames, nor are any frames received on these interfaces considered for forwarding.

3.1.2 Electing the Root, Discovering Root Ports and Designated Ports

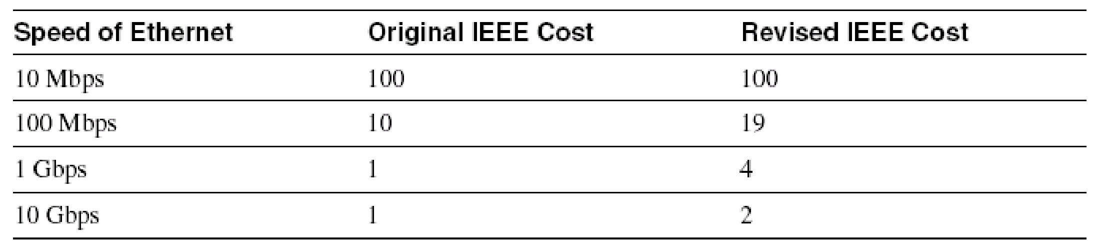
Each bridge begins by claiming to be the root bridge by sending STP messages. STP defines these messages used to exchange information with other bridges, which are called *bridge protocol data units (BPDUs)* . Each bridge begins by sending a BPDU stating the following:

• **The root bridge's bridge ID**
—At the beginning of the process, each bridge claims to be root, so this value is the same as the bridge ID of this bridge.

•**An administratively set priority**
—This is the priority of the root bridge. At the beginning of the process, each bridge claims to be root, so this value is the priority of this bridge.

•**The cost to reach the root from this bridge**
—At the beginning of the process, each bridge claims to be root, so the value is set to 0, which is this bridge's cost to reach itself.

•**The bridge ID of the sender of this BPDU**
—This value is always the bridge ID of the sender of the BPDU, regardless of whether this bridge is the root. The bridges elect a root bridge to begin the process. The root bridge will be the bridge with the lowest priority value. If a tie occurs based on priority, the root bridge with the lowest ID will be the root. The bridge IDs should be unique; bridges and switches use one of their own MAC addresses as their bridge ID, so the bridge IDs are unique because MAC addresses are supposed to be unique.
The message used to identify the root and its priority, ID, and cost is called a hello BPDU. STP elects a root bridge, in a manner not unlike a political election. The process of choosing the root begins with all bridges claiming to be the root by sending hello BPDUs with their bridge IDs and priorities. If a bridge hears of a better candidate, it stops advertising itself as root and starts forwarding the hello sent by the better political candidate, much like a candidate does when leaving a political race: the lesser candidate throws support behind another candidate. Eventually, someone wins and everyone supports the elected switch, which is where the political race analogy falls apart.


Default Port Costs According to IEEE

Speed of Ethernet	Original IEEE Cost	Revised IEEE Cost
10 Mbps	100	100
100 Mbps	10	19
1 Gbps	1	4
10 Gbps	1	2

3.1.3 Reacting to Changes in the Network

After the STP topology has been set, it does not change unless the network topology changes　If network topology changes, the root bridge sends a new BPDU, called a hello, every 2 seconds, by default. Each bridge forwards the hello, changing the cost to reflect that bridge’s added cost to reach the root. When a bridge ceases to receive the hellos, it reacts and starts the process of changing the spanning tree, under the assumption that the reason that the hello BPDUs quit arriving is that something has failed in the network. The hello BPDU defines the timers used by all the bridges when choosing when to react:

•**Hello time** **—**The time that the root waits before sending the periodic hello BPDUs, which then are forwarded by successive switches/bridges. The default is 2 seconds.

• **MaxAge—**The time that any bridge should wait before deciding that the topology has changed. Usually it is a multiple of the hello time; the default is 20 seconds.

• **Forward delay**—Delay that affects the time involved when an interface changes from a blocking state to a forwarding state; this timer is covered in more depth shortly.

When the network is up and no problems are occurring, the process works like this:

1. The root sends a hello BPDU, with a cost of 0, out all its interfaces.
2. The neighboring bridges forward hello BPDUs out their nonroot, designated ports, referring to the root but with their cost added.
3. Step 2 is repeated by each bridge in the network as it receives these hello BPDUs.
4. The root repeats Step 1 every hello time.
5. If a bridge does not get a Hello BPDU in hello time, it continues as normal. If a bridge fails to receive a Hello BPDU in MaxAge time, the bridge reacts.

3.1.4 Spanning-Tree Protocol Summary

Spanning trees accomplish the goals of allowing physical redundancy, but with only one currently active path through a bridged network. Spanning tree uses the following features to accomplish the goal:

• All bridge interfaces eventually stabilize at either a forwarding state or a blocking state. The forwarding interfaces are considered to be a part of the spanning tree.

• One of the bridges is elected as root. The election process includes all bridges claiming to be root, until one is considered best by all. All root bridge interfaces are in forwarding state.

• Each bridge receives hello BPDUs from the root, either directly or forwarded by some other bridge. Each bridge can receive more than one such message on its interfaces, but the port in which the least-cost BPDU is received is called the root port of a bridge, and that port is placed in forwarding state.

• For each LAN segment, one bridge sends the forwarded BPDU with the lowest cost. That bridge is the designated bridge for that segment. That bridge's interface on that segment is placed in forwarding state.

• All other interfaces are placed in blocking state.

• The root sends BPDUs every hello time seconds. The other bridges expect to receive copies of these BPDUs so that they know that nothing has changed. Hello time is defined in the BPDU itself, so all bridges use the same value.

• If a bridge does not receive a BPDU for MaxAge time, it begins the process of causing the spanning tree to change. The reaction can vary from topology to topology. (MaxAge is defined in the BPDU itself, so all bridges use the same value.)

• One or more bridges decide to change interfaces from blocking to forwarding, or vice versa, depending on the change in the network. If moving from blocking to forwarding, the interim listening state is entered first. After forward delay time (another timer defined in the root BPDU), the state is changed to learning. After another forward delay time, the interface is placed in forwarding state.

• The Spanning-Tree Protocol includes these delays to help ensure that no temporary loops occur.

3.2 Virtual LANs

Cisco switches with 10/100 ports autonegotiate by default and place all ports in VLAN 1 by default, which effectively means that the switch behaves as if the concept of VLANs did not exist at all. STP is enabled by default, so even if you cabled the switches in a redundant configuration, it still would have worked. The only thing not spelled out here was whether to use a straight-through or crossover Ethernet cable.
When creating a LAN design, there is a good chance that you will use VLANs. Using VLANs make technological sense and economic sense. A *virtual LAN (VLAN)* is a broadcast domain created by one or more switches. A VLAN is created by configuration in the switch. So, instead of all ports on a switch forming a single broadcast domain, the switch separates them into many, based on configuration.
Two example networks provide enough context to see the great benefit of VLANs. Before VLANs existed, if a design specified three separate broadcast domains, three switches would be used—one for each broadcast domain. And because a switch can forward frames only between devices in the same VLAN, each switch also would be connected to a router so that the router could forward packets among the three broadcast domains. So, if there was a need for three different broadcast domains, three switches would be purchased and three router Ethernet ports would be needed as well.

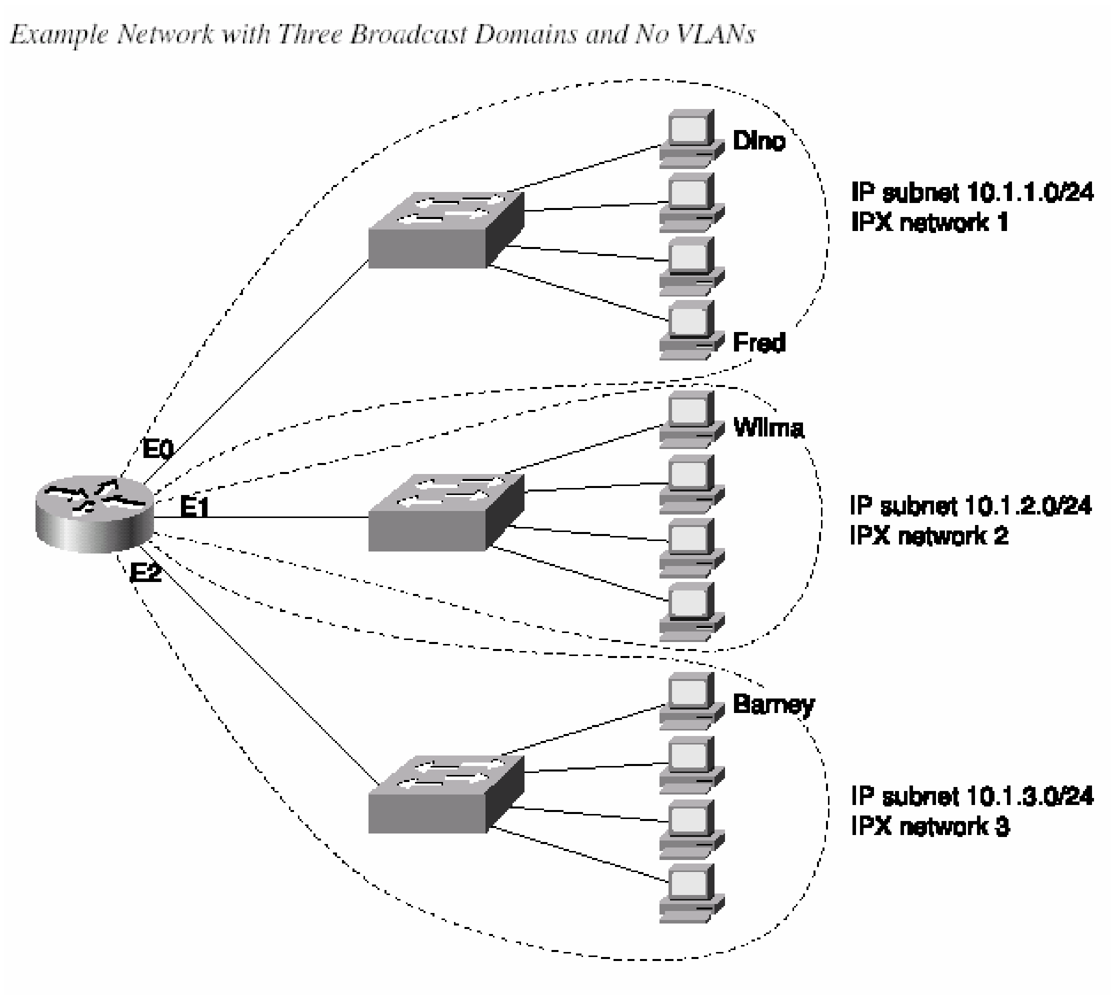


fig 3.2

Three separate hardware switches were required in this network because three broadcast domains were needed. If a switch could logically separate ports into different broadcast domains, only one switch would be required. So, a switch supporting VLANs would be configured to place each port into one of three different groups, with each group being a VLAN. For instance, in following figure , one switch is used; the switch is configured so that each port is considered to be in one of three different broadcast domains or VLANs. In both cases, separate broadcast domains imply separate Layer 3 groupings; a router is needed for forwarding traffic among the different Layer 3 groups.

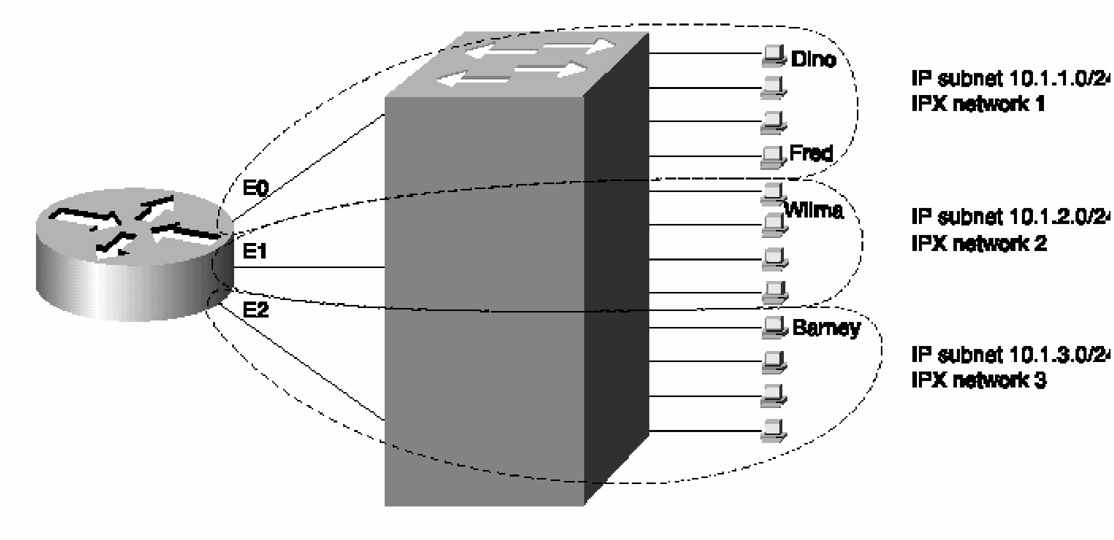


fig 3.3

In both figures, separate broadcast domains imply separate Layer 3 groupings; a router is needed for forwarding traffic among the different Layer 3 groups. The single switch in above figure behaves identically to each of the three switches in previous figure, but for each VLAN. For instance, the top switch in figure forwards broadcasts that it receives from Dino out all ports; similarly, the switch in forwards Dino’s broadcasts out all ports in VLAN 1. The single switch *never* forwards a frame that enters an port that is in one VLAN out another port that is in a different VLAN. Although the switch cannot forward between VLANs, a router can. The switch in above figure forwards frames to the router interfaces only if the frame is a broadcast or is destined for one of the MAC addresses of the router. For example Fred will send an IP packet to Barney, and Fred will encapsulate the IP packet inside an Ethernet frame. Fred encodes a destination MAC address of the router’s E0 MAC address because Fred’s default router should be the router’s E0 interface. The switch forwards this frame to the router, which routes it out E2 to Barney. Of course, the switch can forward frames directly between devices in the same VLAN. For instance, when Fred sends frames to Dino, the destination MAC address of the frame is Dino’s MAC address, and there is no need for the switch to get the router involved. Broadcasts sent by Fred do not go to the other VLANs because each VLAN is a separate broadcast domain, and you know the rule: Switches forward frames only out other ports in the same VLAN.

VLANs allow easy moves, additions, and changes. For example, if Barney moved to a different office that was cabled to a different port on the switch, he still can be configured to be in VLAN 3. No Layer 3 address changes are necessary, which means that no changes need to be made on Barney. Switches create a separate address table for each VLAN. For instance, if a frame is received on a port in VLAN 2, the VLAN 2 address table will be searched. When a frame is received, the source address is checked against the address table so that it can be added if the address is currently unknown. Also, the destination address is checked so that a forwarding decision can be made. In other words, the switch learns addresses and makes forwarding decisions the same way as always, except that it uses a separate address table per VLAN. Implementing VLANs with multiple switches adds more complexity that is not necessarily obvious. Consider following figure , which uses two switches connected with a Fast Ethernet. Two VLANs are configured.

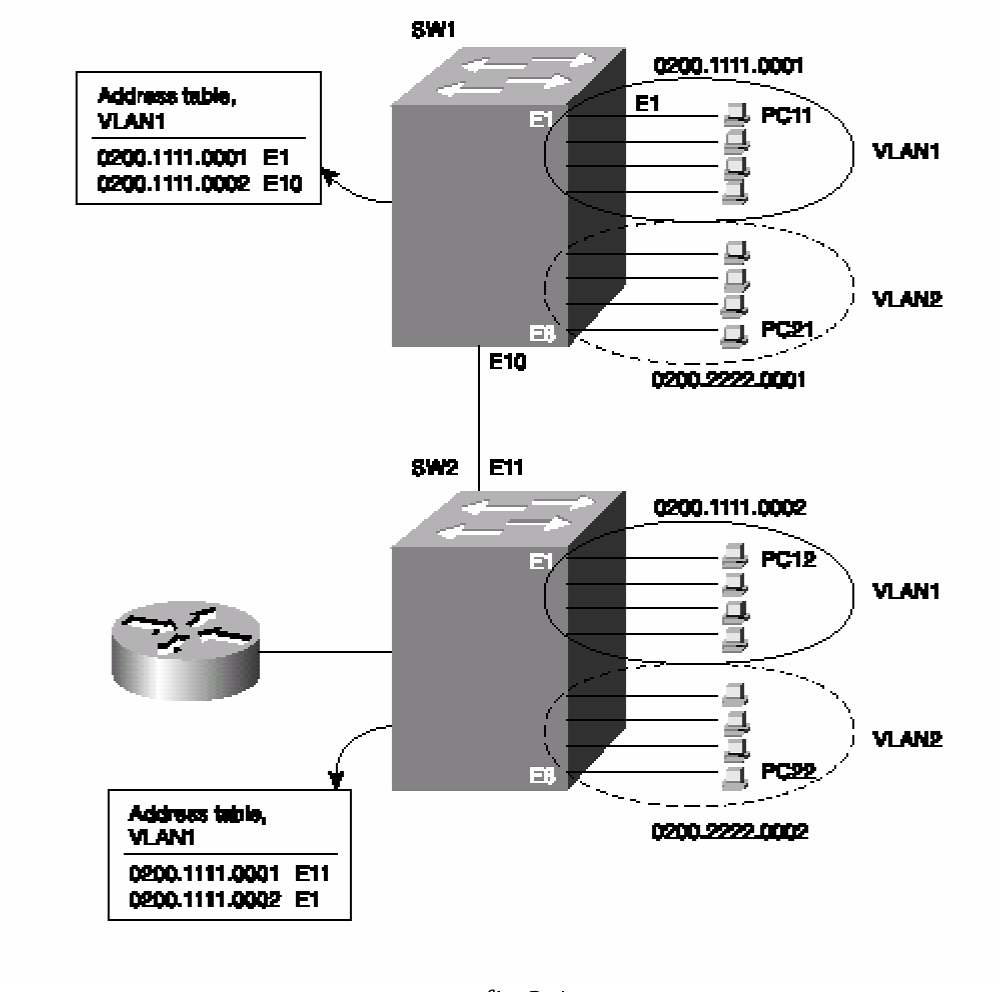


fig 3.4

To see the problem and then the solution, consider a frame sent by PC11, addressed to PC12. The address table for VLAN 1 lists the two MAC addresses for the two PCs. The following steps outline the logic and then give you three different alternatives for how Step 6 could work:

1. PC11 generates the frame with destination MAC 0200.1111.0002.
2. Switch 1 receives the frame on port E1.

3. Switch 1 performs address table lookup in VLAN 1's address table because incoming port E1 is in VLAN 1.
4. Switch 1 forwards the frame out its E10 port.
5. Switch 2 receives the frame in its E11 port.

The creators of LAN switching might have considered the next three options for what should happen next. Just to let you consider some of the issues, there are three alternatives to the next step (Step 6) that the people who made this up might have considered. They did only one of these—but some of the choices for how they could have designed Switch 2 to react to the incoming frame are as follows:

6. Switch 2 considers port E11 to be in VLAN 1, so it performs table lookup for 0200.1111.0002 in that address table and correctly forward the frame to PC12.

or Switch 2 does not consider port E11 to be in any particular VLAN, so it does table lookup in all tables and forwards out all ports matched. Because PC12 is only in VLAN 1, it would possibly match only VLAN 1's address table anyway or Before Switch 1 forwards the frame in Step 4, it adds a header that identifies the VLAN.
Then Switch 2 can look at the frame header to identify the VLAN number and can do table lookup just in that VLAN's address table. LAN switches use the third alternative: frame tagging. The first option would work fine for one VLAN, and it is used when connecting multiple switches without using VLANs. However, the logic in this first option fails when devices in VLAN 2 send frames because their addresses would never be found in VLAN 1's address table. The second option would work well for unicasts, particularly because a unicast address should be found in only a single address table. However, broadcasts would be sent on all interfaces in all VLANs, which would cause horrendous side effects for OSI Layer 3 processes. So, the third option, called *VLAN tagging*, is used. VLAN tagging is the process of adding an additional header to a LAN frame to identify the VLAN to which the frame belongs. Cisco refers to this as *trunking*.

Cisco provides two trunking options on Ethernet. Inter-Switch Link (ISL) tags frames using Cisco-proprietary framing However, a few items are fair game. First, the ISL header encapsulates the LAN frame, which lengthens the frame. IEEE 802.1Q, the alternative trunking protocol on Ethernet, actually modifies the existing Ethernet header to accomplish the same tagging goal. The second important feature is the VLAN ID field, which exists in both ISL and 802.1Q. The VLAN ID simply identifies the VLAN to which the encapsulated frame belongs.

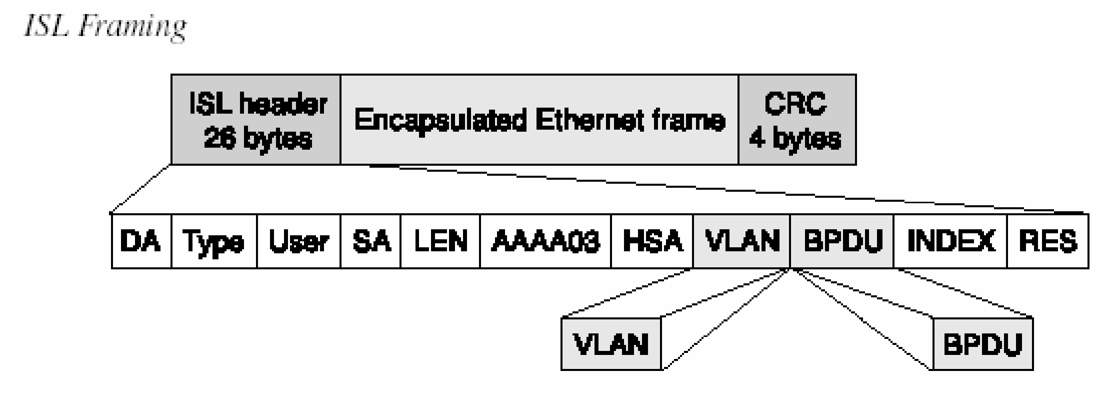


fig 3.5

Back to the original point in Figure 3-4: PC11 sends a frame to PC12, and now ISL trunking is configured between the two switches. The following list outlines what happens:

1 PC11 generates the frame with destination MAC 0200.1111.0002.

2 Switch 1 receives the frame on port E1.

3 Switch 1 performs address table lookup in VLAN 1’s address table because incoming port E1 is in VLAN 1.

4 Switch 1 forwards the frame out its E10 port after adding an ISL header that includes VLAN 1 as the VLAN ID.

5 Switch 2 receives the frame in its E11 port. Expecting an ISL-encapsulated frame, Switch 2 de-encapsulates the original frame, noting that it is in VLAN 1.

6 Switch 2 performs address table lookup in VLAN 1’s address table only and forwards the frame out port E1 based on the table.

Trunking, in this case, simply enables the two switches to identify which VLAN a frame belongs to. Engineers use trunking between two switches, as well as between a switch and a router. Trunking between a switch and a router reduces the number of router interfaces needed. Figure 3-5 shows a router with a single Fast Ethernet interface and a single connection to Switch 2. The same tagging method used between switches is used for frames sent to the router so that the router knows from which VLAN the frame originated. For frames that the router routes between the two VLANs, the incoming frame is tagged with one VLAN ID, and the outgoing frame is tagged with the other VLAN ID by the router before sending the frame back to the switch.

Figure 3-5 shows an example network, with flows from VLAN 1 to VLAN 2. Example below shows the router configuration required to support ISL encapsulation and forwarding between these VLANs.

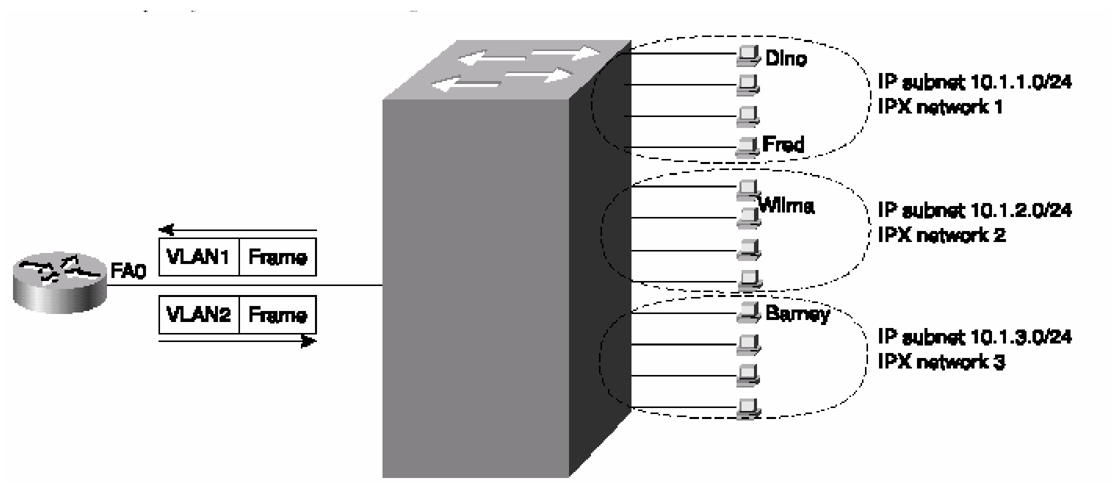


fig 3.6

```
interface fastethernet 0.1
ip address 10.1.1.1 255.255.255.0
encapsulation isl 1
!
interface fastethernet 0.2
ip address 10.1.2.1 255.255.255.0
encapsulation isl 2
!
interface fastethernet 0.3
ip address 10.1.3.1 255.255.255.0
encapsulation isl 3
```

Example above shows the configuration for three subinterfaces of the Ethernet interface on the router. Each is assigned an IP address because the interface is actually a part of three VLANs, implying three IP subnets. So, instead of three physical interfaces, each attached to a different subnet and broadcast domain, there is one physical router interface, with three logical subinterfaces, each attached to a different subnet and broadcast domain. The **encapsulation** command numbers the VLANs, which must match the configuration for VLAN IDs in the switch.

Table below lists the various types of tagging used by Cisco and the types of interfaces on which they are used.

Frame Trunking/Tagging Protocols

Tagging Method	Medium
Inter-Switch Link (ISL)	Fast Ethernet
802.1Q	Fast Ethernet
802.10	FDDI
LAN Emulation (LANE)	ATM

3.3 VLAN Trunking Protocol (VTP)

VTP is a Layer 2 messaging protocol that maintains VLAN configuration consistency throughout a common administration domain. VTP manages the additions, deletions, and name changes of VLANs across multiple switches, minimizing misconfigurations and configuration inconsistencies that can cause problems, such as duplicate VLAN names or incorrect VLANtype specifications.

VTP makes VLAN configuration easier. However, you have not yet seen how to configure VLANs. To appreciate VTP, consider this example. If a network has ten switches that are interconnected, and parts of VLAN 3 were on all ten switches, you would have to type the same config command on all ten switches to “create” the VLAN. With VTP, you would do that once and the other nine switches would learn about VLAN 3 dynamically.

VTP distributes and synchronizes identifying information about VLANs configured throughout a switched network. Configurations made to a single switch, which is called the VTP server, are propagated across trunk links to all switches in the same VTP domain.

VTP allows switched network solutions to scale to large sizes by reducing the manual configuration needs in the network. The VTP domain is created by having each switch in the domain configure the same domain name. The network administrator chooses which switches are in the same domain by deciding which switches share common VLANs. One (or more) switch creates VLANs as the VTP server; then the others are configured as clients for full VTP operation. (VTP transparent mode, a third option, is covered shortly.)

3.3.1 How VTP Works

VTP advertisements are flooded throughout the management domain every 5 minutes, or whenever there is a change in VLAN configurations. Included in a VTP advertisement is a configuration revision number, as well as VLAN names and numbers, and information about which switches have ports assigned to each VLAN. By configuring the details on one server and propagating the information through advertisements, all switches know the names and numbers of all VLANs.

One of the most important cmponents of the VTP advertisements is the *configuration revision number*. Each time a VTP server modifies its VLAN information, it increments the configuration revision number by one. The VTP server then sends out a VTP advertisement that includes the new configuration revision number. When a switch receives a VTP advertisement with a larger configuration revision number, it updates its VLAN configuration. Figure 3-6 illustrates how VTP operates in a switched network.

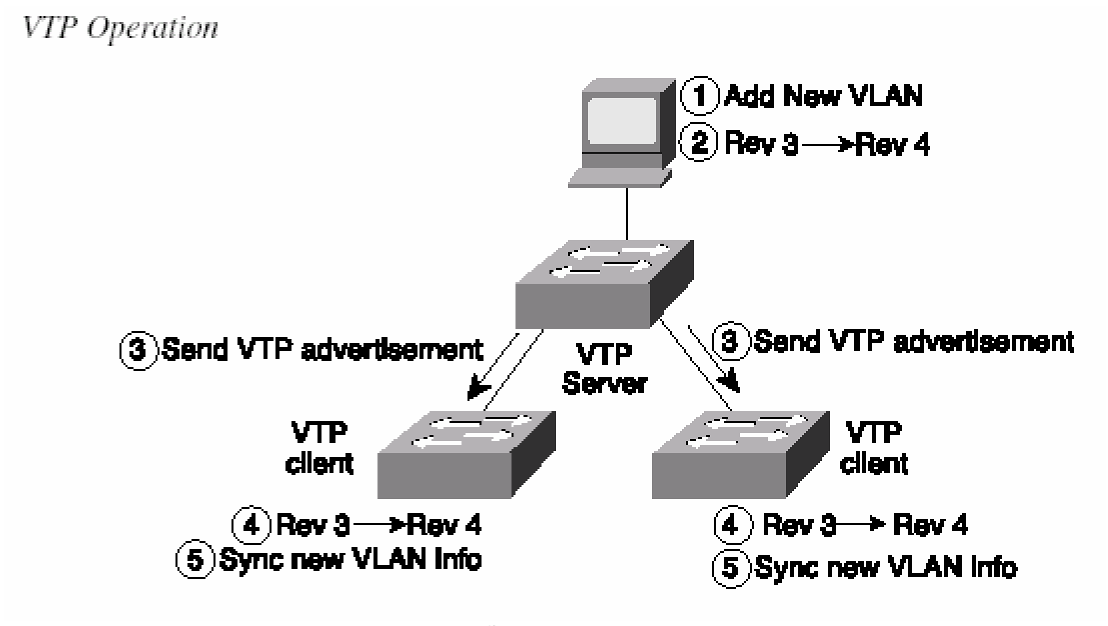


fig 3.7

VTP operates in one of three modes:
• Server mode
• Client mode
• Transparent mode

VTP servers can create, modify, and delete VLANs and other configuration parameters for the entire VTP domain; this information, in turn, is propagated to the VTP clients in that same domain. VTP servers save VLAN configurations in the Catalyst NVRAM, whereas, in clients, the VLAN configuration is not stored. VTP messages are transmitted by the server out all trunk connections.

A VTP client cannot create, change, or delete VLANs, nor can it save VLAN configurations in nonvolatile memory. So, why be a VTP client? Well, if one engineer designs and implementsthe network, it’s a lot more convenient to configure the VLANs in one switch (the VTP server) and have the information propagated to VTP clients.

In some cases, a VLAN exists in multiple switches, but administrative control of those switches is among different departments. VTP transparent mode provides an option so that some switches can use VTP but other switches can ignore VTP, while not stopping the other switches from using it. A switch in transparent mode forwards VTP advertisements received from other switches that are part of the same management domain, while ignoring the information in the VTP message. A switch configured in VTP transparent mode can create, delete, and modify VLANs, but the changes are not transmitted to other switches in the domain; they affect only that local switch. Choosing to use transparent mode is typical when there is a need for distributed administrative control of the switches, in spite of the fact that they each control parts of the same VLANs.

Table below offers a comparative overview of the three VTP modes.

VTP Modes

Function	Server Mode	Client Mode	Transparent Mode
Originates VTP advertisements	Yes	No	No
Processes received advertisements and synchronizes VLAN configuration information with other switches	Yes	Yes	No
Forwards VTP advertisements received in a trunk	Yes	Yes	Yes
Saves VLAN configuration in NVRAM	Yes	No	Yes
Can create, modify, or delete VLANs using configuration commands	Yes	No	Yes

Because ISL trunk lines carry VLAN traffic for all VLANs, some traffic might be needlessly broadcast across links that do not need to carry that traffic. VTP pruning uses VTP advertisements to determine when parts of the network do not have any members in a particular VLAN. By knowing what switches do not have members of a VLAN, VTP can prune some trunks, meaning these trunks do not forward broadcast for that VLAN. By default, a trunk connection carries traffic for all VLANs in the VTP management domain. Commonly, some switches in an enterprise network do not have local ports configured in each VLAN, so VTP pruning provides a way to be ready for future expansion but not waste trunk capacity with needless broadcasts. Figure 5-14 provides an example of VTP pruning.

VTP Pruning Example

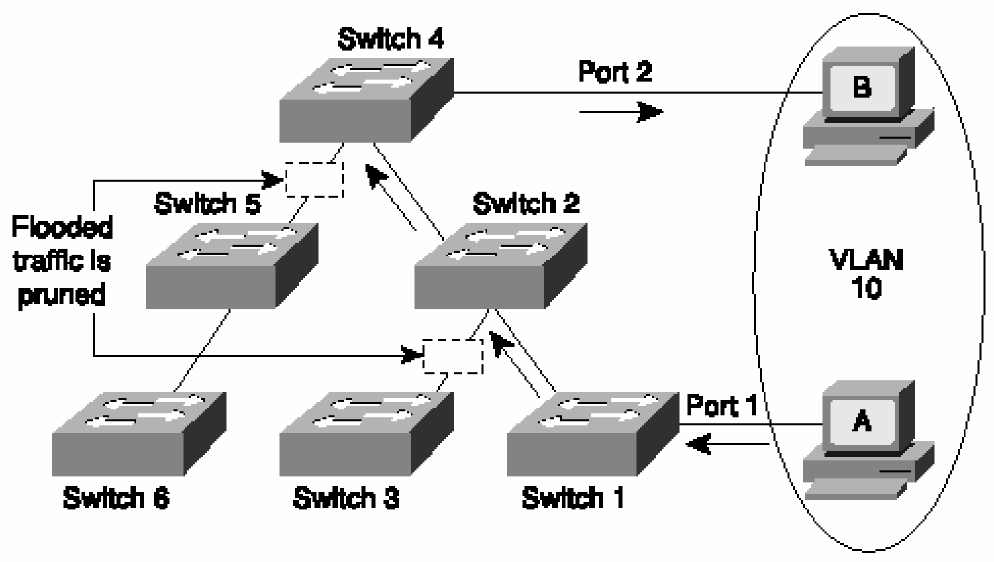


fig 3.8

In Figure 3-8, switches 1 and 4 support ports in VLAN 10. As illustrated, with VTP pruning enabled, when Station A sends a broadcast, the broadcast is flooded only toward any switch with ports assigned to VLAN 10. As a result, broadcast traffic from Station A is not forwarded to switches 3, 5, and 6 because traffic for VLAN 10 has been pruned on the links indicated on switches 2 and 4. VTP pruning increases available bandwidth by restricting flooded traffic, which consists of broadcasts and unknown destination unicasts. VTP pruning is one of the two most compelling reasons to use VTP—the other reason is to make VLAN configuration easier and more consistent.

3.4 VLAN and Trunking Configuration

You can purchase Cisco switches, install devices with the correct cabling, turn on the switches, and see it all work. You would never need to configure the switch and it would work fine, even if you interconnected switches—until you needed more than one VLAN. With redundant links between switches, STP would be needed, but it is enabled by default. So, on the CCNA exam, VLAN configuration is covered partly because it is one of the items that you must configure on a Cisco switch, assuming that you want to use VLANs.

VTP also can be configured, but it is on by default. However, the details of how you might configure VLANs are affected by the VTP configuration, so this is covered here as well. LAN switch configuration on the CCNA exam assumes the use of a 1900 series switch.

3.4.1 Basic VLAN Configuration

You should remember several items before you begin VLAN configuration:

• The maximum number of VLANs is switch-dependent. The Catalyst 1900 supports 64 VLANs with a separate spanning tree per VLAN.

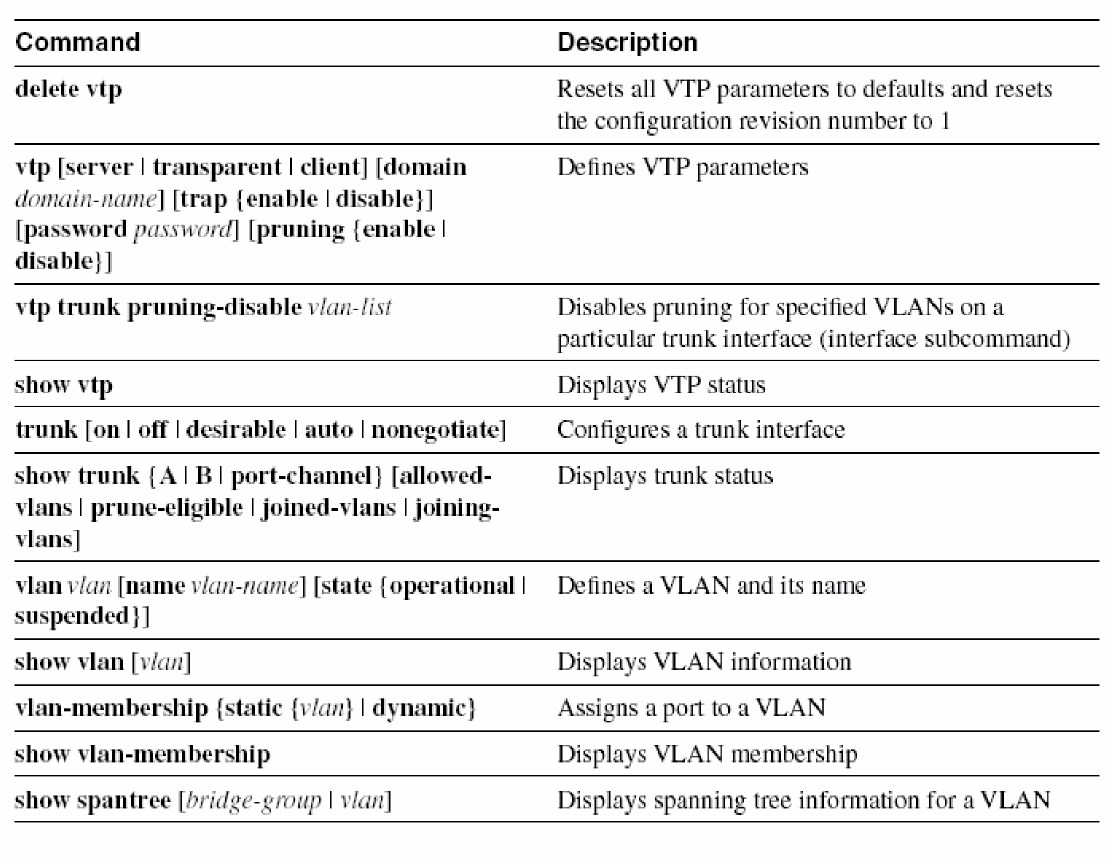
• VLAN 1 is one of the factory-default VLANs.

