This chapter covers the following aspects of router interface configuration:

- Interface connectivity and IP addressing
- Frame Relay overview
- Configuring Frame Relay interfaces
- Configuring point-to-point serial interfaces
- Configuring Ethernet interfaces
- Configuring Token Ring interfaces
- Creating and configuring loopback interfaces
- Cisco Discovery Protocol (CDP)

CHAPTER 7

Router Interface Configuration Methodology

This chapter covers the methodology for configuring the router interfaces. The chapter begins with configuring the Frame Relay interfaces and then continues with configuring any other type of serial interfaces, Ethernet interfaces, and finally Token Ring interfaces. When you are familiar with how to configure the various interface types, you will learn how to create and configure loopback interfaces. The chapter concludes with a brief overview of the Cisco Discovery Protocol (CDP). CDP allows directly connected routers to learn of each other and their capabilities. It is also a very good tool in determining whether you have Layer 1 and Layer 2 connectivity.

Interface Connectivity and IP Addressing

The approach used in this chapter to configuring each of the router interfaces is based on types of interfaces. For example, you will configure all the Frame Relay interfaces first. You will enter all the necessary commands to bring up the Frame Relay connection and assign an IP address to the interface, and then you will verify connectivity to all directly connected interfaces by using the **ping** command.

After the Frame Relay interfaces are configured, you will configure all of the point-to-point serial interfaces and assign them IP addresses; then you'll move on to Ethernet and then Token Ring. This will give you the opportunity to focus on each of the data link layer technologies and how to configure them. This is an important section of the lab because you cannot configure the different routing protocols until you have all interfaces configured with IP addresses and have complete IP connectivity to directly connected interfaces. (*Directly connected interfaces* refers to router interfaces connected to the same physical segment.)

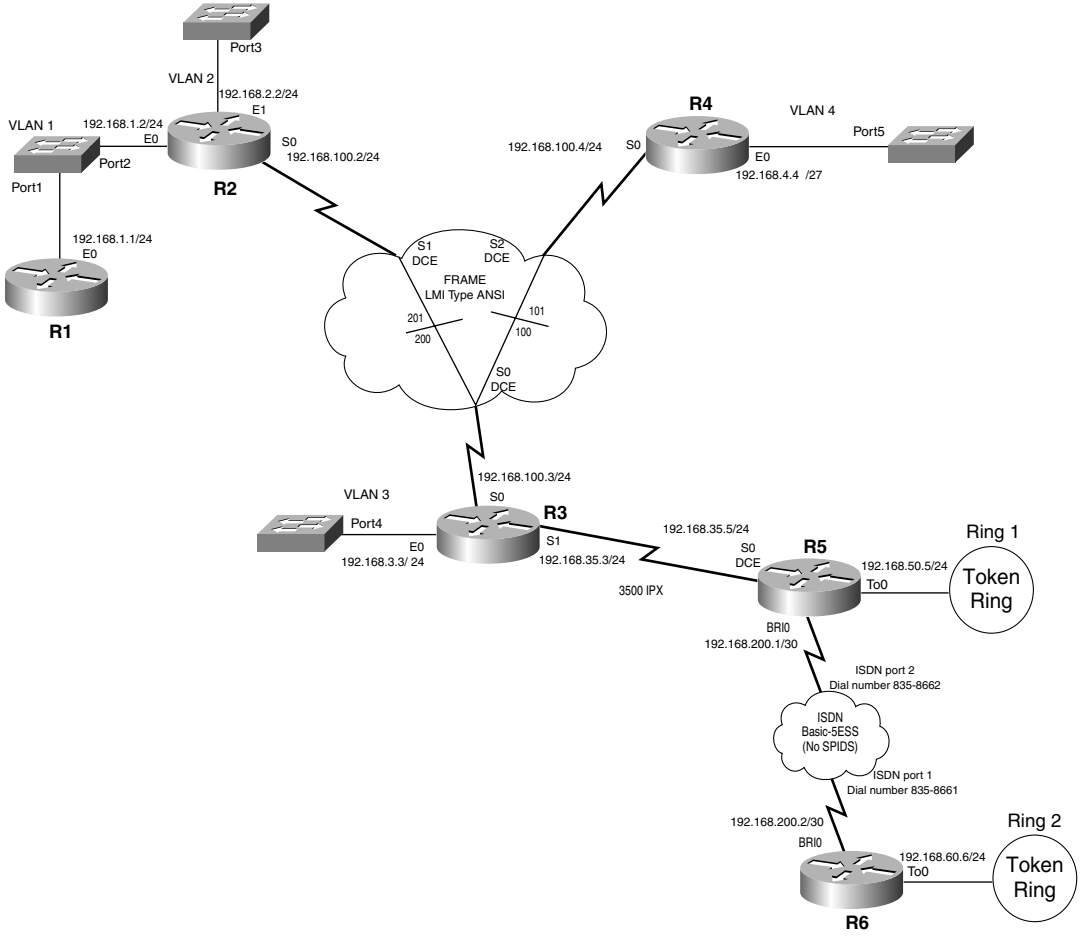
The lab objectives to complete in this chapter are as follows:

- Configure all routers to have a description on all active interfaces (except loopback interfaces) stating the router and interface to which they are connected.
- Create loopback interfaces on all routers. Use IP address 192.169.*X*.*X*/24 (where *X* is the router number). So, R1 would have a loopback address of 192.169.1.1/24, R2 would be 192.169.2.2/24, and so on.

IP Addressing

Refer to the lab diagram in Figure 7-1 for IP addressing assignments. Don't forget to look at the network mask. The BRI ports on R5 and R6 should use the IP addresses 192.168.200.1/30 and 192.168.200.2/30, respectively.

Figure 7-1 *Lab Diagram*



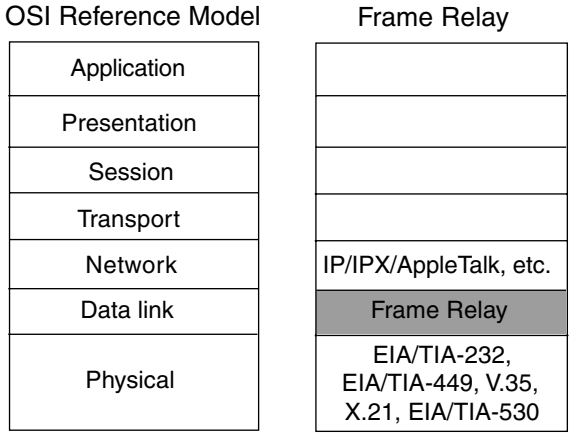
As mentioned earlier, you begin configuring the lab with the routers that have interfaces connected to the Frame Relay network. Before you start configuring the routers, however, you should briefly review Frame Relay and its components.

Frame Relay Overview

Frame Relay is a very popular technology for connectivity across WAN links. Before configuring the interfaces, you probably should gain a basic familiarity with Frame Relay. We will not cover Frame Relay technology in depth; we will just review the essentials so that you understand how Frame Relay is operating in the lab for configuration and troubleshooting tasks. For a complete review of Frame Relay, see Chapter 13 of *Interconnecting Cisco Network Devices* from Cisco Press.

Frame Relay is a physical and data link layer encapsulation technology, as depicted in Figure 7-2.

Figure 7-2 *Frame Relay and OSI Reference Model Correlation*



Frame Relay is an ITU-T (CCITT) and American National Standards Institute (ANSI) defined standard that outlines sending data over a public network.

Frame Relay Components

You should be familiar with several Frame Relay components before going through the Frame Relay interface configuration process:

- **Data-link connection identifier (DLCI)**—This is a number that identifies the logical circuit between the router and the Frame Relay switch. The Frame Relay switch maps the DLCIs between each pair of routers to create a PVC. DLCIs have local significance in that the identifier references the point between the local router and the Frame Relay switch to which it is connected. Thus, the same (or different) DLCI numbers can be used on both ends and it still would work properly.

