CHAPTER

2

CONFIGURING BASIC NETWORKING

After reading this chapter and completing the exercises, you will be able to:

- Describe how network interfaces are initialized
- Configure network interfaces using scripts and text-mode utilities
- Configure Linux networking using popular graphical utilities
- Effectively use networking utilities to test a network and troubleshoot networking problems
- Understand the IPX and AppleTalk protocols

In this chapter, you will learn how the networking principles in Chapter 1 are implemented in Linux, and how to configure Linux networking. This configuration can be accomplished using simple command-line tools or any of several graphical tools. Modern Linux distributions typically include at least one graphical tool.

In the second part of this chapter, you will learn about the basic networking utilities—most of which are command-line tools—and how to test and troubleshoot a network using these tools. Finally, you will learn a little more about other protocols that are occasionally used in Linux to connect with other types of networks. These protocols include IPX and AppleTalk.

Initializing Network Interfaces

When Linux boots up, it will probably recognize your network interface hardware, install the appropriate drivers, and configure the interfaces so they're ready to use. When it doesn't happen this way, you'll have to troubleshoot to find out why. Troubleshooting requires that you know how to identify network hardware and load the drivers.

The first step is to see whether Linux recognized the interface hardware when it booted up. If you're running a Linux kernel earlier than 2.6, you have to look in the `/proc/net/dev` file to see the interfaces. The lines in this file are too long to display in the limited space on this page. Look at the file on your system by entering this command:

```
cat /proc/net/dev
```

You'll see the interface names, such as `lo` and `eth0`, at the start of each of the displayed lines. You can ignore the rest of the lines.

If you're running a 2.6 kernel or later, you can more easily see the network interface names listed in the `/sys/class/net` directory. Here's an example of what you'll see if you display the contents of this directory:

```
eth0 eth1 lo
```

This shows that there are two physical Ethernet interfaces (`eth0` and `eth1`) and a local loopback interface (`lo`). If you know that your computer has a network interface but you don't see its name in one of these files, it means the interface driver is not loaded.

Interface Names

Network interfaces have default names. The first Ethernet interface is called **`eth0`**, the second is called `eth1`, and so on. Token-Ring interfaces are called `tr0`, `tr1`, and so on. Table 2-1 provides a partial list of networking interfaces used by Linux.

Table 2-1 Examples of Linux interface names

Interface Name	Description
`eth0`	Ethernet interface
`tr0`	Token-Ring interface
`lo`	Local loopback interface
`ppp0`	Point-to-Point Protocol
`slip`	Serial Line Internet Protocol
`plip`	Parallel Line Internet Protocol
`arc0`	ARCnet interface
`fddi0`	FDDI interface
`dlci0`	Frame relay interface
`ippp0`	Integrated Services Digital Network (ISDN) modem

You can change the interface names to suit your taste by using the **ip** program. You'll have to be logged on as root to make changes. If you want to rename the `eth0` interface, you must first take the interface down:

```
ip link set eth0 down
```

change its name:

```
ip link set eth0 name inside
```

then bring the renamed interface back up:

```
ip link set inside up
```

You can display information about the renamed interfaces by using either the `ip` or the `ifconfig` programs. Here's an example of using the `ip` program:

```
ip link show inside
2:  inside: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc
pfifo_fast qlen 1000
       link/ether 00:01:02:ed:b3:bb brd ff:ff:ff:ff:ff:ff
```

Here's an example of using the `ifconfig` program:

```
ifconfig inside
inside  Link encap:Ethernet HWaddr 00:01:02:ED:B3:BB
        inet addr:192.168.1.1 Bcast:192.168.1.255
Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:421 errors:0 dropped:0 overruns:0 frame:0
        TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100
        RX bytes:49814 (48.6 Kb) TX bytes:1434 (1.4 Kb)
        Interrupt:11 Base address:0x1400
```

Most Linux users and programmers do not rename their interfaces and many do not know that you can. Consequently, some programs that refer to network interfaces by name only recognize the default names.

Multiple Interfaces

If your computer has only one Ethernet interface, there's no doubt that it will be called `eth0`. If the computer has two or more interfaces, you face the challenge of determining which of the physical connectors belongs to `eth0`, which belongs to `eth1`, and so on. When Linux boots up, it locates the interfaces and loads drivers for them. The order in which the drivers are loaded determines the interface naming.

Normally, you have no choice about which of the interfaces is detected first and, consequently, the names of the physical interfaces. The order in which the interfaces are detected is determined by your computer's Plug and Play logic in the case of PCI interfaces and by hardware settings (base I/O address) in the case of non-Plug and Play interfaces, such as ISA bus cards.

The most reliable way of determining which connectors belong to which interface names is to connect another computer to each interface and see if the computers can ping one another. After you've determined which physical connectors are associated with the interface names and you connect the appropriate cables, they should stay the same when you reboot as long as you do nothing to change the way the interfaces are detected.

Adding a new interface to your computer can cause existing interfaces to be renamed when the system boots up. Whenever you add another interface, you should verify that the connectors are still associated with the same interface names. If you don't, you won't be the first person to spend hours troubleshooting a problem that is solved by moving the cables to the proper connectors.