• CDP and VTP advertisements are sent on VLAN 1.

• Catalyst 1900 IP address is in the VLAN 1 broadcast domain.

• The switch must be in VTP server mode or transparent mode to create, add, or delete VLANs.

Table below represents the commands covered in this section and gives a brief description of each command's function.


VLAN Command List

Command	Description
delete vtp	Resets all VTP parameters to defaults and resets the configuration revision number to 1
vtp [server | transparent | client] [domain *domain-name*] **[trap {enable | disable}] [password** *password*] **[pruning {enable | disable}]**	Defines VTP parameters
vtp trunk pruning-disable *vlan-list*	Disables pruning for specified VLANs on a particular trunk interface (interface subcommand)
show vtp	Displays VTP status
trunk [on | off | desirable | auto | nonegotiate]	Configures a trunk interface
show trunk {A | B | port-channel} [allowed-vlans | prune-eligible | joined-vlans | joining-vlans]	Displays trunk status
vlan *vlan* **[name** *vlan-name*] **[state {operational | suspended}]**	Defines a VLAN and its name
show vlan [*vlan*]	Displays VLAN information
vlan-membership {static {*vlan***} | dynamic}**	Assigns a port to a VLAN
show vlan-membership	Displays VLAN membership
show spantree [*bridge-group* | *vlan*]	Displays spanning tree information for a VLAN

3.4.2 VLAN Configuration for a Single Switch

If only one switch is in use, there is no real benefit to using VTP. However, VTP is on in server mode by default. Because VTP does not help when using a single switch, the first example shows VTP functions being turned off by enabling VTP transparent mode. The steps taken in this example are listed here:

1 Enabling VTP transparent mode

2 Creating the VLAN numbers and names

3 Configuring each port's assigned VLAN

First, use the **vtp** global configuration command to configure VTP transparent mode. Use the**vlan** global command to define each VLAN number (required) and associated name (optional).Then assign each port to its associated VLAN using the **vlan-membership** interface subcommand. Example below shows an example based on Figure 3-10.

Sample Network with One Switch and Three VLANs

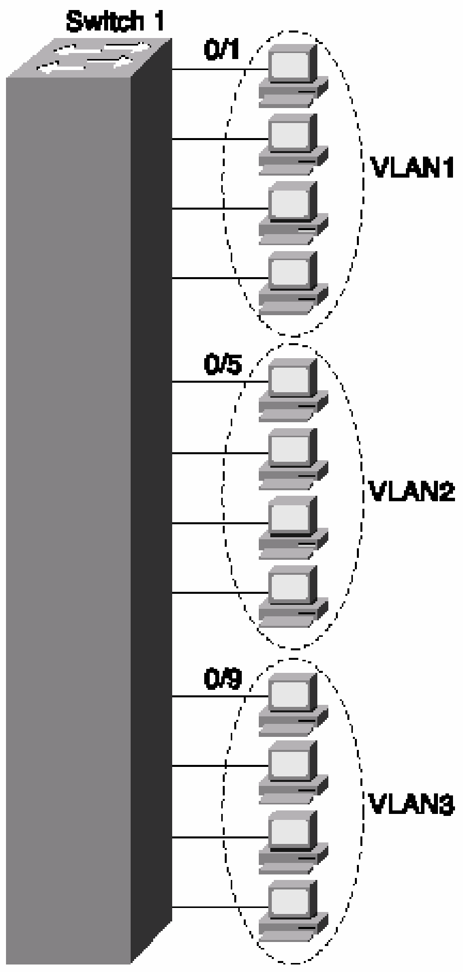
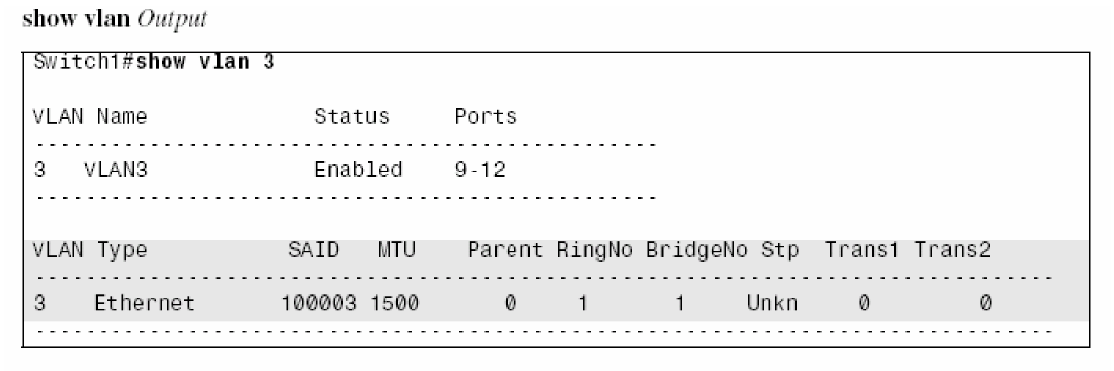


fig 3.10

Single-Switch VLAN Configuration Matching Figure 5-15

```
switch(config)# vtp transparent domain dummy
switch(config)# vlan 2 name VLAN2
switch1(config)# vlan 3 name VLAN3
switch1(config)# interface e 0/5
switch1(config-if)# vlan-membership static 2
switch1(config-if)# interface e 0/6
switch1(config-if)# vlan-membership static 2
switch1(config-if)# interface e 0/7
switch1(config-if)# vlan-membership static 2
switch1(config-if)# interface e 0/8
switch1(config-if)# vlan-membership static 2
switch1(config-if)# interface e 0/9
switch1(config-if)# vlan-membership static 3
switch1(config-if)# interface e 0/10
switch1(config-if)# vlan-membership static 3
switch1(config-if)# interface e 0/11
switch1(config-if)# vlan-membership static 3
switch1(config-if)# interface e 0/12
switch1(config-if)# vlan-membership static 3
```

Notice that some configuration seems to be missing. VLAN 1, with name VLAN 1, is not configured because it is configured automatically. In fact, the name cannot be changed. Also, any ports without a specific static VLAN configuration are considered to be in VLAN 1. Also, the IP address of the switch is considered to be in VLAN 1's broadcast domain. Ports 5 through 8 are statically configured for VLAN 2; similarly, VLAN 3 comprises ports 9 through 12. In addition, VTP is set to transparent mode, with a meaningless domain name of dummy. After the VLANs are configured, the parameters for that VLAN should be confirmed to ensure validity. To verify the parameters of a VLAN, use the **show vlan** *vlan#* privileged exec command to display information about a particular VLAN. Use **show vlan** to show all configured VLANs. Example 5-3 demonstrates the **show** command output, which shows the switch ports assigned to the VLAN.


show vlan *Output*

```
Switch1#show vlan 3

VLAN Name             Status     Ports
-------------------------------------------------
3   VLAN3             Enabled    9-12
-------------------------------------------------

VLAN Type           SAID   MTU    Parent RingNo BridgeNo Stp  Trans1 Trans2
--------------------------------------------------------------------------------
3    Ethernet       100003 1500       0     1       1     Unkn     0        0
--------------------------------------------------------------------------------
```

3.4.3 Sample Configuration for Multiple Switches

To allow VLANs to span multiple switches, you must configure *trunks* to interconnect the switches. Trunks are simply LAN segments that connect switches and use one of two methods of tagging the frames with the VLAN number. Cisco calls the use of a trunking protocol such as ISL or 802.1Q *trunking*, so the command to enable these protocols is **trunk**. Use the **trunk** interface configuration command to set a Fast Ethernet port to trunk mode. On the Catalyst 1900, the two Fast Ethernet ports are interfaces fa0/26 and fa0/27. Enabling and defining the type of trunking protocol can be done statically or dynamically for ISL. The syntax for the **trunk** Fast Ethernet interface configuration subcommand is as follows:

```
switch(config-if)# trunk [on | off | desirable | auto | nonnegotiate]
```

The options for the **trunk** command function are as follows:

• **on**—Configures the port into permanent ISL trunk mode and negotiates with the connected device to convert the link to trunk mode.
• **off**—Disables port trunk mode and negotiates with the connected device to convert the link to nontrunk.
• **desirable**—Triggers the port to negotiate the link from nontrunking to trunk mode. The port negotiates to a trunk port if the connected device is either in the **on**, **desirable**, or **auto** states. Otherwise, the port becomes a nontrunk port.

• **auto**—Enables a port to become a trunk only if the connected device has the state set to
and VTP server configuration. **on** or **desirable**.
• **nonegotiate**—Configures a port to permanent ISL trunk mode, and no negotiation takes place with the partner. As seen in the list, many options exist. Choices for these options are mostly personal preference. Because trunks seldom change, my preference is to configure either **on** or **off**.

Figure 3-11 and Examples below provide an expanded sample network, along with the additional configuration required for trunking

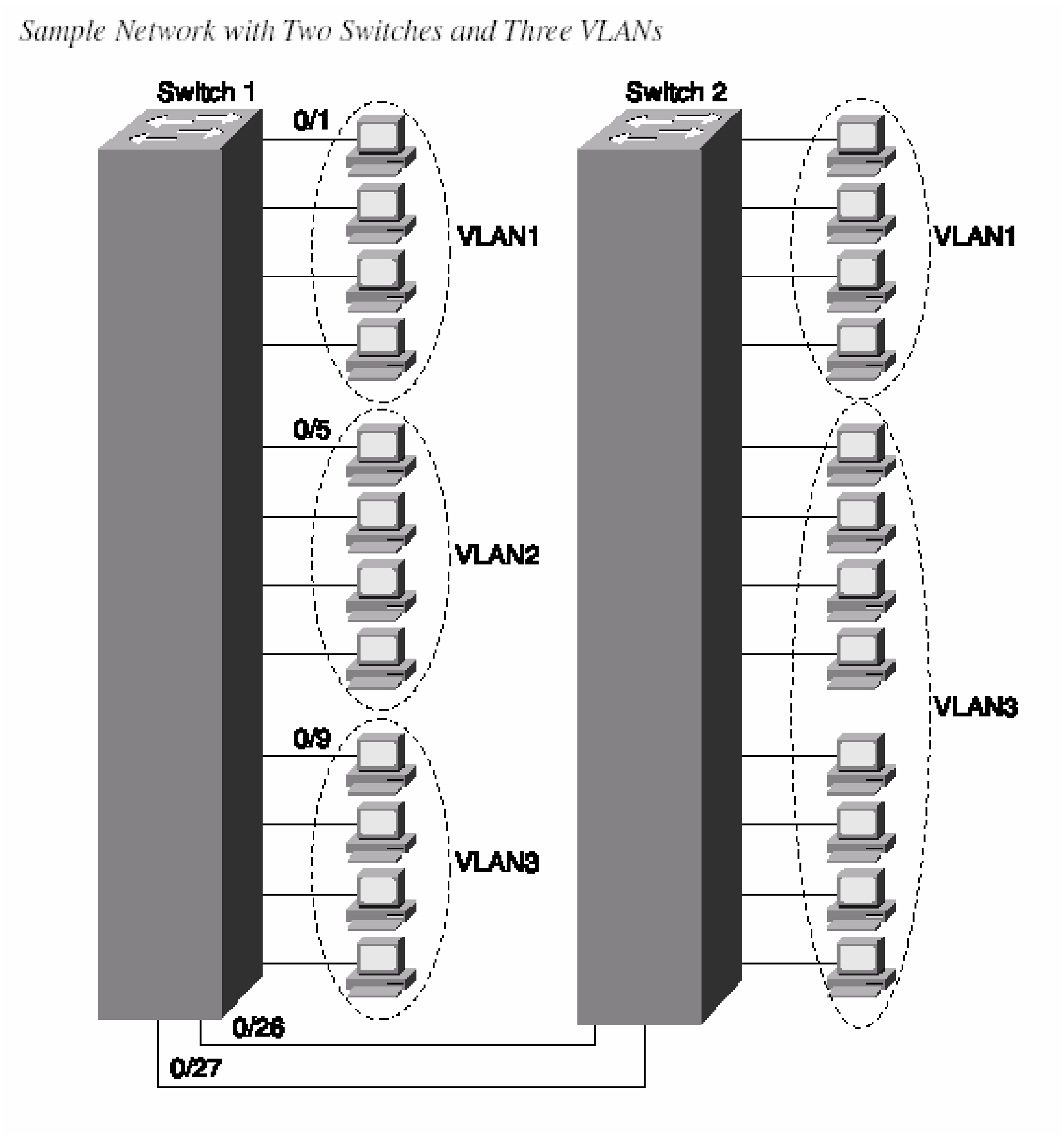
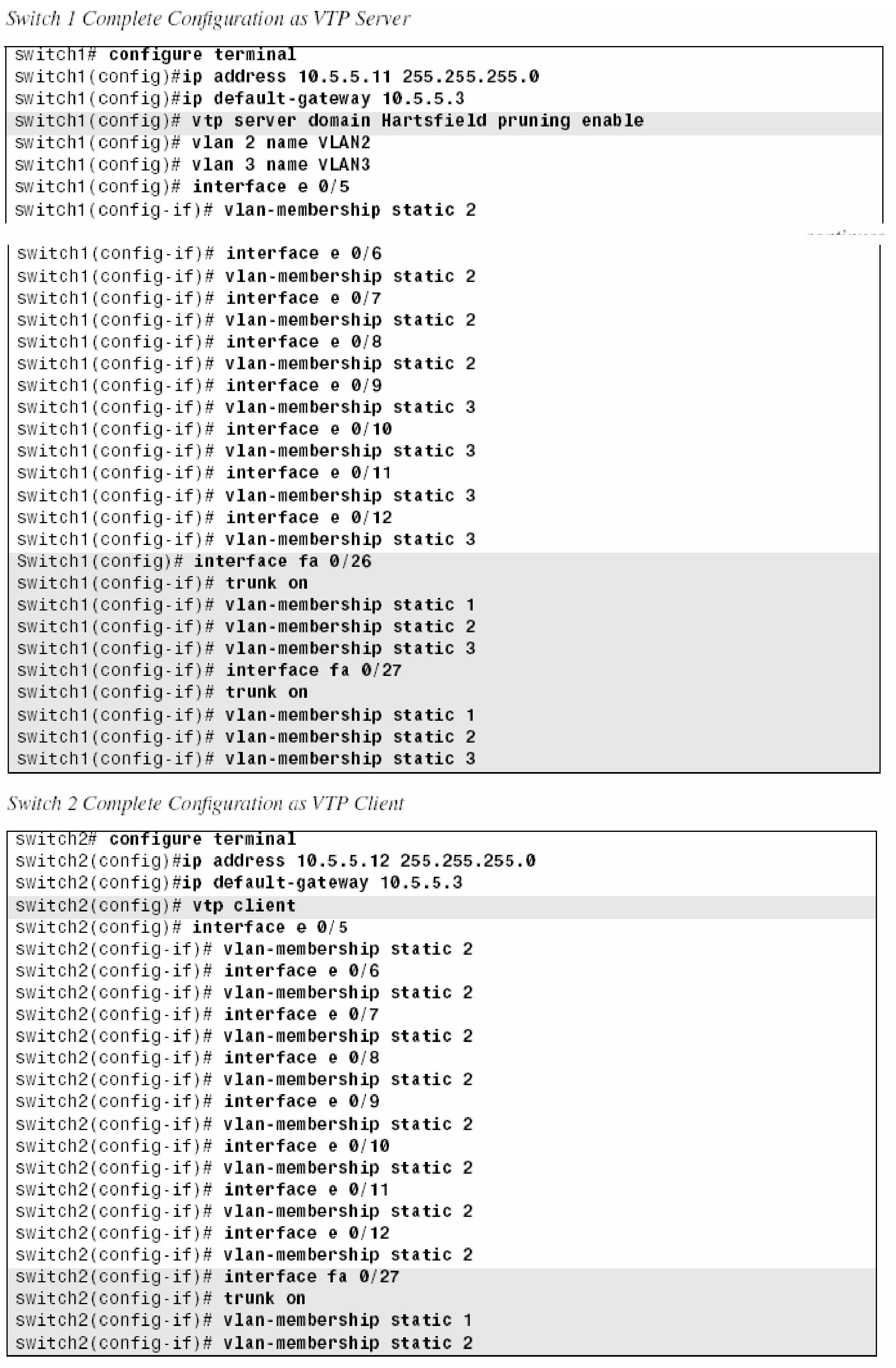


fig 3.11



Switch 1 Complete Configuration as VTP Server

```
switch1# configure terminal
switch1(config)#ip address 10.5.5.11 255.255.255.0
switch1(config)#ip default-gateway 10.5.5.3
switch1(config)# vtp server domain Hartsfield pruning enable
switch1(config)# vlan 2 name VLAN2
switch1(config)# vlan 3 name VLAN3
switch1(config)# interface e 0/5
switch1(config-if)# vlan-membership static 2
```

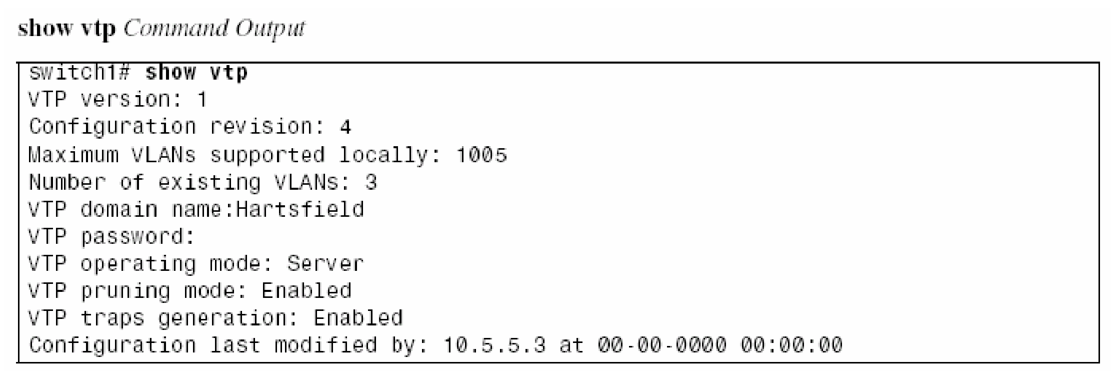
```
switch1(config-if)# interface e 0/6
switch1(config-if)# vlan-membership static 2
switch1(config-if)# interface e 0/7
switch1(config-if)# vlan-membership static 2
switch1(config-if)# interface e 0/8
switch1(config-if)# vlan-membership static 2
switch1(config-if)# interface e 0/9
switch1(config-if)# vlan-membership static 3
switch1(config-if)# interface e 0/10
switch1(config-if)# vlan-membership static 3
switch1(config-if)# interface e 0/11
switch1(config-if)# vlan-membership static 3
switch1(config-if)# interface e 0/12
switch1(config-if)# vlan-membership static 3
Switch1(config)# interface fa 0/26
switch1(config-if)# trunk on
switch1(config-if)# vlan-membership static 1
switch1(config-if)# vlan-membership static 2
switch1(config-if)# vlan-membership static 3
switch1(config-if)# interface fa 0/27
switch1(config-if)# trunk on
switch1(config-if)# vlan-membership static 1
switch1(config-if)# vlan-membership static 2
switch1(config-if)# vlan-membership static 3
```

Switch 2 Complete Configuration as VTP Client

```
switch2# configure terminal
switch2(config)#ip address 10.5.5.12 255.255.255.0
switch2(config)#ip default-gateway 10.5.5.3
switch2(config)# vtp client
switch2(config)# interface e 0/5
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface e 0/6
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface e 0/7
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface e 0/8
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface e 0/9
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface e 0/10
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface e 0/11
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface e 0/12
switch2(config-if)# vlan-membership static 2
switch2(config-if)# interface fa 0/27
switch2(config-if)# trunk on
switch2(config-if)# vlan-membership static 1
switch2(config-if)# vlan-membership static 2
```

Several items are particularly important in these configurations. The **vtp** global command in Example below shows Switch 1 as the server, with domain Hartsfield. No password is used in this case. Switch 2 is not configured with the domain name but will learn it with the first advertisement. Missing from Example 5-5 is the definition of the VLANs, which not only is unnecessary but also is not allowed when in VTP client mode. And because pruning was enabled in the **vtp** command on Switch 1, VTP prunes VLAN 2 from Switch 2 because Switch 2 has no ports in VLAN 3. VLAN 3 broadcasts received by Switch 1 are not forwarded to Switch 2.

Notice that not only was trunking enabled on both Fast Ethernet ports, but each of the three VLANs was statically configured on those ports. By also configuring the VLANs, the switch treats the trunk ports as part of those VLANs. To verify a recent configuration change, or to just view the VTP configuration information, use the **show vtp** privileged exec command, as demonstrated in the Example. Also displayed is the IP address of the device that last modified the configuration and a time stamp of the time the modification was made. VTP has two versions: VTP Version 1 supports only Ethernet; VTP Version 2 supports Ethernet and Token Ring.


show vtp *Command Output*

```
switch1# show vtp
VTP version: 1
Configuration revision: 4
Maximum VLANs supported locally: 1005
Number of existing VLANs: 3
VTP domain name:Hartsfield
VTP password:
VTP operating mode: Server
VTP pruning mode: Enabled
VTP traps generation: Enabled
Configuration last modified by: 10.5.5.3 at 00-00-0000 00:00:00
```

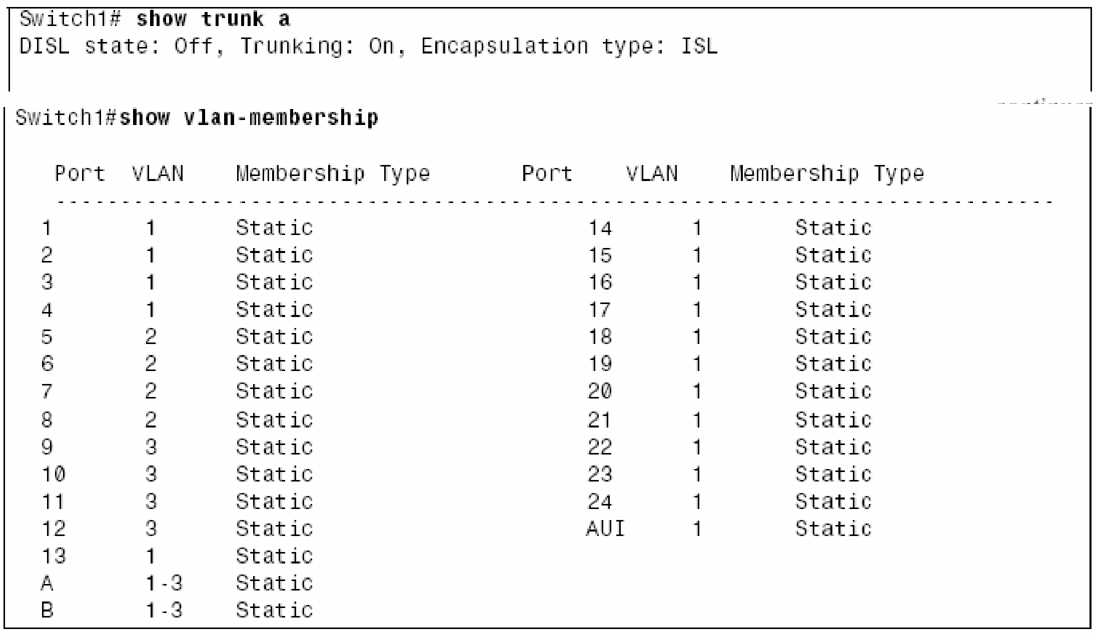
To verify a trunk configuration, use the **show trunk** privileged exec command to display the trunk parameters, as demonstrated in the Example The syntax is as follows:

switch1# show trunk [a | b]

The parameters **a** and **b** represent the Fast Ethernet ports:
• Port a represents Fast Ethernet 0/26.
• Port b represents Fast Ethernet 0/27.

Example 5-7 shows a sample of the **show trunk** command as well as the **show vlanmembership** command.

show trunk *and* **show vlan-membership** *Sample Output*

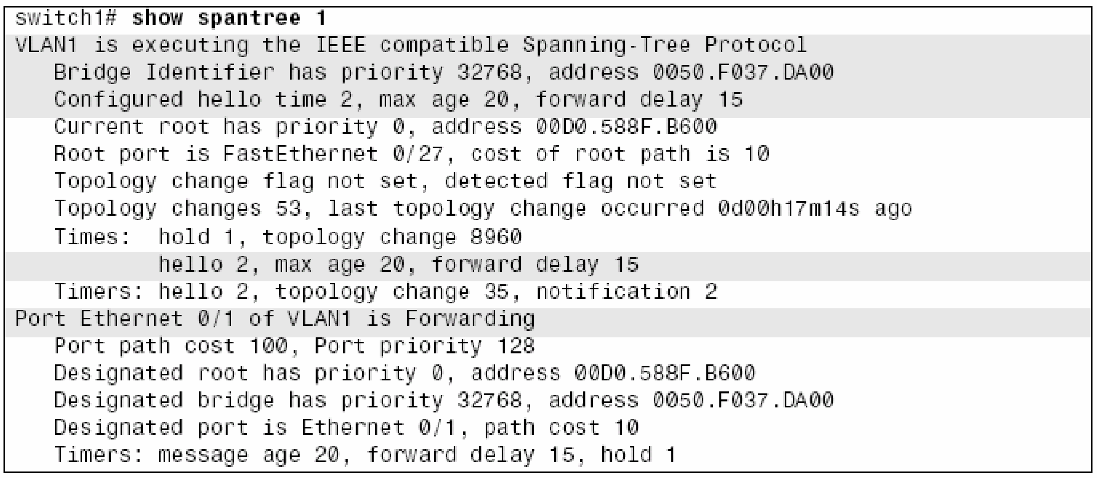


```
Switch1# show trunk a
DISL state: Off, Trunking: On, Encapsulation type: ISL
```

```
Switch1#show vlan-membership

   Port  VLAN    Membership Type       Port    VLAN    Membership Type
   -----------------------------------------------------------------------
  1       1      Static                  14      1       Static
  2       1      Static                  15      1       Static
  3       1      Static                  16      1       Static
  4       1      Static                  17      1       Static
  5       2      Static                  18      1       Static
  6       2      Static                  19      1       Static
  7       2      Static                  20      1       Static
  8       2      Static                  21      1       Static
  9       3      Static                  22      1       Static
  10      3      Static                  23      1       Static
  11      3      Static                  24      1       Static
  12      3      Static                  AUI     1       Static
  13      1      Static
  A       1-3    Static
  B       1-3    Static
```

Although the CCNA exam coverage does not include configuration to tune the behavior of spanning tree, you can see some basic information about STP using the **show spantree** privileged exec command, as demonstrated in Example below.

show spantree *Output*



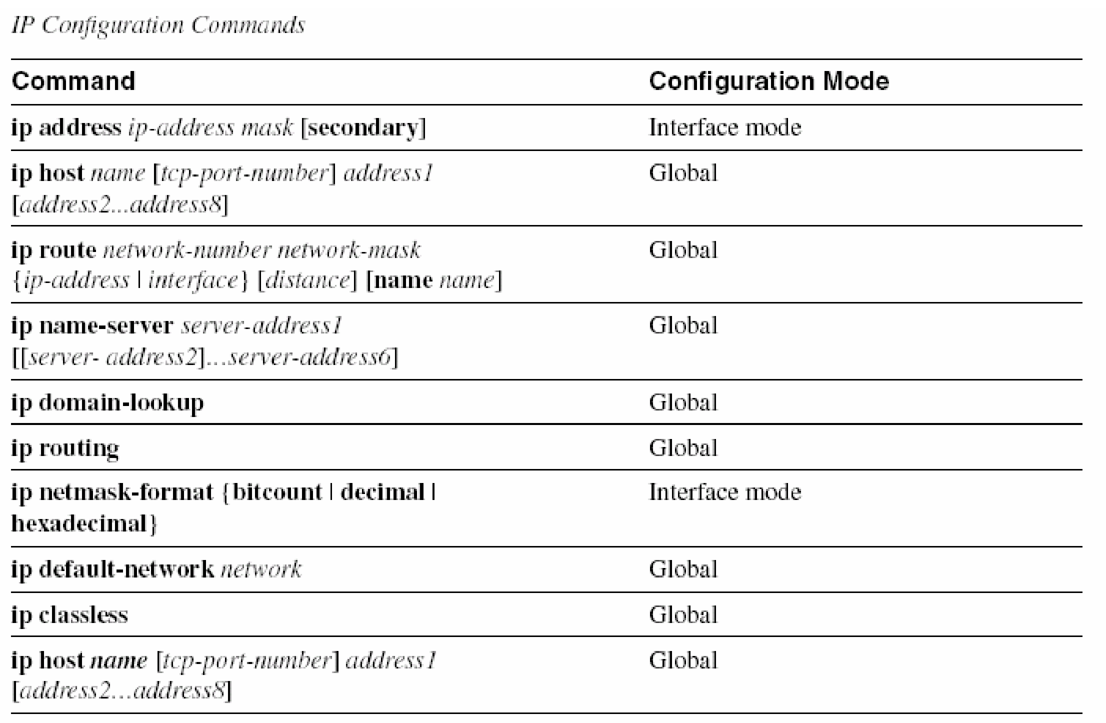
```
switch1# show spantree 1
VLAN1 is executing the IEEE compatible Spanning-Tree Protocol
   Bridge Identifier has priority 32768, address 0050.F037.DA00
   Configured hello time 2, max age 20, forward delay 15
   Current root has priority 0, address 00D0.588F.B600
   Root port is FastEthernet 0/27, cost of root path is 10
   Topology change flag not set, detected flag not set
   Topology changes 53, last topology change occurred 0d00h17m14s ago
   Times:  hold 1, topology change 8960
           hello 2, max age 20, forward delay 15
   Timers: hello 2, topology change 35, notification 2
Port Ethernet 0/1 of VLAN1 is Forwarding
   Port path cost 100, Port priority 128
   Designated root has priority 0, address 00D0.588F.B600
   Designated bridge has priority 32768, address 0050.F037.DA00
   Designated port is Ethernet 0/1, path cost 10
   Timers: message age 20, forward delay 15, hold 1
```

Example above displays various spanning tree information for VLAN 1, including the following:
• Port e0/1 is in the forwarding state for VLAN 1.
• The root bridge for VLAN 1 has a bridge priority of 0, with a MAC address of 00D0.588F.B600.
• The switch is running the IEEE 802.1d Spanning-Tree Protocol.

4 - Configure and Verify IP Addresses

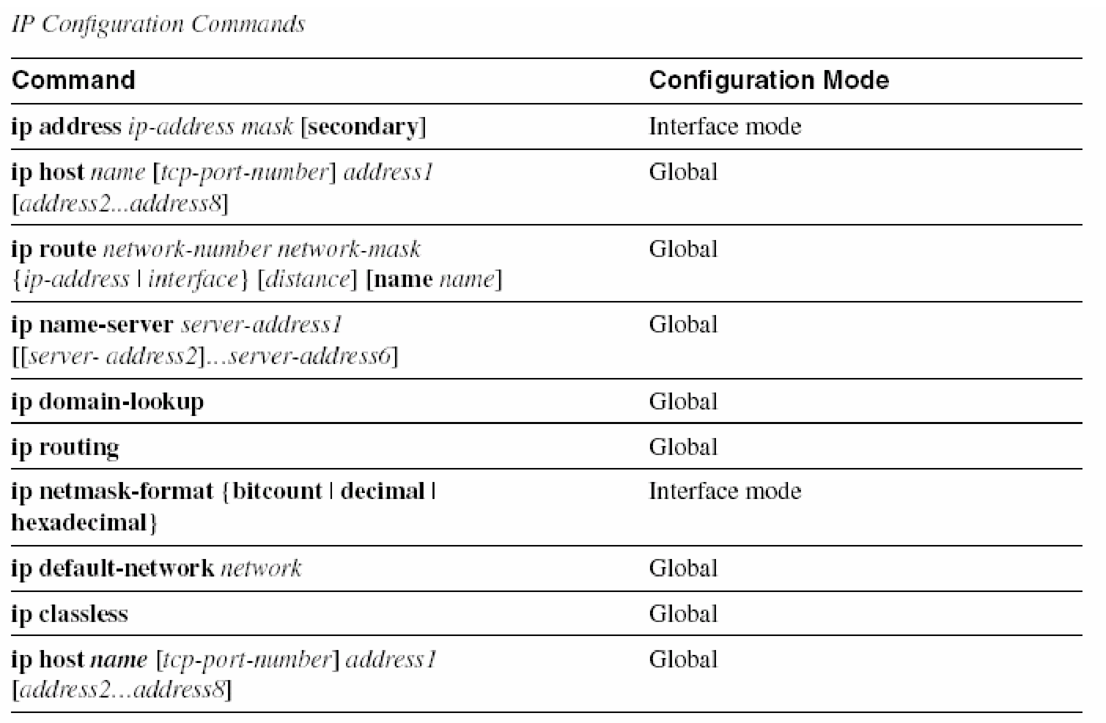
4.1 Configuration Commands

You can easily configure a Cisco router to forward IP traffic when you know the details.The following tables summarize many of the most common commands used for IP configuration and verification. Two sample network configurations, with both configuration and exec command output, follow.



IP Configuration Commands

Command	Configuration Mode
ip address *ip-address mask* [**secondary**]	Interface mode
ip host *name* [*tcp-port-number*] *address1* [*address2...address8*]	Global
ip route *network-number network-mask* {*ip-address* | *interface*} [*distance*] [**name** *name*]	Global
ip name-server *server-address1* [[*server- address2*]…*server-address6*]	Global
ip domain-lookup	Global
ip routing	Global
ip netmask-format {**bitcount** | **decimal** | **hexadecimal**}	Interface mode
ip default-network *network*	Global
ip classless	Global
ip host ***name*** [*tcp-port-number*] *address1* [*address2…address8*]	Global



IP Configuration Commands

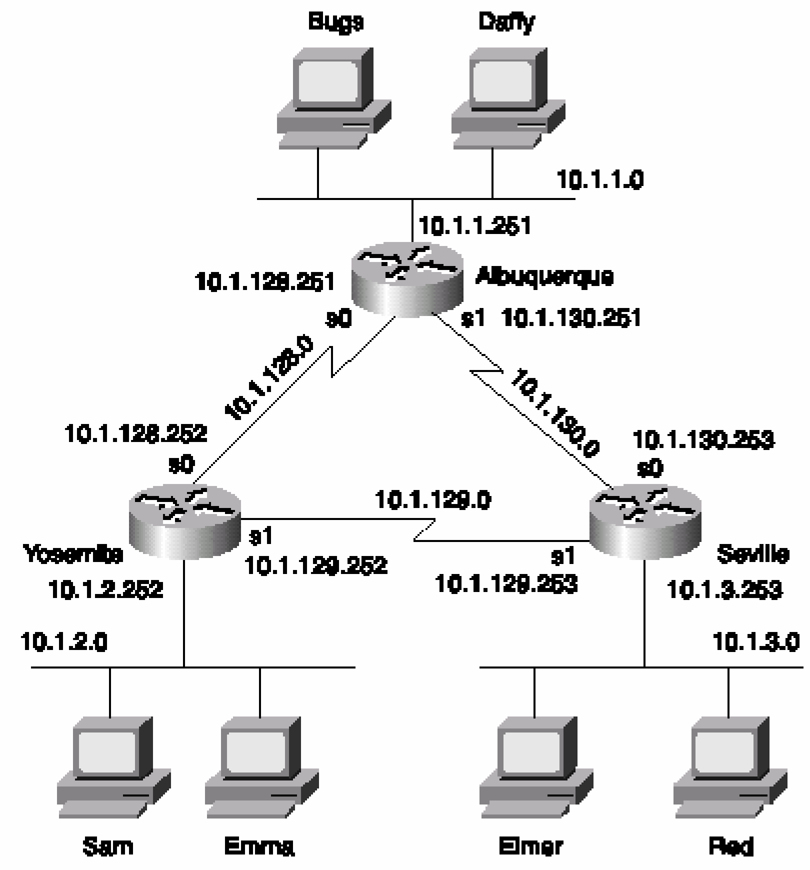
Command	Configuration Mode
ip address *ip-address mask* [**secondary**]	Interface mode
ip host *name* [*tcp-port-number*] *address1* [*address2...address8*]	Global
ip route *network-number network-mask* {*ip-address* | *interface*} [*distance*] [**name** *name*]	Global
ip name-server *server-address1* [[*server- address2*]…*server-address6*]	Global
ip domain-lookup	Global
ip routing	Global
ip netmask-format {**bitcount** | **decimal** | **hexadecimal**}	Interface mode
ip default-network *network*	Global
ip classless	Global
ip host ***name*** [*tcp-port-number*] *address1* [*address2…address8*]	Global

IP Exec Commands (Continued)

Command	Function
terminal ip netmask-format {**bitcount** | **decimal** | **hexadecimal**}	Sets type of display for subnet masks in **show** commands
ping [*protocol* | **tag**] {*host-name* | *system-address*}	Sends and receives ICMP echo messages to verify connectivity
trace [*protocol*] [*destination*]	Sends a series of UDP packets with increasing TTL values to verify the current route to a host

Consider the following figures.
The following site guidelines were used when choosing configuration details:
• Use name servers at 10.1.1.100 and 10.1.2.100.
• The router's IP addresses are to be assigned from the last few valid IP addresses in their attached subnets; use a mask of 255.255.255.0.