- **Permanent virtual circuit (PVC)**—This is a virtual circuit that is permanently established. A PVC is an end-to-end path (that is, router-to-router path through a Frame Relay cloud). One or more DLCIs form a PVC in a Frame Relay network. PVCs save bandwidth associated with the establishment and teardown of circuits.
- **Local Management Interface (LMI)**—This is a signaling standard between the router device and the Frame Relay switch that is responsible for managing the connection and maintaining status between the devices.

NOTE Not all Frame Relay terminology is reviewed here. Frame Relay is a useful technology, and there are several different applications of Frame Relay. Again, we will review only what is needed to complete the lab.

Frame Relay Address Mappings

An important part of Frame Relay is the mapping of IP addresses to DLCIs. Two methods exist for Cisco routers to form these mappings:

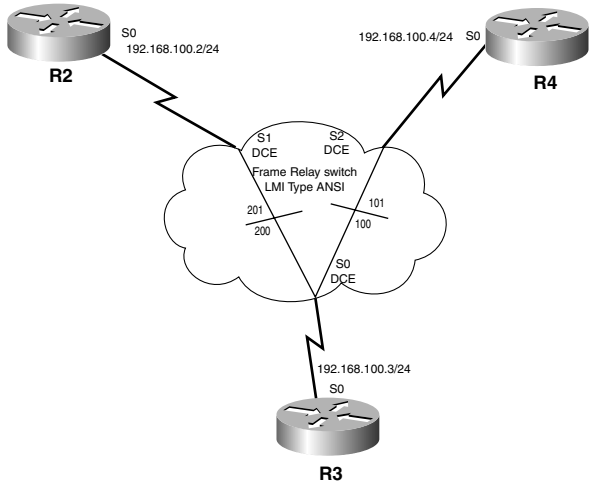
- **Frame Relay Inverse ARP (dynamic)**—This is a method of dynamically associating a network layer address with a DLCI. It allows a router to discover the network address of a device associated with a DLCI. Inverse ARP normally is used in fully meshed networks (networks in which all routers have PVCs to each other). The environment in this lab is a hub-spoke topology, so you will encounter some limitations with Frame Relay Inverse ARP.
- **Static (manual) mappings**—In this method, you manually map a network layer address, such as IP or IPX, to a DLCI. This is done under the interface configuration mode for the serial interface configured for Frame Relay. It is important to note that when you manually map an IP address to a DLCI, Frame Relay Inverse ARP is disabled for that interface. You will have an opportunity to manually map IP addresses to DLCIs in this lab.

Configuring Frame Relay Interfaces

NOTE If you have not configured the Frame Relay Switch router (the Cisco 2523 router), you need to do so before continuing. See Appendix B, "Frame Relay Switch Configuration," for the configuration steps.

Before you configure the Frame Relay interfaces, review the Frame Relay network for the lab environment. Figure 7-3 shows the Frame Relay cloud and the router interfaces that you will be configuring for Frame Relay.

Figure 7-3 *Frame Relay Cloud*



As you can see, R2's S0, R3's S0, and R4's S0 are the three interfaces that you need to configure for Frame Relay. Because R3 is the "hub" router (meaning that all other Frame Relay connections terminate on R3), you will start there.

From the terminal server, let's resume the connection to R3 by typing **3**, as in Example 7-1.

Example 7-1 *Gaining Access to R3*

```
Termserver#3
[Resuming connection 3 to r3 ... ]

R3#
```

If the **R3#** does not appear right away, hit the Enter key several times.

NOTE If for some reason you are not in privileged exec mode, type **enable** and type the password **falcons** before executing any commands.

The first step is to get into global configuration mode by typing **config t**. To get into interface configuration mode, enter the **interface** command with the appropriate interface type and number. The options for the **interface** command are listed here:

```
Router#(config)interface [async | dialer | ethernet | group-async | lex | loopback |
  null | serial | tunnel | virtual-template] number
```

As you can see, Cisco IOS Software supports several different types of interfaces. After you specify the type of interface, you need to indicate the number of that particular interface. In the example, R3 has only two serial interfaces: Serial 0 and Serial 1. Remember, all interface numbering starts at 0.

Example 7-2 illustrates the commands executed on R3.

Example 7-2 *Serial 0 Interface Configuration Mode*

```
R3#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface serial 0
R3(config-if)#
```

Because Frame Relay is a type of encapsulation, you need to change the default encapsulation type of HDLC on S0 to Frame Relay. This is done in interface configuration mode using the following command:

```
Router(config-if)#encapsulation frame-relay [cisco ¦ ietf]
```

Two types of Frame Relay encapsulation exist: Cisco and IEFT. Cisco is the default encapsulation and should be used when connecting to another Cisco router. IETF should be used if you are connecting to a non-Cisco device.

The next step is to specify that the LMI type (provided in Figure 7-3) coming from the Frame Relay switch is ANSI. The command to use is this:

```
Router(config-if)#frame-relay lmi-type [cisco ¦ ansi ¦ q933a]
```

Cisco routers support three different LMI types:

- **cisco**—An LMI type defined together by Cisco, StrataCom, Northern Telecom, and Digital Equipment Corporation
- **ansi–AnnexD**—Defined by the ANSI standard T1.617
- **a33a**—ITU-T Q.933 Annex A

Because the Frame Relay switch was configured for LMI type ansi, that is what you must use on the Frame Relay routers. LMI type is from the local router to the switch. The Frame Relay encapsulation type must be consistent from router to router.

Example 7-3 shows the configuration task completed on R3.

Example 7-3 **frame-relay** *Interface Commands*

```
R3(config-if)#encapsulation frame-relay
R3(config-if)#frame-relay lmi-type ansi
R3(config-if)#
```

Now assign the interface the IP address 192.168.100.3 with a mask of /24 (which is 255.255.255.0 in decimal format) by using the following command:

```
Router(config-if)#ip address A.B.C.D A.B.C.D [secondary]
```

The first four letters represent the IP address in decimal form. The second four letters represent the subnet mask in decimal form. You can specify a secondary IP address on the interface as well by using the **secondary** option at the end of the command line.

By default, physical interfaces on a router are "shut down." To bring the port into service, you must enter the command **no shutdown** in the interface configuration mode.

Example 7-4 shows the completed commands for R3.

Example 7-4 *IP Address Assignment*

```
R3(config-if)#ip address 192.168.100.3 255.255.255.0
R3(config-if)#no shutdown
```

If you have done things correctly, you should receive several messages indicating that the interfaces are up and that the line protocol is up. In addition, you should see LMI messages stating what PVC or DLCIs you are seeing from your Frame Relay switch and stating whether they are active. The messages are shown in Example 7-5.

Example 7-5 *Router Console Messages*

```
R3(config-if)#no shutdown
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed state to up
R3(config-if)#
%LINK-3-UPDOWN: Interface Serial0, changed state to up
R3(config-if)#
%FR-5-DLCICHANGE: Interface Serial0 - DLCI 100 state changed to ACTIVE
%FR-5-DLCICHANGE: Interface Serial0 - DLCI 200 state changed to ACTIVE

%FR-5-DLCICHANGE: Interface Serial0 - DLCI 100 state changed to INACTIVE
%FR-5-DLCICHANGE: Interface Serial0 - DLCI 200 state changed to INACTIVE
```

From the messages in Example 7-5, you know that R3 is seeing two DLCIs, 100 and 200. It is interesting to note that the DLCIs go into an ACTIVE state first and then go back to an INACTIVE a few moments later. This is because no other routers are configured for Frame Relay at this point. When routers R2 and R4 are configured for Frame Relay, the DLCI should stay in an ACTIVE state.