Interface Drivers

Network interfaces use device drivers that are implemented as kernel modules. The modules are usually loaded at system start-up but can be loaded or unloaded while Linux is running. You can see which modules are loaded with the **`lsmod`** command. You'll see a list of modules displayed that include those needed to support the interface. Here's an example of a module used to support Intel Gigabit Ethernet interfaces:

```
e1000          76956    2  (autoclean)
```

How do you know that the e1000 module is used by Intel Gigabit Ethernet interfaces? You learn it with experience. Until you have that experience, you should use Internet search engines, such as Google, to find out. Before you use Google, you must know what network interface you have. If you purchased a network interface card and plugged it in to your computer, you probably know the manufacturer and model. If the interface is built in to your computer, such as a notebook computer, you might not know.

You can see what interfaces are inside your computer by running the `lspci` program. Look for one or more lines that contain the words *Ethernet controller* (in the case of Ethernet). The following is an example of one such line:

```
0000:00:04.0 Ethernet controller: Silicon Integrated Systems
[SiS] SiS900 PCI Ethernet
```

You can then use Google to search for *sis900 ethernet linux*. The SiS900 Ethernet page will probably appear at the top of the list. Click the link and the page will tell you about the SiS900 driver. With this information, you can try loading the driver (module) by using the `modprobe` program:

```
modprobe sis900
```

The **`modprobe`** command automatically handles dependencies. It loads any other modules that are required. The module initializes the interface hardware. If everything is successful, the command prompt returns after the `modprobe` command with no additional output. No news is good news.

The **insmod** command also loads kernel modules but it doesn't handle dependencies. The `modprobe` command is usually the better choice.

Chances are good that the driver/module will load properly and you'll see the interface name appear in the `/sys/class/net` directory or when you run the **ifconfig** or **ip link** commands. One problem you might have is that the module is not present on your system. The obvious solution is to locate a copy of the module by doing another Google search, downloading it, and installing it on your system.

Kernel modules are located in a directory somewhere below the **/lib/modules** directory. The exact location varies according to your Linux distribution and the kernel version you are using. On a Fedora Core system, the interface modules are in the directory:

```
/lib/modules/2.6.10-1.766_FC3/kernel/drivers/net
```

When you load a kernel module, you don't need to specify the path to the module's directory. The `modprobe` command is aware of the location of the modules on your system. You also don't need to include the ".o" or ".ko" file extension that you see in the modules' filenames. All you need is the name of the module, as shown in the following examples.

Interface drivers do not need to be implemented as kernel modules. You can recompile the kernel with the drivers compiled into a monolithic kernel. Directions for doing this are provided in Chapter 4 of the *Guide to Linux Installation and Administration* (Course Technology, ISBN 0-619-13095-4) or the kernel-HOWTO document on *www.tldp.org*.

ISA Bus Interfaces

Network interfaces that are **ISA bus** cards might require that you specify details such as the I/O address, interrupt number, and DMA channel. You can specify these as command-line parameters with the `modprobe` command. For example, to specify interrupt 15 and I/O address 300:

```
modprobe ne2000 irq=15 io=0x300
```

Alternatively, you can specify them in a configuration file that `modprobe` uses. This file may be called `/etc/modprobe.conf` or `/etc/modules.conf`. Older Linux systems may use the `/etc/conf.modules` file. The file might look like this:

```
alias eth0 3c59x
options 3c59x irq=15 io=0x300
```

The statements that you can include in this file, and their syntax, are documented in the man page for the appropriate configuration file. For a system that uses `/etc/modules.conf`, refer to the `modules.conf` man page.

Some modules allow parameters that are unique to the network hardware. To learn about the parameters that may be applicable to the module you need to use, refer to the kernel

documentation at *www.tldp.org/HOWTO/Module-HOWTO*. This document includes a list of networking modules and their valid parameters.

Special-purpose Interfaces

Some of the interface names in Table 2-1 are for special purposes and are described in more detail here:

- **Point-to-Point Protocol (PPP)**—This protocol connects two hosts together. PPP can operate over several types of hardware. The most common use of PPP is to create a network connection over a modem to reach an Internet service provider (ISP). Depending on the type of connection you have, you might have something like PPP over Ethernet. For example, if you are using a digital subscriber line (DSL) or a cable modem from your telephone or cable television company (acting as your ISP), you might be connecting a special modem to your Ethernet card and creating a PPP connection to the ISP. PPP is covered in detail in Chapter 3.
- **Serial Line Internet Protocol (SLIP)**—You can use this protocol to transmit network data over a serial port, rather than a special networking cable. SLIP is used with a modem to connect to an ISP, but has been superseded by PPP, which is more flexible. SLIP is still fully supported in Linux.
- **Parallel Line Internet Protocol (PLIP)**—Similar to SLIP, this protocol uses a parallel port as a network interface. PLIP allows you to connect two computers using a parallel cable. Because this is a parallel interface, bytes are sent—not a serial bit stream. The data rate depends on the speed of both parallel ports, but can reach about 20 kB/s.
- **Integrated Services Digital Network (ISDN)**—This is a special type of telephone service that is widely used in Europe. ISDN is available in much of the United States, but is less favored now that DSL and cable modems provide faster service for lower cost. (ISDN provides 128- or 144-kb/s bandwidth, compared to speeds up to 10 times that or more for DSL and cable modems.) ISDN cards work like modems, using a PPP-type networking device. Linux has good ISDN support. Detailed information, including a list of ISDN hardware supported by Linux, is available at *www.isdn4linux.de/faq*.
- Several other types of high-speed networking connections are supported by Linux, though a complete explanation of them is beyond the scope of this book. For example, you can use a **frame relay** card to connect to an ISP using a **T-1** line from your telephone company. T-1 provides a speed of 1.544 Mb/s, usually for a few hundred dollars per month. A **T-3** line provides 45 Mb/s, but usually costs many thousands of dollars per month.