Sample Network with Three Routers, with Point-to-Point Serial Links



Albuquerque Router Configuration and Exec Commands

```
Albuquerque#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Albuquerque(config)#interface serial 0
Albuquerque(config-if)#ip address 10.1.128.251 255.255.255.0
Albuquerque(config)#interface serial 1
Albuquerque(config-if)#ip address 10.1.130.251 255.255.255.0
Albuquerque(config)#interface ethernet 0
Albuquerque(config-if)#ip address 10.1.1.251 255.255.255.0

Albuquerque#show running-config
Building configuration...

Current configuration : 872 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Albuquerque
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
!
ip name-server 10.1.1.100
ip name-server 10.1.2.100
!
interface Serial0
 ip address 10.1.128.251 255.255.255.0
!
interface Serial1
 ip address 10.1.130.251 255.255.255.0
!
interface Ethernet0
 ip address 10.1.1.251 255.255.255.0
!
no ip http server

banner motd ^C
  Should've taken a left turn here! This is Albuquerque...  ^C
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

continues

Albuquerque Router Configuration and Exec Commands (Continued)

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

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0

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

Gateway of last resort is not set

     10.0.0.0 255.255.255.0 is subnetted, 3 subnets
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial1

C       10.1.128.0 is directly connected, Serial0
Albuquerque#
```

Yosemite Router Configuration and Exec Commands

```
Yosemite#show running-config
Building configuration...

Current configuration : 867 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Yosemite
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
!
ip name-server 10.1.1.100
```

Yosemite Router Configuration and Exec Commands (Continued)

```
ip name-server 10.1.2.100
!
interface Serial0
 ip address 10.1.128.252 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 10.1.129.252 255.255.255.0
!
interface Ethernet0
 ip address 10.1.2.252 255.255.255.0
!
no ip http server

banner motd ^C
   This is the Rootin-est Tootin-est Router in these here parts!  ^C
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end

Yosemite#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
Serial0                10.1.128.252    YES manual up                    up
Serial1                10.1.129.252    YES manual up                    up
Ethernet0              10.1.2.252      YES manual up                    up
Yosemite#
```

Seville Router Configuration and Exec Commands

```
Seville#show running-config
Building configuration...

Current configuration : 869 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Seville
!
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
ip name-server 10.1.1.100
```

continues

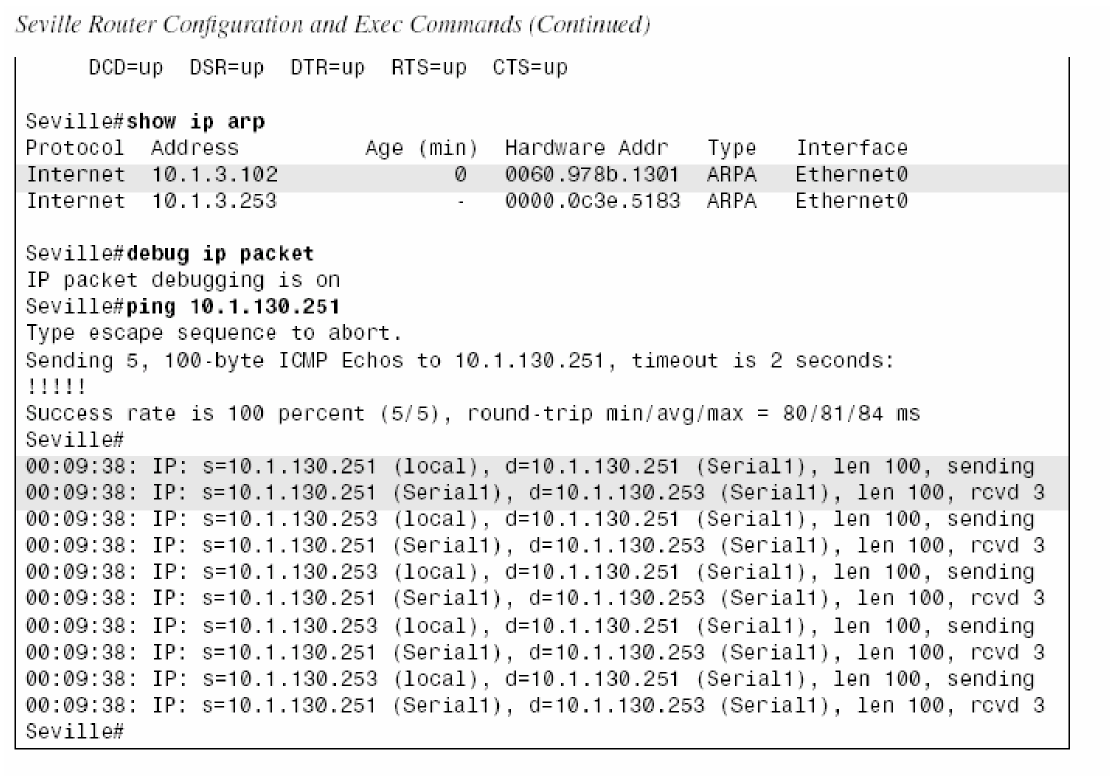
Seville Router Configuration and Exec Commands (Continued)

```
ip name-server 10.1.2.100
!
interface Serial0
 ip address 10.1.130.253 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 10.1.129.253 255.255.255.0
!
Ethernet0
 ip address 10.1.3.253 255.255.255.0
!
no ip http server
banner motd ^C
  Take a little off the top, Wabbit!  (Elmer)   ^C
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end

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


Gateway of last resort is not set


10.0.0.0/24 is subnetted, 3 subnets
C       10.1.3.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial0
C       10.1.129.0 is directly connected, Serial1


Seville#show ip interface serial 1
Serial1 is up, line protocol is up
  Internet address is 10.1.129.253/24
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
```

Seville Router Configuration and Exec Commands (Continued)

```
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is enabled
  Security level is default
  Split horizon is disabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is enabled
  IP fast switching on the same interface is enabled
  IP Flow switching is disabled
  IP Feature Fast switching turbo vector
  IP multicast fast switching is disabled
  IP multicast distributed fast switching is disabled
  IP route-cache flags are Fast
  Router Discovery is disabled
  IP output packet accounting is disabled
  IP access violation accounting is disabled
  TCP/IP header compression is disabled
  RTP/IP header compression is disabled
  Probe proxy name replies are disabled
  Policy routing is disabled
  Network address translation is disabled
  WCCP Redirect outbound is disabled
  WCCP Redirect inbound is disabled
  WCCP Redirect exclude is disabled
  BGP Policy Mapping is disabled

Seville#show interface serial 0
Serial0 is up, line protocol is up
  Hardware is HD64570
  Internet address is 10.1.130.253/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

continues


Seville Router Configuration and Exec Commands (Continued)

```
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

Seville#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.1.3.102               0   0060.978b.1301  ARPA   Ethernet0
Internet  10.1.3.253               -   0000.0c3e.5183  ARPA   Ethernet0

Seville#debug ip packet
IP packet debugging is on
Seville#ping 10.1.130.251
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.130.251, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 80/81/84 ms
Seville#
00:09:38: IP: s=10.1.130.251 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
Seville#
```

As you see in Figure, the IP addresses chosen for the interfaces are shown. At the beginning of example, the engineer uses configuration mode to configure the IP addresses.

The **show running-config** command displays the results of the configuration, along with some other details that were already configured.

Notice that the configuration matches the output of the **show interface**, **show ip interface**, and **show ip interface brief** commands. In example, the IP addresses in the configuration match the output of **show ip interface brief**. If these details did not match, one common oversight would be that you are looking at the configuration in NVRAM, not in RAM. Be sure to use the **show running-config** or **write terminal** commands to see the active configuration. The subnet mask in the output of **show** commands uses prefix notation. For example, 10.1.4.0/ 24 means 24 network and subnet bits, leaving 8 host bits with this subnetting scheme. The **terminal ip netmask** command can be used to change this formatting for screen output during this session.
Other example shows the ARP cache generated by the **show ip arp** output. The first entry shows the IP address (10.1.3.102) and MAC address of another host on the Ethernet. The timer value of 0 implies that the entry is very fresh—the value grows with disuse and eventually times out. One entry is shown for the router’s Ethernet interface itself, which never times out of the ARP table.

The **debug ip packet** output in example lists one entry per IP packet sent and received. This command is a very dangerous one—it could crash almost any production router because of the added overhead of processing the debug messages. Imagine a router that forwards 50,000 packets per second, needing to send 50,000 messages per second to a console that’s running at 9600 bps! The router would buffer the messages and exhaust all its memory doing so, and the router would crash. Also notice that the output shows both the source and destination IP addresses. As compared with a working, real configuration in a production network, these examples omit the needed routing protocol configuration. The routing table in example does not list all subnets because the routing protocol configuration has not been added—notice that all the routers have a value of C beside them, which, according to the legend shown at the beginning of the command, means that the route describes a connected subnet. However, because you also need to know how to configure static routes rather than show the configuration of a routing protocol next, the **ip route** commands in other example have been added to Albuquerque, which adds static routes. Last examples contain **show** commands executed after the new configuration was added.

Static Routes Added to Albuquerque

```
ip route 10.1.2.0 255.255.255.0 10.1.128.252
ip route 10.1.3.0 255.255.255.0 10.1.130.253
```

Albuquerque Router Exec Commands, After Adding Static Routes for 10.1.2.0 and 10.1.3.0

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


Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 5 subnets
S       10.1.3.0 [1/0] via 10.1.130.253
S       10.1.2.0 [1/0] via 10.1.128.252
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0
Albuquerque#ping 10.1.128.252

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.128.252, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/8 ms
```

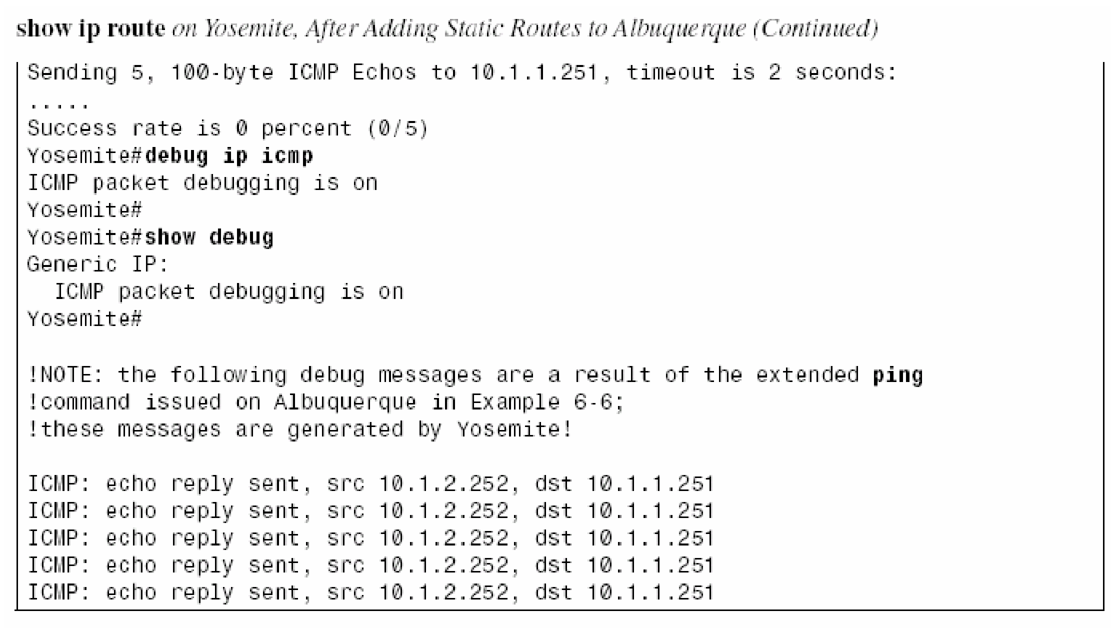
continues

Albuquerque Router Exec Commands, After Adding Static Routes for 10.1.2.0 and 10.1.3.0 (Continued)

```
! Note: the following extended ping command will result in some debug messages
! on Yosemite in Example 6-7.

Albuquerque#ping
Protocol [ip]:
Target IP address: 10.1.2.252
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 10.1.1.251
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.252, timeout is 2 seconds:
. . . . .
Success rate is 0 percent (0/5)
Albuquerque#
```

show ip route *on Yosemite, After Adding Static Routes to Albuquerque*

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

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
C       10.1.2.0 is directly connected, Ethernet0
C       10.1.129.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0
Yosemite#ping 10.1.128.251

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.128.251, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/8 ms
Yosemite#ping 10.1.1.251

Type escape sequence to abort.
```



show ip route *on Yosemite, After Adding Static Routes to Albuquerque (Continued)*

```
Sending 5, 100-byte ICMP Echos to 10.1.1.251, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Yosemite#debug ip icmp
ICMP packet debugging is on
Yosemite#
Yosemite#show debug
Generic IP:
  ICMP packet debugging is on
Yosemite#

!NOTE: the following debug messages are a result of the extended ping
!command issued on Albuquerque in Example 6-6;
!these messages are generated by Yosemite!

ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
```

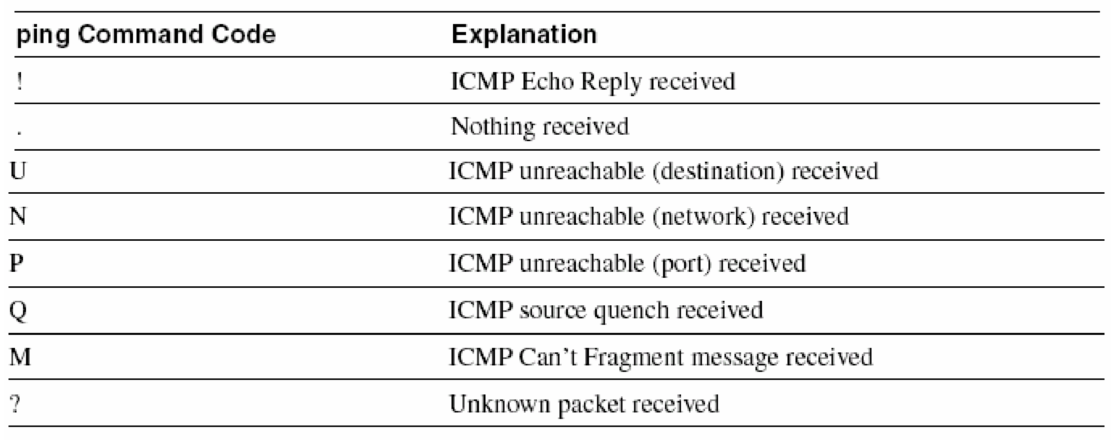
First, you should examine the static routes in example. On Albuquerque, one route defines a route to 10.1.2.0, off Yosemite, so the next-hop IP address is configured as 10.1.128.252, which is Yosemite’s Serial 0 IP address. Similarly, a route to 10.1.3.0, the subnet off Seville, points to Seville’s Serial 0 IP address, 10.1.130.253. Presumably, Albuquerque can forward packets to these subnets now; as seen in other example ’s first **ping** command, the ping works. However, if you look at just the syntax but not the rest of these two examples, you will miss an important concept about routing and some necessary details about the **ping** command. First, two subtleties of the **ping** command are used in these two example console dialogs of Examples.

• Cisco **ping** commands use the output interface’s IP address as the source address of the packet, unless otherwise specified in an extended ping. The first ping in example uses a source of 10.1.128.251; the extended ping uses the source address that the user typed in (10.1.1.251).

• ICMP Echo Reply messages (ping responses) reverse the IP addresses used in the ICMP Echo Request to which they are responding. To make the ping options appear more obvious, this configuration does not contain routes on Yosemite or Seville, pointing back to the subnet off Albuquerque—namely, 10.1.1.0. In a real network, routing protocols would be used instead. If static routes were used, you would need routes pointing in both directions. But because the needed static route on Yosemite, pointing back to 10.1.1.0, is missing, packets from subnet 10.1.1.0 can get to 10.1.2.0 but cannot get back. When you troubleshoot this network, you can use the extended **ping** command to act like you issued a ping from a computer on that subnet, without having to call a user and ask him to type a **ping** command for you on his PC. The extended version of the **ping** command can be used to more fully refine the underlying cause of the problem. In fact, when a ping from a router works but a ping from a host does not, the extended ping could help in re-creating the problem without needing to work with the end user on the phone. For example, the extended **ping** command on Albuquerque sent an Echo Request from 10.1.1.251 (Albuquerque’s Ethernet) to 10.1.2.252 (Yosemite’s Ethernet); no response was received by

Albuquerque. Normally, the echoes are sourced from the IP address of the outgoing interface; with the use of the extended **ping** source address option, the source IP address of the echo packet can be changed. Because the ICMP echo generated by the extended ping is sourced from an address in 10.1.1.0, it looks more like a packet from an end user in that subnet. It appears that the ICMP Echo Requests generated by the extended ping were received by Yosemite because the debug messages on Yosemite imply that it sent ICMP Echo Replies back to 10.1.1.251. Somewhere between Yosemite creating the ICMP echo replies and Albuquerque receiving them, a problem occurred. The problem, as mentioned earlier, is that Yosemite has no routes that tell it how to forward packets back to 10.1.1.0. An examination of the steps after the Echo Replies were created by Yosemite is needed to understand the problem in this example. ICMP asks the IP software in Yosemite to deliver the packets. The IP code performs IP routing table lookup to find the correct route for these packets, whose destination is 10.1.1.251. However, the **show ip route** command output in Example 6-7 shows that Yosemite has no route to subnet 10.1.1.0. It seems that Yosemite created the Echo Reply messages but failed to send them because it has no route to 10.1.1.0/24. This is just one example in which the route in one direction is working fine, but the route in the reverse direction is not. Other options for extended ping are also quite useful. The Don't Fragment (DF) bit can be set, along with the amount of data to send in the echo so that the MTU for the entire route can be discovered through experimentation. Echo packets that are too large to pass over a link because MTU restrictions will be discarded because the DF bit is set. The timeout value can be set so that the **ping** command will wait longer than the default two seconds for an Echo Reply. Furthermore, not only can a single size for the ICMP Echo be set, but a range of sizes can be used to give a more realistic set of packets. One key to troubleshooting with the **ping** command is understanding the various codes the command uses to signify the various responses that it can receive. Table 6-50 lists the various codes that the Cisco IOS Software **ping** command can supply.

Explanation of the Codes That the **ping** *Command Receives in Response to Its ICMP Echo Request*

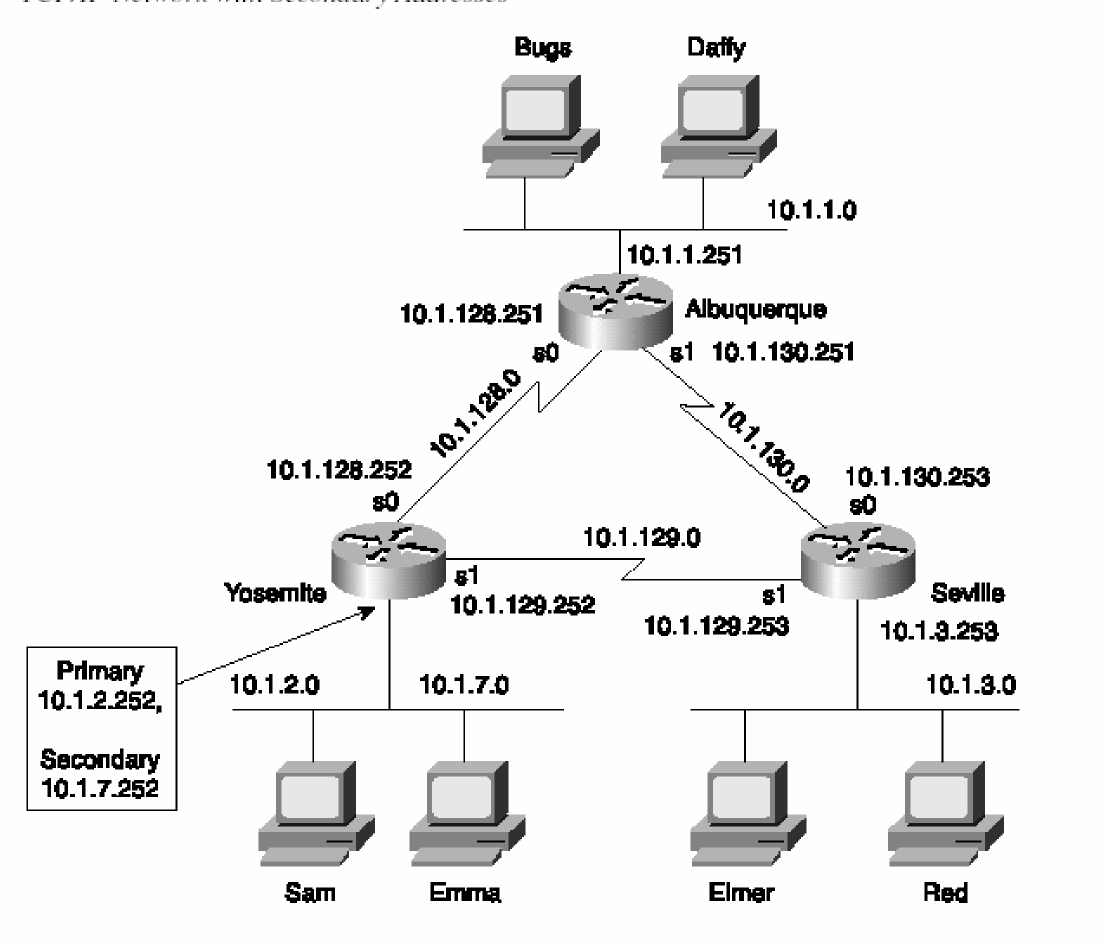


ping Command Code	Explanation
!	ICMP Echo Reply received
.	Nothing received
U	ICMP unreachable (destination) received
N	ICMP unreachable (network) received
P	ICMP unreachable (port) received
Q	ICMP source quench received
M	ICMP Can't Fragment message received
?	Unknown packet received

4.2 Secondary Addresses

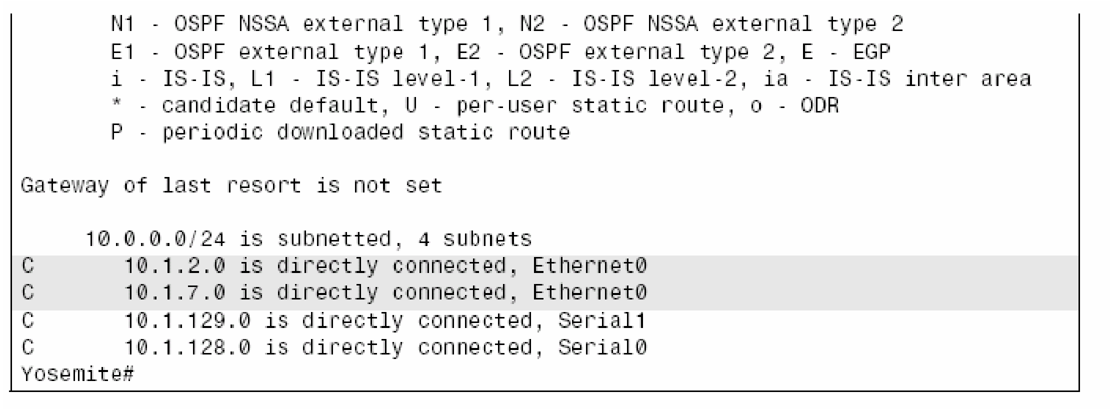
One issue involves when planning IP addresses in a network, what to do when there are no more unassigned IP addresses in a subnet. One alternative solution is to change the mask used on that subnet, making the existing subnet larger. However, changing the mask could cause an overlap. For example, if 10.1.4.0/24 is running out of addresses and you make a change to mask 255.255.254.0 (9 host bits, 23 network/subnet bits), an overlap can occur. 10.1.4.0/23 includes addresses 10.1.4.0 to 10.1.5.255; this is indeed an overlap with a different existing subnet, 10.1.5.0/24. If subnet 10.1.5.0/24 already exists, using 10.1.4.0/23 would not work. Another alternative for continued growth is to place all the existing addresses in the mostly full subnet into another larger subnet. There must be a valid subnet number that is unassigned, that does not create an overlap, and that is larger than the old subnet. However, this solution causes administrative effort to change the IP addresses. In either case, both solutions that do not use secondary addressing imply a strategy of using different masks in different parts of the network. Use of these different masks is called variable-length subnet masking (VLSM), which brings up another set of complex routing protocol issues. The issue of running out of addresses in this subnet can be solved by the use of IP secondary addressing. Secondary addressing uses multiple subnets on the same data link. Secondary IP addressing is simple in concept. Because more than one subnet is used on the same medium, the router needs to have more than one IP address on the interface attached to that medium. For example, Figure 6-35 has subnet 10.1.2.0/24; assume that the subnet has all IP addresses assigned. Assuming secondary addressing to be the chosen solution, subnet 10.1.7.0/24 also could be used on the same Ethernet. example shows the configuration for secondary IP addressing on Yosemite.

TCP/IP Network with Secondary Addresses



Secondary IP Addressing Configuration and **show ip route** *Command on Yosemite*

```
! Excerpt from show running-config follows...
Hostname Yosemite
ip domain-lookup
ip name-server 10.1.1.100 10.1.2.100
interface ethernet 0
ip address 10.1.7.252  255.255.255.0 secondary
ip address 10.1.2.252  255.255.255.0
interface serial 0
ip address 10.1.128.252  255.255.255.0
interface serial 1
ip address 10.1.129.252  255.255.255.0

Yosemite#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```


```
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 4 subnets
C       10.1.2.0 is directly connected, Ethernet0
C       10.1.7.0 is directly connected, Ethernet0
C       10.1.129.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0
Yosemite#
```

The router has routes to subnets 10.1.2.0/24 and 10.1.7.0/24, so it can forward packets to each subnet. The router also can receive packets from hosts in one subnet and can forward the packets to the other subnet using the same interface.

4.3 IP Addressing with Frame Relay Subinterfaces

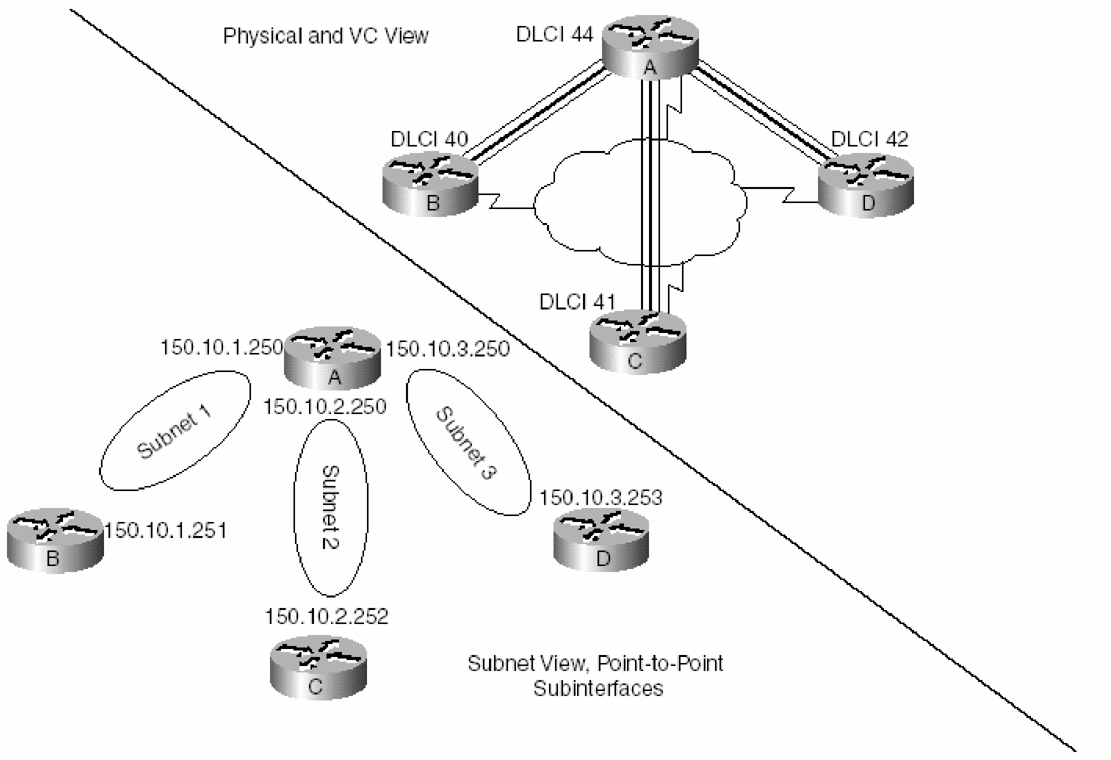
Frame Relay behaves like a WAN in some ways and more like a LAN in other ways. Cisco provides three different ways to configure IP addresses on Frame Relay serial interfaces:

1 Configure the IP addresses on the normal physical interface, just like for other interfaces. By doing so, all routers on the Frame Relay network are in the same subnet.

2 Configure IP addresses on a point-to-point subinterface of the physical interface. A subinterface is a logical subdivision of the physical interface, which, in this case, allows you to correlate a single VC to the point-to-point subinterface. The IP address is configured under the subinterface. Every VC results in a separate subnet.

3 Configure IP addresses on a multipoint subinterface of the physical interface. A subinterface is a logical subdivision of the physical interface, which, in this case, allows you to correlate multiple VCs to the multipoint subinterface.

The following figures give a graphical representation of the first two options for the same network. The following examples show the configurations on routers A, B, C, and D when point-to-point subinterfaces are used, respectively.



Frame Relay Subnets with Point-to-Point Subinterfaces

Router B Configuration

```
hostname routerB
!
interface serial 0
encapsulation frame-relay
!
interface serial 0.1 point-to-point
ip address 150.10.1.251   255.255.255.0
frame-relay interface-dlci 44
description this is for the VC to site A
```

Router C Configuration

```
hostname routerC
!
interface serial 0
encapsulation frame-relay
!
```

continues

Router C Configuration (Continued)

```
interface serial 0.2 point-to-point
ip address 150.10.2.252   255.255.255.0
frame-relay interface-dlci 44
description this is for the VC to site A
```

Router D Configuration

```
hostname routerD
!
interface serial 0
encapsulation frame-relay
!
interface serial 0.3 point-to-point
ip address 150.10.3.253   255.255.255.0
frame-relay interface-dlci 44
description this is for the VC to site A
```

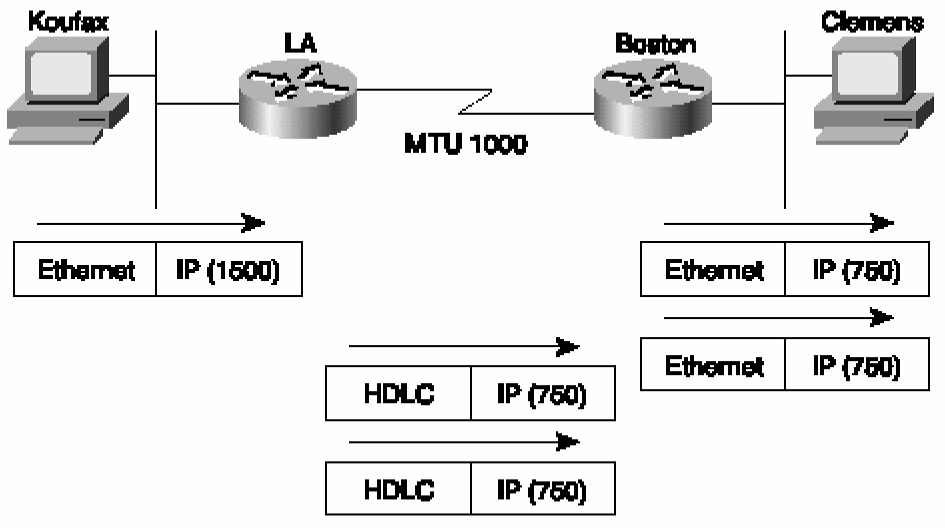
4.4 MTU and Fragmentation

The maximum transmission unit (MTU) is a concept that implies the largest Layer 3 packet that can be forwarded out an interface. The maximum MTU value allowed is based on the data-link protocol; essentially, the maximum size of the data portion of the data-link frame (where the packet is placed) is the maximum setting for the MTU on an interface. The default MTU value on Ethernet and serial interfaces is 1,500.

If an interface’s MTU is smaller than a packet that must be forwarded, fragmentation is performed by the router. *Fragmentation* is the process of simply breaking the packet into smaller packets, each of which is less than or equal to the MTU value. For example, consider the following figure, with a point-to-point serial link whose MTU has been lowered to 1000. As figure illustrates, Koufax threw a 1500-byte packet toward Router LA. LA removed the Ethernet header but could not forward the packet because it was 1500 bytes and the HDLC link supported only an MTU of 1000. LA fragmented the original packet into two packets. After forwarding the two packets, Boston receives the packets and forwards them, without reassembling them—reassembly is done by the endpoint host, which, in this case, is Clemens. The IP header contains fields useful for reassembly of the fragments into the original packet.

The IP header includes an ID value that is the same in each fragmented packet, as well as an offset value that defines which part of the original packet is held in each fragment. Fragmented packets arriving out of order can be identified as part of the same original packet and can be reassembled into the correct order using the offset field in each fragment.

IP Fragmentation



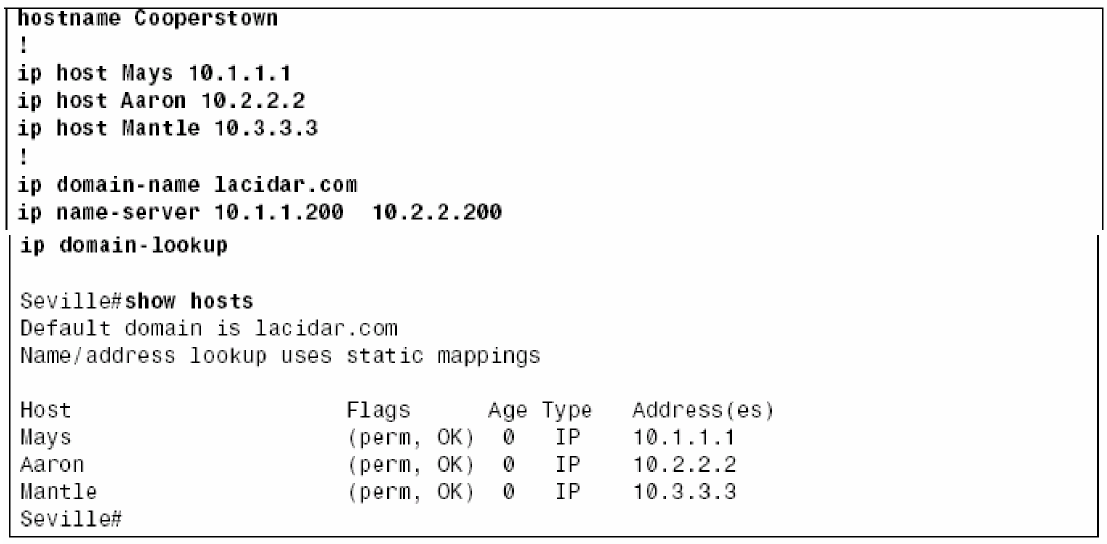
Two configuration commands can be used to change the IP MTU size on an interface: the **mtu** interface subcommand and the **ip mtu** interface subcommand. The **mtu** command sets the MTU for all Layer 3 protocols; unless there is a need to vary the setting per Layer 3 protocol, this command is preferred. If a different setting is desired for IP, the **ip mtu** command sets the value used for IP. A few nuances relate to the two MTU-setting commands. If both are configured on an interface, the IP MTU setting takes precedence on the interface. However, if the **mtu** command is configured after the **ip mtu** is configured, the **ip mtu** value is reset to the same value as that of the **mtu** command. Care must be taken when changing these values.

4.5 IP Naming Commands and Telnet

When using the IOS CLI, you will want to refer to names instead of IP addresses. Particularly for the **trace**, **ping**, and **telnet** commands, the IP address or host name must be supplied. This section describes the use of host names on an IOS-based device. Along the way, some nuances of the use of Telnet are covered.

The IOS can use statically configured names as well as refer to one or more DNSs. The following example shows some names statically configured, with configuration pointing to two different DNSs.