The **show frame-relay pvc** command is useful for troubleshooting because it displays more information regarding the DLCIs, as demonstrated in Example 7-6.

Example 7-6 **show frame-relay pvc** *Command Output*

```
R3#show frame-relay pvc

PVC Statistics for interface Serial0 (Frame Relay DTE)

DLCI = 100, DLCI USAGE = LOCAL, PVC STATUS = INACTIVE, INTERFACE = Serial0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 0          out bcast bytes 0
  pvc create time 00:00:44, last time pvc status changed 00:00:23

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = INACTIVE, INTERFACE = Serial0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 0          out bcast bytes 0
  pvc create time 00:00:47, last time pvc status changed 00:00:16
R3#
```

With R3 having multiple DLCIs, one going to R2 and the other to R4, this command separates DLCI numbers and gives you individual statistics for each DLCI. If you could not communicate with the Frame Relay switch, you would not have any entries in the output. Remember, LMI is communicating with the Frame Relay switch to provide information on what, if any, DLCIs are advertised on each interface.

The highlighted text in Example 7-6 shows some important information gleaned from this **show** command. The output shows the DLCI number(s), the PVC state, and the interfaces on which these DLCI were received. Another important piece of information is **pvc create time** and **last time pvc status changed**. From these fields, you can determine when the router or the PVC to the switch went down, which is very useful information for troubleshooting purposes.

Another useful troubleshooting command is **show frame-relay lmi**, as demonstrated in Example 7-7.

Example 7-7 **show frame-relay pvc** *Command Output*

```
R3(config-if)#end
R3#show frame-relay lmi

LMI Statistics for interface Serial0 (Frame Relay DTE) LMI TYPE = ANSI
  Invalid Unnumbered info 0              Invalid Prot Disc 0
  Invalid dummy Call Ref 0               Invalid Msg Type 0
  Invalid Status Message 0               Invalid Lock Shift 0
  Invalid Information ID 0               Invalid Report IE Len 0
  Invalid Report Request 0               Invalid Keep IE Len 0
  Num Status Enq. Sent 13            Num Status msgs Rcvd 10
  Num Update Status Rcvd 0               Num Status Timeouts 5
R3#
```

The highlighted text in Example 7-7 depicts some important information. You can see the interface that is receiving LMI (Serial0) and the LMI type (ANSI). You also see the number of LMI keepalives being sent and received. Usually these values will not be equal. It takes a few moments for the DLCI to become active, so a few LMI timeouts will occur.

show frame-relay map is an additional helpful command for troubleshooting. There are no options for this command. The output from this command reveals the mapping of network address (IP or IPX) to DLCI. If you issued this command at this point, you would not see any mappings because no other routers are configured for Frame Relay. Demonstration of this command appears later after the configuration of R2 for Frame Relay.

Next, place a description on the interface that indicates to whom this interface connects.

The **description** command is issued under interface configuration mode:

```
Router#(config-if)description {text}
```

This command is very helpful in managing your routers. By placing a description on each active interface with information such as which router the interface connects to, the private line number or circuit number, DLCI numbers, and other helpful information, you can reduce the time to resolve issues because you have the information located on the router and not on some spreadsheet tucked away somewhere.

Example 7-8 demonstrates this task.

Example 7-8 *Interface Description Task*

```
R3#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface serial 0
R3(config-if)#description This interface connects to R2's S0 (DLCI 200) and R4's S0
  (DLCI 100)
```

You will be placing descriptions on the interfaces at the same time that you assign IP addresses. Before leaving R3 and configuring R2 and R4, take a look at the running-config

to see what the configuration looks like for interface Serial 0. Example 7-9 reveals only the interface section of the running-config.

Example 7-9 **show running-config** *Command Output*

```
R3#show running-config
Building configuration...

Current configuration:
.
.
.
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
description This interface connects to R2's S0 (DLCI 200) and R4's S0 (DLCI 100)
ip address 192.168.100.3 255.255.255.0
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
interface Serial1
 no ip address
 shutdown
```

The highlighted text in Example 7-9 illustrates what the configuration should look like. You do not see any Frame Relay DLCI information in the configuration. This is because Frame Relay Inverse ARP will take care of the DLCI mappings on R3. This will not be the case for R2 and R4, as explained later in the chapter.

Next, save the configuration to NVRAM and move on to R2.

TIP It is always good to save your configuration before leaving the router. Remember, **Ctrl-Shift-6, x** will take you back to the terminal server.

Example 7-10 walks you through the process.

Example 7-10 *Save Configuration and Return to Term Server*

```
R3#copy running-config startup-config
Building configuration...
[OK]
R3#<ctrl-shft-6><x>
    Termserver#2
[Resuming connection 2 to r2 ... ]

R2#
```

From the lab diagram that hopefully you have printed and are updating, you know that you need to configure R2's Serial 0 interface for Frame Relay. Configuring R2's S0 interface for Frame Relay works the same way as what you did for R3's S0 interface. After you change the encapsulation to Frame Relay and specify the Frame Relay LMI type, you can assign the IP address to the main interface and remove it from shutdown state. Example 7-11 demonstrates this process.

Example 7-11 *R2's Serial 0 Frame Relay Configuration*

```
R2#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface serial 0
R2(config-if)#encapsulation frame-relay
R2(config-if)#frame-relay lmi-type ansi
R2(config-if)#ip address 192.168.100.2 255.255.255.0
R2(config-if)#description This interface connects to R3's S0 (DLCI 201)
R2(config-if)#no shutdown
```

Again, if the configuration is correct, you should see console messages like those shown in Example 7-12.

Example 7-12 *R2 Console Messages*

```
2w3d: %LINK-3-UPDOWN: Interface Serial0, changed state to up
R2(config-if)#
2w3d: %FR-5-DLCICHANGE: Interface Serial0 - DLCI 201 state changed to ACTIVE
2w3d: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed state to
  up
R2(config-if)#
```

As the output in Example 7-12 reveals, both the interface and the line protocol came up. As you can see, the LMI reveals that DLCI 201 is being advertised from the Frame Relay switch. This matches the diagram of the Frame Cloud in Figure 7-3. R2 should be receiving DLCI 201. IP connectivity to R3 should be established. To **ping** R3, you need to have an IP address-to-DLCI mapping. Issuing the **show frame-relay map** command reveals whether

Frame Relay Inverse ARP is working. Example 7-13 contains the output from the **show frame-relay map** command.

Example 7-13 show frame-relay map *Command Output*

```
R2#show frame-relay map
Serial0 (up): ip 192.168.100.3 dlci 201(0xC9,0x3090), dynamic,
              broadcast,, status defined, active
R2#
```

The highlighted sections of Example 7-13 illustrate the important information of this command. The output shows that the interface that the mapping was learned on (Serial 0) and its status (up), the IP address of R3 (192.168.100.3), and the DLCI number (201). In addition, "dynamic" tells you that Frame Relay Inverse ARP was used to discover this mapping. If you used a **frame-relay map** statement to manually map the IP address to the DLCI, the output would show "static" where "dynamic" is now. One more piece of information tells you whether the mapping is active or inactive. If the PVC is to be disconnected, you would see "deleted" in that field. In short, the IP address-to-DLCI mapping tells the router that any packets destined to the IP address, or where the IP address is the next hop to the destination address in the IP packet, is to be sent out DLCI 201 (or to R3).