Configuring Networking with Command-line Utilities

Linux distributions use the same commands and similar scripts to configure and control networking. The two traditional commands are `ifconfig` and `route`. These are replaced by the newer `ip` command; the `ip` program is part of the new IPROUTE2 package.

Using the `ifconfig` Command

The `ifconfig` program is the traditional way of viewing and controlling network interfaces. If you enter the `ifconfig` command with no command-line parameters, you'll see a list of all the interfaces that have drivers loaded and their status:

```
ifconfig
eth0 Link encap:Ethernet HWaddr 00:10:5A:9E:9D:6F
   inet addr:192.168.100.10 Bcast:192.168.100.255 Mask:255.255.255.0
   UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
   RX packets:421 errors:0 dropped:0 overruns:0 frame:0
   TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:100
   RX bytes:49814 (48.6 Kb) TX bytes:1434 (1.4 Kb)
   Interrupt:11 Base address:0x1400

lo  Link encap:Local Loopback
   inet addr:127.0.0.1 Mask:255.0.0.0
   UP LOOPBACK RUNNING MTU:16436 Metric:1
   RX packets:52 errors:0 dropped:0 overruns:0 frame:0
   TX packets:52 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:0
   RX bytes:3428 (3.3 Kb) TX bytes:3428 (3.3 Kb)
```

If you haven't yet loaded a kernel module for your interface, it will not appear in the output. You should still see the loopback interface, as indicated by the **`lo`** in the left column of the output. The loopback interface doesn't exist physically—it's a logical interface used for testing and by daemons or applications for communicating within the computer. The loopback interface typically uses the 127.0.0.1 IP address.

The status information may seem cryptic. Here's an explanation.

- `Link encap`—Stands for link encapsulation. For Ethernet interfaces, it typically shows "Ethernet."
- `HWaddr`—Indicates the hardware address or MAC address of the interface.
- `inet addr`—Indicates the IPv4 address of the interface.
- `Bcast`—Indicates the broadcast address of the interface.
- `Mask`—Indicates the subnet or network mask of the interface.
- `UP`—Indicates the interface is up. You can take the interface down with an `ifconfig` command.

- `BROADCAST`—Indicates that the interface supports broadcasting.
- `MULTICAST`—Indicates that the interface supports multicasting.
- `LOOPBACK`—Indicates the interface is a loopback device.
- `MTU`—Stands for maximum transmission unit—the maximum size of a frame (packet) the interface supports. The MTU for Ethernet always defaults to 1500, though you could change it to avoid fragmentation if most of your traffic had to pass through a network segment with a smaller MTU.
- **`Metric`**—Determines the cost of a route that uses this interface. It is normally set to 1, but it can be set to a higher value to make the route less attractive. It is used for routing purposes.
- `RX packets`—Indicates the number of packets that have been received while the interface has been up.
- `TX packets`—Indicates the number of packets that have been transmitted while the interface has been up.
- `collisions`—The number of specific collision errors (remember that Ethernet packets collide if multiple NICs try to use the cable at the same instant).
- `txqueuelen`—Indicates how many packets the network interface can store for transmission while waiting to actually send the packets on a busy network.
- `RX bytes`:—Indicates the number of bytes that have been received by the interface. Following that number, you may see another number in parentheses that displays the same information in a more convenient unit, such as MB for megabytes. Recent versions of the `ifconfig` program display MiB, which stands for mibibytes—a measure of megabytes that is based on binary, not decimal. A mibibyte is 1,048,576 bytes, not 1,000,000.
- `TX bytes`—Indicates the number of bytes that have been transmitted by the interface.
- `Interrupt`—Indicates the interrupt or IRQ of the interface hardware.
- `Base address`—Indicates the I/O address of the interface hardware.
- `Memory`: The memory address range of the interface hardware. This is not displayed if the hardware does not use memory for communicating with the operating system.

You can display the status of only one interface by specifying it like this:

```
ifconfig eth0
```

You can configure and control an interface by specifying the interface name followed by an option. For example, this command stops the `eth0` interface:

```
ifconfig eth0 down
```

To start it again, enter this command:

```
ifconfig eth0 up
```

You can configure the interface IP settings like this:

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0 broadcast
192.168.0.255
```

You can use numerous options with `ifconfig`. Refer to the man page for more details.

When you change the status of an interface that is used to reach your default gateway, the default gateway is deleted. It has to be configured again with the `route` or `ip` command.

Using the `route` Command

The **`route`** program is the traditional way to view and configure the routing table. The routing table determines where packets are to be sent so they can reach their destination. Using the `route` command with no parameters displays the routing table. Here's an example for a host with one Ethernet interface (`eth0`), an IP address of 10.0.1.1, and a subnet mask of 255.255.255.0:

```
route
Kernel IP routing table
Destination  Gateway    Genmask        Flags Metric Ref Use Iface
10.0.1.0     *          255.255.255.0  U     0      0   0   eth0
127.0.0.0    *          255.0.0.0      U     0      0   0   lo
default      10.0.1.3   0.0.0.0        UG    0      0   0   eth0
```

The output consists of three lines (after the column headings):

- A line defining where to send traffic for the 10.0.1.0 network—the network to which this computer is directly connected.
- A line defining where to send traffic for the 127.0.0.0 network—the loopback interface.
- A line defining where to send any packet with a destination address on a network other than the two just mentioned. These packets must go to the gateway because this system doesn't know how to reach any other networks.

Now look at the columns in the output of the `route` command:

- `Destination`—The network or host to which the routing table entry applies. If the destination address of an IP packet is part of the network listed on a line, that entry will be used to route the packet.
- `Gateway`—The IP address of the host that should receive a packet destined for the specified network. An asterisk (*) is displayed to indicate which network the host is a part of; no routing is needed.

- `Genmask`—The network mask of the routing table entry. The entry for the default gateway is supposed to be 0.0.0.0.
- `Flags`—The nine single-letter flags that indicate information about this routing table entry. A `U` indicates that the route is up; a `G` indicates that the route refers to a gateway. An `H` indicates that the route refers to a host. The other six flags are only displayed when the host is running a dynamic routing protocol.
- `Metric`—The number of hops this route represents.
- `Ref`—The number of references made to this route. This is not used by Linux.
- `Use`—The number of times this route has been looked up by the routing software. This gives a rough measure of how much traffic is headed for the specified network.
- `Iface`—The interface on which packets destined for the specified network should be sent.

In the following examples of the `route` command's output, the Ref and Use columns are not shown to save space on the page.

The routing table is normally handled automatically. When an interface is assigned an IP address, the Linux kernel determines the network address and an entry for that network is placed in the routing table. Look at the following example. Assume that no interfaces are active (up). The routing table is empty:

```
route
Kernel IP routing table
Destination  Gateway     Genmask       Flags Metric Iface
```

Bring up the `eth0` interface and assign it an IP address of 10.0.1.3 and a mask of 255.255.255.0 (a /24 network):

```
ifconfig eth0 10.0.1.3 netmask 255.255.255.0
```

Here's what happens to the routing table:

```
route
Kernel IP routing table
Destination  Gateway     Genmask       Flags Metric Iface
10.0.1.0     *           255.255.255.0 U     0      eth0
```

The Linux kernel automatically determined the network address is 10.0.1.0 and created the routing table entry. It did not create an entry for the default gateway because it has no way to automatically determine the address of the gateway. This must be done manually or by a script. Assuming the gateway address is 10.0.1.1, here's the syntax of the `route` command that adds a default gateway into the routing table:

```
route add default gw 10.0.1.1
```

The default gateway is added to the routing table:

```
route
Kernel IP routing table
Destination  Gateway     Genmask        Flags Metric Iface
10.0.1.0     *           255.255.255.0  U     0      eth0
default      10.0.1.1    0.0.0.0        UG    0      eth0
```

Figure 2-1 shows a typical routing scenario. It shows a network with two Ethernet segments that connect to the local router—Host C. Host C also has an interface that connects to the Internet via an ISP's router. Host A and B have their default gateway set to 10.0.1.1. Host D has its default gateway set to 10.0.2.1. Host C also has a default gateway setting that points to the ISP's router—69.30.87.1.

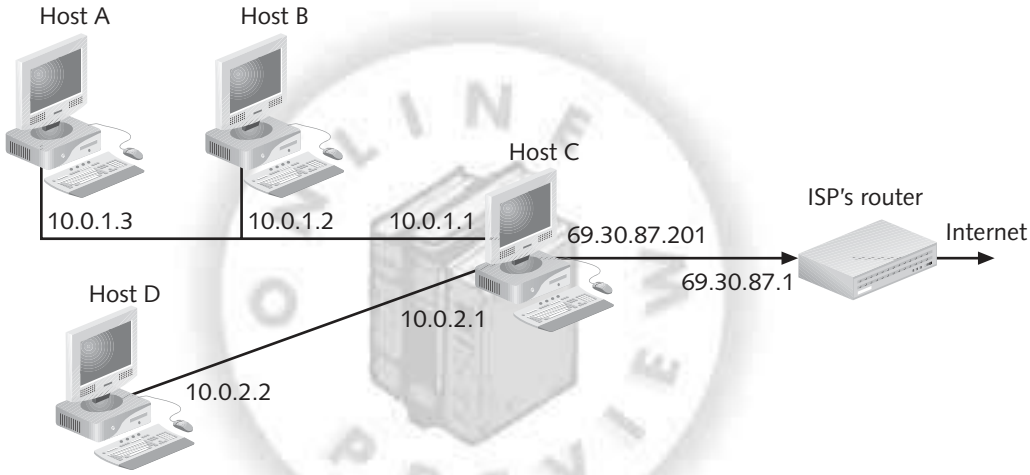

Figure 2-1 A two-segment network with an Internet connection

This example is relatively simple with regard to the routing table. You just need to ensure that all hosts point to the correct default router. The Linux kernel determines the rest. Things are more complex with networks that have more than one gateway. Figure 2-2 shows that Host B is now a router—the gateway to the Internet instead of Host C. Yet, Host C is still a router. It connects networks 10.0.1.0 and 10.0.2.0 together.

To what should Host A's default gateway be set? If it's set to 10.0.1.2, it will be able to reach the Internet but it won't be able to reach network 10.0.2.0. If it's set to 10.0.1.1, it will be able to reach network 10.0.2.0 but it won't be able to reach the Internet.

The solution is to add another route to the host's routing table. Its default gateway will be set to 10.0.1.2. Another route is needed for network 10.0.2.0. You can add this route with this command:

```
route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.1.1 dev eth0
```

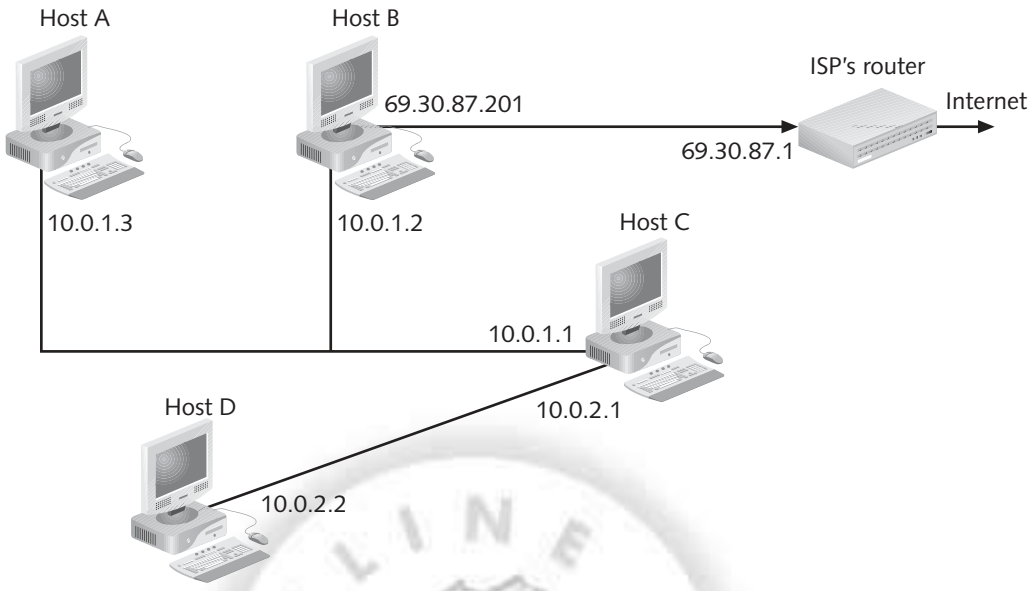

Figure 2-2 A two-segment network with two gateways

Host A can now reach the Internet and network 10.0.2.0. Its routing table now includes two gateways:

```
route
Kernel IP routing table
Destination  Gateway     Genmask         Flags Metric Iface
10.0.1.0     *           255.255.255.0   U     0      eth0
10.0.2.0     10.0.1.1    255.255.255.0   UG    0      eth0
default      10.0.1.2    0.0.0.0         UG    0      eth0
```

Host B has its default gateway set to 69.30.87.1—the ISP's router. However, it doesn't know about network 10.0.2.0. It also needs to have another route added to its routing table, just like Host A. The `route` command is the same as for Host A. Host B's routing table also has two gateways:

```
route
Kernel IP routing table
Destination  Gateway     Genmask         Flags Metric Iface
10.0.1.0     *           255.255.255.0   U     0      eth0
10.0.2.0     10.0.1.1    255.255.255.0   UG    0      eth0
default      69.30.87.1  0.0.0.0         UG    0      eth0
```

Host C already has routes for networks 10.0.1.0 and 10.0.2.0. The only change is to set its default gateway to 10.0.1.2 so it can reach the Internet. The change from Figure 2-1 to Figure 2-2 is invisible to Host D, which uses the same routing table as before.

Using the ip Command

The ip command is a recent addition to Linux. It is part of the IPROUTE2 package that most distributions now include. It's meant to be a replacement for the ifconfig, route, and arp commands. These excerpts from the Linux Advanced Routing & Traffic Control HOWTO explain why:

- *Most Linux distributions … use the arp, ifconfig, and route commands. While these tools work, they show some unexpected behavior under Linux 2.2 and up.*
- *As new networking concepts have been invented, people have found ways to plaster them on top of the existing framework…This constant layering of cruft has led to networking code that is filled with strange behavior…*

One of the best features of the ip command is that it allows you to use CIDR notation. This was covered in Chapter 1 to a degree, but this section takes a closer look at it. The classful notation that ifconfig requires is awkward—you need to specify three parameters to ensure that you've properly configured an interface: IP address, subnet mask, and broadcast address. The following is an ifconfig command that illustrates this using a subnetted address:

```
ifconfig eth0 10.1.1.204 netmask 255.255.255.248 broadcast 10.1.1.207
```

This is a lot to remember and a lot to type. The following is the equivalent ip command using CIDR notation:

```
ip addr add 10.1.1.204/29 dev eth0
```

Instead of three parameters, there's only one: 10.1.1.204/29. From this single piece of information, the network address and the broadcast address are easily derived.

You can use the ip program to work with the routing table—eliminating the need for the route program. To display the routing table, use the ip route command:

```
ip route
10.1.1.200/29 dev eth0 proto kernel scope link src 10.1.1.204
default via 10.1.1.201 dev eth0
```

The second line that starts with default is the default gateway. You can add a default route with a command like this:

```
ip route add default dev eth0 via 10.1.1.201
```

The ip program does not display interface statistics, such as the number of packets transmitted and received, unless you ask for them:

```
ip –statistics link dev eth0
eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
link/ether 00:11:43:35:c0:cc brd 00:00:00:00:00:00
RX: bytes  packets  errors  dropped  overrun mcast
1578604995 12828220 7       0        0       0
```

```
TX: bytes  packets  errors  dropped  carrier collsns
212168783  217086443 0     0     0     0
```

The `ip` program has much more functionality that's not described here. You can refer to the `ip` man page for more detail, but it's not easy reading. You might have better results searching the Internet for tutorial information on the command.

Adding IP Addresses to Interfaces

Sometimes you need to assign more than one IP address to an interface. This is often called **IP aliasing**. You'll read about a practical application for this in Chapter 6. You have two ways to add addresses to interfaces that are quite different from one another. One way uses the `ifconfig` program and the other uses the `ip` program.

When you use `ifconfig`, the aliases have names. The names are typically created by appending a colon followed by a number to the interface name. If the interface is `eth0`, the first alias would be called `eth0:0`. The second alias would be `eth0:1` and so on. The following is an example of creating an alias using `ifconfig`:

```
ifconfig eth0:0 10.0.0.2 netmask 255.0.0.0 broadcast
10.255.255.255
```

When you run the `ifconfig` program with no command-line parameters, you can see both the `eth0` interface and the `eth0:0` interface. Alias names do not have to use a number following the interface name. You could use a short string of alphabetic characters such as this:

```
ifconfig eth0:web1 10.0.0.2 netmask 255.0.0.0 broadcast
10.255.255.255
```

However, some programmers do not know that you can do this. Consequently, some programs do not work with aliases that don't use numbers.

When you use the `ip` program to create aliases, the aliases do not have names. You simply add an IP address to the interface. The following command is equivalent to the previous examples:

```
ip addr 10.0.0.3/8 dev eth0
```

When you create aliases using `ip`, they do not appear in an `ifconfig` listing. However, Linux knows about these aliases and uses them. If you create an alias using `ifconfig`, you can see it in an `ip` listing.

Using the `ip` command is generally the better way to create aliases.

PCMCIA and PC Card Interfaces

Mobile computers can use PCMCIA cards or PC Cards for their network interfaces. These interfaces are hot-pluggable—they can be inserted and removed anytime you want. Linux needs to be able to load and unload their drivers on the fly. When Linux support for these

hot-pluggable devices was introduced, there was no generic hotplug mechanism. So, a hotplug mechanism that used scripts and configuration files specific to PCMCIA and PC Cards was developed.

Kernel modules for the cards are kept in a directory separate from the non-hotplug interfaces. On a Fedora Core system, these interface modules are in the directory:

```
/lib/modules/2.6.10-1.766_FC3/kernel/drivers/net/pcmcia
```

These cards have their own configuration files and scripts in the `/etc/pcmcia` directory. To use a PCMCIA or PC Card interface, edit the `/etc/pcmcia/network.opts` file. Then create a hotplug event by removing and inserting the card. The scripts should run and activate the interface with the settings you placed in the `network.opts` file. If this doesn't work, you have some troubleshooting on your hands. This may involve having to deal with the details of card services. You can visit the Linux PCMCIA Information Page at *http://pcmcia-cs.sourceforge.net* for more information.

PCMCIA Card Services allow you to create named configurations that let the interface behave differently depending on whether you're at home, in the office, or elsewhere.

Since PCMCIA and PC Card interface support was integrated into Linux, interfaces based on USB and FireWire have been introduced. These are also hot-pluggable devices. We need a more generic and intelligent hotplug system. The new `/sys` filesystem that was introduced with kernel version 2.6 now makes this possible, and work is progressing at a rapid rate. Sometime in the future, the `/etc/pcmcia` directory will go away.

Wireless Interfaces

Wireless interfaces, such as wireless Ethernet, have their own unique capabilities and requirements. They have a radio receiver and transmitter that need to be configured and monitored. Instead of adding wireless support to programs like `ifconfig`, additional wireless-specific programs were developed. These are **`iwconfig`** and `iwlist`.

The `iwconfig` program is the primary tool for configuring wireless interfaces. It is used to display and configure parameters that are peculiar to wireless interfaces, such as frequency, transmit power, data rate, encryption key, and selecting between ad hoc or access point mode. You still need to use the `ifconfig` or `ip` program to set the interface's network parameters, such as IP address.

The `iwconfig` syntax is:

```
iwconfig interface option
```

where *interface* is the name of the interface and *option* is one of the following:

- `essid` *name*—Sets the Extended Service Set Identifier or ESSID. This is the name of the overall wireless network, which might consist of more than one cell. This is needed so a user can roam between cells. The name is case sensitive and limited to 32 characters.
- `nwid` *name*—Sets the network ID of the cell. This is needed only if the WLAN is running in ad hoc mode. If you use an access point, you don't need to define the network ID.
- `freq` *number*—Sets the transmitter's frequency. You can enter the frequency in Hz but you'll want to use the G suffix (2.46 G) instead.
- `mode` *mode*—Sets the operating mode of the interface, where *mode* can be Ad-Hoc, Managed, Master, Repeater, Secondary, Monitor, and Auto.
- `ap` *x*—Sets the access point to be used, where *x* is the MAC address of the access point you prefer or *any* chooses the best access point.
- `key` *key*—Sets the encryption key, where *key* is in a numeric form using hex digits (1234-5678-90ab or 1234567890ab) or in string form (s:mysecretkey). If you need to use more than one key, refer to the `iwconfig` man page.

Not all `iwconfig` options are shown here. Refer to the `iwconfig` man page for the lesser-used options.

The `iwlist` program displays detailed information about the interface's radio. The syntax is:

```
iwlist interface option
```

where *interface* is the name of the interface and *option* is one of these:

- `event`—Lists the wireless events supported by the interface
- `frequency`—Lists the interface's available frequencies
- `key`—Lists the encryption key sizes that the interface supports
- `power`—Lists the power management attributes and modes
- `rate`—Lists the data rates supported
- `retry`—Lists the transmit retry limits
- `scanning`—Lists the access points and ad hoc cells that are in range
- `txpower`—Lists the transmit power levels the interface supports

Using `proc`

You can view and set many network configuration parameters by using the **`proc` filesystem**. This is a virtual filesystem that allows you to view and modify kernel settings. Many network-related parameters can only be modified by using the `proc`

filesystem. To view the `proc` filesystem, go to the `/proc` directory. All the files and directories you see there don't exist on your hard disk. They are kernel settings made to look like a filesystem's directory structure so you can use ordinary text-oriented tools to view and set them.

For example, enter the command `cat /proc/cpuinfo`. You see information regarding your computer's CPU. Network-related kernel settings are in two `/proc` directories: `/proc/net` and `/proc/sys/net`. The `/proc/net` directory structure contains status information—you can only view this information. The `/proc/sys/net` directory structure contains network-related settings that you can modify.

You can tell which `proc` files you can modify by doing a long directory listing. For example, enter `ls -l /proc/net/` and you see entries like this:

```
-r--r--r-- 1 root root 0 Feb 25 09:28 anycast6
```

If you're up to speed on Linux/UNIX permissions, you can see that you only have read access to this "file." Permissions are covered in a later chapter. If you look at files in the `/proc/sys/net` directory structure, you see that the root user has write permission to the files there.

To see how you can modify network parameters, enter `cat /proc/sys/net/ipv4/ip_local_port_range`. For a default Fedora Core installation, you see this:

```
32768    61000
```

These numbers are the lower and upper limits for the pool of port numbers that the kernel assigns to TCP and UDP clients. In Chapter 1, you discovered that these are called dynamic ports and learned that the proper range is from 49,152 to 65,535. Fedora Core and most other Linux distributions don't use the proper range. You can fix this by setting the proper values using this command:

```
echo "49152 65535" > /proc/sys/net/ipv4/ip_local_port_range
```

The change takes place immediately. Verify that this worked by entering `cat /proc/sys/net/ipv4/ip_local_port_range` and noticing that the proper numbers are displayed. All new client connections will now use ports from this range. Existing connections will continue to use the ports assigned to them in the past.

The change is made to the running kernel. If you reboot your computer, the change will be lost. You must put the `echo` command into one of the start-up scripts to have it take effect when you boot your computer. The `/etc/rc.d/rclocal` script is one such script. It runs after all other start-up scripts.

Another `proc` setting that you should know about is `/proc/sys/net/ipv4/ip_forward`. This needs to be set to 1 if you have more than one interface and you want to route packets between the interfaces (you want the computer to behave as a router). Your Linux distribution probably sets this automatically, but if it doesn't you'll have problems. You can set it with this command:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Many network-related files are in the `proc` filesystem. The filenames and subdirectories you see vary depending on the features compiled into the kernel and the kernel modules that are loaded. You'll take a look at other `proc` settings throughout this book.

Using ARP

In Chapter 1, you learned about the Address Resolution Protocol (ARP), which obtains the hardware address of a host given its IP address. Now that you are more familiar with the `route` command, you can try the **`arp`** command. You'll use the `arp` command mainly for troubleshooting network problems.

The ARP protocol maintains a table in memory called the **ARP cache**, which is a mapping of IP addresses to interface hardware addresses. You can view the ARP table with the command `arp -a` (the "a" is for "all"):

```
arp-a
fire.course.com (10.0.1.1) at 00:0D:87:F0:76:1A (ether) on eth0
igor.course.com (10.0.1.3) at 00:40:63:C2:8A:64 (ether) on eth0
```

The first entry says that the host whose name is *fire.course.com* and IP address is 10.0.1.1 has an Ethernet interface with a MAC address of 00:0D:87:F0:76:1A. The second entry says that the host whose name is *igor.course.com* and IP address is 10.0.1.3 has an Ethernet interface with a MAC address of 00:40:63:C2:8A:64.

ARP table entries are dynamic and are discarded if not referenced within two minutes. You can see this by pinging some hosts on your network and on the Internet. Then run the `arp -a` command and note the entries. Wait a few minutes and run the `arp -a` command again. You'll see that some of the table entries have been deleted.

You can use the `arp` command to add an entry manually, but there are few practical reasons for doing so.

System Networking Scripts

Linux distributions provide numerous scripts and configuration files to set up your interfaces and to make starting and stopping the interfaces easy. The networking scripts follow the model used for most system services on UNIX-based computers. The simplistic view is that networking on most Linux distributions is controlled by a single script: `/etc/init.d/network` or a similar name. You can start up networking with this command:

```
/etc/rc.d/init.d/network start
```

Using this command stops networking:

```
/etc/rc.d/init.d/network stop
```

These two steps are combined with this command:

```
/etc/rc.d/init.d/network restart
```

Elegant though it might be, this grand network script begs the question: How does it do all of this? The answer is that more scripts and configuration files lie behind the scenes. All of these are located in the directory **/etc/sysconfig**, under the subdirectories `network-scripts` and in the file `networking`.

The `network-scripts` subdirectory contains scripts that start and stop individual network interfaces. In that directory, you see two main scripts called **ifup** and **ifdown**. These are used to control more common interfaces such as localhost and Ethernet. Other specialized scripts can also be found in the subdirectory for interfaces, such as IPv6, PPP, or ISDN.

A question should be forming in your mind: Why use a script instead of just the `up` and `down` options with `ifconfig`? The short answer is this: The scripts can be more intelligent in handling the network interfaces. For example, they can check whether a firewall configuration needs to be adjusted, whether a wireless interface is involved, whether default routes must be updated, and so forth. You can obtain the more complex answer if you have any experience with shell scripts: Open the files in a text editor and study them.

Scripts that control individual network interfaces do not contain the actual configuration data. That information is stored in separate files in the `/etc/sysconfig/network-scripts` subdirectory according to the name of the interface. The file you're most likely to see is `/etc/sysconfig/network-scripts/ifcfg-eth0`, which contains information to configure the first Ethernet interface. The contents of that file are shown here:

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.100.255
IPADDR=192.168.100.10
NETMASK=255.255.255.0
NETWORK=192.168.100.0
ONBOOT=yes
```

If you are using DHCP to obtain IP address information, your `ifcfg-eth0` file will look different from the one shown here.