IP Naming Configuration and **show ip host** *Command*



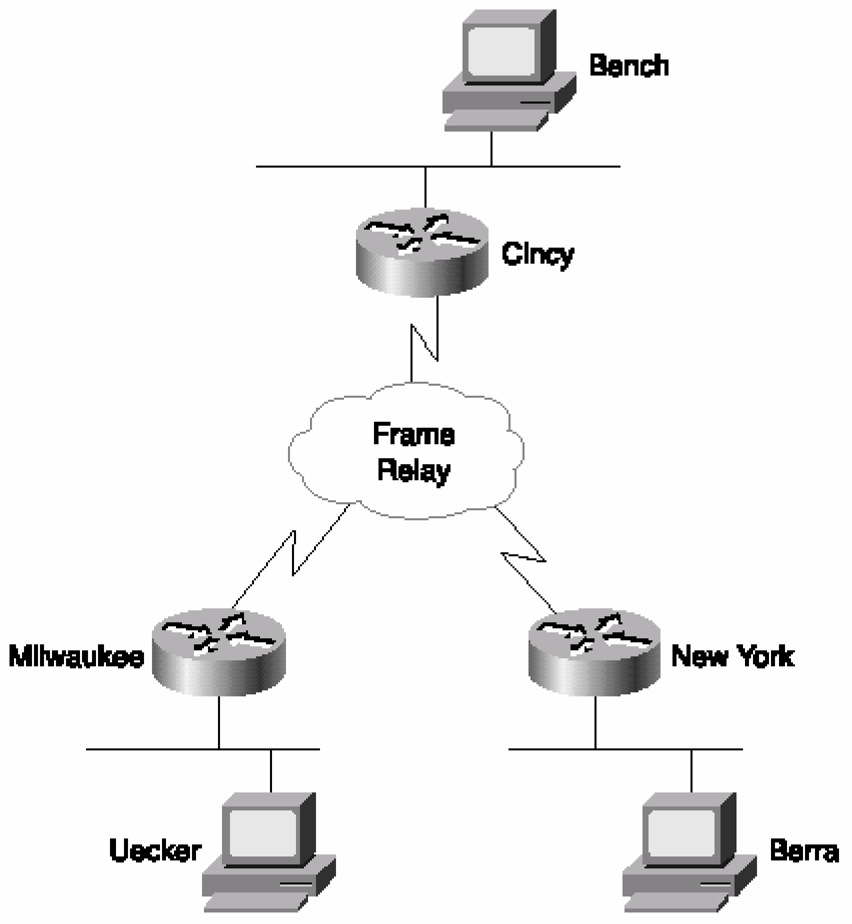
```
hostname Cooperstown
!
ip host Mays 10.1.1.1
ip host Aaron 10.2.2.2
ip host Mantle 10.3.3.3
!
ip domain-name lacidar.com
ip name-server 10.1.1.200  10.2.2.200
ip domain-lookup

Seville#show hosts
Default domain is lacidar.com
Name/address lookup uses static mappings

Host                    Flags      Age Type   Address(es)
Mays                    (perm, OK)  0   IP    10.1.1.1
Aaron                   (perm, OK)  0   IP    10.2.2.2
Mantle                  (perm, OK)  0   IP    10.3.3.3
Seville#
```

Router Cooperstown will use any of the three statically configured host name–to–IP address mappings. Three names are statically configured in this case—Mays, Aaron, and Mantle. Any command referring to Mays, Aaron, or Mantle will resolve into the IP addresses shown in the **ip host** command. Router Cooperstown also will ask a DNS for name resolution if it does not know the name and IP address already. The DNS configuration is shown toward the end of the configuration. The IP addresses of the name servers are shown in the **ip name-server** command. Up to six DNSs can be listed; they are searched for each request sequentially based on the order in the command. Finally, the **ip domain-lookup** command enables IOS to ask a name server. IP domain lookup is the default; **no ip domain-lookup** disables the DNS client function. For names that do not include the full domain name, the **ip domain-name** command defines the domain name that should be assumed by the router. The **show ip host** command lists the static entries, in addition to any entries learned from a DNS request. Only the three static entries were in the table, in this case. The term *perm* in the output implies that the entry is static.

4.6 Telnet and Suspend

The **telnet** IOS exec command allows you to Telnet from one Cisco device to another; in practical use, it is typically to another Cisco device. One of the most important features of the **telnet** command is the *suspend* feature. To understand the suspend function, you will need to refer to the network diagram in following figure.

Telnet Suspension



In the figure, the router administrator is using Bench to Telnet into the Cincy router. When in Cincy, the user Telnets to Milwaukee. When in Milwaukee, the user suspends the Telnet by pressing Ctrl-Shift-6, followed by pressing the letter *x*. The user then Telnets to New York and again suspends the connection. The example begins with Bench already logged into Cincy. Following example shows example output, with annotations to the side.

Telnet Suspensions

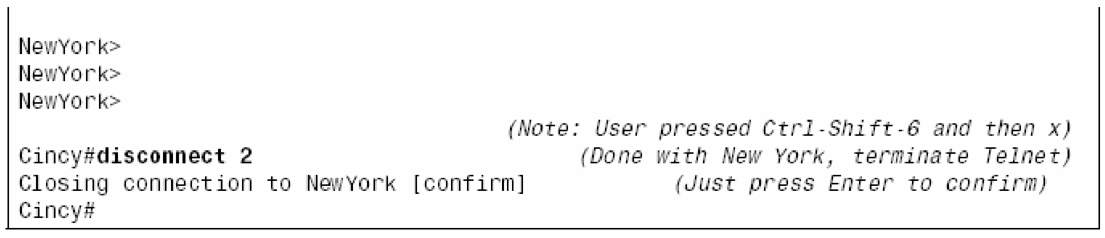
```
Cincy#telnet milwaukee            (User issues command to Telnet to Milwaukee)
Trying Milwaukee (10.1.4.252)... Open


User Access Verification

Password:                 (User plugs in password, can type commands at Milwaukee)
```

continues

Telnet Suspensions (Continued)

```
Milwaukee>
Milwaukee>
Milwaukee>
                         (Note: User pressed Ctrl-Shift-6 and then x)
Cincy#telnet NewYork            (User back at Cincy because Telnet was suspended)
Trying NewYork (10.1.6.253)... Open
               (User is getting into New York now, based on telnet NewYork command)


User Access Verification

Password:
NewYork>                                (User can now type commands on New York)
NewYork>
NewYork>
NewYork>
                                    (Note: User pressed Ctrl-Shift-6 and then x)

Cincy#show sessions               (This command lists suspended Telnet sessions)
Conn Host                Address             Byte  Idle Conn Name
   1 milwaukee           10.1.4.252             0     0 milwaukee
*  2 NewYork             10.1.6.253             0     0 NewYork

Cincy#where                      (where does the same thing)
Conn Host                Address             Byte  Idle Conn Name
   1 milwaukee           10.1.4.252             0     0 milwaukee
*  2 NewYork             10.1.6.253             0     0 NewYork

Cincy#resume 1       (Resume connection 1 (see show session) to Milwaukee)
[Resuming connection 1 to milwaukee ... ]

Milwaukee>                         (User can type commands on Milwaukee)
Milwaukee>
Milwaukee>
(Note: User pressed Ctrl-Shift-6 and then x)    (User wants to go back to Cincy)
Cincy#         (WOW! User just pressed Enter and resumes the last Telnet)
 [Resuming connection 1 to milwaukee ... ]

Milwaukee>
Milwaukee>
Milwaukee>
                                 (Note: User pressed Ctrl-Shift-6 and then x)
                               (Tired of Milwaukee again - can't imagine why!)
Cincy#disconnect 1          (No more need to use Milwaukee - Telnet terminated!)
Closing connection to milwaukee [confirm]         (User presses Enter to confirm)
Cincy#
[Resuming connection 2 to NewYork ... ]
                  (Pressing Enter resumes most recently suspended active Telnet)
```

Telnet Suspensions (Continued)



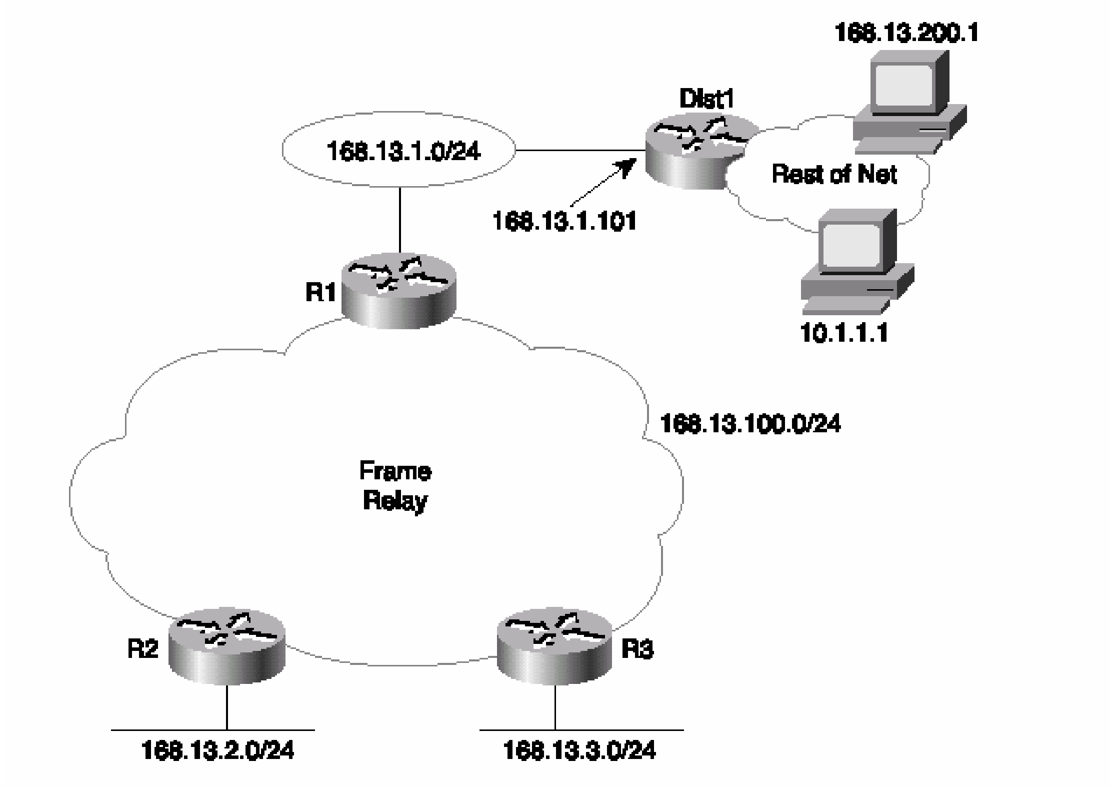
```
NewYork>
NewYork>
NewYork>
                                        (Note: User pressed Ctrl-Shift-6 and then x)
Cincy#disconnect 2                          (Done with New York, terminate Telnet)
Closing connection to NewYork [confirm]          (Just press Enter to confirm)
Cincy#
```

The play-by-play notes in the example explain most of the details. The example begins with the Cincy command prompt that would be seen in Bench's Telnet window because the user at Bench Telnetted into Cincy first. After Telnetting to Milwaukee, the Telnet connection was suspended. Then, after Telnetting to New York, that connection was suspended. The two connections can be suspended or resumed easily. The **resume** command can be used to resume either connection; however, the **resume** command requires a connection ID, which is shown in the **show sessions** command. (The **where** command provides the same output.) The interesting and potentially dangerous nuance here is that if a Telnet session is suspended and you simply press Enter, *Cisco IOS Software resumes the connection to the most recently suspended Telnet connection*. That is fine, until you realize how much you tend to press the Enter key occasionally to clear some of the clutter from the screen. With a suspended Telnet connection, you also just happened to reconnect to another router. This is particularly dangerous when you are changing the configuration or using potentially damaging exec commands—be careful about what router you are actually using when you have suspended Telnet connections!

4.7 Default Routes and the ip classless Command

When a router needs to route a packet and there is no route matching that packet's destination in the routing table, the router discards the packet. Default routing lets the router forward the packet to some default next-hop router. Default routing is that simple! However, two configuration options for default routing make it a little tricky. Also one other option changes the algorithm of how the router decides whether there is a routing table match, which affects when the default route is used. First, default routes work best when there is one path to a part of the network. In following figure, R1, R2, and R3 are connected to the rest of this network only through R1's Token Ring interface. All three routers can forward packets to the rest of the network, as long as the packets get to R1, which forwards them to Dist1.

Example Network Using a Default Route

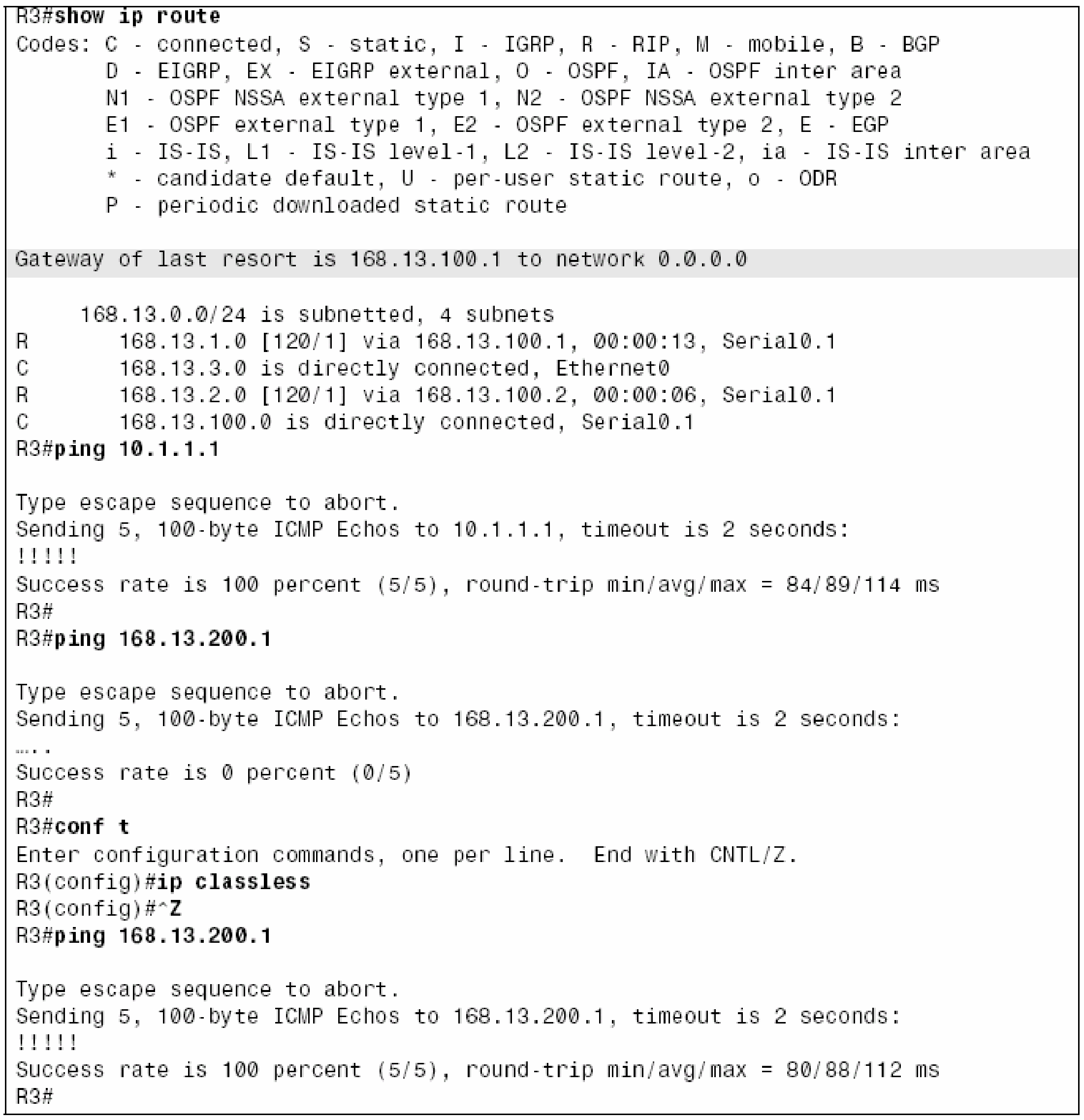


By coding a default route on R1 that points to router Dist1 in figure and having R1 advertise the default to R2 and R3, default routing can be accomplished. Following examples show a default route on R1.

R1 Static Default Route Configuration and Routing Table

```
R1(config)#ip route 0.0.0.0 0.0.0.0 168.13.1.101

R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.1.101 to network 0.0.0.0

     168.13.0.0/24 is subnetted, 4 subnets
C       168.13.1.0 is directly connected, TokenRing0
R       168.13.3.0 [120/1] via 168.13.100.3, 00:00:05, Serial0.1
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:21, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
S*   0.0.0.0/0 [1/0] via 168.13.1.101
R1#
```

R3—Nuances with Successful Use of Static Route on R1



```
R3#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.100.1 to network 0.0.0.0

     168.13.0.0/24 is subnetted, 4 subnets
R       168.13.1.0 [120/1] via 168.13.100.1, 00:00:13, Serial0.1
C       168.13.3.0 is directly connected, Ethernet0
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:06, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
R3#ping 10.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 84/89/114 ms
R3#
R3#ping 168.13.200.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 168.13.200.1, timeout is 2 seconds:
…..
Success rate is 0 percent (0/5)
R3#
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#ip classless
R3(config)#^Z
R3#ping 168.13.200.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 168.13.200.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 80/88/112 ms
R3#
```

The default route is defined with a static **ip route** command on R1, with destination 0.0.0.0, mask 0.0.0.0. This route matches all destinations, by convention. R1 advertises to R2 and R3, as seen in the output of the **show ip route** command on R3 in Example 6-16. So the **ping 10.1.1.1** command on R3 works, just like it should. However, the **ping 168.13.200.1** command does not work—why? The key to knowing why one ping worked and one did not is based on what Cisco IOS Software thinks is a "match" of the routing table. If the router first matches the Class A, B, or C network number that a destination resides in and then looks for the specific subnet, the router is considered to be *classful*. If the router simply looks for the best subnet match, ignoring Class A, B, and C rules, the router is *classless*. What you need in this case is a router with no class!

Seriously, here's R3's logic when the ping failed:

1 I need to send a packet to 168.13.200.1.

2 I match Class B network 168.13.0.0, so there is a match.

3 I do not match a specific subnet that contains 168.13.200.1.

4 I use the default route only if there is no match, and there was a match, so I discard the packet. If you make R3 act in a classless manner, the ping will work, as in the second **ping 168.13.200.1** command in the example. The logic works something like this:

1 I need to send a packet to 168.13.200.1.

2 I do not match a specific subnet that contains 168.13.200.1.

3 I use the default route only if there is no match, and there was no match, so I use the default route.

The **no ip classless** command makes the router behave as a classful router. The **ip classless** command makes it act as a classless router, which would be preferred in this case. It seems like classless would always be better, but it is not. What if the networks on the other side of Dist1 were on the Internet and 168.13.0.0 was your registered Class B network? Well, there should not be any part of 168.13.0.0 to the right of Dist1, so it would be pointless to send packets for unknown subnets of 168.13.0.0 to Dist1 because they will be discarded at some point anyway. There are uses for both modes—just be aware of how each works. The gateway of last resort, highlighted in the **show ip route** command output, sounds like a pretty desperate feature. There are worse things than having to discard a packet in a router, and "gateway of last resort" simply references the current default route. It is possible that several default routes have been configured and then distributed with a routing protocol; the gateway of last resort is the currently used default on a particular router. Be careful—multiple defaults can cause a routing loop.

Another style of configuration for the default route uses the **ip default-network** command. This command is used most typically when you want to reach other Class A, B, or C networks by default, but all the subnets of your own network are expected to be in your own routing tables. For example, imagine that the cloud next to Dist1 in the figure has subnets of network 10.0.0.0 in it as well as other networks. (Dist1 could be an ISP router.) The network in the figure is still in use, but instead of using the **ip route 0.0.0.0 0.0.0.0 168.13.1.101** command, the **ip default-network 10.0.0.0** command is used on R1. R1 uses its route to network 10.0.0.0 as its default and advertises this route as a default route to other routers. The following examples show several details on R1 and R3.

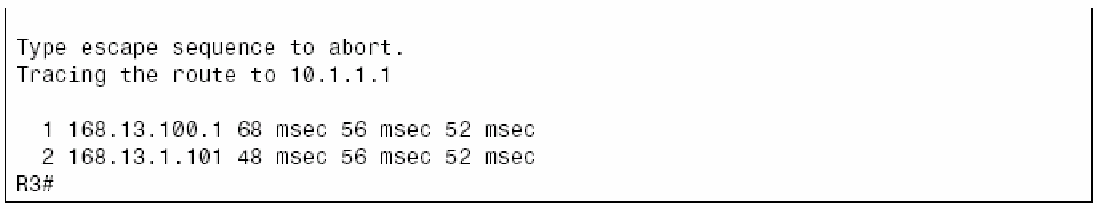
R1's Use of the **ip default-network** *Command*

```
R1#configure terminal
R1(config)#ip default-network 10.0.0.0
R1(config)#exit
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.1.101 to network 10.0.0.0

     168.13.0.0/24 is subnetted, 5 subnets
R       168.13.200.0 [120/1] via 168.13.1.101, 00:00:12, TokenRing0
C       168.13.1.0 is directly connected, TokenRing0
R       168.13.3.0 [120/1] via 168.13.100.3, 00:00:00, Serial0.1
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:00, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
R*   10.0.0.0/8 [120/1] via 168.13.1.101, 00:00:12, TokenRing0
R1#
```

R3 Routing Table and **trace** *Command Samples*

```
R3#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.100.1 to network 0.0.0.0

     168.13.0.0/24 is subnetted, 5 subnets
R       168.13.200.0 [120/2] via 168.13.100.1, 00:00:26, Serial0.1
R       168.13.1.0 [120/1] via 168.13.100.1, 00:00:26, Serial0.1
C       168.13.3.0 is directly connected, Ethernet0
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:18, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
R    10.0.0.0/8 [120/2] via 168.13.100.1, 00:00:26, Serial0.1
R*   0.0.0.0/0 [120/2] via 168.13.100.1, 00:00:26, Serial0.1
R3#trace 168.13.222.2

Type escape sequence to abort.
Tracing the route to 168.13.222.2

  1 168.13.100.1 68 msec 56 msec 52 msec
  2 168.13.1.101 52 msec 56 msec 52 msec
R3#trace 10.1.1.1
```

continues



```
Type escape sequence to abort.
Tracing the route to 10.1.1.1

  1 168.13.100.1 68 msec 56 msec 52 msec
  2 168.13.1.101 48 msec 56 msec 52 msec
R3#
```

Both R1 and R3 have default routes, but they are shown differently in their respective routing tables. R1 shows a route to network 10.0.0.0 with an *, meaning that it is a candidate to be the default route. In R3, 0.0.0.0 shows up in the routing table as the candidate default route. The reason that R3 shows this information differently is that RIP advertises default routes using network number 0.0.0.0. If IGRP or EIGRP were in use, there would be no route to 0.0.0.0 on R3, and network 10.0.0.0 would be the candidate default route. That's because IGRP and EIGRP would flag 10.0.0.0 as a candidate default route in their routing updates rather than advertise the special case of 0.0.0.0. The default route on R3 is used for destinations in network 168.13.0.0, 10.0.0.0, or any other network because **ip classless** is still configured. The **trace** commands in example , which show destinations in two different networks, both succeed. The **trace** commands each show that the first router in the route was R1, then comes Dist1, and then the command finished. If *n* other routers had been present in the network of above figure , these routers could have shown up in the **trace** output as well. (In each case, the destination address was the address of some loopback interface in Dist1, so there were no routers beyond Dist1.) **ip classless** still was configured; it is recommended that you configure **ip classless** if using any form of default routes.

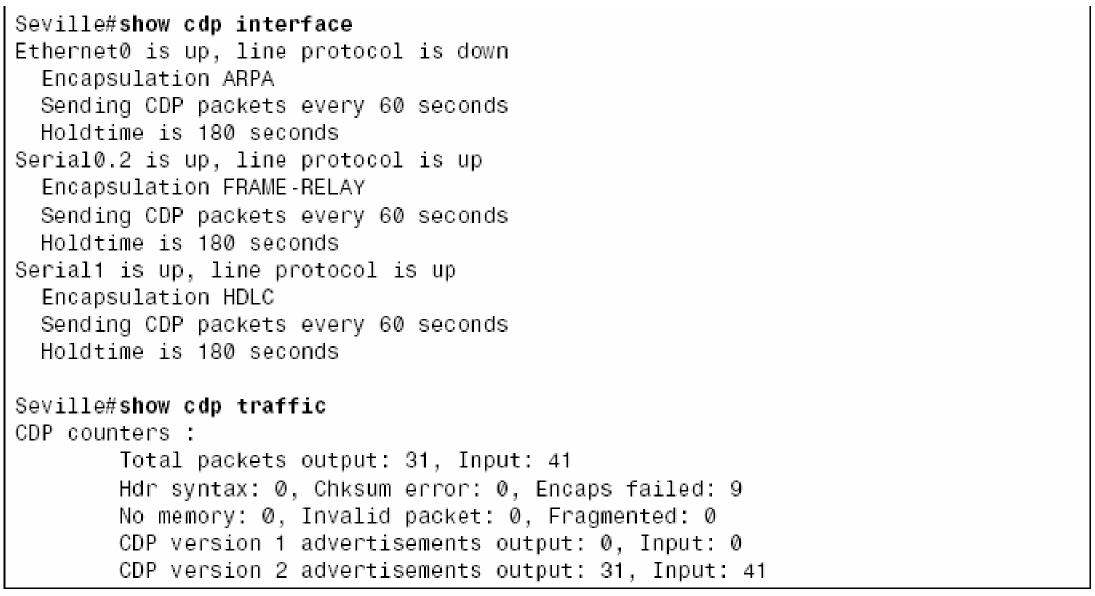
4.8 Cisco Discovery Protocol

The Cisco Discovery Protocol (CDP) discovers basic information about neighboring routers and switches, without needing to know the passwords for the neighboring devices. CDP supports any LAN, HDLC, Frame Relay, and ATM interface. CDP supports any interface that supports the use of SNAP headers. The router or switch can discover Layer 2 and Layer 3 addressing details of neighboring routers without even configuring that Layer 3 protocol—this is because CDP is not dependent on any particular Layer 3 protocol. CDP discovers several useful details from the neighboring device:

• **Device identifier**—Typically the host name
• **Address list**—Network and data-link addresses
• **Port identifier**—Text that identifies the port, which is another name for an interface
• **Capabilities list**—Information on what type of device it is—for instance, a router or a switch
• **Platform**—The model and OS level running in the device CDP is enabled in the configuration by default. The **no cdp run** global command disables CDP
for the entire device, and the **cdp run** global command re-enables CDP. Likewise, the **no cdp enable** interface subcommand disables CDP just on that interface, and the **cdp enable** command switches back to the default state of CDP being enabled. A variety of **show cdp** command options are available. The following example lists the output of the commands, with some commentary.

show cdp *Command Options*

```
Seville#show cdp neighbor
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
fred                Ser 1          172            R        2500      Ser 1
Yosemite            Ser 0.2        161            R        2500      Ser 0.2

Seville#show cdp entry fred
-------------------------
Device ID: fred
Entry address(es):
  IP address: 163.5.8.3
Platform: cisco 2500,  Capabilities: Router
Interface: Serial1,  Port ID (outgoing port): Serial1
Holdtime : 168 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(3), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Wed 18-Jul-01 21:10 by pwade

advertisement version: 2

Seville#show cdp neighbor detail
-------------------------
Device ID: fred
Entry address(es):
  IP address: 163.5.8.3
Platform: cisco 2500,  Capabilities: Router
Interface: Serial1,  Port ID (outgoing port): Serial1
Holdtime : 164 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(3), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Wed 18-Jul-01 21:10 by pwade

advertisement version: 2

-------------------------
Device ID: Yosemite
Entry address(es):
  IP address: 10.1.5.252
  Novell address: 5.0200.bbbb.bbbb
Platform: cisco 2500,  Capabilities: Router
Interface: Serial0.2,  Port ID (outgoing port): Serial0.2
Holdtime : 146 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(3), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Wed 18-Jul-01 21:10 by pwade

advertisement version: 2
```

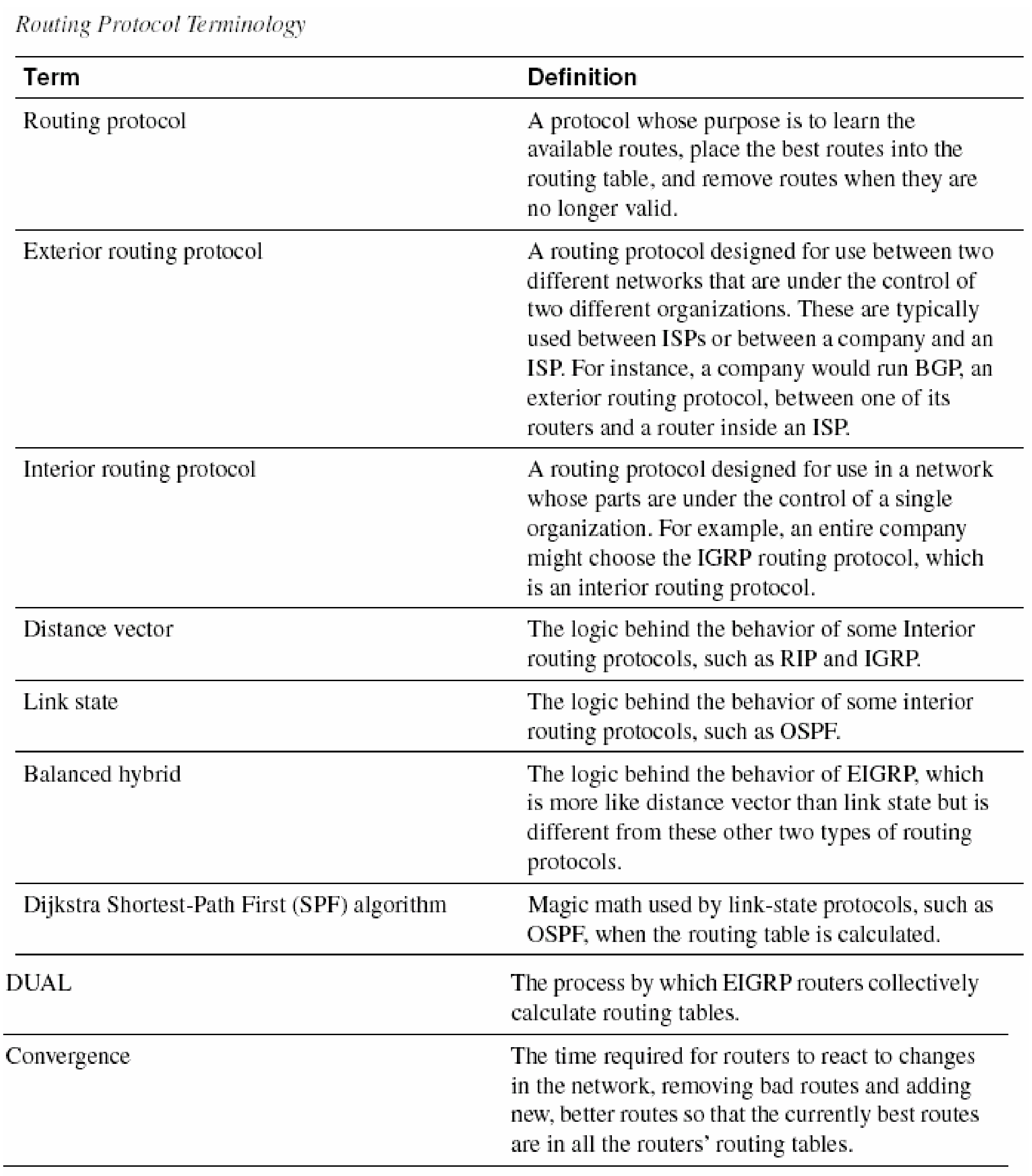


```
Seville#show cdp interface
Ethernet0 is up, line protocol is down
  Encapsulation ARPA
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial0.2 is up, line protocol is up
  Encapsulation FRAME-RELAY
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial1 is up, line protocol is up
  Encapsulation HDLC
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds

Seville#show cdp traffic
CDP counters :
        Total packets output: 31, Input: 41
        Hdr syntax: 0, Chksum error: 0, Encaps failed: 9
        No memory: 0, Invalid packet: 0, Fragmented: 0
        CDP version 1 advertisements output: 0, Input: 0
        CDP version 2 advertisements output: 31, Input: 41
```

The commands provide information about both the neighbors and the behavior of the CDP protocol itself. In the **show cdp entry fred** command in example , all the details learned by CDP are shown and highlighted. To know that fred is the device identifier of a neighbor, the **show cdp neighbor** command can be used to summarize the information about each neighbor. **Show cdp neighbor detail** lists the detail of all neighbors, in the same format as **show cdp entry**. In addition, **show cdp traffic** lists the overhead that CDP introduces to perform its functions.

5 - Configuration of Routing Protocols

5.1 Terminologies and Features



Routing Protocol Terminology

Term	Definition
Routing protocol	A protocol whose purpose is to learn the available routes, place the best routes into the routing table, and remove routes when they are no longer valid.
Exterior routing protocol	A routing protocol designed for use between two different networks that are under the control of two different organizations. These are typically used between ISPs or between a company and an ISP. For instance, a company would run BGP, an exterior routing protocol, between one of its routers and a router inside an ISP.
Interior routing protocol	A routing protocol designed for use in a network whose parts are under the control of a single organization. For example, an entire company might choose the IGRP routing protocol, which is an interior routing protocol.
Distance vector	The logic behind the behavior of some Interior routing protocols, such as RIP and IGRP.
Link state	The logic behind the behavior of some interior routing protocols, such as OSPF.
Balanced hybrid	The logic behind the behavior of EIGRP, which is more like distance vector than link state but is different from these other two types of routing protocols.
Dijkstra Shortest-Path First (SPF) algorithm	Magic math used by link-state protocols, such as OSPF, when the routing table is calculated.
DUAL	The process by which EIGRP routers collectively calculate routing tables.
Convergence	The time required for routers to react to changes in the network, removing bad routes and adding new, better routes so that the currently best routes are in all the routers' routing tables.

The following table lists interior IP routing protocols and their types. A column referring to whether the routing protocol includes subnet mask information in the routing updates is listed for future reference.

Interior IP Routing Protocols and Types

Routing Protocol	Type	Loop-Prevention Mechanisms	Mask Sent in Updates, Which Allows VLSM?
RIP-1	Distance vector	Hold-down timer, split horizon	No
RIP-2	Distance vector	Hold-down timer, split horizon	Yes
IGRP	Distance vector	Hold-down timer, split horizon	No
EIGRP	Balanced hybrid	DUAL and feasible successors	Yes
OSPF	Link-state	Dijkstra SPF algorithm and full topology knowledge	Yes

Issues Relating to Distance Vector Routing Protocols in a Network with Multiple Paths

Issue	Solution
Multiple routes to the same subnet have equal metrics	Implementation options involve either using the first route learned or putting multiple routes to the same subnet in the routing table.
Routing loops occur due to updates passing each other over a single link	**Split horizon**—The routing protocol advertises routes out an interface only if they were not learned from updates entering that interface. **Split horizon with poison reverse**—The routing protocol uses split-horizon rules unless a route fails. In that case, the route is advertised out all interfaces, but with infinite-distance metrics.
Routing loops occur due to updates passing each other over alternative paths	**Route poisoning**—When a route to a subnet fails, the subnet is advertised with an infinite-distance metric.
Counting to infinity	**Hold-down timer**—After finding out that a route to a subnet has failed, a router waits a certain period of time before believing any other routing information about that subnet. **Triggered updates**—When a route fails, an update is sent immediately rather than waiting on the update timer to expire. Used in conjunction with route poisoning, this ensures that all routers know of failed routes before any hold-down timers can expire.

RIP and IGRP Feature Comparison

Feature	RIP (Default)	IGRP (Default)
Update timer	30 seconds	90 seconds
Metric	Hop count	Function of bandwidth and delay (the default). Can include reliability, load, and MTU.
Hold-down timer	180	280
Flash (triggered) updates	Yes	Yes
Mask sent in update	No for RIP-1; yes for RIP-2	No
Infinite-metric value	16	4,294,967,295

5.2 Routing Protocol Commands

IP RIP and IGRP Configuration Commands

Command	Configuration Mode
router rip	Global
router igrp *as-number*	Global
network *net-number*	Router subcommand
passive-interface [**default**] {*interface-type interface-number*}	Router subcommand
maximum-paths *number-paths*	Router subcommand
variance *multiplier*	Router subcommand
traffic-share {*balanced* | *min*}	Router subcommand

IP RIP and IGRP EXEC Commands

Command	Description
show ip route [*ip-address* [*mask*] [**longer-prefixes**]] | [*protocol* [*process-id*]]	Shows the entire routing table, or a subset if parameters are entered.
show ip protocols	Shows routing protocol parameters and current timer values.
debug ip rip	Issues log messages for each RIP update.
debug ip igrp transactions [*ip-address*]	Issues log messages with details of the IGRP updates.
debug ip igrp events [*ip-address*]	Issues log messages for each IGRP packet.
ping [*protocol* | **tag**] {*host-name* | *system-address*}	Sends and receives ICMP echo messages to verify connectivity.
trace [*protocol*] [*destination*]	Sends a series of ICMP echoes with increasing TTL values to verify the current route to a host.

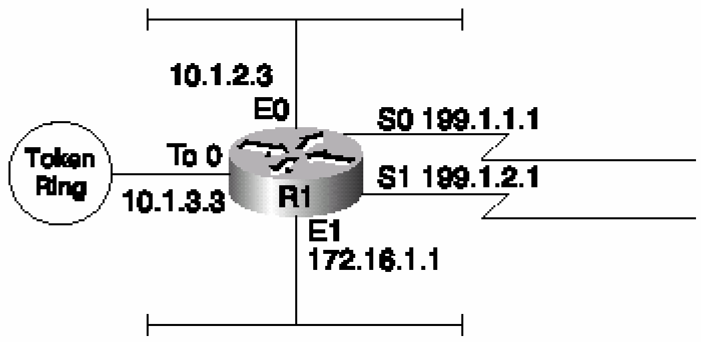
5.3 Basic RIP and IGRP Configuration

Each **network** command enables RIP or IGRP on a set of interfaces. You must understand the subtleties of the **network** command, as explained in this section. However, what “enables” really means in this case is not obvious from the Cisco IOS Software documentation. Also, the parameters for the **network** command are not intuitive to many people who are new to Cisco IOS configuration commands. Therefore, routing protocol configuration, including the **network** command, is a likely topic for tricky questions on the exam. The **network** command “matches” one or more interfaces on a router. For each interface, the **network** command causes the router to do three things:
• The router broadcasts or multicasts routing updates out an interface.
• The router listens for incoming updates on that same interface.
• The router, when sending an update, includes the subnet off that interface in the routing update.

All you need to know is how to match interfaces using the **network** command. The router matches interfaces with the **network** command by asking this simple question: Which of my interfaces have IP addresses with the same network number referenced in this **network** subcommand?

For all interfaces that match the **network** command, the router does the three things just listed. Examples give you a much better understanding of the **network** command, so examine the following figure.