With a valid IP address-to-DLCI mapping, try to **ping** R3's Serial 0 IP address (192.168.100.3). The command syntax is shown here:

```
Router#ping [word | apollo | appletalk | clns | decnet | ip | ipx |vines | xns ] {address}
```

ping is actually an acronym for Packet INternet Groper. It is a mechanism for devices to test Layer 3b (network) connectivity to other devices. The Cisco implementation of the **ping** command supports several different protocols. For this example, you want to test IP connectivity. The router is smart enough to understand the different protocol address format, so you need to type only the command and the destination IP address that you want to test. When testing IP connectivity, the router sends an ICMP echo-request packet; if the destination router receives the packet, it will send an ICMP echo-reply packet. The command will display a "!" when it receives an echo-reply packet. If it never receives the packet, it displays a "." (period) indicating that it waited 2 seconds and did not receive a response to its ICMP echo-request. This will be a very useful command throughout the lab.

Example 7-14 shows the result of issuing the **ping** command on R3's Serial 0 IP address (192.168.100.3).

Example 7-14 *R2-to-R3* **ping** *Result*

```
R2#ping 192.168.100.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/34/36 ms
R2#
```

You got 100 percent success! Looks like you are on the right track. Before leaving R2 to configure R4, take a look at the running-config for R2 and save the configuration to NVRAM. Example 7-15 illustrates the running-config output.

Example 7-15 *R2's Running-Config*

```
R2#show run
Building configuration...

Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R2
.
.
.
!
interface Ethernet0
 no ip address
 no ip directed-broadcast
 shutdown
!
interface Ethernet1
 no ip address
 no ip directed-broadcast
 shutdown
!
interface Serial0
description This interface connects to R3's S0 (DLCI 201)
 ip address 192.168.100.2 255.255.255.0
 no ip directed-broadcast
 encapsulation frame-relay
 no ip mroute-cache
 frame-relay lmi-type ansi
!
.
.
.
```

continues

Example 7-15 *R2's Running-Config (Continued)*

```
R2#copy running-config startup-config
Building configuration...
[OK]
R2#
```

The **encapsulation frame-relay** and **frame-relay lmi-type** commands are present along with the IP address. You have saved the running-config to the startup-config, so move on to R4.

To get to R4, go back to the terminal server, resume the reverse Telnet session to R4, and go into global configuration mode and then interface configuration mode for interface Serial 0. Example 7-16 takes you through the commands to do this.

Example 7-16 *R4 Interface Configuration Mode*

```
R2#
R2#<ctrl-shft-6><x>
    Termserver#4
[Resuming connection 4 to r4 ... ]

R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface serial 0
R4(config-if)#
```

Now that you are in interface configuration mode for R4's S0, go ahead and configure Frame Relay on S0; assign it an IP address and remove it from shutdown mode. The commands are exactly like R2's, except that the IP address changes to 192.168.100.4 with a network mask of 255.255.255.0. See Example 7-17.

Example 7-17 *R4 Frame Relay Configuration*

```
R4(config-if)#encapsulation frame-relay
R4(config-if)#frame-relay lmi-type ansi
R4(config-if)#ip address 192.168.100.4 255.255.255.0
R4(config-if)#description This interface connects to R3's S0 (DLCI 101)
R4(config-if)#no shutdown
R4(config-if)#
```

The highlighted text in Example 7-17 shows the commands for Frame Relay and IP address assignment. You should see console messages, as you did on R2, to inform you that the

interface and line protocols came up and that LMI is discovering DLCIs. Example 7-18 illustrates what you should see when the interface becomes active.

Example 7-18 *R4 Console Messages*

```
%LINK-3-UPDOWN: Interface Serial0, changed state to up
R4(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed state to up
R4(config-if)#
%FR-5-DLCICHANGE: Interface Serial0 - DLCI 101 state changed to ACTIVE
R4(config-if)#
```

The interface is up and the DLCI is active. You now should be capable of **ping**ing R3. Example 7-19 shows the result.

Example 7-19 *R4-to-R3* **ping** *Result*

```
R4(config-if)#end
R4#
%SYS-5-CONFIG_I: Configured from console by console
R4#ping 192.168.100.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/60/60 ms
R4#
```

Okay, 100 percent success! Now try to **ping** R2. After all, it is configured for Frame Relay, right? Example 7-20 illustrates what happens when you try to **ping** R2's S0 interface IP address.

Example 7-20 *R4-to-R2* **ping** *Result*

```
R4#ping 192.168.100.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R4#
```

What happened? You can **ping** R3, right? R2 is on the same IP segment (IP subnet 192.168.100.0), right? Why is the **ping** to R2 from R4 not successful? Frame Relay Inverse ARP is the reason. Remember, for Inverse ARP to operate correctly, you need to have a fully meshed network. The Frame Relay network in this lab is a hub-spoke topology. This means that R2 and R3 have a PVC connecting each other; R4 and R3 do, too, but R2 and R4 do not. If two routers do not have a direct PVC linking them, Frame Relay Inverse ARP

will not resolve the IP address. As demonstrated in Example 7-21, you can issue a **show frame-relay map** command to take a look at the Frame Relay DLCI mappings to see if a mapping to R2's IP address exists.

Example 7-21 show frame-relay map *Output*

```
R4#show frame-relay map
Serial0 (up): ip 192.168.100.3 dlci 101(0x65,0x1850), dynamic,
              broadcast,, status defined, active
R4#
```

As you can see from Example 7-21, there is a mapping for R3's S0, but not for R2's. You can resolve this problem by also mapping R2's IP address to DLCI 101. It is important to understand, however, that R4 does not send packets directly to R2. R4 sends any packet destined to R2 out DLCI 101, which terminates on R3, and then R3 redirects the packet to R2. Remember, just because you make a manual DLCI-to-IP-address mapping, you have not created a PVC to R2; you are just forwarding everything to R3. Another important item to remember is that when you issue a manual **map** statement on an interface, you disable Frame Relay Inverse ARP for that interface. This means that any mappings that were discovered using Frame Relay Inverse ARP are removed when the router is rebooted or a **clear frame-relay inverse arp** command is issued. So, you need to map not only R2's IP address, but R3's as well, or you will lose connectivity to R3.

The command to manually map IP addresses to DLCIs is as follows:

```
Router(config-if)#frame-relay map [ip | apollo | appletalk | bridge | clns |
  decnet | dlsw | ip | ipx | llc2 | qllc | rsrb | stun | vines | xns]
  {a.b.c.d}  {dlci-number} [active | broadcast | cisco | ietf | nocompress |
  payload-compression | tcp]
```

The first option in the command is the protocol type. The **frame-relay map** command supports several protocols, but the only one to be concerned with in this chapter is IP. The second option in the command syntax (*a.b.c.d*) is the destination address. If you specify IP as the protocol that you want to map, the command expects an IP address. If you choose IPX (as covered in Chapter 13, "IPX"), the command expects an IPX address. The third option in the command is the DLCI number. This is the local DLCI number that you want any packets destined for the address to be forwarded out this DLCI. The last option provides several options, but you need to be concerned with only the **broadcast** option. Frame Relay, by default, will not forward Layer 3 broadcasts. You can override that default value by specifying the **broadcast** option here. This is important because, later in the lab, you will be configuring routing protocols over the Frame Relay network that will need to use broadcast packets to exchange routing tables (RIP or IGRP) or to establish neighbor adjacencies (EIGRP).

Example 7-22 demonstrates how to manually map an IP address to a DLCI. Because the DLCI is being advertised to the interface, you create the IP address-to-DLCI mapping on that interface (in this case, interface S0).

Example 7-22 *Frame Relay Manual Map*

```
R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface serial 0
R4(config-if)#frame-relay map ip 192.168.100.2 101 broadcast
R4(config-if)#frame-relay map ip 192.168.100.3 101 broadcast
R4(config-if)#
```

Before trying to **ping** R2 again, go back and place the **frame-relay map** statements of R2's Serial 0 interface. Example 7-23 walks you through the process.