Sample Router with Five Interfaces

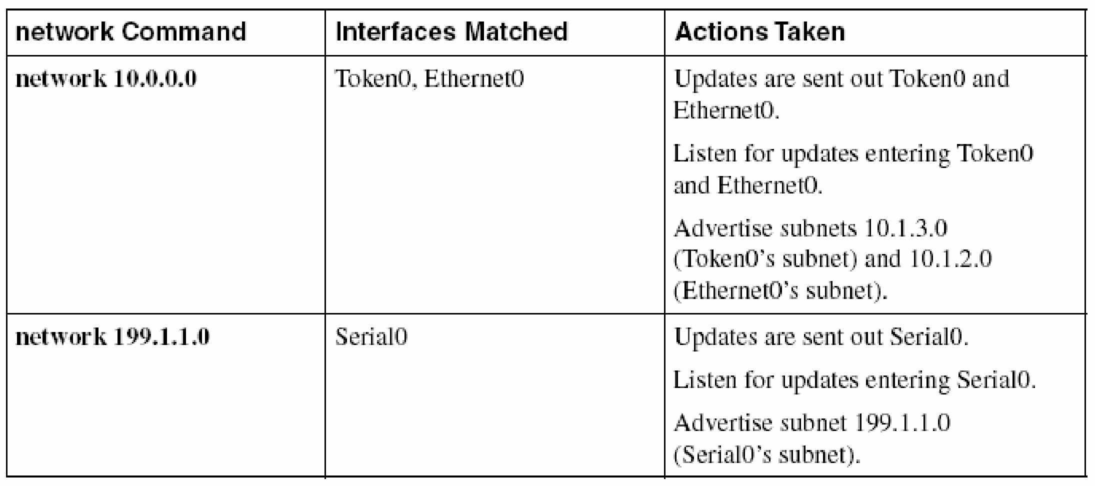


Sample Router Configuration with RIP Partially Enabled

```
interface ethernet 0
ip address 10.1.2.3 255.255.255.0
interface ethernet 1
ip address 172.16.1.1 255.255.255.0
interface tokenring 0
ip address 10.1.3.3 255.255.255.0
interface serial 0
ip address 199.1.1.1 255.255.255.0
interface serial 1
ip address 199.1.2.1 255.255.255.0
!
router rip
network 10.0.0.0
network 199.1.1.0
```

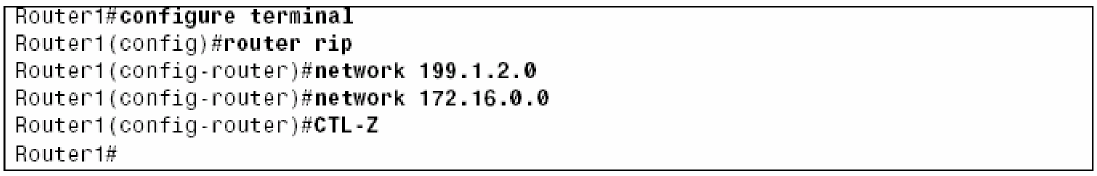
The RIP configuration includes three commands in this case. The **router rip** global command moves the user from global configuration mode to RIP configuration mode. Then, two network commands appear, each with a different Class A, B, or C network number. So what interfaces were matched, and what did this accomplish? Well, if the goal is to enable RIP on *all* interfaces, the configuration is incomplete. Following table summarizes what this configuration accomplishes and what it does not.

What Happens with the RIP Configuration Shown in Example 7-1



network Command	Interfaces Matched	Actions Taken
network 10.0.0.0	Token0, Ethernet0	Updates are sent out Token0 and Ethernet0. Listen for updates entering Token0 and Ethernet0. Advertise subnets 10.1.3.0 (Token0's subnet) and 10.1.2.0 (Ethernet0's subnet).
network 199.1.1.0	Serial0	Updates are sent out Serial0. Listen for updates entering Serial0. Advertise subnet 199.1.1.0 (Serial0's subnet).

For any interfaces that have IP addresses with the same network number referenced in this **network** subcommand, routing updates are broadcast and listened for, and the connected subnet is advertised. The **network** command requires a network number, not a subnet number, for the parameter. Interestingly, you can type a subnet number in the command, and the Cisco IOS Software changes the parameter to the network number in which that subnet resides. If the goal was to configure RIP for all interfaces, a common mistake was made in this example. No **network** command matches interfaces Serial1 and Ethernet1. Following example shows the configuration process to add the additional network commands.



```
Router1#configure terminal
Router1(config)#router rip
Router1(config-router)#network 199.1.2.0
Router1(config-router)#network 172.16.0.0
Router1(config-router)#CTL-Z
Router1#
```

5.3.1 IGRP Configuration

You configure IGRP just like RIP, except that the **router igrp** command has an additional parameter—the AS number. All that is needed is for all routers to use the same process-id in order for IGRP to work. In following example , a complete sample IGRP configuration causes the router to advertise all connected subnets, to listen on all interfaces for IGRP updates, and to advertise on all interfaces.

Sample IGRP Configuration and **show ip route** *Command Output*

```
interface ethernet 0
ip address 10.1.2.3 255.255.255.0
interface ethernet 1
ip address 172.16.1.1 255.255.255.0
interface tokenring 0
ip address 10.1.3.3 255.255.255.0
interface serial 0
ip address 199.1.1.1 255.255.255.0
interface serial 1
ip address 199.1.2.1 255.255.255.0
!
router igrp 1
 network 10.0.0.0
 network 199.1.1.0
 network 199.1.2.0
 network 172.16.0.0

Router1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
C       10.1.3.0 is directly connected, TokenRing0
C       10.1.2.0 is directly connected, Ethernet0
  I       10.1.4.0 [100/8539] via 10.1.2.14, 00:00:50, Ethernet0
     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Ethernet1
I       172.16.2.0 [100/6244] via 172.16.1.44, 00:00:20, Ethernet1
C    199.1.1.0/24 is directly connected, Serial0
C    199.1.2.0/24 is directly connected, Serial1
```

IGRP configuration begins with the **router igrp 1** global configuration command. Then, four consecutive network commands match all the interfaces on the router, so that IGRP is fully enabled. In fact, the **network** commands are identical to the **network** commands in the complete RIP configuration.

IGRP Metrics

IGRP uses a composite metric. This metric is calculated as a function of bandwidth, delay, load, and reliability. By default, only bandwidth and delay are considered; the other parameters are considered only if they are enabled via configuration. Delay and bandwidth are not measured values but are set via the **delay** and **bandwidth** interface subcommands. (The same formula is used to calculate the metric for EIGRP, but with a scaling factor so that the actual metric values are larger, allowing more granularity in the metric.)

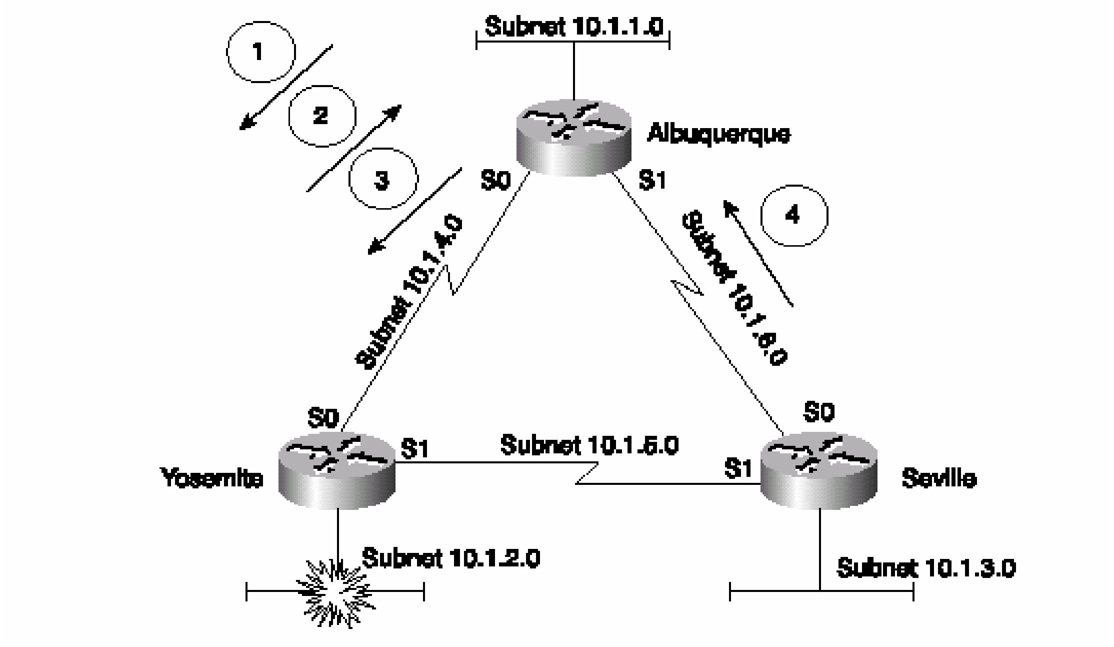
The **show ip route** command in the example shows the IGRP metric values in brackets. For example, the route to 10.1.4.0 shows the value [100/8539] beside the subnet number. The metric 8539 is a single value, as calculated based on bandwidth and delay. The metric is calculated (by default) as the sum of the inverse of the minimum bandwidth, plus the cumulative delay on all links in the route. *In other* *words, the higher the bandwidth, the lower the metric; the lower the cumulative delay, the lower the metric.*

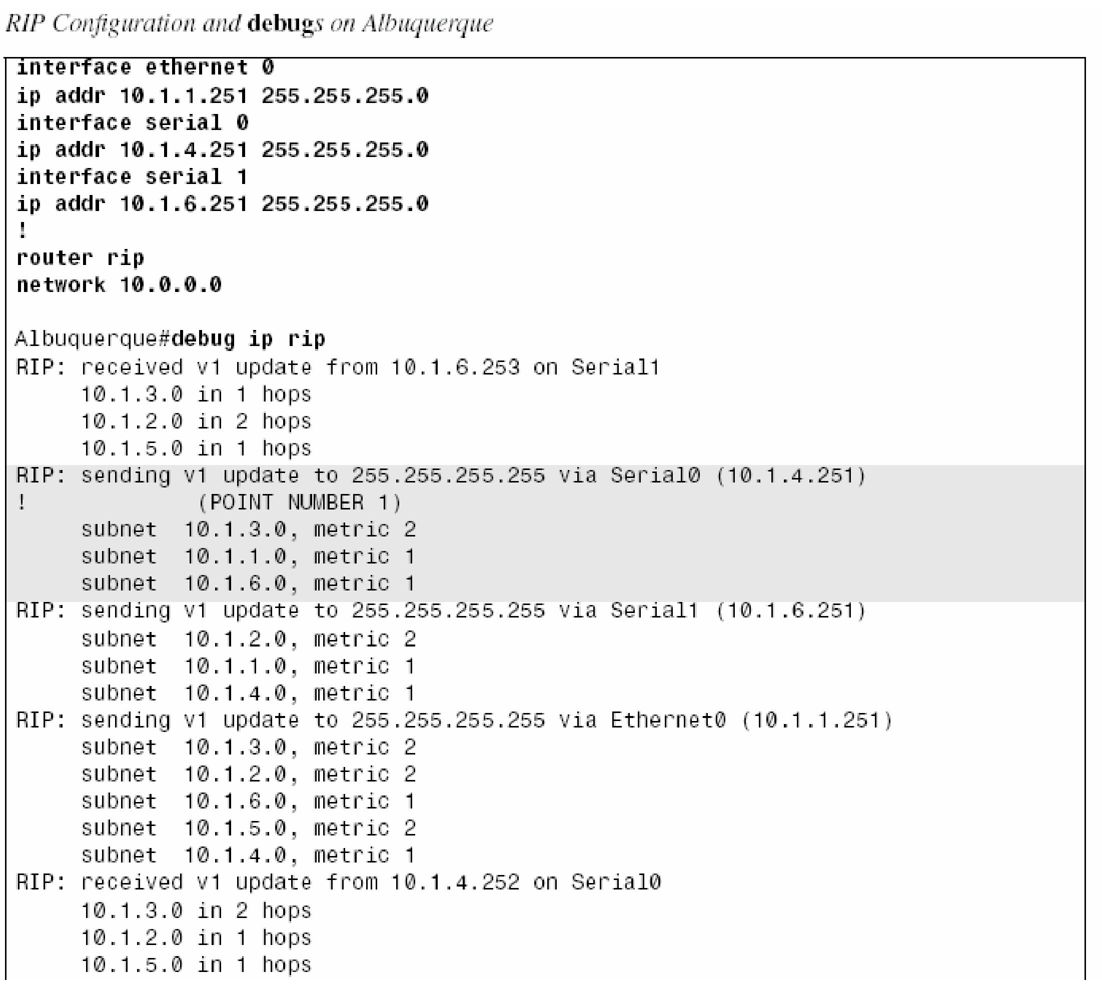
5.4 Examination of RIP and IGRP debug and show Commands

This section on basic RIP and IGRP configuration closes with one more sample network that is first configured with RIP and then is configured with IGRP. Advanced distance vector protocol concepts, such as split horizon and route poisoning, become more obvious when you look at these examples. RIP and IGRP implement split horizon and route poisoning. You can better understand them by examining the upcoming **debug** messages.

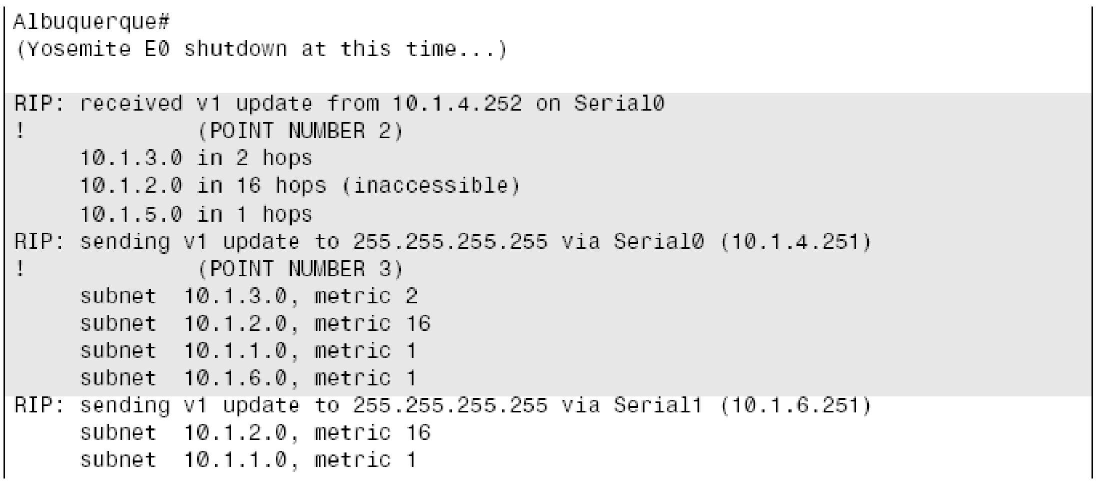
Following , figure and example show a stable RIP network with split-horizon rules that affect the RIP updates. Then Ethernet 0 on Yosemite is shut down, and Yosemite advertises an infinite distance route to 10.1.2.0 because route poisoning is in effect, as shown in other example. The numbered arrows in the figure represent routing updates. The numbers are referred to with comments inside example.

Sample Three-Router Network with Subnet 10.1.2.0 Failing




RIP Configuration and **debugs** *on Albuquerque*

```
interface ethernet 0
ip addr 10.1.1.251 255.255.255.0
interface serial 0
ip addr 10.1.4.251 255.255.255.0
interface serial 1
ip addr 10.1.6.251 255.255.255.0
!
router rip
network 10.0.0.0

Albuquerque#debug ip rip
RIP: received v1 update from 10.1.6.253 on Serial1
     10.1.3.0 in 1 hops
     10.1.2.0 in 2 hops
     10.1.5.0 in 1 hops
RIP: sending v1 update to 255.255.255.255 via Serial0 (10.1.4.251)
!             (POINT NUMBER 1)
     subnet  10.1.3.0, metric 2
     subnet  10.1.1.0, metric 1
     subnet  10.1.6.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Serial1 (10.1.6.251)
     subnet  10.1.2.0, metric 2
     subnet  10.1.1.0, metric 1
     subnet  10.1.4.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (10.1.1.251)
     subnet  10.1.3.0, metric 2
     subnet  10.1.2.0, metric 2
     subnet  10.1.6.0, metric 1
     subnet  10.1.5.0, metric 2
     subnet  10.1.4.0, metric 1
RIP: received v1 update from 10.1.4.252 on Serial0
     10.1.3.0 in 2 hops
     10.1.2.0 in 1 hops
     10.1.5.0 in 1 hops
```



```
Albuquerque#
(Yosemite E0 shutdown at this time...)

RIP: received v1 update from 10.1.4.252 on Serial0
!             (POINT NUMBER 2)
     10.1.3.0 in 2 hops
     10.1.2.0 in 16 hops (inaccessible)
     10.1.5.0 in 1 hops
RIP: sending v1 update to 255.255.255.255 via Serial0 (10.1.4.251)
!             (POINT NUMBER 3)
     subnet  10.1.3.0, metric 2
     subnet  10.1.2.0, metric 16
     subnet  10.1.1.0, metric 1
     subnet  10.1.6.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Serial1 (10.1.6.251)
     subnet  10.1.2.0, metric 16
     subnet  10.1.1.0, metric 1
     subnet  10.1.4.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (10.1.1.251)
     subnet  10.1.3.0, metric 2
     subnet  10.1.2.0, metric 16
     subnet  10.1.6.0, metric 1
     subnet  10.1.5.0, metric 2
     subnet  10.1.4.0, metric 1
RIP: received v1 update from 10.1.6.253 on Serial1
!              (POINT NUMBER 4)
     10.1.3.0 in 1 hops
     10.1.2.0 in 16 hops (inaccessible)
     10.1.5.0 in 1 hops
```

RIP Configuration on Yosemite

```
interface ethernet 0
ip addr 10.1.2.252 255.255.255.0
interface serial 0
ip addr 10.1.4.252 255.255.255.0
interface serial 1
ip addr 10.1.5.252 255.255.255.0

router rip
network 10.0.0.0
```

First, examine the configuration on Albuquerque and Yosemite. Because all interfaces on each router are part of network 10.0.0.0, RIP needs only a single **network** command on each router, so the configuration is relatively easy. For the rest of the explanation, refer to the phrase “Point Number *X*”.

The following list describes what happens at each point in the process:

• **Point Number 1**—Albuquerque sends an update out Serial0, obeying split-horizon rules. Notice that 10.1.2.0, Yosemite’s Ethernet subnet, is not in the update sent out Albuquerque’s S0 interface.

• **Point Number 2**—This point begins right after Yosemite’s E0 has been shut down, simulating a failure. Albuquerque receives an update from Yosemite, entering Albuquerque’s S0 interface. The route to 10.1.2.0 has an infinite metric, which, in this case, is 16.

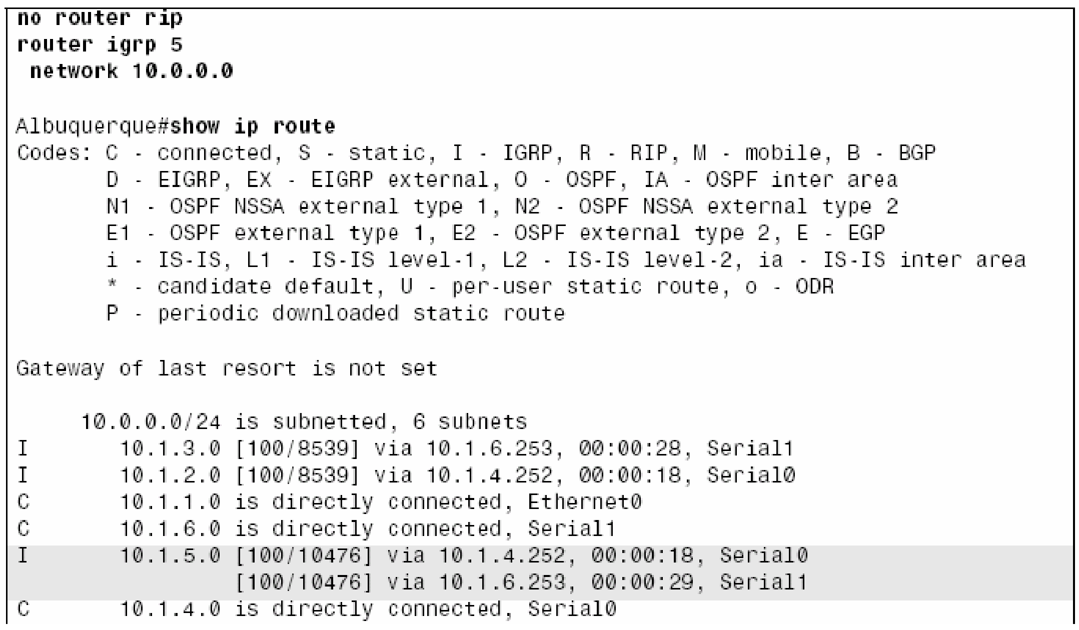
• **Point Number 3**—Albuquerque formerly did not mention subnet 10.1.2.0 due to split horizon rules (point 1). The update at point 3 includes a poisoned route for 10.1.2.0 with metric 16. This is an example of split horizon with poison reverse.

• **Point Number 4**—Albuquerque receives an update in S1 from Seville. The update includes a metric 16 (infinite) route to 10.1.2.0. Seville does not suspend any split-horizon rules in order to send this route because it saw the advertisement of that route earlier, so this is a simple case of route poisoning.

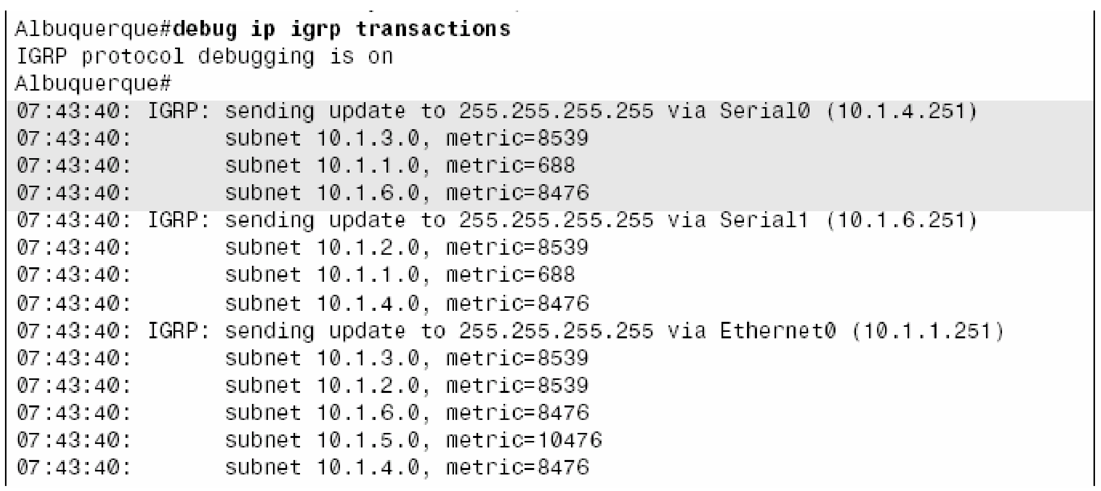
Example shows the steps needed to migrate to IGRP. It also lists some **debug** and **show** commands. Example lists the configuration added to each of the three routers shown in figure to migrate to IGRP. The logic of the **network** commands works just like with RIP. The output of the **show** and **debug** commands provides some insight into the differences between RIP and IGRP.

The following configuration commands would be used on all three routers.

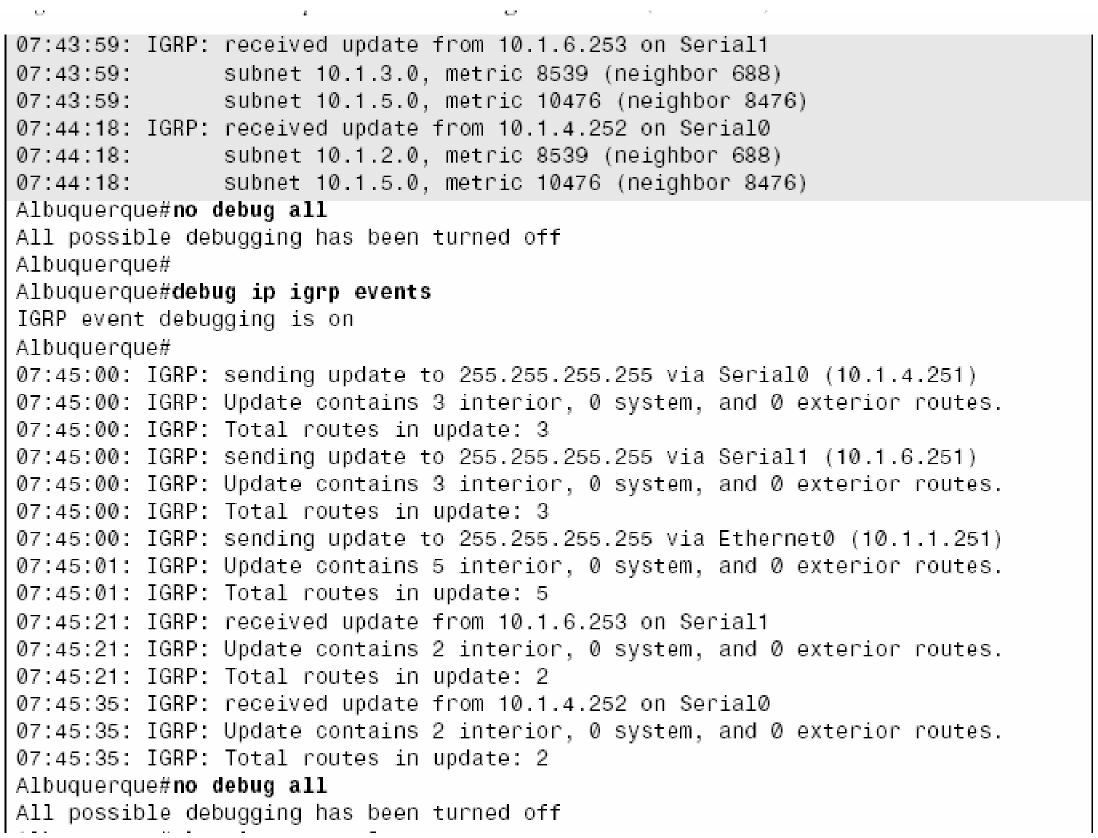
Migration to IGRP with Sample **show** *and* **debug** *Commands*



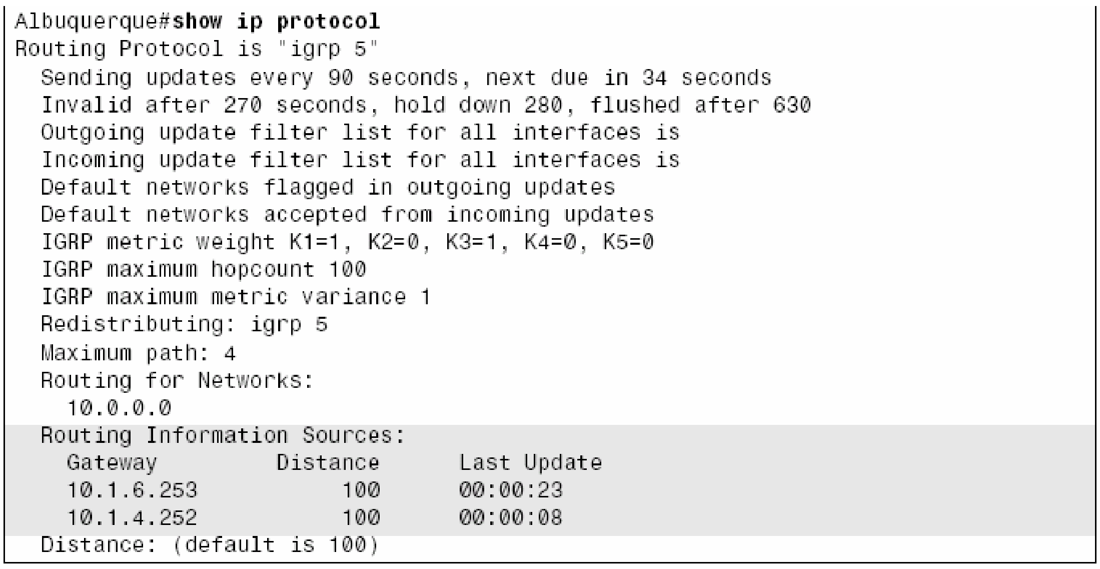
```
no router rip
router igrp 5
 network 10.0.0.0

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

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 6 subnets
I       10.1.3.0 [100/8539] via 10.1.6.253, 00:00:28, Serial1
I       10.1.2.0 [100/8539] via 10.1.4.252, 00:00:18, Serial0
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.6.0 is directly connected, Serial1
I       10.1.5.0 [100/10476] via 10.1.4.252, 00:00:18, Serial0
                 [100/10476] via 10.1.6.253, 00:00:29, Serial1
C       10.1.4.0 is directly connected, Serial0
```



```
Albuquerque#debug ip igrp transactions
IGRP protocol debugging is on
Albuquerque#
07:43:40: IGRP: sending update to 255.255.255.255 via Serial0 (10.1.4.251)
07:43:40:       subnet 10.1.3.0, metric=8539
07:43:40:       subnet 10.1.1.0, metric=688
07:43:40:       subnet 10.1.6.0, metric=8476
07:43:40: IGRP: sending update to 255.255.255.255 via Serial1 (10.1.6.251)
07:43:40:       subnet 10.1.2.0, metric=8539
07:43:40:       subnet 10.1.1.0, metric=688
07:43:40:       subnet 10.1.4.0, metric=8476
07:43:40: IGRP: sending update to 255.255.255.255 via Ethernet0 (10.1.1.251)
07:43:40:       subnet 10.1.3.0, metric=8539
07:43:40:       subnet 10.1.2.0, metric=8539
07:43:40:       subnet 10.1.6.0, metric=8476
07:43:40:       subnet 10.1.5.0, metric=10476
07:43:40:       subnet 10.1.4.0, metric=8476
```



```
07:43:59: IGRP: received update from 10.1.6.253 on Serial1
07:43:59:       subnet 10.1.3.0, metric 8539 (neighbor 688)
07:43:59:       subnet 10.1.5.0, metric 10476 (neighbor 8476)
07:44:18: IGRP: received update from 10.1.4.252 on Serial0
07:44:18:       subnet 10.1.2.0, metric 8539 (neighbor 688)
07:44:18:       subnet 10.1.5.0, metric 10476 (neighbor 8476)
Albuquerque#no debug all
All possible debugging has been turned off
Albuquerque#
Albuquerque#debug ip igrp events
IGRP event debugging is on
Albuquerque#
07:45:00: IGRP: sending update to 255.255.255.255 via Serial0 (10.1.4.251)
07:45:00: IGRP: Update contains 3 interior, 0 system, and 0 exterior routes.
07:45:00: IGRP: Total routes in update: 3
07:45:00: IGRP: sending update to 255.255.255.255 via Serial1 (10.1.6.251)
07:45:00: IGRP: Update contains 3 interior, 0 system, and 0 exterior routes.
07:45:00: IGRP: Total routes in update: 3
07:45:00: IGRP: sending update to 255.255.255.255 via Ethernet0 (10.1.1.251)
07:45:01: IGRP: Update contains 5 interior, 0 system, and 0 exterior routes.
07:45:01: IGRP: Total routes in update: 5
07:45:21: IGRP: received update from 10.1.6.253 on Serial1
07:45:21: IGRP: Update contains 2 interior, 0 system, and 0 exterior routes.
07:45:21: IGRP: Total routes in update: 2
07:45:35: IGRP: received update from 10.1.4.252 on Serial0
07:45:35: IGRP: Update contains 2 interior, 0 system, and 0 exterior routes.
07:45:35: IGRP: Total routes in update: 2
Albuquerque#no debug all
All possible debugging has been turned off
```



```
Albuquerque#show ip protocol
Routing Protocol is "igrp 5"
  Sending updates every 90 seconds, next due in 34 seconds
  Invalid after 270 seconds, hold down 280, flushed after 630
  Outgoing update filter list for all interfaces is
  Incoming update filter list for all interfaces is
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  IGRP maximum hopcount 100
  IGRP maximum metric variance 1
  Redistributing: igrp 5
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.1.6.253           100      00:00:23
    10.1.4.252           100      00:00:08
  Distance: (default is 100)
```

You can migrate from RIP to IGRP in this case with only three configuration commands per router. As highlighted in above example , the **no router rip** command removes all RIP configuration on the router, including any **network** subcommands. The three routers each must use the same IGRP process-id (5, in this case), and because all interfaces on each of the routers are in network 10.0.0.0, only a single **network 10.0.0.0** subcommand is needed.

The **show ip route** command provides the most direct view into what a routing protocol does. First, the legend at the beginning of the **show ip route** command in example defines the letter codes that identify the source of the routing information—for example, **C** for connected routes, **R** for RIP, and **I** for IGRP. Each of the Class A, B, and C networks is listed, along with each of the subnets of that network. If a static mask is used within that network, the mask is shown only in the line referring to the network. Each routing entry lists the subnet number and the outgoing interface. In most cases, the next-hop router's IP address is also listed. IGRP learns the same routes that RIP learned, but using different metrics. The output of the **show ip route** command lists six subnets, just as it did when RIP was used. Also, notice the two routes to 10.1.5.0/24—one through Yosemite and one through Seville. Both routes are included in the routing table, because the default setting for **ip maximum-paths** is 4, and because the routes have an equal metric. Looking further into the output of the **debug ip igrp transactions** command, you can see the equal-cost routes being advertised. One route is seen in the update received on Serial1; the other route in the update is received on Serial0. The output of the **debug ip igrp transactions** command shows the details of the routing updates, whereas the **debug ip igrp events** command simply mentions that routing updates have been received. Finally, the **show ip protocol** command lists several important details about the routing protocol. The update timer is listed, shown with the time remaining until the next routing update is to be sent. Also, the elapsed time since an update was received from each neighboring router is listed at the end of the output. This command also lists each of the neighbors from which routing information has been received. If you are in doubt as to whether updates have been received during the recent past and from what routers, the **show ip protocol** command is the place to find out.

5.5 Advanced RIP and IGRP Configuration

RIP-1 and IGRP do not transmit the subnet mask in the routing updates, as seen in the **debug** output in earlier examples. Cisco expects you to be able to explain why routing protocols that do not transmit a mask can have problems in some networks. This section explains these problems, which all originate from the same root cause.

NOTE Routers must assume the subnet mask that should be used with a subnet number listed in a routing update. Routing protocols that do not transmit masks, such as RIP and IGRP, behave predictably:

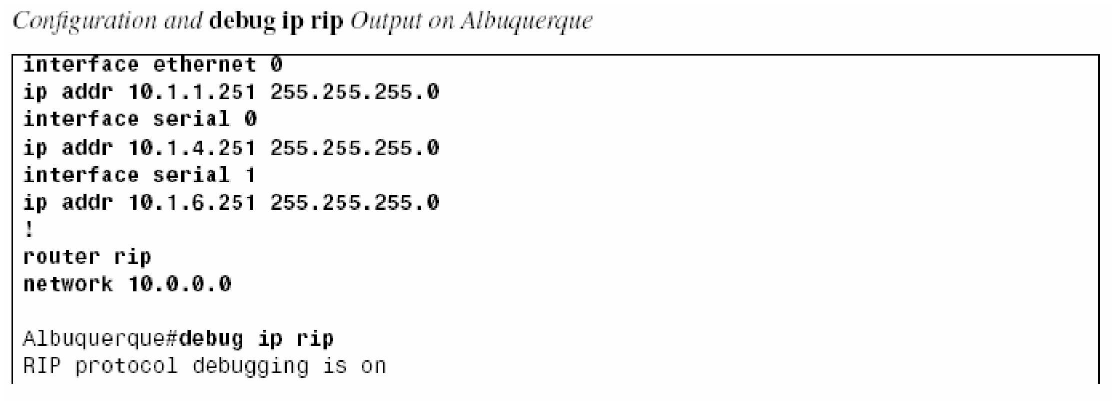
• Updates sent out an interface in network X, when containing routes about subnets ofnetwork X, contain the subnet numbers of the subnets of network X but not the corresponding masks.

• Updates sent out an interface in network X, when containing routes about subnets of network Y, contain one route about the entire network Y, but not any routes about subnets of network Y.

• When receiving a routing update containing routes referencing subnets of network X, the receiving router assumes that the mask in use is the same mask it uses on an interface with an address in network X.

• When receiving an update about network X, if the receiving router has no interfaces in network X, it treats the route as a route to the entire Class A, B, or C network X.

Following examples contain **show** and **debug** command output on Albuquerque, Yosemite, and Seville with the effects described in the preceding list. The network of above figure s still in use, but the subnet on Seville’s Ethernet has been changed from 10.1.3.0/24 to 10.1.3.192/26. Because RIP-1 does not send the mask in the update, Seville chooses *not* to address 10.1.3.192/26 onto its serial links (which use mask 255.255.255.0), because the update would be ambiguous.


Configuration and **debug ip rip** *Output on Albuquerque*

```
interface ethernet 0
ip addr 10.1.1.251 255.255.255.0
interface serial 0
ip addr 10.1.4.251 255.255.255.0
interface serial 1
ip addr 10.1.6.251 255.255.255.0
!
router rip
network 10.0.0.0

Albuquerque#debug ip rip
RIP protocol debugging is on
```

Configuration and **debug ip rip** *Output on Albuquerque (Continued)*

```
Albuquerque#
00:38:23: RIP: received v1 update from 10.1.4.252 on Serial0
00:38:23:      10.1.2.0 in 1 hops
00:38:23:      10.1.5.0 in 1 hops
00:38:33: RIP: sending v1 update to 255.255.255.255 via Serial0 (10.1.4.251)
00:38:33:      subnet  10.1.1.0, metric 1
00:38:33:      subnet  10.1.6.0, metric 1
00:38:33: RIP: sending v1 update to 255.255.255.255 via Serial1 (10.1.6.251)
00:38:33:      subnet  10.1.2.0, metric 2
00:38:33:      subnet  10.1.1.0, metric 1
00:38:33:      subnet  10.1.4.0, metric 1
00:38:33: RIP: sending v1 update to 255.255.255.255 via Ethernet0 (10.1.1.251)
00:38:33:      subnet  10.1.2.0, metric 2
00:38:33:      subnet  10.1.6.0, metric 1
00:38:33:      subnet  10.1.5.0, metric 2
00:38:33:      subnet  10.1.4.0, metric 1
00:38:40: RIP: received v1 update from 10.1.6.253 on Serial1
00:38:40:      10.1.2.0 in 2 hops
00:38:40:      10.1.5.0 in 1 hops
undebug all
All possible debugging has been turned off
Albuquerque#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 5 subnets
R       10.1.2.0 [120/1] via 10.1.4.252, 00:00:26, Serial0
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.6.0 is directly connected, Serial1
R       10.1.5.0 [120/1] via 10.1.4.252, 00:00:27, Serial0
                 [120/1] via 10.1.6.253, 00:00:10, Serial1
C       10.1.4.0 is directly connected, Serial0
Albuquerque#
(Suspended telnet resumed to Seville....)

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

Gateway of last resort is not set
```

continues

Configuration and **debug ip rip** *Output on Albuquerque (Continued)*

```
     10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
R       10.1.2.0/24 [120/1] via 10.1.5.252, 00:00:19, Serial1
R       10.1.1.0/24 [120/1] via 10.1.6.251, 00:00:22, Serial0
C       10.1.6.0/24 is directly connected, Serial0
C       10.1.5.0/24 is directly connected, Serial1
R       10.1.4.0/24 [120/1] via 10.1.6.251, 00:00:22, Serial0
                    [120/1] via 10.1.5.252, 00:00:19, Serial1
C       10.1.3.192/26 is directly connected, Ethernet0
Seville#
```

Configuration on Yosemite

```
interface ethernet 0
ip addr 10.1.2.252 255.255.255.0
interface serial 0
ip addr 10.1.4.252 255.255.255.0
interface serial 1
ip address 10.1.2.252 255.255.255.0

router rip
network 10.0.0.0
```

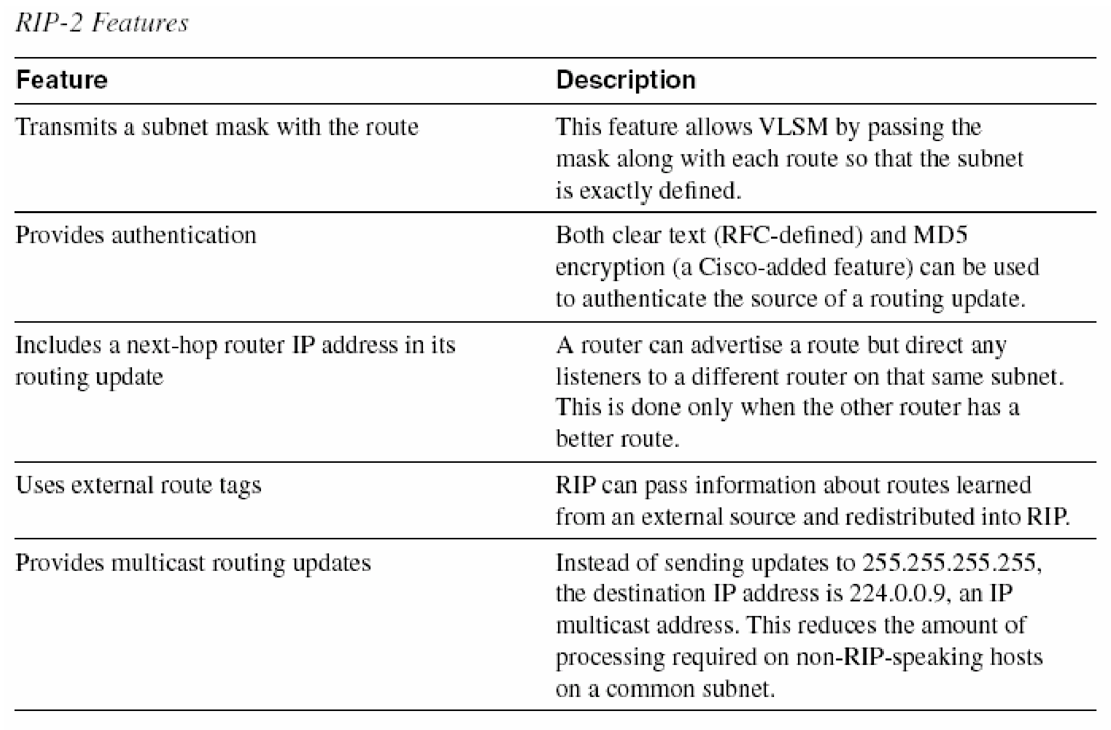
Configuration on Seville

```
interface ethernet 0
ip addr 10.1.3.253  255.255.255.192
interface serial 0
ip addr 10.1.6.253  255.255.255.0
interface serial 1
ip address 10.1.5.253  255.255.255.0
!
router rip
network 10.0.0.0
```

As shown in the highlighted portions of example subnet 10.1.3.192/26 is not advertised by Seville, as seen in its update received into Albuquerque’s Serial1 interface. Essentially, *RIP does not advertise the route with a mask of 255.255.255.192 out an interface that is in the same network but that has a different mask*. If RIP on Seville had advertised the route to 10.1.3.192, Albuquerque and Yosemite would have believed there was a problem, because the subnet number is 10.1.3.192, which is not a subnet number with the mask that Albuquerque and Yosemite think is in use (255.255.255.0). So, RIP and IGRP simply do not advertise the route into the same network on an interface that uses a different mask. The use of different masks in parts of the same network is called *variable-length subnet masking* (VLSM). As shown in this example, VLSM is not supported by RIP-1 or IGRP.

5.5.1 RIP Version 2

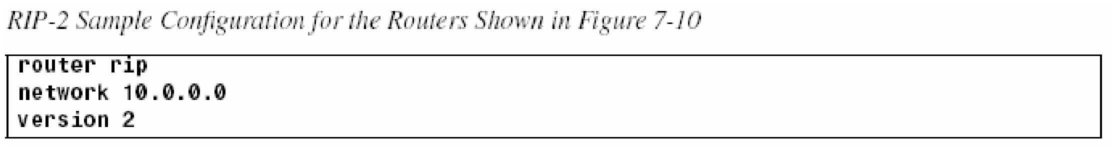
RIP-2, defined by RFC 1723, adds advanced features to RIP-1. Many features are the same: Hop count is still used for the metric, it is still a distance vector protocol, and it still uses hold-down timers and route poisoning. Several features have been added. They are listed in following table.



RIP-2 Features

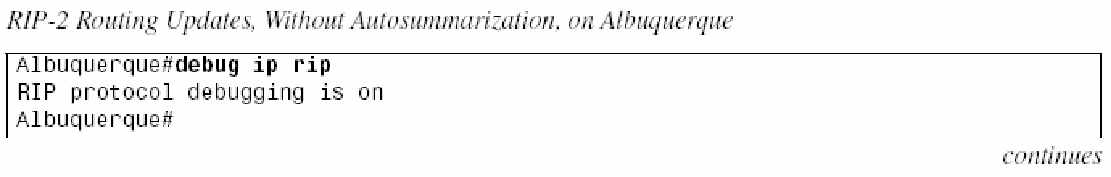
Feature	Description
Transmits a subnet mask with the route	This feature allows VLSM by passing the mask along with each route so that the subnet is exactly defined.
Provides authentication	Both clear text (RFC-defined) and MD5 encryption (a Cisco-added feature) can be used to authenticate the source of a routing update.
Includes a next-hop router IP address in its routing update	A router can advertise a route but direct any listeners to a different router on that same subnet. This is done only when the other router has a better route.
Uses external route tags	RIP can pass information about routes learned from an external source and redistributed into RIP.
Provides multicast routing updates	Instead of sending updates to 255.255.255.255, the destination IP address is 224.0.0.9, an IP multicast address. This reduces the amount of processing required on non-RIP-speaking hosts on a common subnet.

RIP-2 supports the new features listed in the table, but most importantly, it supports the use of VLSM through transmitting mask information. For instance, the preceding example showed a problem using VLSM on Seville (subnet 10.1.3.192/26). RIP-2 works fine in the same network, as shown in following example. This example lists the RIP-2 configuration on each of the three routers, and other example shows a sample RIP **debug** on Albuquerque.



RIP-2 Sample Configuration for the Routers Shown in Figure 7-10

```
router rip
network 10.0.0.0
version 2
```



RIP-2 Routing Updates, Without Autosummarization, on Albuquerque

```
Albuquerque#debug ip rip
RIP protocol debugging is on
Albuquerque#
```

continues

RIP-2 Routing Updates, Without Autosummarization, on Albuquerque (Continued)

```
00:36:04: RIP: received v2 update from 10.1.4.252 on Serial0
00:36:04:      10.1.2.0/24 -> 0.0.0.0 in 1 hops
00:36:04:      10.1.5.0/24 -> 0.0.0.0 in 1 hops
00:36:04:      10.1.3.192/26 -> 0.0.0.0 in 2 hops
00:36:08: RIP: sending v2 update to 224.0.0.9 via Serial0 (10.1.4.251)
00:36:08:      10.1.1.0/24 -> 0.0.0.0, metric 1, tag 0
00:36:08:      10.1.6.0/24 -> 0.0.0.0, metric 1, tag 0
00:36:08:      10.1.3.192/26 -> 0.0.0.0, metric 2, tag 0
00:36:08: RIP: sending v2 update to 224.0.0.9 via Serial1 (10.1.6.251)
00:36:08:      10.1.2.0/24 -> 0.0.0.0, metric 2, tag 0
00:36:08:      10.1.1.0/24 -> 0.0.0.0, metric 1, tag 0
00:36:08:      10.1.4.0/24 -> 0.0.0.0, metric 1, tag 0
00:36:08: RIP: sending v2 update to 224.0.0.9 via Ethernet0 (10.1.1.251)
00:36:08:      10.1.2.0/24 -> 0.0.0.0, metric 2, tag 0
00:36:08:      10.1.6.0/24 -> 0.0.0.0, metric 1, tag 0
00:36:08:      10.1.5.0/24 -> 0.0.0.0, metric 2, tag 0
00:36:08:      10.1.4.0/24 -> 0.0.0.0, metric 1, tag 0
00:36:08:      10.1.3.192/26 -> 0.0.0.0, metric 2, tag 0
00:36:20: RIP: received v2 update from 10.1.6.253 on Serial1
00:36:20:      10.1.2.0/24 -> 0.0.0.0 in 2 hops
00:36:20:      10.1.5.0/24 -> 0.0.0.0 in 1 hops
00:36:20:      10.1.3.192/26 -> 0.0.0.0 in 1 hops
00:36:30: RIP: received v2 update from 10.1.4.252 on Serial0
00:36:30:      10.1.2.0/24 -> 0.0.0.0 in 1 hops
00:36:30:      10.1.5.0/24 -> 0.0.0.0 in 1 hops
00:36:30:      10.1.3.192/26 -> 0.0.0.0 in 2 hops

Albuquerque#no debug all
All possible debugging has been turned off

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

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
R       10.1.2.0/24 [120/1] via 10.1.4.252, 00:00:09, Serial0
C       10.1.1.0/24 is directly connected, Ethernet0
C       10.1.6.0/24 is directly connected, Serial1
R       10.1.5.0/24 [120/1] via 10.1.4.252, 00:00:09, Serial0
                    [120/1] via 10.1.6.253, 00:00:19, Serial1
C       10.1.4.0/24 is directly connected, Serial0
R       10.1.3.192/26 [120/1] via 10.1.6.253, 00:00:19, Serial1
Albuquerque#
```

A couple of important items should be noted in the **debug** output of example . (As always, the specific portions that are referred to are highlighted.) The updates sent by Albuquerque are sent to multicast IP address 224.0.0.9 as opposed to a broadcast address; this allows the devices that are not using RIP-2 to ignore the updates and not waste processing cycles. The **show ip route** output on Albuquerque lists the previously missing subnet, 10.1.3.192/26; this is expected, as highlighted in the **debug ip rip** messages received by Albuquerque from Seville (10.1.6.253). The subnet masks are shown in the prefix style, with /26 representing mask 255.255.255.192. Also, note the **debug** output designating **tag 0**. This means that all the external route tags have value 0, which is the default.

Migrating from RIP-1 to RIP-2 requires some planning. RIP-1 sends updates to the broadcast address, whereas RIP-2 uses a multicast. A RIP-1-only router and a RIP-2-only router will not succeed in exchanging routing information. To migrate to RIP-2, one option is to migrate all routers at the same time. This might not be a reasonable political or administrative option, however. If not, some coexistence between RIP-1 and RIP-2 is required.

The **ip rip send version** command can be used to overcome this problem. Essentially, the configuration tells the router whether to send RIP-1-style updates, RIP-2-style updates, or both for each interface. Consider the familiar Figure 7-10 network, with RIP-1 still configured on all three routers. If two of the routers are migrated—for instance, Albuquerque and Seville—they can communicate with RIP-2 easily. However, by default these two routers now send only RIP-2 updates, which Yosemite cannot understand, because it is still running RIP-1. The configurations shown in Examples overcome this problem by having Albuquerque and Seville send only RIP-1 updates to Yosemite.

Configuration on Albuquerque

```
interface ethernet 0
ip addr 10.1.1.251 255.255.255.0
interface serial 0
ip addr 10.1.4.251 255.255.255.0
ip rip send version 1
ip rip receive version 1
interface serial 1
ip address 10.1.6.251 255.255.255.0
!
router rip
network 10.0.0.0
version 2
```

Configuration on Yosemite

```
interface ethernet 0
ip addr 10.1.2.252 255.255.255.0
interface serial 0
ip addr 10.1.4.252 255.255.255.0
```

continues

Configuration on Yosemite (Continued)

```
interface serial 1
ip address 10.1.5.252 255.255.255.0
!
router rip
network 10.0.0.0
```

Configuration on Seville

```
interface ethernet 0
ip addr 10.1.2.252 255.255.255.0
interface serial 0
ip addr 10.1.4.252 255.255.255.0
interface serial 1
ip address 10.1.5.252 255.255.255.0
ip rip send version 1
ip rip receive version 1
!
router rip
network 10.0.0.0
version 2
```

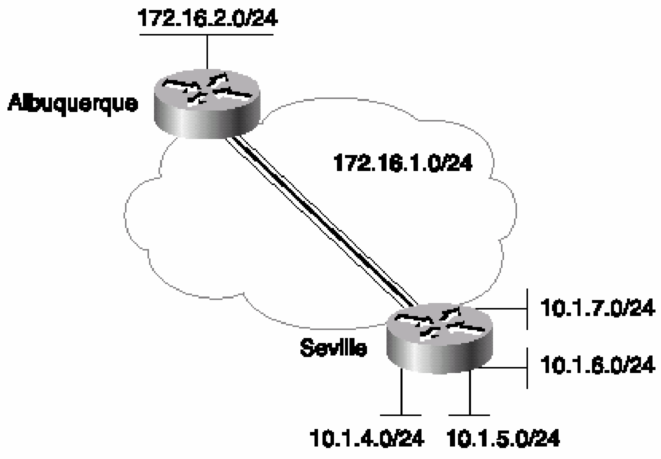
The RIP-2 configuration logic works just like RIP-1. Updates are sent and received on each interface that is matched by a **network** command. But because Yosemite sends and receives only RIP-1 updates, the other two routers need the appropriate interface subcommands to tell the router to send and receive RIP-1 updates to and from Yosemite. Both Albuquerque and Seville continue to send and receive RIP-2 updates on all interfaces, so when Yosemite upgrades to RIP-2, no immediate configuration changes are required in Albuquerque and Seville.

5.6 Autosummarization and Route Aggregation

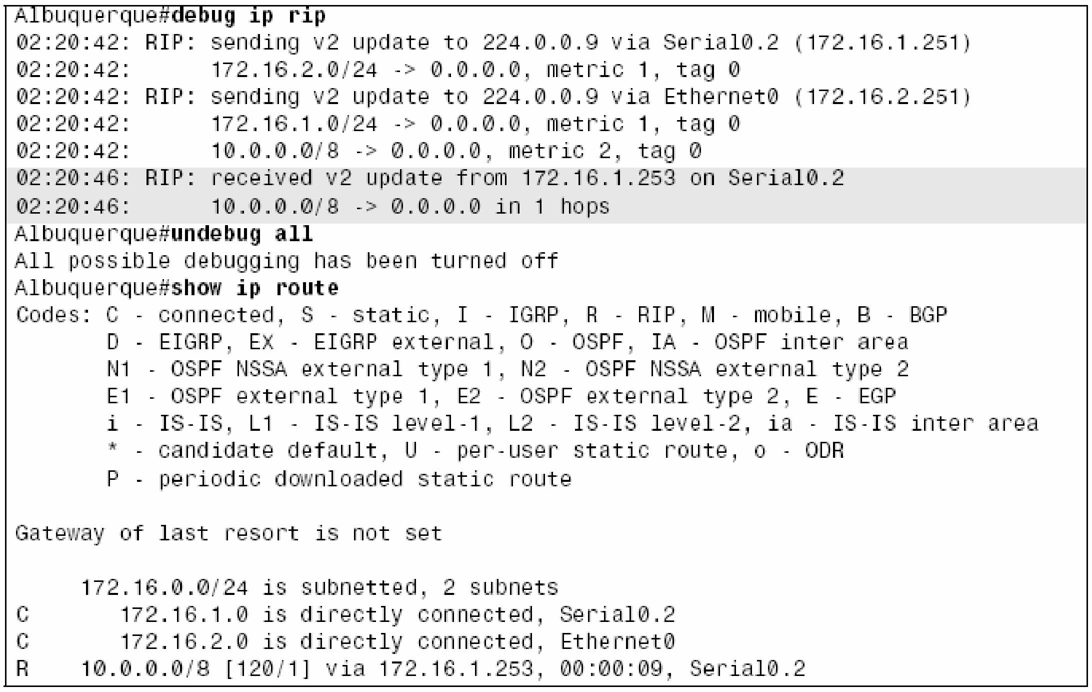
Cisco IOS Software is optimized to perform routing as fast as possible. Most of the Layer 3 routing performance improvement in the brief history of routers has been through improved algorithms. Many times those improved algorithms later have been implemented in hardware to provide even lower latency. Although these improvements have been a great benefit, it is typically true that any algorithm that searches a list runs more quickly if the list is short, as compared to searching a similar list that is long. Autosummarization and route aggregation (also known as route summarization) are two Cisco IOS Software features that reduce the size of the IP routing table, thereby reducing latency per packet. Autosummarization is a routing protocol feature that operates using this rule:
When advertised on an interface whose IP address is not in network X, routes about subnets in network X are summarized and advertised as one route. That route is for the entire Class A, B, or C network X. RIP and IGRP perform autosummarization, and it cannot be disabled. Essentially, it must be a side effect of routing protocols that transmit the mask. For RIP-2 and EIGRP, autosummarization can be enabled or disabled.

As usual, an example makes the concept much clearer. Consider above figure , which shows two networks in use: 10.0.0.0 and 172.16.0.0. Seville has four (connected) routes to subnets of network 10.0.0.0. Example 7-15 shows the output of a **show ip route** command on Albuquerque, as well as RIP-2 **debug ip rip** output.

Autosummarization



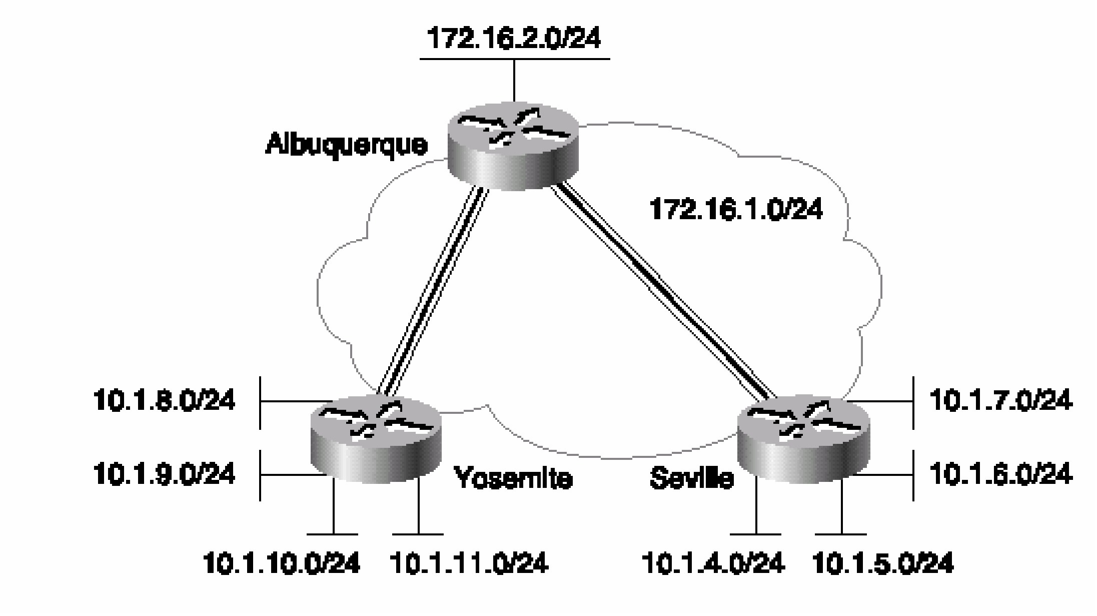
Configuration on Seville



```
Albuquerque#debug ip rip
02:20:42: RIP: sending v2 update to 224.0.0.9 via Serial0.2 (172.16.1.251)
02:20:42:      172.16.2.0/24 -> 0.0.0.0, metric 1, tag 0
02:20:42: RIP: sending v2 update to 224.0.0.9 via Ethernet0 (172.16.2.251)
02:20:42:      172.16.1.0/24 -> 0.0.0.0, metric 1, tag 0
02:20:42:      10.0.0.0/8 -> 0.0.0.0, metric 2, tag 0
02:20:46: RIP: received v2 update from 172.16.1.253 on Serial0.2
02:20:46:      10.0.0.0/8 -> 0.0.0.0 in 1 hops
Albuquerque#undebug all
All possible debugging has been turned off
Albuquerque#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Serial0.2
C       172.16.2.0 is directly connected, Ethernet0
R    10.0.0.0/8 [120/1] via 172.16.1.253, 00:00:09, Serial0.2
```

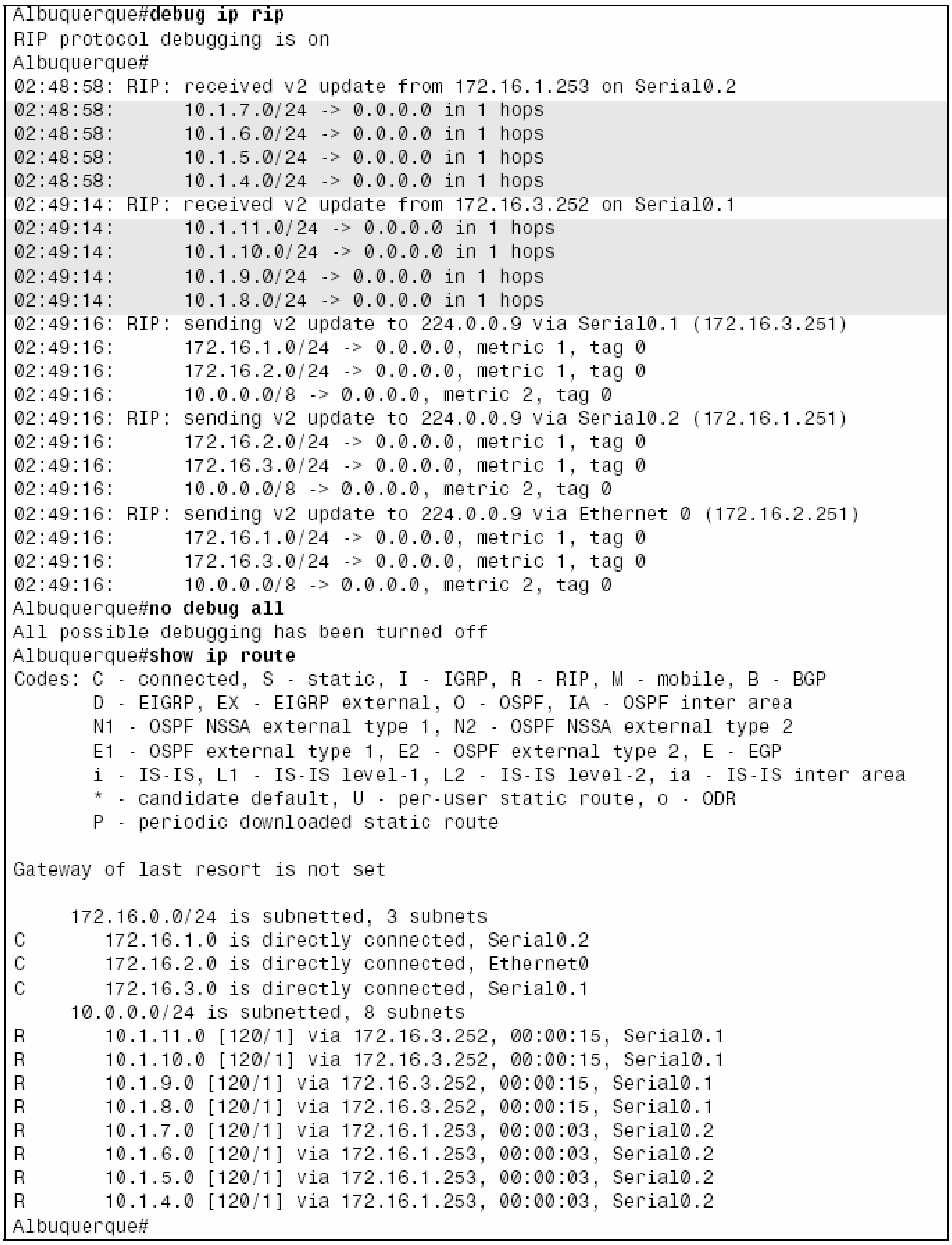
Notice, as highlighted in example , that Albuquerque's received update on Serial0.2 from Seville advertises only the entire Class A network 10.0.0.0/8 because autosummarization is enabled on Seville (by default). The IP routing table lists just one route to network 10.0.0.0. This works fine, as long as network 10.0.0.0 is contiguous. Consider above figure , in which Yosemite also has subnets of network 10.0.0.0 but has no connectivity to Seville other than through Albuquerque.

Autosummarization Pitfalls



IP subnet design traditionally has not allowed *discontiguous networks*. A *contiguous network* is a single Class A, B, or C network for which all routes to subnets of that network pass through only other subnets of that same single network. *Discontiguous networks* refers to the concept that, in a single Class A, B, or C network, there is at least one case in which the only routes to one subnet pass through subnets of a different network. An easy analogy for residents of the United States is the term *contiguous 48,* referring to the 48 states besides Alaska and Hawaii. To drive to Alaska from the contiguous 48, for example, you must drive through another country(Canada, for the geographically impaired!), so Alaska is not contiguous with the 48 states—in other words, it is discontiguous. Figure breaks that rule. In this figure, there could be a PVC between Yosemite and Seville that uses a subnet of network 10.0.0.0, but that PVC might be down, causing the discontiguous network. The temporarily discontiguous network can be overcome with the use of a routing protocol that transmits masks, because the rule of discontiguous subnets can be ignored if no autosummarization, or configured summarization, is performed. Consider the routing updates and routing table on Albuquerque in following example , where RIP-2 is used so that autosummarization is disabled on all routers.

Albuquerque's Routing Table When Seville Is Not Summarizing



```
Albuquerque#debug ip rip
RIP protocol debugging is on
Albuquerque#
02:48:58: RIP: received v2 update from 172.16.1.253 on Serial0.2
02:48:58:      10.1.7.0/24 -> 0.0.0.0 in 1 hops
02:48:58:      10.1.6.0/24 -> 0.0.0.0 in 1 hops
02:48:58:      10.1.5.0/24 -> 0.0.0.0 in 1 hops
02:48:58:      10.1.4.0/24 -> 0.0.0.0 in 1 hops
02:49:14: RIP: received v2 update from 172.16.3.252 on Serial0.1
02:49:14:      10.1.11.0/24 -> 0.0.0.0 in 1 hops
02:49:14:      10.1.10.0/24 -> 0.0.0.0 in 1 hops
02:49:14:      10.1.9.0/24 -> 0.0.0.0 in 1 hops
02:49:14:      10.1.8.0/24 -> 0.0.0.0 in 1 hops
02:49:16: RIP: sending v2 update to 224.0.0.9 via Serial0.1 (172.16.3.251)
02:49:16:      172.16.1.0/24 -> 0.0.0.0, metric 1, tag 0
02:49:16:      172.16.2.0/24 -> 0.0.0.0, metric 1, tag 0
02:49:16:      10.0.0.0/8 -> 0.0.0.0, metric 2, tag 0
02:49:16: RIP: sending v2 update to 224.0.0.9 via Serial0.2 (172.16.1.251)
02:49:16:      172.16.2.0/24 -> 0.0.0.0, metric 1, tag 0
02:49:16:      172.16.3.0/24 -> 0.0.0.0, metric 1, tag 0
02:49:16:      10.0.0.0/8 -> 0.0.0.0, metric 2, tag 0
02:49:16: RIP: sending v2 update to 224.0.0.9 via Ethernet 0 (172.16.2.251)
02:49:16:      172.16.1.0/24 -> 0.0.0.0, metric 1, tag 0
02:49:16:      172.16.3.0/24 -> 0.0.0.0, metric 1, tag 0
02:49:16:      10.0.0.0/8 -> 0.0.0.0, metric 2, tag 0
Albuquerque#no debug all
All possible debugging has been turned off
Albuquerque#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 3 subnets
C       172.16.1.0 is directly connected, Serial0.2
C       172.16.2.0 is directly connected, Ethernet0
C       172.16.3.0 is directly connected, Serial0.1
     10.0.0.0/24 is subnetted, 8 subnets
R       10.1.11.0 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
R       10.1.10.0 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
R       10.1.9.0 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
R       10.1.8.0 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
R       10.1.7.0 [120/1] via 172.16.1.253, 00:00:03, Serial0.2
R       10.1.6.0 [120/1] via 172.16.1.253, 00:00:03, Serial0.2
R       10.1.5.0 [120/1] via 172.16.1.253, 00:00:03, Serial0.2
R       10.1.4.0 [120/1] via 172.16.1.253, 00:00:03, Serial0.2
Albuquerque#
```

As highlighted in example , the routing updates include the individual subnets. Therefore, Albuquerque can see routes to all subnets of network 10 and can route packets to the correct destinations in Seville and Yosemite.

With autosummarization enabled, Albuquerque would think that both Seville and Yosemite had an equal-metric route to network 10.0.0.0, and some packets would be routed incorrectly. Route summarization (also called route aggregation) works like autosummarization, except that there is no requirement to summarize into a Class A, B, or C network. Consider the network shown in figure. Albuquerque has eight routes to subnets of network 10.0.0.0; four of those routes are learned from Seville. Consider the subnet, broadcast, and assignable addresses in each of the subnets, as shown in following table .

Route Aggregation Comparison of Subnet Numbers

Subnet	Mask	Broadcast	Assignable Addresses
10.1.4.0	255.255.255.0	10.1.4.255	10.1.4.1 to 10.1.4.254
10.1.5.0	255.255.255.0	10.1.5.255	10.1.5.1 to 10.1.5.254
10.1.6.0	255.255.255.0	10.1.6.255	10.1.6.1 to 10.1.6.254
10.1.7.0	255.255.255.0	10.1.7.255	10.1.7.1 to 10.1.7.254

Now consider the concept of a subnet 10.1.4.0 with mask 255.255.252.0. In this case, 10.1.4.0/22 (the same subnet written differently) has a subnet broadcast address of 10.1.7.255 and assignable addresses of 10.1.4.1 to 10.1.7.254. Because 10.1.4.0/22 includes all the assignable addresses of the original four subnets, a single route to 10.1.4.0/22 is just as good as the four separate routes, assuming that the next-hop information is the same for each of the original four routes. Route aggregation is simply a tool used to tell a routing protocol to advertise a single, larger subnet rather than individual, smaller subnets. In this case, the routing protocol advertises 10.1.4.0/22 rather than the four individual subnets. Albuquerque's routing table is then smaller. EIGRP and OSPF are the only interior IP routing protocols that support route aggregation. Following example shows route summarization of the subnets off Seville. Still using the network shown in the figure, the routers are all migrated to EIGRP. Following example shows the EIGRP configuration on Albuquerque, the EIGRP configuration on Seville, and the resulting IP routing table on Albuquerque. (Yosemite is migrated to EIGRP as well; the configuration is not shown because the example shows only aggregation by Seville.)

Route Aggregation Example Using EIGRP

```
On Seville:
router eigrp 9
 Network 10.0.0.0
 Network 172.16.0.0
 No auto-summary
!
interface serial 0.1 point-to-point
 ip address 172.16.1.253 255.255.255.0
 frame-relay interface-dlci 901
 ip summary-address eigrp 9 10.1.4.0 255.255.252.0
```

```
On Albuquerque:
router eigrp 9
  network 172.16.0.0
  no auto-summary
Albuquerque#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 3 subnets
C       172.16.1.0 is directly connected, Serial0.2
C       172.16.2.0 is directly connected, Ethenet0
C       172.16.3.0 is directly connected, Serial0.1
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
D       10.1.11.0/24 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
D       10.1.10.0/24 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
D       10.1.9.0/24 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
D       10.1.8.0/24 [120/1] via 172.16.3.252, 00:00:15, Serial0.1
D       10.1.4.0/22 [90/2185984] via 172.16.1.253, 00:00:58, Serial0.2
```

The **ip summary-address** interface subcommand on Seville's Serial0.1 interface is used to define the superset of the subnets that should be advertised. Notice the route in Albuquerque's routing table, which indeed shows 10.1.4.0/22 rather than the four individual subnets. When summarizing, the superset of the original subnets can actually be smaller than the Class A, B, or C network; larger than the network; or exactly matched to a network. For instance, 192.168.4.0, 192.168.5.0, 192.168.6.0, and 192.168.7.0 can be summarized into 192.168.4.0/22, which represents four consecutive Class C networks. Summarizing when the summarized group is a set of networks is sometimes called *supernetting*. Following table lists the features for summarizing the interior IP routing protocols.

Interior IP Routing Protocol Summarization Features

Routing Protocol	Autosummarization Enabled?	Can Autosummarization Be Disabled?	Route Aggregation/ Summarization Allowed?
RIP-1	Yes, by default	No	No
IGRP	Yes, by default	No	No
RIP-2	Yes, by default	Yes	No
Enhanced IGRP	Yes, by default	Yes	Yes
OSPF	No, but the equivalent can be done with aggregation	N/A	Yes

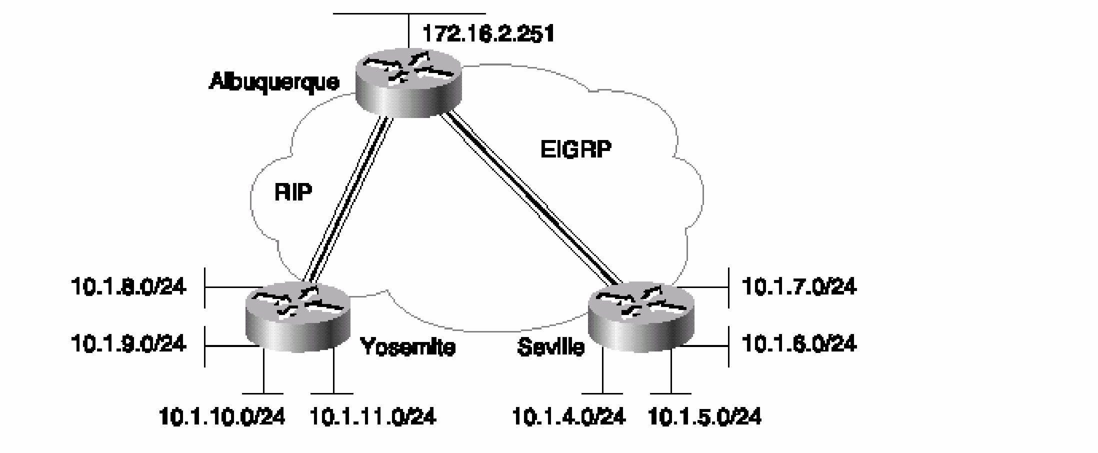
5.6.1 Multiple Routes to the Same Subnet

What if a router learns two routes whose metrics are equal? By default, the Cisco IOS Software supports four equal-cost routes to the same IP subnet in the routing table at the same time. The traffic is balanced across the equal-metric routes on a per-destination address basis by default. You can change the number of equal-cost routes to between 1 and 6 using the **ip maximum-paths** *x* router configuration subcommand, where *x* is the maximum number of routes to any subnet. The metric formula used for IGRP (and EIGRP) poses an interesting problem when considering equal-metric routes. IGRP can learn more than one route to the same subnet with different metrics; however, the metrics are very likely to never be exactly equal. The routers **variance** command is used to define how variable the metrics can be in order for routes to be considered to have equal metrics. For example, if the metric for the better of two routes is 100, and the variance is set to 2, a second route with a metric less than 200 would be considered equal-cost and would be added to the routing table. For many years, equal-cost routes were treated equally—which seems to make sense. But what if the routes are not really equal and you use variance to add them? Well, with one more router subcommand, you can make the router either always use the truly best route or balance across the routes based on the metrics' ratios. In other words, the **traffic-share min** router IGRP subcommand tells the router to ignore all equal-metric routes in the routing table, except the route that truly has the smallest metric. So why not just add only the truly lowest metric route to the routing table? Well, if the other pretty good routes are in the table, and the best one fails, convergence time is practically instantaneous! An alternative to using the single, truly best route, even when multiple routes are in the routing table, is to use the **traffic-share balanced** router subcommand. It tells the router to use all the routes proportionally based on the metrics for each route.

5.7 Troubleshooting Routing and Routing Protocols

This section gives you some final insights into some tricky problems with routing protocols. The **show ip route** command has a myriad of options that are helpful when you're troubleshooting a large network. The **show ip protocol** command also can provide some useful information when you're troubleshooting a routing problem. With a small network, most of the options with the **show ip route** command are unnecessary. However, knowing the options and what each can do is useful for your work with larger networks. Refer the previous figure shows the network; it should look familiar from previous examples. In this case, EIGRP is used between Albuquerque and Seville, and RIP-2 is used between Albuquerque and Yosemite. There is no PVC between Yosemite and Seville.