Example 7-23 *Adding* **frame-relay map** *Statements on R2*

```
R4#copy running-config startup-config
Building configuration...
[OK]
R4#<ctrl-shft-6><x>
    Termserver#2
[Resuming connection 2 to r2 ... ]
```

```
R2#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface serial 0
R2(config-if)#frame-relay map ip 192.168.100.4 201 broadcast
R2(config-if)#frame-relay map ip 192.168.100.3 201 broadcast
R2(config-if)#
```

Now that you have manually mapped the IP addresses to the DLCI, take a look at how the mapping table changed from the output in Example 7-24.

Example 7-24 **show frame-relay map** *Command Output*

```
R2#show frame-relay map
Serial0 (up): ip 192.168.100.3 dlci 201(0xC9,0x3090), static,
              broadcast,
              CISCO, status defined, active
Serial0 (up): ip 192.168.100.4 dlci 201(0xC9,0x3090), static,
              broadcast,
              CISCO, status defined, active
R2#
```

The output of Example 7-24 shows a couple of changes. First, now two mappings exist—one to 192.168.100.3 (R3) and one to 192.168.100.4 (R4). Both go out DLCI 201. The other major change is that the mapping changed from dynamic (meaning that they were learned by Inverse ARP) to static (meaning that the IP address to the DLCI under the interface was manually mapped). With the correct mappings, you should be capable of **ping**ing R3 as well as R4. Example 7-25 shows the result of the **ping** command to R3 and R4.

Example 7-25 *R2-to-R3 and R2-to-R4* **ping** *Result*

```
R2#ping 192.168.100.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/36/44 ms

R2#ping 192.168.100.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 88/94/112 ms
R2#
```

With R2 capable of **ping**ing both routers, full IP connectivity through the Frame Relay network now exists.

NOTE R3 will not need manual **frame-relay map** statements because R3 has a direct PVC to both R2 and R4. Frame Relay Inverse ARP will operate correctly for R3.

Now you can move on and configure the point-to-point serial link between R3 and R5.

Configuring Point-to-Point Serial Interfaces

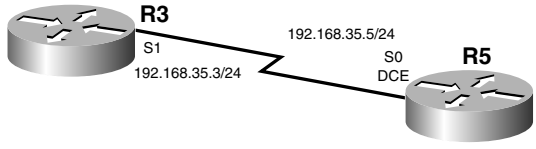
The point-to-point serial link is a little different than the Frame Relay serial link. As you recall, Frame Relay is used in many cases in a point-to-multipoint environment. To create a point-to-point connection between two routers, you can use other types of WAN encapsulations, such as HDLC, PPP, and SLIP. For a complete review of these encapsulations, refer to Chapter 11 of the ICDN book. We will use the default serial encapsulation, which is HDLC.

For two serial interfaces to communicate, you must provide the clock rate. The clock rate provides bit synchronization and has other uses that are beyond the scope of this chapter's

purposes. The device that provides this clock rate is the data circuit-terminating equipment (DCE). The other device is denoted as the data terminal equipment (DTE). In the real world, the router is the DTE. The telecom equipment, such as the Frame Relay switch, is the DCE. Because the lab scenario bypasses the telecom equipment, you need to specify which router is the DTE and which one is the DCE; you also need to provide the clock rate. To bypass the telecom equipment, you need to directly connect the routers with a DTE–DCE crossover cable. Each end of this cable is labeled as DTE or DCE. Whichever router is plugged into the DCE end of the cable will need to provide the clock rate. In the Frame Relay network, the Frame Relay switch is the DCE, so none of the Frame Relay routers needs to provide clock rate. For more information on WAN serial cabling and signaling, refer to Chapter 2 of *Interconnecting Cisco Network Devices* from Cisco Press.

First, review the routers that you are going to use to configure a point-to-point serial connection. Figure 7-4 shows routers R3 and R5. This is the only point-to-point connection in the lab.

Figure 7-4 *Point-to-Point Serial Connection*



In the figure, you can see the R5 has been cabled as the DCE, so it will need to supply the clock rate.

Begin by configuring the serial link and assign IP addresses to the interfaces. Start with R5.

You last configured R2, so you need to go back to the terminal server and resume the connection to R5. When there, you need to go into global configuration mode and then into the appropriate interface configuration. For R5, that would be Serial 0. Example 7-26 walks you through these initial configuration steps.

Example 7-26 *R5 Serial Configuration*

```
R2#
R2#<crtl-shft-6-x>

    Termserver#5
[Resuming connection 5 to r5 ... ]

R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface serial 0
R5(config-if)#
```

Now that you are in interface configuration mode for R5's Serial 0, you can execute the necessary configuration commands. The first thing to do is make the encapsulation type HDLC for the interface. Because HDLC is the default encapsulation method, you really don't need to execute the command. However, just for the sake of practice, and so that you understand that there is a data link layer configuration command for the serial link, specify HDLC as the encapsulation by entering it as a command option. This is the same command issued previously when specifying the encapsulation type for routers R2, R3, and R4; the only difference is that you specify the **hdlc** option instead of **frame-relay**.

This is the command for R2's S0, R3's S0, and R4's S0:

```
Router(config-if)#encapsulation frame-relay [cisco | ietf]
```

This is the command for R3's S1 and R5's S0:

```
Router(config-if)#encapsulation hdlc
```

Unlike Frame Relay, there aren't any different types of HDLC encapsulation. After you specify the encapsulation type as HDLC, you can assign the appropriate IP address to the interface.

NOTE We will not review previous commands that already have been demonstrated. Refer back to the previous examples if you are unsure of the command syntax, or use the help menu in Cisco IOS Software.

Example 7-27 shows the commands executed on R5.

Example 7-27 *R5 Configuration Commands*

```
R5(config-if)#encapsulation hdlc
R5(config-if)#ip address 192.168.35.5 255.255.255.0
R5(config-if)#
```

Before removing the interface from shutdown mode, you need to provide the clock rate to R3 using the following command:

```
Router(config-if)#clock rate {300-8000000 bps}
```

The only option in this command is to give the speed of the link in bits per second. Because this is a T1 or E1 interface, you can specify an easy-to-remember value of 2,000,000. This is the equivalent of an E1 link, which will work for this lab environment. As mentioned earlier, in the real world, you will not have to configure this parameter. The telecom service provider

will set this value on its equipment. After you set this value, give the interface a description and remove the interface from shutdown mode, as demonstrated in Example 7-28.

Example 7-28 clock rate *Command*

```
R5(config-if)#clock rate 2000000
R5(config-if)#description This interface connects to R3's S1 (DTE)
R5(config-if)#no shutdown
R5(config-if)#
%LINK-3-UPDOWN: Interface Serial0, changed state to down
R5(config-if)#
```

At first glance, you might get a little nervous that the interface did not come up, but that is normal. R3's serial interface has not been configured yet, so the R5 interface is not receiving any signaling from R3; thus, the interface will remain in the down state until R3 is configured and removed from shutdown mode. Before you get too far into this configuration, you should know about a very helpful **show** command:

```
Router#show interfaces [bri | null | serial | tokenring | accounting | crb | irb]
  {number}
```

This command is very useful in troubleshooting and verifying interface configuration. The first option is to choose which type of interface you would like to see; the second option is to select the number of the interface. If you do not select any type of interface, the command shows you all the interfaces that the router has. Example 7-29 demonstrates sample output of the command on R5.