Network Environment for Use with the **show ip route** *Options*



Albuquerque Configuration for the **show ip route** *Options in Example 7-21*

```
Albuquerque#show running-config
Current configuration : 964 bytes
!
version 12.2
```

continues

```
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Albuquerque
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
enable password fred
!
ip subnet-zero
no ip domain-lookup
!
interface Serial0
 no ip address
 no ip directed-broadcast
 encapsulation frame-relay IETF
 clockrate 56000
 frame-relay lmi-type cisco
!
interface Serial0.1 point-to-point
 ip address 172.16.3.251 255.255.255.0
 no ip directed-broadcast
 frame-relay interface-dlci 902
!
interface Serial0.2 point-to-point
 ip address 172.16.1.251 255.255.255.0
 no ip directed-broadcast
 frame-relay interface-dlci 903
!
interface Serial1
 no ip address
 no ip directed-broadcast
 shutdown
!
interface Ethernet0
 ip address 172.16.2.251 255.255.255.0
 no ip directed-broadcast
!
router eigrp 9
 passive-interface Serial0.1
 network 172.16.0.0
 no auto-summary
!
router rip
 version 2
 passive-interface Serial0.2
 network 172.16.0.0
 no auto-summary
!
ip classless
no ip http server
!
access-list 1 permit 10.0.0.0 0.255.255.255
```

Yosemite Configuration for the **show ip route** *Options*

```
Yosemite#show running-config
Current configuration : 968 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Yosemite
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
ip subnet-zero
no ip domain-lookup
!
interface Serial0
 no ip address
 no ip directed-broadcast
 encapsulation frame-relay IETF
 no fair-queue
 frame-relay lmi-type cisco
!
interface Serial0.1 point-to-point
 ip address 172.16.3.252 255.255.255.0
 no ip directed-broadcast
 frame-relay interface-dlci 901
!
interface Serial1
 no ip address
 no ip directed-broadcast
 shutdown
!
!
interface Ethernet0
 ip address 10.1.8.253 255.255.255.0
!
interface Ethernet1
 ip address 10.1.9.253 255.255.255.0
!
interface Ethernet2
 ip address 10.1.10.253 255.255.255.0
!
interface Ethernet3
 ip address 10.1.11.253 255.255.255.0
!
router rip
 version 2
 network 10.0.0.0
 network 172.16.0.0
 no auto-summary
!
ip classless
no ip http server
```

Seville Configuration for the **show ip route** *Options*

```
Seville#show running-config
Current configuration : 960 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Seville
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
ip subnet-zero
no ip domain-lookup
!
interface Serial0
 no ip address
 no ip directed-broadcast
 encapsulation frame-relay IETF
 no fair-queue
 frame-relay lmi-type cisco
!
interface Serial0.1 multipoint
 ip address 172.16.1.253 255.255.255.0
 no ip directed-broadcast
 ip summary-address eigrp 9 10.1.4.0 255.255.252.0
 frame-relay interface-dlci 901
!
interface Serial1
 no ip address
 no ip directed-broadcast
 shutdown
!
interface Ethernet0
 ip address 10.1.4.253 255.255.255.0
!
interface Ethernet1
 ip address 10.1.5.253 255.255.255.0
!
interface Ethernet2
 ip address 10.1.6.253 255.255.255.0
!
interface Ethernet3
 ip address 10.1.7.253 255.255.255.0
!
router eigrp 9
 network 10.0.0.0
 network 172.16.0.0
 no auto-summary
!
ip classless
no ip http server
```

show ip route: *Albuquerque*

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

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 3 subnets
C       172.16.1.0 is directly connected, Serial0.2
C       172.16.2.0 is directly connected, Ethernet0
C       172.16.3.0 is directly connected, Serial0.1
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
R       10.1.11.0/24 [120/1] via 172.16.3.252, 00:00:17, Serial0.1
R       10.1.10.0/24 [120/1] via 172.16.3.252, 00:00:17, Serial0.1
R       10.1.9.0/24 [120/1] via 172.16.3.252, 00:00:17, Serial0.1
R       10.1.8.0/24 [120/1] via 172.16.3.252, 00:00:17, Serial0.1
D       10.1.4.0/22 [90/2185984] via 172.16.1.253, 00:28:01, Serial0.2

Albuquerque#show ip route ?
   Hostname or A.B.C.D  Network to display information about or hostname
  bgp                   Border Gateway Protocol (BGP)
  connected             Connected
  egp                   Exterior Gateway Protocol (EGP)
  eigrp                 Enhanced Interior Gateway Routing Protocol (EIGRP)
  igrp                  Interior Gateway Routing Protocol (IGRP)
  isis                  ISO IS-IS
  list                  IP Access list
  mobile                Mobile routes
  odr                   On Demand stub Routes
  ospf                  Open Shortest Path First (OSPF)
  profile               IP routing table profile
  rip                   Routing Information Protocol (RIP)
  static                Static routes
  summary               Summary of all routes
  supernets-only        Show supernet entries only
  vrf                   Display routes from a VPN Routing/Forwarding instance
  |                     Output modifiers


Albuquerque#show ip route 10.1.5.8
Routing entry for 10.1.4.0/22
  Known via "eigrp 9", distance 90, metric 2185984, type internal
  Redistributing via eigrp 9
  Last update from 172.16.1.253 on Serial0.2, 00:28:36 ago
  Routing Descriptor Blocks:
  * 172.16.1.253, from 172.16.1.253, 00:28:36 ago, via Serial0.2
      Route metric is 2185984, traffic share count is 1
      Total delay is 20630 microseconds, minimum bandwidth is 1544 Kbit
```

continues

show ip route: *Albuquerque (Continued)*

```
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1

Albuquerque#show ip route rip
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
R       10.1.11.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
R       10.1.10.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
R       10.1.9.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
R       10.1.8.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
Albuquerque#show ip route igrp

Albuquerque#show ip route eigrp
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
D       10.1.4.0/22 [90/2185984] via 172.16.1.253, 00:29:42, Serial0.2

Albuquerque#show ip route connected
     172.16.0.0/24 is subnetted, 3 subnets
C       172.16.1.0 is directly connected, Serial0.2
C       172.16.2.0 is directly connected, Ethernet0
C       172.16.3.0 is directly connected, Serial0.1

Albuquerque#show ip route list 1
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
R       10.1.11.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
R       10.1.10.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
R       10.1.9.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
R       10.1.8.0/24 [120/1] via 172.16.3.252, 00:00:22, Serial0.1
D       10.1.4.0/22 [90/2185984] via 172.16.1.253, 00:29:58, Serial0.2

Albuquerque#show ip route summary
Route Source    Networks    Subnets     Overhead    Memory (bytes)
connected       0           3           156         420
static          0           0           0           0
rip             0           4           208         560
eigrp 9         0           1           52          140
internal        2                                   2320
Total           2           8           416         3440

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

Gateway of last resort is not set
```

The **show ip route** command with no options has been seen many times in this book. A review of some of the more important bits of the output is in order; most comments refer to a highlighted portion. First, the legend at the beginning of last example defines the letter codes that identify the source of the routing information—for example, **C** for connected routes, **R** for RIP, and **I** for IGRP. Each of the Class A, B, and C networks is listed, along with each of the subnets of that network. If a static mask is used within that network, the mask is shown only in the line referring to the network (as is the case in Example 7-21, network 172.16.0.0). If the network uses VLSM, as network 10.0.0.0 appears to do because of the route summarization done by Seville, the mask information is listed on the lines referring to each of the individual subnets. Each routing entry lists the subnet number and the outgoing interface. In most cases, the next hop router's IP address is also listed. The outgoing interface is needed so that the router can choose the type of data link header to use to encapsulate the packet before transmission on that interface. The next-hop router's IP address is needed on interfaces for which the router needs the IP address so that it can find the associated data-link address to put in the newly built data link header. For instance, knowing the next-hop IP address of 172.16.3.252, Yosemite's IP address on the Frame Relay VC allows Albuquerque to find the corresponding DLCI in the Frame Relay map. The numbers in brackets in the **show ip route** output for each route are interesting. The second number in brackets represents the metric value for this route. The first number defines the administrative distance.

Administrative distance is important only if multiple IP routing protocols are in use in a single router. When this is true, both routing protocols can learn routes to the same subnets. Because their metric values are different (for example, hop count or a function of bandwidth and delay), there is no way to know which routing protocol's routes are better. Therefore, Cisco supplies a method of defining which routing protocol's routes are better. The Cisco IOS Software implements this concept using something called *administrative distance*. Administrative distance is an integer value; a value is assigned to each source of routing information. The lower the administrative distance, the better the source of routing information. IGRP's default is 100, OSPF's is 110, RIP's is 120, and EIGRP's is 90. The value 100 in brackets in the **show ip route** output signifies that the administrative distance used for IGRP routes is 100. In other words, the default value is in use. So, if RIP and IGRP are both used, and if both learn routes to the same subnets, only IGRP's routing information for those subnets is added to the routing table. If RIP learns about a subnet that IGRP does not know about, that route is added to the routing table Moving down example, the **show ip route ?** command lists several options, many of which are shown in the ensuing commands in the example. You can limit the **show ip route** output to the routes learned by a particular routing protocol by referring to that routing protocol. Likewise, the output can be limited to show just connected routes.

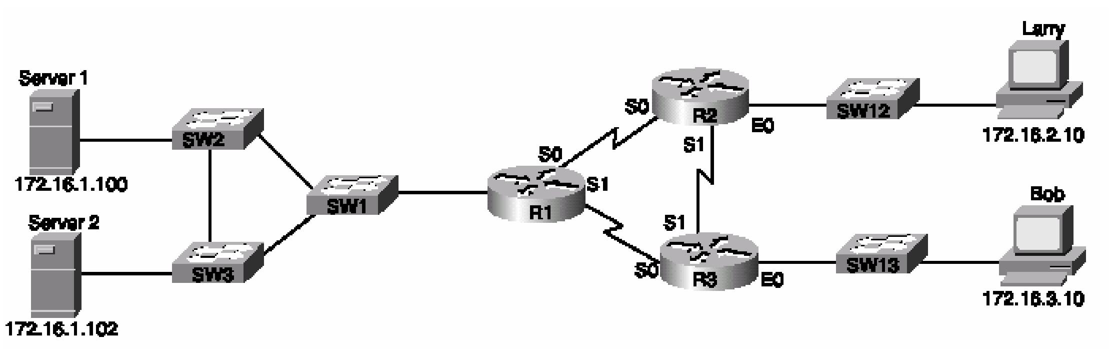
One of the more important options for the **show ip route** command is to simply pass an IP address as the last parameter. This tells the router to perform routing table lookup, just as it would for a packet destined for that address. In example , **show ip route 10.1.5.8** returns a set of messages, the first of which identifies the route to 10.1.4.0/22 as the route matched in the routing table. The route that is matched is listed so that you can always know the route that would be used by this router to reach a particular IP address.

Finally, another feature of **show ip route** that is useful in large networks is filtering the command's output based on an access list. Notice the command **show ip route list 1** in Example 7-21. Access list 1 is configured so that any route with information about network 10.0.0.0 is matched (permitted by the access list) and all others are denied. By referring to the access list, the **show ip route** output is filtered, showing only a portion of the routes. This is particularly useful when there are many routes in the routing table. The many options of the **show ip route** command can be particularly useful for troubleshooting larger networks.

6 - Access List Security

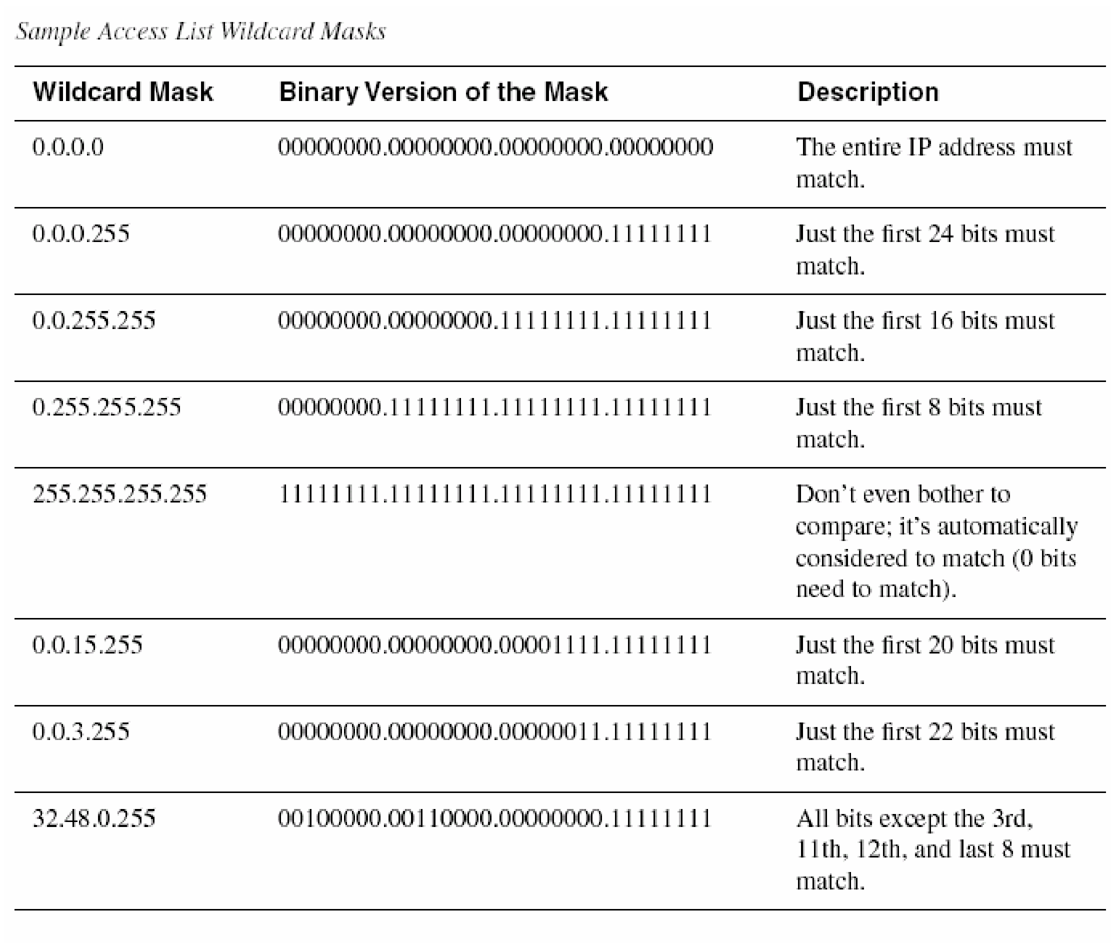
6.1 Standard Access Lists

IP access lists cause a router to discard some packets based on criteria defined by the network engineer. The goal of these filters is to prevent unwanted traffic in the network—whether to prevent hackers from penetrating the network, or just to prevent employees from using systems that they should not be using. Access lists should simply be part of an organization's security policy, but for CCNA study purposes, we do not need to consider the business goals that drive the security policy. By the way, IP access lists can also be used to filter routing updates, to match packets for prioritization, and to match packets for implementing quality of service features.As soon as you know what needs to be filtered, the next goal is to decide where to filter the traffic. Following figure serves as an example. In this case, imagine that Bob is not allowed to access Server1, but Larry is.



The matching criteria available to access lists is based on fields inside the IP, TCP, and UDP headers. Extended access lists can check source and destination IP addresses, as well as source and destination port numbers, along with several other fields. However, standard IP access lists can examine only the source IP address. You can configure the portion of the IP address that is checked by the **access-list** command. For instance, if you wanted to stop Bob from sending packets to Server1, you would look at the entire IP address of Bob and Server1 in the access list. But what if the criteria were to stop all hosts in Bob's subnet from getting to Server1? Because all hosts in Bob's subnet have the same numbers in their first three octets, the access list could just check the first three octets of the address in order to match all packets with a single **access-list** statement.

Cisco *wildcard masks* are access list parameters that define the portion of the IP address that should be examined. For example, suppose that one mask implies that the whole packet should be checked and another implies that only the first three octets need to be examined. To perform this matching, Cisco access lists use wildcard masks. Following table lists some of the more popular wildcard masks, as well as a few that are not quite as common.


Sample Access List Wildcard Masks

Wildcard Mask	Binary Version of the Mask	Description
0.0.0.0	00000000.00000000.00000000.00000000	The entire IP address must match.
0.0.0.255	00000000.00000000.00000000.11111111	Just the first 24 bits must match.
0.0.255.255	00000000.00000000.11111111.11111111	Just the first 16 bits must match.
0.255.255.255	00000000.11111111.11111111.11111111	Just the first 8 bits must match.
255.255.255.255	11111111.11111111.11111111.11111111	Don't even bother to compare; it's automatically considered to match (0 bits need to match).
0.0.15.255	00000000.00000000.00001111.11111111	Just the first 20 bits must match.
0.0.3.255	00000000.00000000.00000011.11111111	Just the first 22 bits must match.
32.48.0.255	00100000.00110000.00000000.11111111	All bits except the 3rd, 11th, 12th, and last 8 must match.

The first several examples show the typical use of the wildcard mask. As you can see, it is not a subnet mask. A wildcard of 0.0.0.0 means that the entire IP address must be examined, and be equal, in order to be considered a match. 0.0.0.255 means that the last octet automatically matches, but the first three must be examined, and so on. More generally, the wildcard mask means the following: Bit positions of binary 0 mean that the access list compares the corresponding bit position in the IP address and makes sure it is equal to the same bit position in the address configured in the **access-list** statement. Bit positions of binary 1 are wildcards—those bit positions are immediately considered to be a match.

The next two rows of table show two reasonable but not obvious wildcard masks. 0.0.5.255, as seen in binary, is 20 0s followed by 12 1s. This means that the first 20 bits must match. Similarly, 0.0.3.255 means that the first 22 bits must be examined to find out if they match. Why are these useful? If the subnet mask is 255.255.240.0, and you want to match all hosts in the same subnet, the 0.0.15.255 wildcard means that all network and subnet bits must be matched, and all host bits are automatically considered to match. Likewise, if you want to filter all hosts in a subnet that uses subnet mask 255.255.252.0, the wildcard mask 0.0.3.255 matches the network and subnet bits. In general, if you want a wildcard mask that helps you match all hosts in a subnet, invert the subnet mask, and you have the correct wildcard mask.

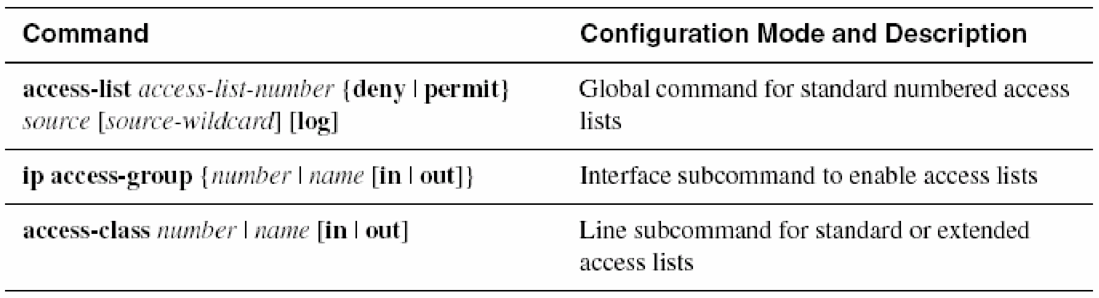
The last entry in table is unreasonable for real networks, but it is included to make a point. The wildcard mask just defines which bits must be compared and which are automatically assumed to match. You should not expect such strange masks on the exam! The point is that although subnet masks must use a sequential set of binary 1s followed by only binary 0s, wildcard masks do not have to follow any such rule.

6.2 Standard IP Access List Configuration

Standard IP access list configuration works much like a simple programming language. The logic is something like this: If statement 1 is matched, carry out the action defined in that statement. If it isn't, examine the next statement. If it matches, carry out the action it defines. Continue looping through the list until a statement is matched or until the last statement in the list is not matched. A standard access list is used to match a packet and then take the directed action. Each standard access list can match all or only part of the packet's source IP address. The only two actions taken when an **access-list** statement is matched are to either deny (discard) or permit (forward) the packet.

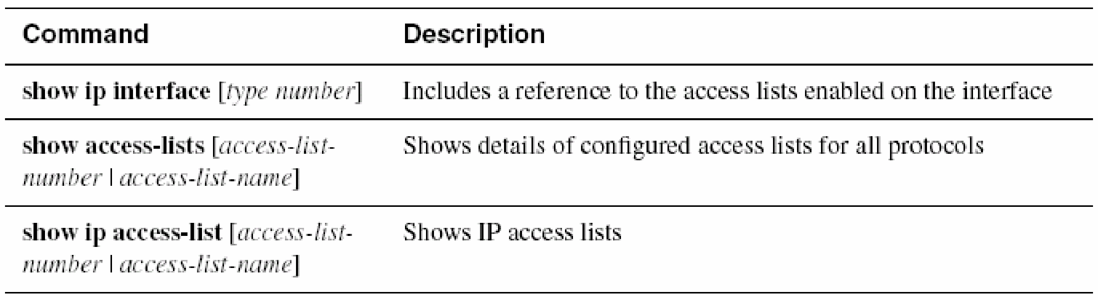
Following table lists the configuration commands related to standard IP access lists. Next table lists the related EXEC commands. Several examples follow the lists of commands.

Standard IP Access List Configuration Commands



Command	Configuration Mode and Description
access-list *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**]	Global command for standard numbered access lists
ip access-group {*number* | *name* [**in** | **out**]}	Interface subcommand to enable access lists
access-class *number* | *name* [**in** | **out**]	Line subcommand for standard or extended access lists

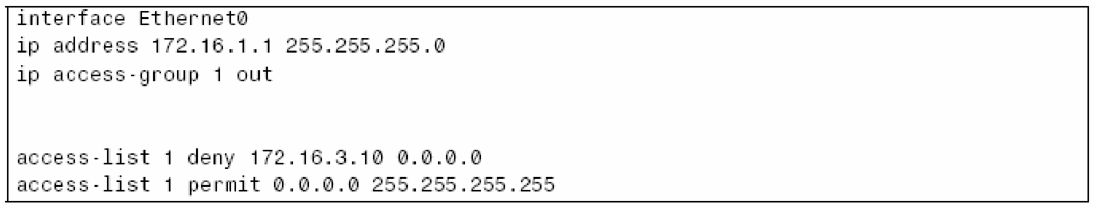
Standard IP Access List EXEC Commands



Command	Description
show ip interface [*type number*]	Includes a reference to the access lists enabled on the interface
show access-lists [*access-list-number* | *access-list-name*]	Shows details of configured access lists for all protocols
show ip access-list [*access-list-number* | *access-list-name*]	Shows IP access lists

The first example is basic in order to cover the statements' syntax. As shown in following figure , Bob is not allowed to access Server1, but Larry is allowed. In following example , the access list is enabled for all packets going out R1's Ethernet0 interface. Example 8-1 shows the configuration on R1.

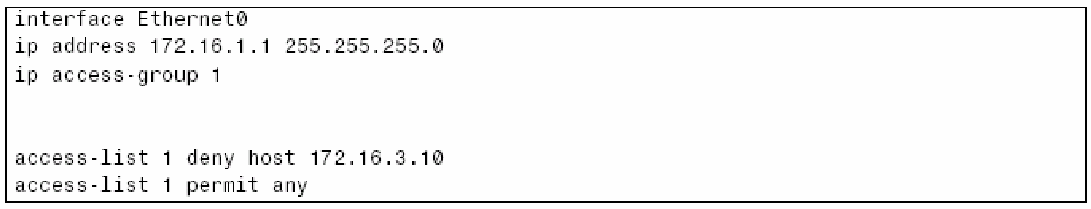
Standard Access List Stopping Bob from Reaching Server



```
interface Ethernet0
ip address 172.16.1.1 255.255.255.0
ip access-group 1 out


access-list 1 deny 172.16.3.10 0.0.0.0
access-list 1 permit 0.0.0.0 255.255.255.255
```

There are several small details in this example. Standard IP access lists use a number between 1 and 99, inclusive. Number 1 is used here for no particular reason, other than it's in the right range. The **access-list** command is a global configuration command, *not* a subcommand under the Ethernet0 interface. (The **access-list** commands do appear toward the end of the configuration file, after the interfaces.) The **ip access-group** command enables the logic on Ethernet0 for packets going out. Access list 1 stops packets sent by Bob from exiting R1's Ethernet interface based on the matching logic of the first **access-list** statement. It forwards any other packets based on the matching logic of the second statement. The configuration in example is not what shows up in the output of the **show running-config** command. Following example shows what would actually be placed in the configuration file.

Standard Access List Stopping Bob from Reaching Server1: Revised

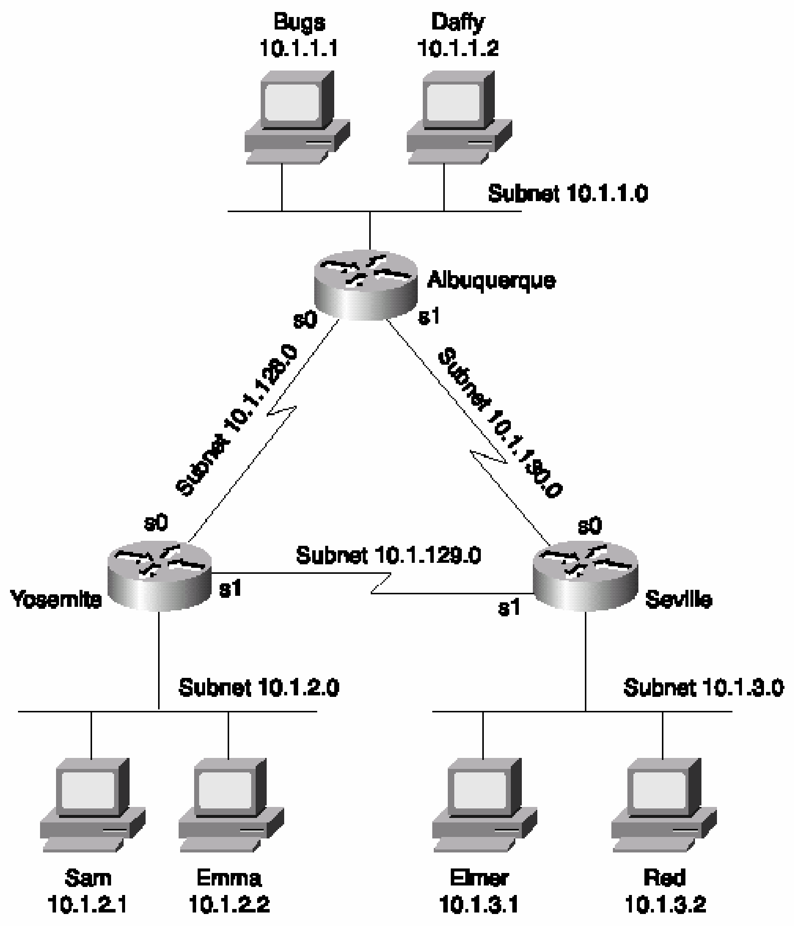


```
interface Ethernet0
ip address 172.16.1.1 255.255.255.0
ip access-group 1


access-list 1 deny host 172.16.3.10
access-list 1 permit any
```

The commands in first example are changed based on three factors. First, "out" is the default direction for access lists, so the router would omit the **out** keyword of the **ip access-group** command. Second, the use of a wildcard mask of 0.0.0.0 is the old way to configure an access list to match a specific host's IP address. The new style is to code the **host** keyword in front of the IP address. When you type a wildcard of 0.0.0.0, the router replaces the configuration with the newer **host** keyword. Finally, when you use an IP address and a wildcard mask of 255.255.255.255, the keyword **any** is used to replace both parameters. **any** simply means that any IP address is matched. The second example is more involved.

Following figure and examples show a basic use of standard IP access lists, with two typical oversights in the first attempt at a complete answer. The criteria for the access lists are as follows:
• Sam is not allowed access to Bugs or Daffy.
• Hosts on the Seville Ethernet are not allowed access to hosts on the Yosemite Ethernet.
• All other combinations are allowed.

Network Diagram for Standard Access List Example



Yosemite Configuration for Standard Access List Example

```
interface serial 0
ip access-group 3
!
access-list 3 deny host 10.1.2.1
access-list 3 permit any
```

Seville Configuration for Standard Access List Example

```
interface serial 1
ip access-group 4
!
access-list 4 deny 10.1.3.0   0.0.0.255
access-list 4 permit any
```

At first glance, these two access lists seem to perform the desired function. Criterion 1 is met in Yosemite. In Yosemite, the packets from Sam are filtered before leaving S0 using access list 3. Criterion 2 is met in Seville: Packets from 10.1.3.0/24 are filtered before leaving Seville’s S1 toward Yosemite, using access list 4.

Both routers meet criterion 3: A wildcard **permit any** is used at the end of each access list to override the default, which is to discard all other packets. So, all the criteria appear to be met. One subtle problem prevents this example from actually meeting the stated goals. If certain links fail, new routes are learned. For example, if the link from Albuquerque to Yosemite fails, Yosemite learns a route to 10.1.1.0/24 through Seville. Packets from Sam, forwarded by Yosemite and destined for hosts in Albuquerque, would leave Yosemite's serial1 interface without being filtered. Similarly, if the link from Albuquerque to Yosemite failed, Seville would route packets through Albuquerque, routing around the access list enabled on Seville. Following example illustrates an alternative answer to the stated problem. The access list has been removed from Seville, and all filtering is performed on Yosemite.

Yosemite Configuration for Standard Access List Example

```
interface serial 0
ip access-group 3
!
interface serial 1
ip access-group 3
!
interface ethernet 0
ip access-group 4
!
access-list 3 deny host 10.1.2.1
access-list 3 permit any
!
access-list 4 deny 10.1.3.0   0.0.0.255
access-list 4 permit any
```

The configuration in example solves the problem of the earlier example, but it creates another problem. It denies all traffic that should be denied, but it also denies more traffic than the first of the three criteria says it should! In many cases, the meaning of the criteria for the access lists greatly affects your configuration choices. In this example, the problem of Sam's traffic going through Seville to reach Albuquerque when the link directly to Albuquerque is down is solved. The access list denies traffic from Sam (10.1.2.1) in an outbound access list on both of Yosemite's serial interfaces. However, that also prevents Sam from communicating with anyone outside Yosemite. This does not meet the spirit of the filtering goals, because it filters more than it should. An alternative would be to use the same **access-list 3** logic but use it as an inbound access list on Albuquerque's serial interfaces. However, that achieves the real goal only if there are no other servers in Albuquerque that Sam should be allowed to access. And if that were the case, criterion 1 should be rewritten to say something like "Sam is not allowed to access devices on the Albuquerque Ethernet."

The main point is this: With three simple criteria and three routers, the configuration was simple. However, it is easy to introduce problems that are not obvious. As shown in example , **access-list 4** does an effective job of meeting the second of the three criteria. Because the goal was to stop Seville hosts from communicating with Yosemite's hosts, and because the only LAN hosts off Yosemite are the ones on the local Ethernet, the access list is effective in stopping packets from exiting Ethernet 0.

6.3 Extended IP Access Lists

Extended IP access lists are almost identical to standard IP access lists in their use. The key difference between the two is the variety of fields in the packet that can be compared for matching by extended access lists. To pass the CCNA exam, you must remember all the items that an extended IP access list can check to make a match. As with standard lists, extended access lists are enabled for packets entering or exiting an interface. The list is searched sequentially; the first statement matched stops the search through the list and defines the action to be taken. All these features are true of standard access lists as well. The matching logic, however, is different than that used with standard access lists and makes extended access lists much more complex.

Following table lists the configuration commands associated with creating extended IP access lists. Next table lists the associated EXEC commands. Several examples follow the lists of commands.

Extended IP Access List Configuration Commands

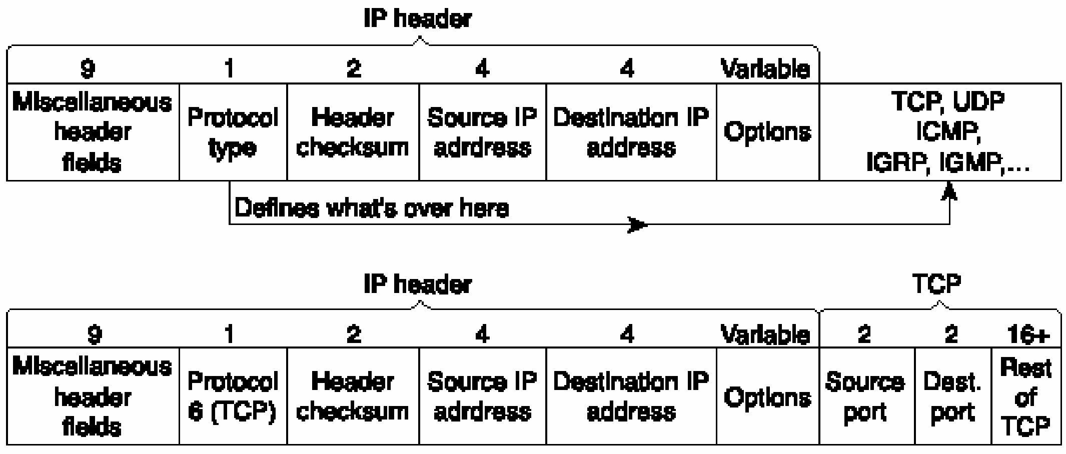
Command	Configuration Mode and Description
access-list *access-list-number* [**dynamic** *dynamic-name* [**timeout** *minutes*]] {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**tos** *tos*] [**log** | **log-input**] [**time-range** *time-range-name*]	Global command for extended numbered access lists
ip access-group {*number* | *name* [**in** | **out**]}	Interface subcommand to enable access lists
access-class *number* | *name* [**in** | **out**]	Line subcommand for standard or extended access lists

Extended IP Access List EXEC Commands

Command	Description
show ip interface [*type number*]	Includes a reference to the access lists enabled on the interface
show access-lists [*access-list-number* | *access-list-name*]	Shows details of configured access lists for all protocols
show ip access-list [*access-list-number* | *access-list-name*]	Shows IP access lists

Extended access lists create powerful matching logic by examining many parts of a packet. Following figure shows several of the fields in the packet headers that can be matched. The top set of headers shows the IP protocol type, which identifies what header follows the IP header. The source and destination IP addresses are also shown. In the second set of headers, an example with a TCP header following the IP header is shown. The TCP source and destination port numbers are listed in the abbreviated TCP header. Following table provides the complete list of items that can be matched with an IP extended access list.

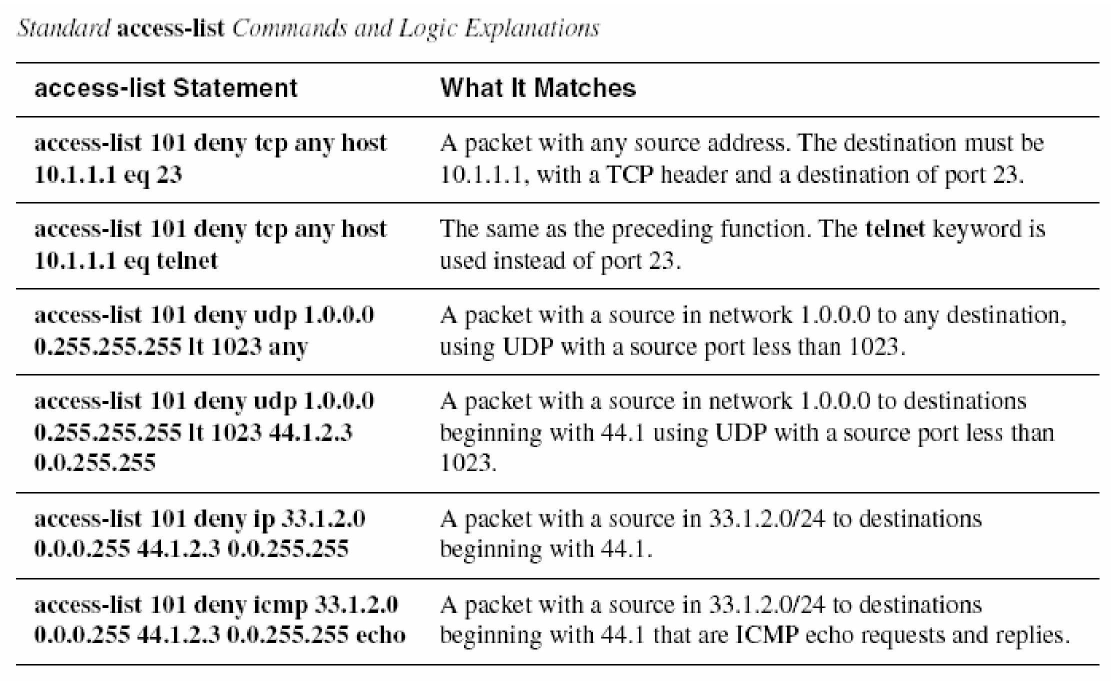
Extended Access List Matching Options



Standard and Extended IP Access Lists: Matching

Type of Access List	What Can Be Matched
IP standard	Source IP address
	Portions of the source IP address using a wildcard mask
IP extended	Source IP address
	Portions of the source IP address using a wildcard mask
	Destination IP address
	Portions of the destination IP address using a wildcard mask
	Protocol type (TCP, UDP, ICMP, IGRP, IGMP, and others)
	Source port
	Destination port
	Established—matches all TCP flows except the first
	IP TOS
	IP precedence

A statement is considered to match if all options in the statement match. If one option does not match, the statement is skipped, and the next entry in the list is examined. Following table provides several sample **access-list** statements.

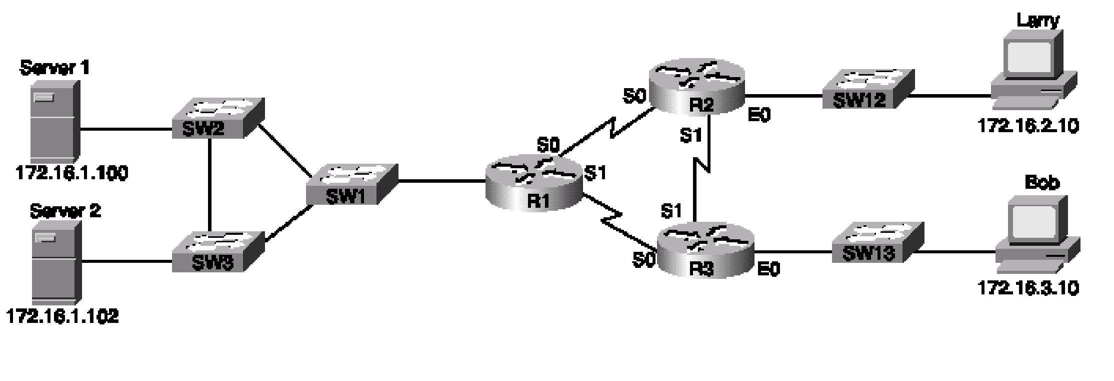

Standard **access-list** *Commands and Logic Explanations*

access-list Statement	What It Matches
access-list 101 deny tcp any host 10.1.1.1 eq 23	A packet with any source address. The destination must be 10.1.1.1, with a TCP header and a destination of port 23.
access-list 101 deny tcp any host 10.1.1.1 eq telnet	The same as the preceding function. The **telnet** keyword is used instead of port 23.
access-list 101 deny udp 1.0.0.0 0.255.255.255 lt 1023 any	A packet with a source in network 1.0.0.0 to any destination, using UDP with a source port less than 1023.
access-list 101 deny udp 1.0.0.0 0.255.255.255 lt 1023 44.1.2.3 0.0.255.255	A packet with a source in network 1.0.0.0 to destinations beginning with 44.1 using UDP with a source port less than 1023.
access-list 101 deny ip 33.1.2.0 0.0.0.255 44.1.2.3 0.0.255.255	A packet with a source in 33.1.2.0/24 to destinations beginning with 44.1.
access-list 101 deny icmp 33.1.2.0 0.0.0.255 44.1.2.3 0.0.255.255 echo	A packet with a source in 33.1.2.0/24 to destinations beginning with 44.1 that are ICMP echo requests and replies.

The sequence of the parameters is very important—and very tricky, in some cases. When checking port numbers, the parameter on the **access-list** command checking the port checks the source port number when placed immediately after the check of the source IP address. Likewise, if the port parameter follows the check of the destination address, the logic matches the destination port. For example, the command **access-list 101 deny tcp any eq telnet any** matches all packets that use TCP and whose source TCP port is 23 (Telnet). Likewise, the command **access-list 101 deny tcp any any eq telnet** matches all packets that use TCP and whose destination TCP port is 23 (Telnet).

6.3.1 Extended IP Access List – example

The first example is basic in order to cover the statements'' syntax. In this case, Bob is denied access to all FTP servers on R1's Ethernet, and Larry is denied access to Server1's Web server. Following figure is a reminder of the network topology. In the example , an access list is created on R1. Example 8-6 shows the configuration on R1.



R1's Extended Access List:

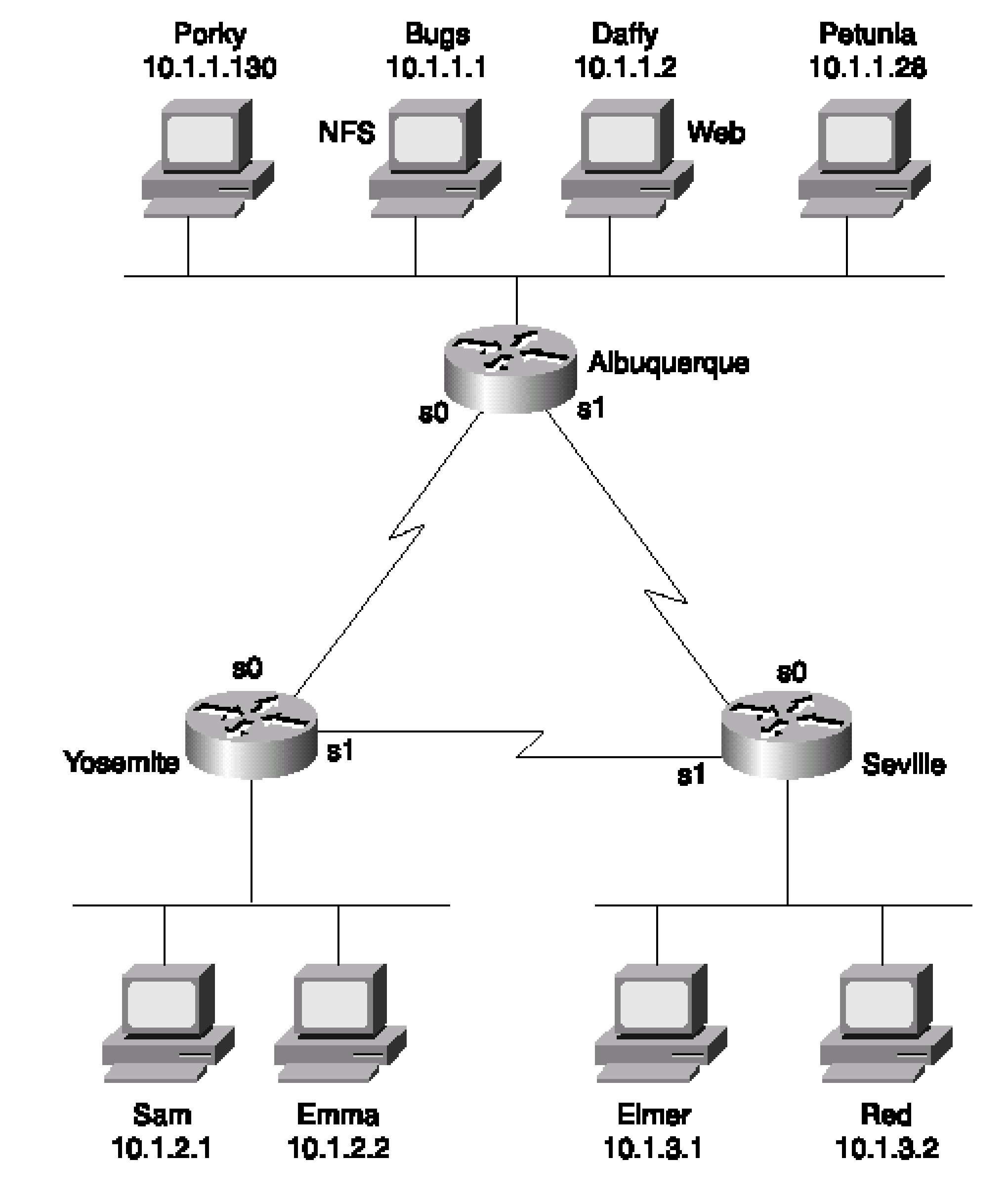
```
interface Serial0
ip address 172.16.12.1 255.255.255.0
ip access-group 101 in

interface Serial1
ip address 172.16.13.1 255.255.255.0
ip access-group 101 in

access-list 101 deny tcp host 172.16.3.10 172.16.1.0 0.0.0.255 eq ftp
access-list 101 deny tcp host 172.16.2.10 host 172.16.1.100 eq http
access-list 101 permit ip any any
```

Focusing on the syntax for a moment, there are several new items to review. First, the access list number for extended access lists is from 100 to 199, inclusive. A protocol parameter is the first option after the permit or deny action. When checking for TCP or UDP port numbers, the TCP or UDP protocol must be specified. The **eq** parameter means "equals." It implies that you are checking the port numbers—in this case, the destination port numbers. You can use the numeric values—or, for the more popular options, a more obvious text version is valid. (If you were to type **eq 80**, the config would show **eq http**.)

R3's Extended Access List Stopping Bob from Reaching FTP Servers Near R1

```
interface Serial0
ip address 172.16.13.3 255.255.255.0
ip access-group 101 out

interface Serial1
ip address 172.16.12.3 255.255.255.0
ip access-group 101 out

access-list 101 deny tcp host 172.16.3.10 172.16.1.0 0.0.0.255 eq ftp
access-list 101 permit ip any any
```

Example 2



• The Web server (Daffy) is available to all users.
• UDP-based clients and servers on Bugs are unavailable to hosts whose IP addresses are in the upper half of the valid IP addresses in each subnet. (The subnet mask used is 255.255.255.0.)
• Packets between hosts on the Yosemite Ethernet and the Seville Ethernet are allowed only if packets are routed across the direct serial link.
• Clients Porky and Petunia can connect to all hosts except Red.
• Any other connections are permitted.
(To be done at the class)

6.4 Named IP Access Lists

Named IP access lists allow the same logic to be configured as with numbered standard and extended access lists. As a CCNA, you will need to remember the configuration commands' syntax differences and also be able to create both numbered and named lists with the same logic. The key differences between numbered and named IP access lists are as follows:

• A name is a more intuitive reminder of a list's function.

• Names allow for more access lists than 99 standard and 100 extended, which is the restriction with numbered access lists.

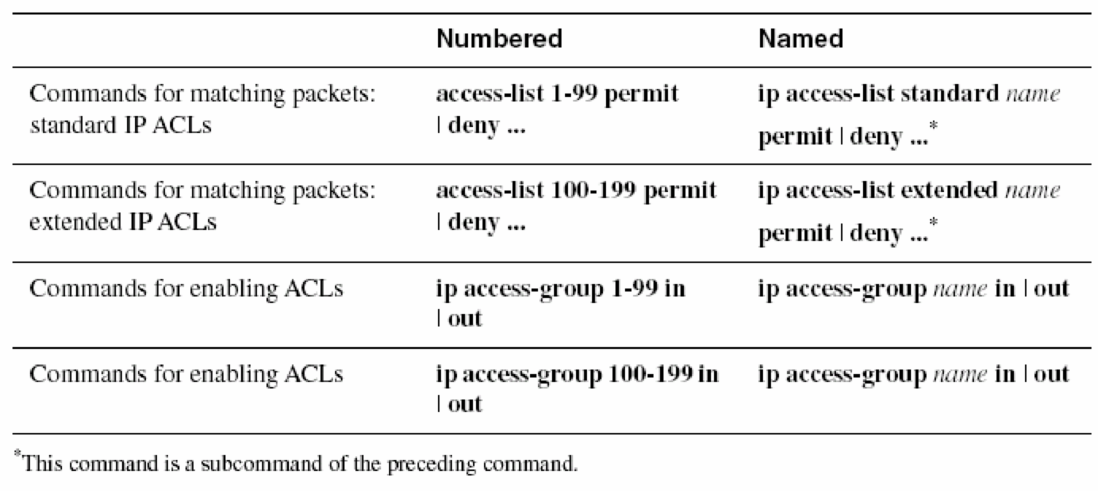
• Named access lists allow individual statements to be deleted. Numbered lists allow only for the deletion of the entire list. Insertion of the new statement into a named list requires the deletion and re-addition of all statements that should be later in the list than the newly added statement.

• The actual names used must be unique across all named access lists of all protocols and types on an individual router. Names can be duplicated on different routers.

The configuration syntax is very similar between named and numbered IP access lists. The items that can be matched with a numbered standard IP access list are identical to the items that can be matched with a named standard IP access list. Likewise, the items are identical with both numbered and named extended IP access lists. Two important differences exist between numbered and named access lists. One key difference is that named access lists use a global command, which moves the user into a named IP access list submode under which the matching and permit/deny logic is configured. The other key difference is that when a named matching statement is deleted, only that one statement is deleted. With numbered lists, the deletion of any statement in the list deletes all the statements in the list. (This feature is demonstrated in more detail in an upcoming example.)

Following table lists the key configuration commands and shows their differences and similarities.

Comparison of Named and Numbered IP Access List Configuration Commands

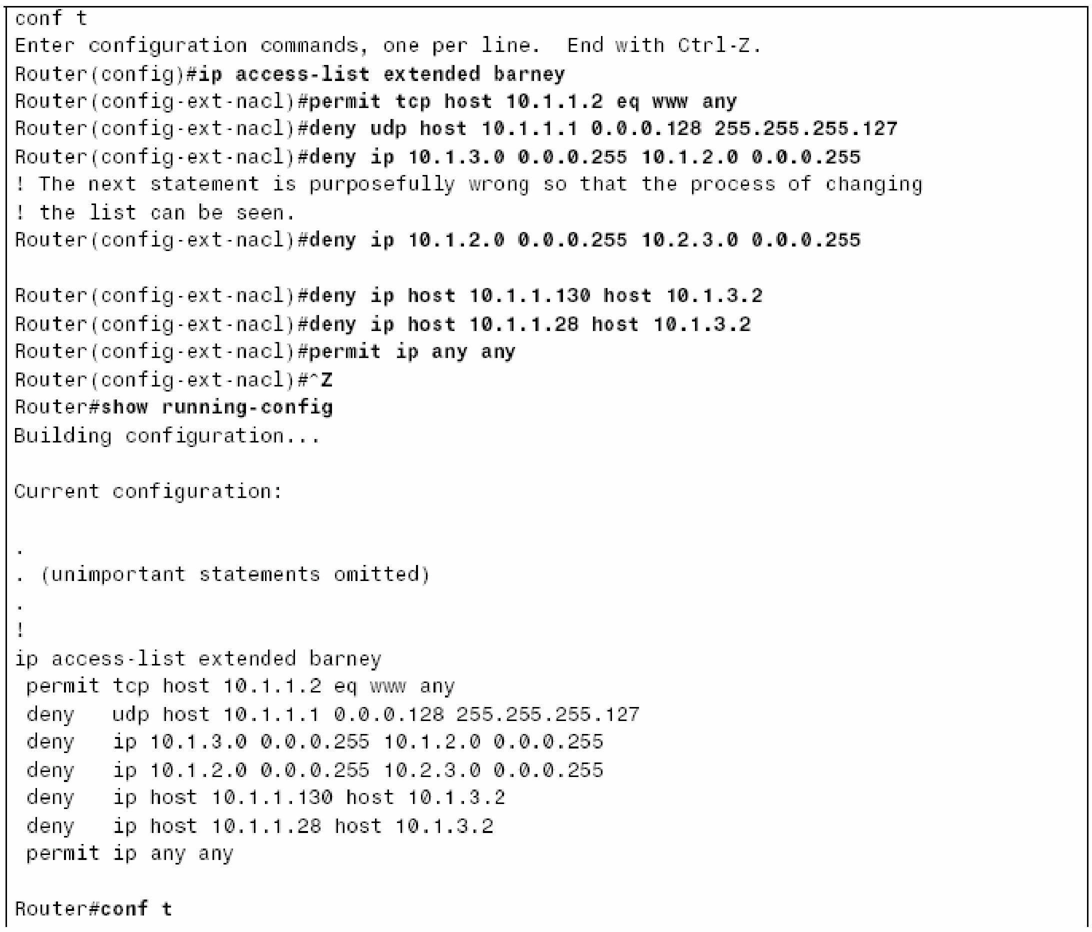


	Numbered	Named
Commands for matching packets: standard IP ACLs	**access-list 1-99 permit | deny ...**	**ip access-list standard** *name* **permit | deny ...**[*]
Commands for matching packets: extended IP ACLs	**access-list 100-199 permit | deny ...**	**ip access-list extended** *name* **permit | deny ...**[*]
Commands for enabling ACLs	**ip access-group 1-99 in | out**	**ip access-group** *name* **in | out**
Commands for enabling ACLs	**ip access-group 100-199 in | out**	**ip access-group** *name* **in | out**

[*]This command is a subcommand of the preceding command.

The word *name* represents a name created by the administrator. This name must be unique among all named access lists of all types in this router. Also, note that because the named list does not imply standard or extended by the value of the list's number, the command explicitly states the type of access list. Also, the ... represents all the matching parameters, which are identical in meaning and syntax when comparing the respective numbered and named IP access lists. Also note that the same command is used to enable the list on an interface for both numbered and named lists. One difference between the two types of lists is that individual matching statements can be removed from named lists. Following example shows the configuration mode output when entering the access list used on Albuquerque in access list 112 of previous example , but this time as a named access list instead of a numbered access list. One typo is shown in the original creation of the access list in Example 8-13, with changes made to delete and add the statement shown later in this same example. (The statement that is a typo is **deny ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255**. It is a typo because there is no subnet 10.2.3.0; the intent was to configure 10.1.3.0 instead.)

Named Access List Configuration



```
conf t
Enter configuration commands, one per line.  End with Ctrl-Z.
Router(config)#ip access-list extended barney
Router(config-ext-nacl)#permit tcp host 10.1.1.2 eq www any
Router(config-ext-nacl)#deny udp host 10.1.1.1 0.0.0.128 255.255.255.127
Router(config-ext-nacl)#deny ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
! The next statement is purposefully wrong so that the process of changing
! the list can be seen.
Router(config-ext-nacl)#deny ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255

Router(config-ext-nacl)#deny ip host 10.1.1.130 host 10.1.3.2
Router(config-ext-nacl)#deny ip host 10.1.1.28 host 10.1.3.2
Router(config-ext-nacl)#permit ip any any
Router(config-ext-nacl)#^Z
Router#show running-config
Building configuration...

Current configuration:

.
. (unimportant statements omitted)
.
!
ip access-list extended barney
 permit tcp host 10.1.1.2 eq www any
 deny   udp host 10.1.1.1 0.0.0.128 255.255.255.127
 deny   ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
 deny   ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255
 deny   ip host 10.1.1.130 host 10.1.3.2
 deny   ip host 10.1.1.28 host 10.1.3.2
 permit ip any any

Router#conf t
```

Named Access List Configuration (Continued)

```
Enter configuration commands, one per line.  End with Ctrl-Z.
Router(config)#ip access-list extended barney
Router(config-ext-nacl)#no deny ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255
Router(config-ext-nacl)#^Z
Router#show access-list

Extended IP access list barney
    permit tcp host 10.1.1.2 eq www any
    deny   udp host 10.1.1.1 0.0.0.128 255.255.255.127
    deny   ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
    deny   ip host 10.1.1.130 host 10.1.3.2
    deny   ip host 10.1.1.28 host 10.1.3.2
    permit ip any any
Router#conf t
Enter configuration commands, one per line.  End with Ctrl-Z.
Router(config)#ip access-list extended barney
Router(config-ext-nacl)#no permit ip any any
Router(config-ext-nacl)#no deny ip host 10.1.1.130 host 10.1.3.2
Router(config-ext-nacl)#no deny ip host 10.1.1.28 host 10.1.3.2
Router(config-ext-nacl)#deny ip 10.1.2.0 0.0.0.255 10.1.3.0 0.0.0.255
Router(config-ext-nacl)#deny ip host 10.1.1.130 host 10.1.3.2
Router(config-ext-nacl)#deny ip host 10.1.1.28 host 10.1.3.2
Router(config-ext-nacl)#permit ip any any
Router(config-ext-nacl)#^Z
Router#show ip access-list

Extended IP access list barney
    permit tcp host 10.1.1.2 eq www any
    deny   udp host 10.1.1.1 0.0.0.128 255.255.255.127
    deny   ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
    deny   ip 10.1.2.0 0.0.0.255 10.1.3.0 0.0.0.255
    deny   ip host 10.1.1.130 host 10.1.3.2
    deny   ip host 10.1.1.28 host 10.1.3.2
    permit ip any any
```

If an access list is not configured but is enabled on an interface with the **ip access-group** command, no packets are filtered because of this command. After the access list's first command is configured, Cisco IOS software implements the access list's logic. This is true of IP standard access lists as well as extended and named access lists. Access lists that filter other types of packets follow this same logic.

6.5 Controlling vty Access with IP Access Lists

Access into and out of the virtual terminal line (vty) ports of the Cisco IOS software can be controlled by IP access lists. (vty is used for Telnet access to and from the Cisco IOS software.) The inbound case is the more obvious case. For instance, imagine that only hosts in subnet 10.1.1.0/24 are supposed to be capable of Telnetting into any of the Cisco routers in a network.

In such a case, the configuration in following example could be used on each router to deny access from IP addresses not in that one subnet.

vty Access Control Using the **access-class** *Command*

```
line vty 0 4
 login
 password cisco
 access-class 3 in
!
! Next command is a global command
 access-list 3 permit 10.1.1.0 0.0.0.255
```

The **access-class** command refers to the matching logic in **access-list 3**. The keyword **in** refers to packets that are entering the router when you are trying to Telnet to that router’s vtys. The **out** keyword is used both with outbound Telnet from a router and when using the reverse Telnet feature of the Cisco IOS software (which is unlikely to be on the exam). The **out** keyword implies that the packets originated by the Telnet client in the router are checked using the packets’ destination address.

7 - Network Security

7.1 Introduction

The network security mainly depends on the network architecture and the network devices and servers are to be located at correct positions.

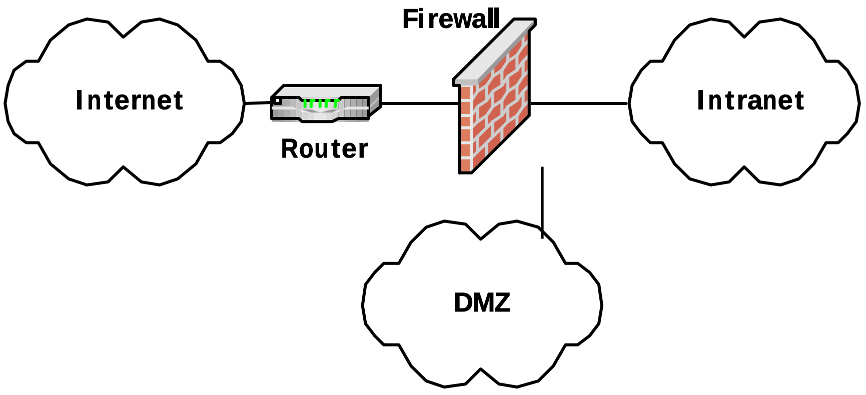


Figure 7-1 Typical Network Architecture

In the network architecture shown in **figure 7-1** has three main areas or zones for the firewall. All internal servers are placed in the Intranet. The servers which are accessed by internal users as well as external users are placed in the De-Maliterized Zone [DMZ].

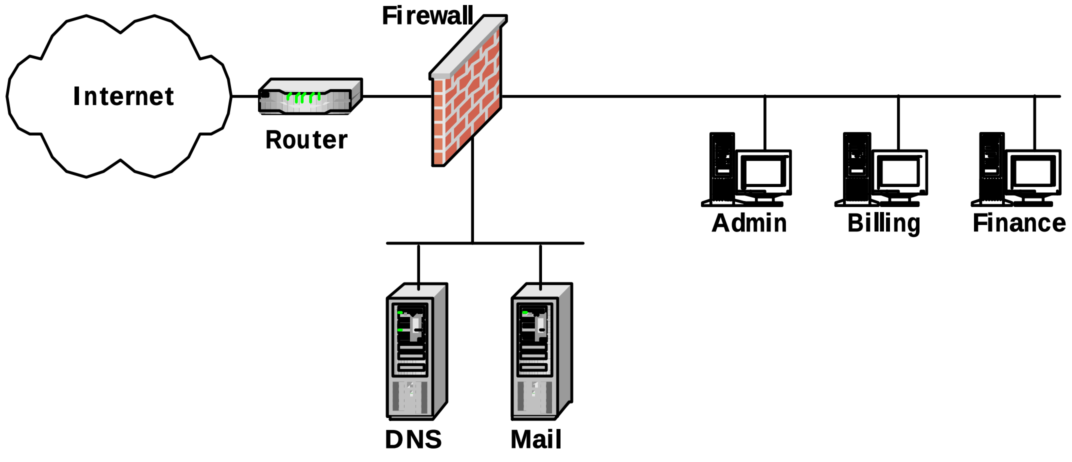


Figure 7-2 Typical Server Setting

A typical security setting is shown in the **figure 7-2**. The Administration, Billing, Finance, Inventory Control Servers are in the Intranet. The Mail Server, DNS Server can be located in the DMZ

We can improve the security by using more than one firewall as shown in the figure 7-3.

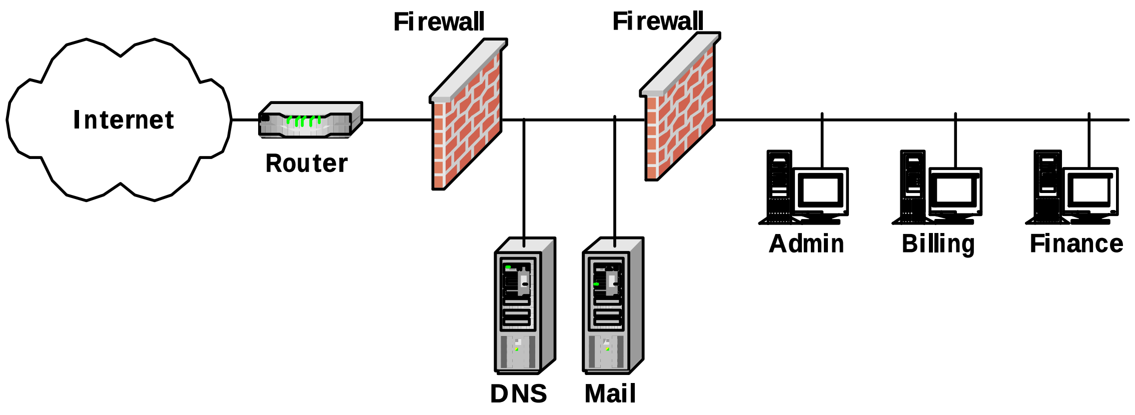


Figure 7-3 More Secured Setting

The network security can be established in different ways by using following concepts or devices.

- Firewall
- Intrusion Detection System
- Routers with ACLS.
- Switches with port security
- VLANs.
- Directory Services.
- Single sign on.

7.2 Firewalls

Firewalls are used to create security checkpoints at the boundaries of private networks. By providing the routing function among Intranet, DMZ and Internet, firewall inspect all packets passing among the networks and either pass or drop the packets depending on the programmed policy rules.

7.3 TCP/IP Model

In order to understand the configuration of Firewall clear understanding of TCP header and IP header are very important.

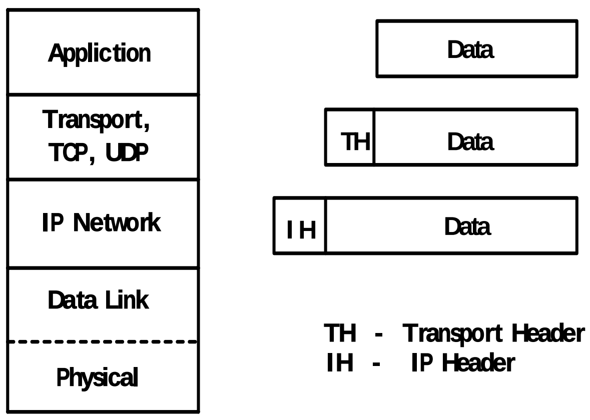


Figure 7-4 TCP/IP Model

CP/IP protocol suite is shown in the **figure 7-4** TCP/IP is a hierarchical protocol made up of interactive modules each of which provides a specific functionality. The application layer sends data to Transport Layer. It adds the TCP header or UDP header which depends on the Application. This will send to the Network Layer. It adds the IP header and sends to Data Link Layer [DLL] which depends on the LAN or WAN protocol

7.3.1 TCP Header

The standard size of TCP header is 20 bytes. The two important fields for firewall is, source port number and destination port number. Different applications are given different port numbers. The **table 1-1** shows some of the port numbers and their corresponding applications.

TCP Port Number	Application
20	FTP Data
23	Telnet

25	SMTP
80	HTTP

Table 5-1 TCP Ports

7.3.2 IP Header

The standard size of IP header is 20 bytes.The three important fields for firewall are, source IP address, destination IP address, and protocol. Some of the protocols are indicted in the **table 7-2**.

Protocol Number	Protocol
1	ICMP
2	IGMP
6	TCP
8	UDP
89	OSPF

Table 7-2 IP Protocols

Some of the security protocols used in TCP/IP are given in **table 7-3**

Protocol	Definition
AFJ	Authenticated firewall traversal
CAJ	Common authentication technology
DNSSEC	Domain Name Service Security
IPSec	Internet Security Protocol.
IDWG	Intrusion Detection Working Group
TLS	Transport Layer Security
WTS	Web Transaction Security

Table 7-3 TCP/IP Security Protocols

7.4 Firewall Functions

Firewalls primarily function using three fundamental methods.

- Packet Filtering.
- Network Address Translation [NAT]

Most firewalls also perform two other important security services.

- Encrypted Authentication
- Encrypted Tunnels.

7.4.1 Packet Filtering

There are two types of packet filtering.

- Standard or stateless packet filtering.
- Stateful inspection packet filters.

7.4.2 Stateless Packet Filters

In this method it is decided whether or not to forward a packet based on information contained in the header of individual packet. Normally the following fields are used to filter.

7.5 Protocol Filtering

The filtering is based on the protocol field of IP packet. The normally used protocols are,

- ICP
- UDP
- ICMP
- IGMP

7.6 IP Address Filtering

This filtering is based on the source or destination IP addresses. It deny or permit access based on the IP address.

7.7 TCP/UDP Ports

TCP or UDP port information is the most commonly used information filter on because this data field indicates most specifically what the packet is for. Some of the common protocols that can be filtered based on TCP or UDP port are, FTP, Telnet, SMTP, HITP, DNS, SNMP, POP and NFS.

In addition to the standard fields, headers contain other information that can be used to determine whether or not a packet should be passed. Source routing and fragmentation are two techniques supported by the IP protocol.

Source routing is the process of defining the exact route a packet must take between hosts in an IP connection. This feature is now frequently used by hackers. Unless use the source routing in the network the firewall should be configured to drop any source routed packets

The problem with fragmentation comes from the fact that the most useful filter data, the TCP or UDP port number is only provided at the beginning of an IP packet, so it will only be contained in fragment 0.

Packet filters suffer from two problems.

- They cannot check the payload of packers.
- They do not retain the state of connection.

7.7.1 Stateful Packet Filters

Stateful packet filters remember the state of connections at the network and session layers by recording the session establishment information that passes through the firewall. The filters then use that information to discriminate valid packets from invalid connection attempts or hacking.

7.7.2 Network Address Translation

Network Address Translation [NAT] converts private IP addresses in the network to public IP addresses as shown in **figure** 7-5

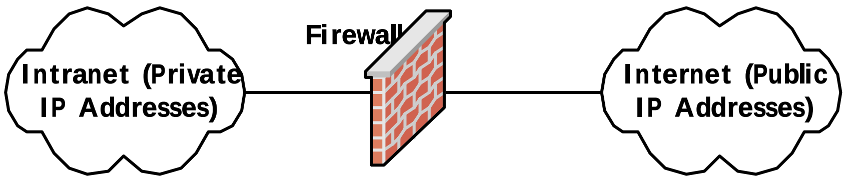

Intranet (Private IP Addresses)
Firewall
Internet (Public IP Addresses)

Figure 7-5 Network Address Translation

NAT effectively hides all TCP/IP level information about internal hosts. There are two types of NATs.

Static NAT

One to one permanent mapping of an private IP Address to public IP address is called static mapping. If the outsiders wants to access to internal servers [normally located in DMZ] needs this type of NAT.

Dynamic NAT

Private IP address arbitrarily select a public IP address from the pool is called dynamic mapping. In this way the public IP addresses can be used effectively.

7.8 Security Analysis Tools

Security analysis tools scan target hosts and determine which known bugs or vulnerabilities our machines susceptible to. Some of them are,

- SATAN
- WS-Ping

- Sniffer Basic
- Nessus
- LANguard

7.9 Commercial Firewalls

There are many types of Commercial Firewalls. Some of them are given below.

- Pix firewall – CISCO
- Lucent Managed firewall
- SonicWALL
- NetScreen 10x100
- AltaVista Firewall – Digital Equipment Cooperation.
- SecurIT Firewall
- Watchguard FireBox II

7.10 Intrusion Detection System (IDS)

7.10.1 What is intrusion detection?

Intrusion detection is the process of identifying and responding to malicious activity targeted at computing and networking resources.

The processes involve time and interaction among entities. The identification can be done before, during or after the malicious activity proceeds. If the identification is done before the damage can be prevented. If the identification is done during the occurrence, then the decisions must be made about whether to allow target activity to proceed and whether to create alarms. If the identification is done after the malicious activity completes, then questions of what damage might have been done and how such actions could have occurred must be addressed.

The main goals of an IDS are,

- Defect a wide variety of intrusions.
- Defect intrusions in a timely fashion.

- Present the analysis in a simple easy to understand format.
- Be accurate.

7.10.2 Basic concept in Intrusion Detection

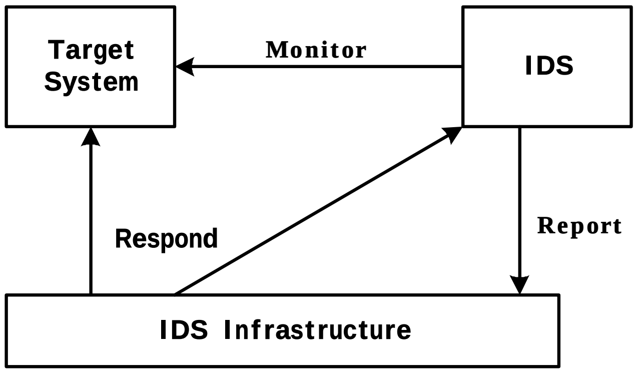


Figure7-6 Simple Depiction of Intrusion Detection Concept

The diagram in **figure** 7-6. Shows three article functions in the basic intrusion detection concept.

Monitor

Intrusion detection system examine and process information about target system.

Report

Intrusion detection system report information about monitored systems into a system security and protection infrastructure.

Respond

When risk related information is made available by the intrusion detection system, an associated response function initiates mitigation activities.

7.10.3 Architecture

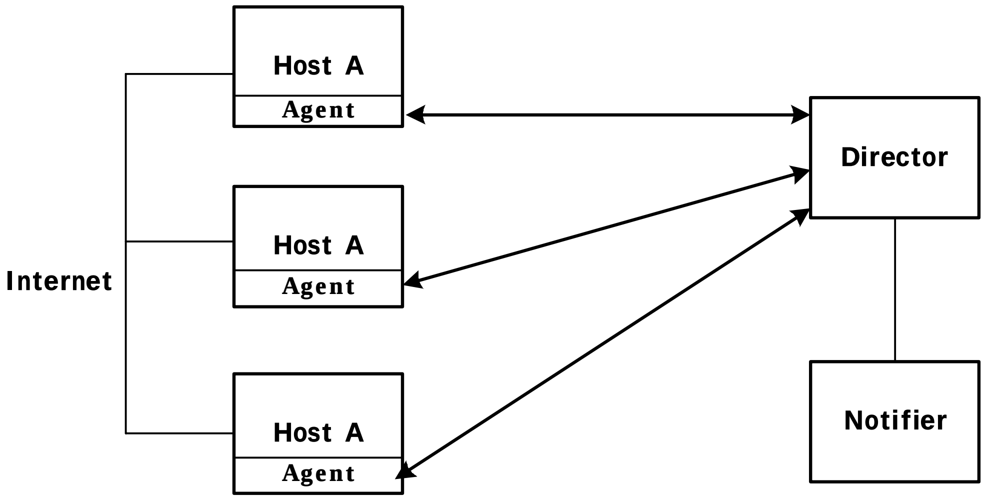


Figure 7-7 Agent , Director and Notifier

An intrusion detection system is also an automated auditing mechanism. It consists of three parts, Agent, Director and Notifier as shown in figure 7-7.

An agent obtains information from a data source. The source may be a log file, another process or a network. The information, once acquired, may be send directly to the director.

The director may determine that if needs more information from a particular information source. In that case, the director can instruct the agent to collect additional data, or to process the data it collects differently. It then uses an analysis engine may use any of, or a mixture of, several techniques to perform its analysis.

The notifier accepts information from the director and takes the appropriate action.

7.10.4 Intrusion Handling

Once the intrusion is defeated, how can the system be protected? The intrusion response deals with this problem. Handling the intrusion means restoring the system to comply with the site security policy and taking actions against the attacker that the policy specifies. Intrusion handling consists of six phases.

- Preparation for an attack.
- Identification of an attack.
- Containment of the attack.
- Eradication of the attack.
- Recovery from the attack.
- Follow up to the attack

7.11 Switches with port security

The simplest network device is the switch. Its ports can be configured to provide security.

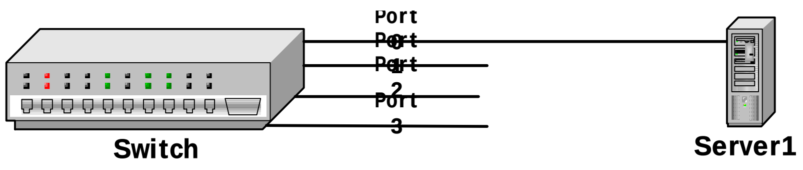


Figure 5-8 L2 Switch with port security

As shown in the **figure 5-8** it can be configured that port 0 is connected only to Server 1. Also the Server 1 is allowed to access from a particular port or ports.

7.12 VLANS

In a large LAN similarly type of users are not in the same location. For instance the Finance Department can be spreaded in Floor 1, Floor 2 etc. In such case a Virtual LAN (VLAN) can be created. A VLAN groups a number of ports on one or more switches together into a logical domain. VLANs offer a way to protect traffic between nodes in the same network. There are two types of VLANs; port based or policy based. A port based VLAN groups ports together regardless of the physical location. A policy based VLAN groups ports together after examining the contents of the packets slowing in to a port. Grouping are based on, for example, the source MAC address of a packet. A VLAN may be identified with a tag [a VLAN ID] as per the IEEE 802.12 specification, which enables spanning of a VLAN across multiple switches. Regardless of the physical location member of the same VLAN can easily communicate with one another. Thus, VLANs provide security through logical partitioning of a physical network into membership domains.

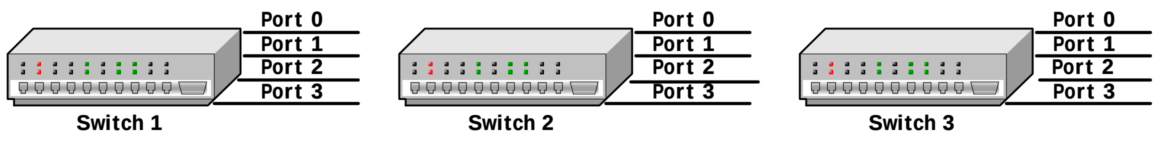


Figure 7-9 VLAN Configuration with three L2 switches

The **figure 7-9** shows three switches in a network. For example the switch 1 ports 1,2, switch 2 ports 3,4 and switch 3 port 1 can be established one VLAN.