Example 7-29 show interfaces serial 0 *Command Output*

```
R5#show interfaces serial 0
Serial0 is down, line protocol is down
  Hardware is HD64570
  Description: This interface connects to R3's S1 (DTE)
  Internet address is 192.168.35.5/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input never, output 2w5d, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  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
     212 packets output, 18206 bytes, 0 underruns
     0 output errors, 0 collisions, 37557 interface resets
     0 output buffer failures, 0 output buffers swapped out
     111 carrier transitions
     DCD=up  DSR=up  DTR=down  RTS=down  CTS=up
```

The highlighted text reveals some important information regarding the interface Serial 0. The first thing that you see is the state in which the interface resides: "Serial0 is down, line protocol is down." The first "down" (this is referred to as *interface* or the physical layer state) tells you that there is a physical problem. A physical problem might result from a cable not being plugged in, or the connected device might not be receiving any electrical signaling, which is the case here. The "line protocol down" means that Layer 2 is not functional, meaning that HDLC is not operating correctly for some reason. It is important to note that the line protocol will never be up if the interface is in the down state. Next, the output shows the description placed on the Serial 0 interface. You see the IP address that you assigned earlier. This is a good place to review your configuration and make sure that what you typed in the interface configuration mode was correct. You also see the encapsulation type here. For a complete review of the output, refer to Chapter 3 of *Interconnecting Cisco Network Devices*. You will see this command again after configuring R3's serial interface to see what changes.

To configure R3, you need to go back to the terminal server and resume the session with R3, but don't forget to save the configuration before leaving. When at R3, you need to enter global configuration mode and then go into interface configuration mode for Serial 1. Remember, you will configure Serial 0 for Frame Relay. Serial 1 connects to R5's S0 interface. (Refer to your lab diagram.) See Example 7-30.

Example 7-30 *R3 Interface Configuration Mode*

```
R5#copy running-config startup-config
Building configuration...
[OK]
R5#<ctrl-shft-6><x>
    Termserver#3
[Resuming connection 3 to r3 ... ]
R3#
R3#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface serial 1
R3(config-if)#
```

Now you are in interface configuration mode for Serial 1 on R3, and you can assign the appropriate IP address and mask. After that, don't forget to remove the interface from shutdown mode. Example 7-31 illustrates the commands.

Example 7-31 *R3 Serial 1 Configuration Commands*

```
R3(config-if)#encapsulation hdlc
R3(config-if)#ip address 192.168.35.3 255.255.255.0
R3(config-if)#description This interface connects to R5's S0 (DCE)
R3(config-if)#no shutdown
R3(config-if)#
%LINK-3-UPDOWN: Interface Serial1, changed state to up
```

Example 7-31 *R3 Serial 1 Configuration Commands (Continued)*

```
R3(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to up
R3(config-if)#
```

As you can see, the interface came up, and so did the line protocol. Return to R5 and see how the **show interface** command output has changed. Example 7-32 shows the changes in the output.

Example 7-32 *R5* **show interface serial 0** *Command Output*

```
R3(config-if)#end
%SYS-5-CONFIG_I: Configured from console by console
R3#copy running-config startup-config
Building configuration...
[OK]
R3#<ctrl-shft-6-x>
    Termserver#5
[Resuming connection 5 to r5 ... ]

R5#
R5#show interface serial 0
Serial0 is up, line protocol is up
  Hardware is HD64570
  Description: This interface connects to R3's S1 (DTE)
  Internet address is 192.168.35.5/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input 00:00:01, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     25 packets input, 1865 bytes, 0 no buffer
     Received 25 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     236 packets output, 20009 bytes, 0 underruns
     0 output errors, 0 collisions, 37629 interface resets
     0 output buffer failures, 0 output buffers swapped out
     112 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
R5#
```

Great! You now should be capable of **ping**ing R3 from R5, as demonstrated in Example 7-33.

Example 7-33 *R5* **ping** *Result*

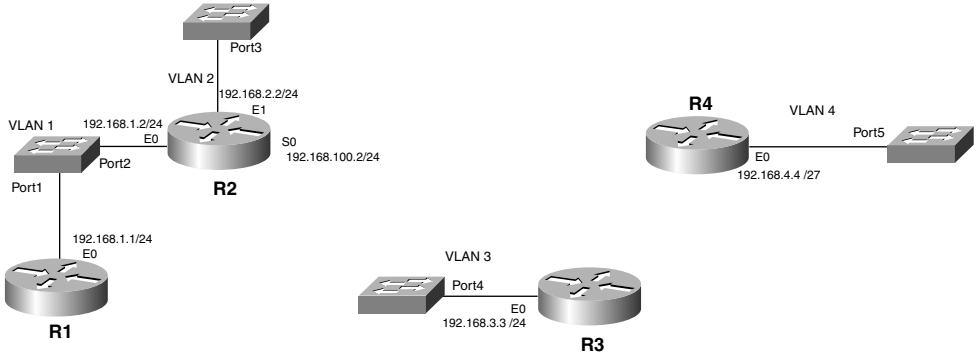
```
R5#ping 192.168.35.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.35.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/8 ms
R5#
```

You got 100 percent success! R3 and R5 have full IP connectivity. Question: Should R5 be capable of **ping**ing R3's Serial 0 IP address? Why not? Even though R5 can reach R3 through interface Serial 1, R5 cannot **ping** R3's Serial 0 interface. This is because you do not have any routing protocols configured to let R5 know about the 192.168.100.0 network, to which R3's Serial 0 interface belongs. When you have configured all the interfaces on all the routers, you will start configuring the routing protocols. Then you should be capable of **ping**ing any interface on any router.

Configuring Ethernet Interfaces

The configuration tasks for Ethernet interfaces are quite simple. In fact, you only need to assign the IP address and remove the interface from shutdown mode. Because the configuration tasks are so straightforward, an overview of Ethernet technology is not really necessary here. Begin with a review of the routers that have Ethernet interfaces that you will need to configure. Figure 7-5 illustrates the Ethernet interfaces on the routers.

Figure 7-5 *Ethernet Routers*



Start with configuring R1, then configure R2, and go up to R4. To assign the IP address and remove the interface from shutdown mode, you need to be in interface configuration mode for the Ethernet interface. Example 7-34 takes you through the process of configuring the

Ethernet interface on R1. Refer to Figure 7-5 for the interface number, IP address, and subnet mask.

Example 7-34 *R1 Ethernet Configuration*

```
R5#<ctrl-shft-6-x>
    Termserver#1
[Resuming connection 1 to r1 ... ]

R1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface ethernet0
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#description This interface connects to R2's E0
R1(config-if)#no shut
R1(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to up
R1(config-if)#
R1#
%SYS-5-CONFIG_I: Configured from console by console
R1(config-if)#
%LINK-3-UPDOWN: Interface Ethernet0, changed state to up
```

R1's Ethernet 0 came up fine, so go to R2. Example 7-35 demonstrates the steps in configuring R2's Ethernet interfaces.

Example 7-35 *R2 Ethernet Interface Configuration*

```
R1#copy running-config startup-config
Building configuration...
[OK]
R1#<ctrl-shft-6><x>
    Termserver#2
[Resuming connection 2 to r2 ... ]

R2#
R2#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface ethernet 0
R2(config-if)#ip address 192.168.1.2 255.255.255.0
R2(config-if)#description This interface connects to R1's E0
R2(config-if)#no shutdown
R2(config-if)#
1d17h: %LINK-3-UPDOWN: Interface Ethernet0, changed state to up
1d17h: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state
to up
R2(config-if)#
R2(config-if)#exit
R2(config)#interface ethernet 1
R2(config-if)#ip address 192.168.2.2 255.255.255.0
R1(config-if)#description This interface does not connect with another IP device
```

continues

Example 7-35 *R2 Ethernet Interface Configuration (Continued)*

```
R2(config-if)#no shutdown
R2(config-if)#
1d17h: %LINK-3-UPDOWN: Interface Ethernet1, changed state to up
1d17h: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet1, changed state
to up
R2(config-if)#
```

Both Ethernet interfaces came up. You should be capable of **ping**ing R1's Ethernet 0 interface for R2, as demonstrated in Example 7-36.

Example 7-36 *R2 to R1* **ping** *Results*

```
R2(config-if)#end
R2#ping 192.168.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/1 ms
R2#
```

Okay, R1 and R2 have IP connectivity. You can verify that Ethernet 1 is functional by looking at the interface. Example 7-37 shows the output of the **show interfaces** command, which displays all of R2's interfaces; however, only the Ethernet interfaces are of interest for this part of the lab.

Example 7-37 **show interfaces** *Command Output on R2*

```
R2#show interfaces
Ethernet0 is up, line protocol is up
  Description: This interface connects to R1's E0
  Hardware is QUICC Ethernet, address is 0010.7bf9.4912 (bia 0010.7bf9.4912)
  Internet address is 192.168.1.2/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:20, output 00:00:06, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     41 packets input, 4110 bytes, 0 no buffer
     Received 36 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     159 packets output, 16101 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
```

Example 7-37 **show interfaces** *Command Output on R2 (Continued)*

```
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
Ethernet1 is up, line protocol is up
  Hardware is QUICC Ethernet, address is 0010.7bf9.4913 (bia 0010.7bf9.4913)
  Description: This interface does not connect with another IP device
  Internet address is 192.168.2.2/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:23, output 00:00:09, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     19 packets input, 1729 bytes, 0 no buffer
     Received 19 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     136 packets output, 13770 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
  --More--
```

The most important information right now is to see that both interfaces are up. This signifies that link keepalives are being exchanged between the interfaces and the switch. No other devices exist off Ethernet 1, so you cannot verify connectivity. However, because both interfaces are up, you can assume that they are configured and working properly.

Example 7-38 consolidates the configuration of both R3 and R4 to save time. Make sure that you see the console messages stating that the interfaces are up, but there is no need to **ping** anything at this point. Be sure to look at the subnet mask on R4's Ethernet 0. It has a mask of /27. That is a 255.255.255.224 mask in decimal notation.

Example 7-38 *R3 and R4 Ethernet Configuration*

```
R2#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
R2#<ctrl-shft-6><x>
    Termserver#3
[Resuming connection 3 to r3 ... ]

R3#config t
Enter configuration commands, one per line.  End with CNTL/Z.
```

continues

Example 7-38 *R3 and R4 Ethernet Configuration (Continued)*

```
R3(config)#interface ethernet 0
R3(config-if)#ip address 192.168.3.3 255.255.255.0
R3(config-if)#description This interface does not connect with another IP device
R3(config-if)#no shutdown
R3(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to up
R3(config-if)#
%LINK-3-UPDOWN: Interface Ethernet0, changed state to up
R3(config-if)#end
R3#
%SYS-5-CONFIG_I: Configured from console by console
R3#copy running-config  startup-config
Building configuration...
[OK]
R3#<ctrl-shft-6><x>
    Termserver#4
[Resuming connection 4 to r4 ... ]

R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface  ethernet 0
R4(config-if)#ip address 192.168.4.4 255.255.255.224
Bad mask /27 for address 192.168.4.4
R4(config-if)#
```

Notice the error message "Bad mask /27 for address 192.168.4.4." Why is /27 (or 255.255.255.2250) a bad mask? In IP subnetting, you cannot use the first group of IP addresses or the last group of a subnetted address space. Does 192.168.4.4 fall into the first group of addresses? To find out, break up the /27 bit mask (255.255.255.224). The result is eight different groups of IP addresses:

0 to 31
32 to 63
64 to 95
96 to 127
128 to 159
160 to 191
192 to 123
224 to 255

The address of 192.168.4.4 does fall into the first group. Cisco has a command that will overcome this limitation:

```
Router(config)#ip subnet-zero
```

There are no options on this command, and it is executed under global configuration mode. This command enables you to use the first and last groups of a subnetted address space. Example 7-39 uses this command to configure R4.

NOTE In Cisco IOS Software Release 12.*x*, the command **ip subnet-zero** is on by default. If you are using this version, you will not see the error, nor will you need to execute the command.

Example 7-39 *R4 Configuration for* **ip subnet-zero**

```
R4(config-if)#exit
R4(config)#ip subnet-zero
R4(config)#interface ethernet 0
R4(config-if)#ip address 192.168.4.4 255.255.255.224
R4(config)-if)#$cription This interface does not connect with another IP device
R4(config-if)#no shutdown
R4(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to up
R4(config-if)#
%LINK-3-UPDOWN: Interface Ethernet0, changed state to up
R4(config-if)#end
%SYS-5-CONFIG_I: Configured from console by console
R4#copy running-config startup-config
Building configuration...
[OK]
R4#
```

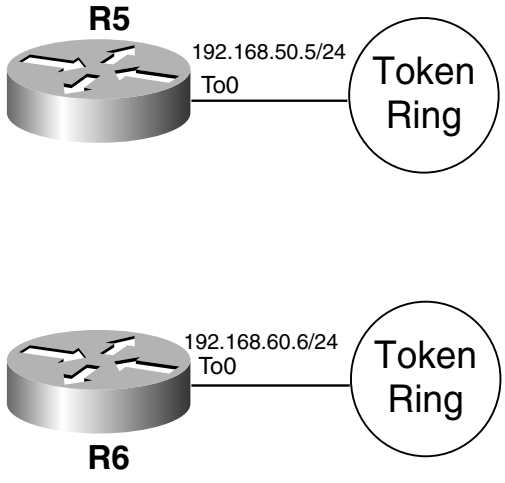
You get no error messages this time. The IP address successfully is assigned to the interface and is removed from the shutdown state. Console messages indicate that the interface and line protocol for Ethernet 0 are up. The configuration is saved, and you are ready to configure the Token Ring interfaces on R5 and R6.

Configuring Token Ring Interfaces

Token Ring interfaces have similar configuration tasks to Ethernet, but the technologies are very different. For lab purposes, those differences are out of the scope of this book.

The only configuration difference in Token Ring versus Ethernet interfaces is that, on Token Ring, you need to specify a ring speed, either 4 or 16 Mbps. Begin by reviewing the routers that you will be configuring as Token Ring, as shown in Figure 7-6.

Figure 7-6 *Router Interfaces to Configure as Token Ring*



To set the ring speed, you must be in interface configuration mode for the Token Ring interface and must use this command:

```
Router(config-if)#ring-speed {4 ¦16}
```

As mentioned earlier, Token Ring supports two speeds: 4 and 16 Mbps. For this lab, use 16 Mbps. For Token Ring to operate correctly, every device belonging to a certain Ring must be configured for the same ring speed. After you set the speed, you can assign an IP address to the interface. Example 7-40 completes the commands on R5.

Example 7-40 *R5 Token Ring Interface Configuration*

```
R4#<ctrl-shft-6><x>
    Termserver#5
[Resuming connection 5 to r5 ... ]

R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface tokenRing 0
R5(config-if)#ring-speed 16
R5(config-if)#ip address 192.168.50.5 255.255.255.0
R5(config-if)#$iption This interface does not connect with another IP device
R5(config-if)#no shutdown
R5(config-if)#
%LINK-5-CHANGED: Interface TokenRing0, changed state to initializing
R5(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface TokenRing0, changed state to up
R5(config-if)#
%LINK-3-UPDOWN: Interface TokenRing0, changed state to up
R5(config-if)#
```

Example 7-40 shows a console message a little differently than on other interfaces. "%LINK-5-CHANGED: Interface TokenRing0, changed state to initializing" means that the router is trying to insert the Token Ring into the main network ring. If that is successful, you will get the interface up and line protocol up messages. This looks good, so save the configuration and complete R6. See Example 7-41.

Example 7-41 *Token Ring Interface Configuration on R6*

```
R5#copy running-config startup-config
Building configuration...
[OK]
R5#<ctrl-shft-6><x>
    Termserver#6
[Resuming connection 6 to r6 ... ]

R6#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#interface tokenRing 0
R6(config-if)#ring-speed 16
R6(config-if)#ip address 192.168.60.6 255.255.255.0
R6(config-if)#$iption This interface does not connect with another IP device
R6(config-if)#no shutdown
R6(config-if)#
%LINK-5-CHANGED: Interface TokenRing0, changed state to initializing
R6(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface TokenRing0, changed state to up
R6(config-if)#
%LINK-3-UPDOWN: Interface TokenRing0, changed state to up
R6(config-if)#
```

Everything looks good. Next you will learn how to create and configure loopback interfaces.

Creating and Configuring Loopback Interfaces

A loopback interface is a virtual interface that resides on a router. It is not connected to any other device. Loopback interfaces are very useful because they will never go down, unless the entire router goes down. This helps in managing routers because there will always be at least one active interface on the routers, the loopback interface.

To create a loopback interface, all you need to do is enter configuration mode for the interface:

```
Router(config)interface loopback {number}
```

The only option on this command is to specify a number between 0 and 2,147,483,647. Cisco IOS Software gives you plenty of loopback interfaces, if you want to use all of them. When entering this command, Cisco IOS Software automatically creates the loopback

interface, places you into interface configuration mode, and removes the interface from shutdown mode. When that is complete, you only need to assign an IP address to the interface. The criteria for the IP addresses of the loopback interfaces is as follows:

> Create loopback interfaces on all routers using IP address 192.169.*X*.*X*/24 (where *X* is the router number). So, R1 would have a loopback address of 192.169.1.1/24, R2 would be 192.169.2.2/24, and so on.

Because you are already on R6, create and configure the loopback interface on R6; then go to R1, R2, and so on, and create and configure all the loopback interfaces and assign appropriate IP addresses. Example 7-42 takes you through the process on R6.

Example 7-42 *R6 Loopback Interfaces Configuration*

```
R6(config)#interface loopback 0
R6(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0, changed state to up
R6(config-if)#ip address 192.169.6.6 255.255.255.0
```

Because the router automatically removes the loopback from shutdown state, you receive the console message indicating that the interface is up.

Now configure the rest of the routers, starting with R1. Don't forget to save the running-config to NVRAM (startup-config) before leaving the routers. See Example 7-43.

Example 7-43 *Loopback Interface Configuration*

```
R6#<ctrl-shft-6><x>
    Termserver#1
[Resuming connection 1 to r1 ... ]
[OK]
```

```
R1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface loopback 0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0, changed state to up
R1(config-if)#ip address 192.169.1.1 255.255.255.0
R1(config-if)#end
%SYS-5-CONFIG_I: Configured from console by console
R1#copy running-config startup-config
Building configuration...
[OK]
R1#<ctrl-shft-6><x>
Termserver#2
 [Resuming connection 2 to r2 ... ]
```

```
R2#
R2#config t
```

Example 7-43 *Loopback Interface Configuration (Continued)*

```
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface loopback 0
R2(config-if)#ip address 192.169.2.2 255.255.255.0
R2(config-if)#end
R2#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
R2#<ctrl-shft-6><x>
    Termserver#3
[Resuming connection 3 to r3 ... ]
[OK]
```

```
R3#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface loopback 0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0, changed state to up
R3(config-if)#ip address 192.169.3.3 255.255.255.0
R3(config-if)#end
R3#copy running-config startup-config
Building configuration...
[OK]
R3#<ctrl-shft-6><x>
    Termserver#4
[Resuming connection 4 to r4 ... ]
```

```
R4#config t
R4(config)#interface loopback 0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0, changed state to up
R4(config-if)#ip address 192.169.4.4 255.255.255.0
R4(config-if)#end
R4#copy running-config startup-config
Building configuration...
[OK]
R4#<ctrl-shft-6><x>
Termserver#5
[Resuming connection 5 to r5 ... ]
```

```
R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface loopback 0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0, changed state to up
R5(config-if)#ip address 192.169.5.5 255.255.255.0
R5(config-if)#end
R5#copy running-config startup-config
Building configuration...
[OK]
R5#
```

Everything looks okay. A router interface description is not necessary here because no other type of device can connect to a loopback interface. Now that you have configured all the interfaces and have IP connectivity, you should familiarize yourself with the Cisco Discovery Protocol (CDP). CDP is an information-gathering tool that enables you to discover directly connected Cisco devices and their network layer addresses.

Cisco Discovery Protocol (CDP)

CDP is a Cisco proprietary data link layer protocol that operates over any medium that supports the Subnetwork Access Protocol (SNAP) encapsulation (LANs, most WANs, and ATM). It is important to understand that because CDP operates at Layer 2 (data link layer of the OSI model), it functions independently of the Layer 3 (network) protocol (IP or IPX). CDP is on by default, but it can be disabled. In many cases, CDP is disabled on dial backup links, such as ISDN, so as to not keep the link up constantly. Chapter 4 of *Interconnecting Cisco Network Devices* provides more detailed information about CDP.

To display what CDP has discovered, issue this command:

```
Router#show cdp [entry | interface | neighbors | traffic]
```

This command offers several options. For purposes here, look at the **interface** and **neighbors** options only, but feel free to become familiar with the other options.

The first option to look at is the **interface** option:

```
Router#show cdp interface [ethernet | loopback | null | serial] [number]
```

The first option on this command is to specify the type of interface that you want to see CDP information on. The last option is to specify the interface number. Example 7-44 uses the **show cdp interface** command to examine R5's Serial 0 interface.

Example 7-44 show cdp interface serial 0 *Command Output*

```
R5#show cdp interface serial 0
Serial0 is up, line protocol is up
  Encapsulation HDLC
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
R5#
```

It does not give a whole lot of information, but it is a "quick and dirty" way of seeing some CDP information. The second option of the **show cdp** command that we are going to look at is this:

```
Router#show cdp neighbors [bri | loopback | null | serial | tokenring | detail]
```

As you can see, this command provides the option to see CDP information by interface, but the last option, **detail**, gives a complete summary of all Cisco devices that CDP was capable

of discovering and displays information about those devices. Example 7-45 displays the output of the **show cdp neighbors detail** command.

Example 7-45 show cdp neighbors detail *Command Output*

```
R5#show cdp neighbors detail
-------------------------
Device ID: R3
Entry address(es):
  IP address: 192.168.35.3
Platform: cisco 2500,  Capabilities: Router
Interface: Serial0,  Port ID (outgoing port): Serial1
Holdtime : 164 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-JS-L), Version 11.2(17), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Mon 04-Jan-99 17:27 by ashah

R5#
```

The first piece of information that the output shows is the device ID, usually the hostname of the device. As you can see, R5 has discovered R3.

The second field is the IP address of R3. This is very useful if you have several routers and you are not sure of the IP address of the desired router. You can use this command to find that instead of trying to track down a network map.

The third field is the platform and capability. R3 is a 2500 and is a router. If you issue this same command on R1, you will see the Catalyst 1900 switch in the summary as well.

The fourth item is the interface on R5 that the device was discovered and the port on R3 to which R5 is connected. The command also displays the Cisco IOS Software version of your neighbor.

You will see this command revisited throughout the rest of the book, to demonstrate the different ways to utilize this command to help configure and troubleshoot the network.

Now that you have all the interfaces configured and have established IP connectivity, you can move on to the next chapter and start configuring the different routing protocols.