With this file, you can finally see how the data to configure the interface is actually stored. If you were to change the IP address in this file and use the command `/etc/rc.d/init.d/network restart`, the IP address of your system would be changed. Most Linux systems add graphical configuration utilities to let you change IP addresses, add network interfaces, and so forth. Behind the scenes, those utilities may be changing the data files and using the system scripts that control networking.

An earlier section in this chapter discussed how to add an IP address to an interface. This was called IP aliasing and you learned how to do it with the `ifconfig` and `ip` commands. The scripts used by Fedora Core are designed around all aliases having a name. This makes

the `ifconfig` command the better one to use in these scripts. The scripts would have to be redesigned to make use of the `ip` command.

To add an IP alias, you create a new file in the `/etc/sysconfig/network-scripts` directory with the name `ifcfg-eth0:0`. The easiest way is to copy the existing file `ifcfg-eth0`. Then change the values of the `DEVICE` and `IPADDR` lines to refer to `eth0:0` and the new IP address. Now restart networking to make the additional IP address automatically active. Every time you start up Linux, the additional IP address is started. To stop this, delete the additional file that you created and restart networking. This experiment is one of the Hands-On Projects at the end of this chapter.

Configuring Networking Using Graphical Tools

A good system administrator knows how things work "under the hood," rather than just relying on the basic tools. By knowing about the networking scripts and the `ifconfig`, `route`, and `ip` commands, you'll have a better grasp of how to diagnose any problems that arise on your network. You'll also be better prepared to do well on a Linux certification test. After you understand these things, however, you might find it easier to use graphical tools to configure your network. Every major version of Linux includes graphical utilities to help you manage networking. This section describes the utilities included with some popular versions of Linux.

Fedora Core includes the graphical **Network Configuration Tool** for managing network interfaces. Start the tool from the graphical menu of the GNOME desktop by choosing Applications, System Settings, and then Network. You can also start it from a command line by entering the utility name: **`neat`**. Figure 2-3 shows the main screen of this tool.

This section only explores the Hardware and Devices tabs. The Hosts and DNS tabs are covered later.

When the tool starts, the Devices tab is selected. It displays a list of each Linux networking device name and its status. Select one of the interfaces and click Edit. You'll see the Ethernet Device window shown in Figure 2-4. The General tab allows you to perform the following tasks:

- Activate the interface when the computer boots up. Normally, this is enabled.
- Allow all users to control the interface. Normally, this is not enabled because of security concerns about giving users such power.
- Enable IPv6 for the interface. Enable this only if you need IPv6 support.

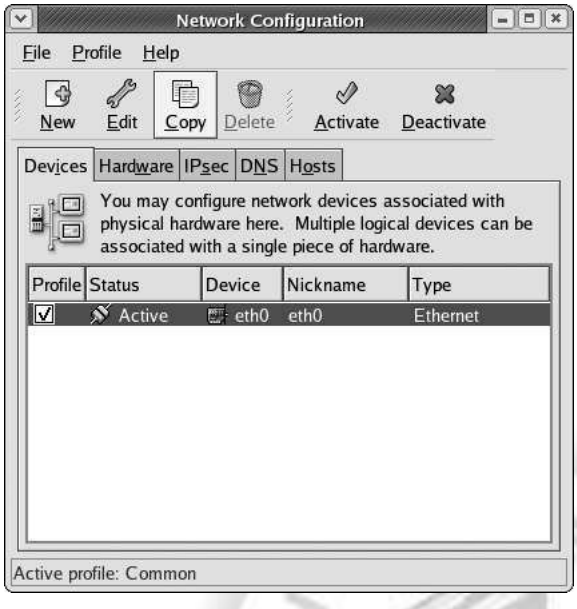

Figure 2-3 The main screen of the Fedora Core Network Configuration Tool

- Choose to obtain an IP address lease from a DHCP or BOOTP server. You'll learn about DHCP in Chapter 3.
- Choose to use a static IP address and gateway.

The Route tab of the Ethernet Device window lets you define a list of static routes that will take effect when you start or restart networking. You should only need to do this if you have nonstandard routing needs caused by working on a large network with multiple gateways or points through which you can reach the Internet. The Hardware Device tab is ill-named. It actually allows you to define IP aliases for the interfaces. Click OK to return to the main screen.

The Hardware tab of the Network Configuration Tool (see Figure 2-5) shows a list similar to the Devices tab, but the manufacturer and model of the interface are shown. Click Edit to reveal the Network Adapters Configuration window (see Figure 2-6), which allows you to configure the hardware settings for the interface. Click New to define another interface. You can select the type of hardware for which you want to add support to the kernel. The supported options are Ethernet, Token-Ring, ISDN, Modem, and Wireless. Linux supports many other types of networking, but the options shown are the ones you can manage using this utility. To configure others, you must use the command line.

After you make any changes in the dialog boxes of the `neat` utility, click the OK button to close each dialog box. Then click the Close Window button and confirm that you want to save your changes and update the system with the new information you've entered.

Figure 2-4 The Ethernet Device window in the Fedora Core Network Configuration Tool

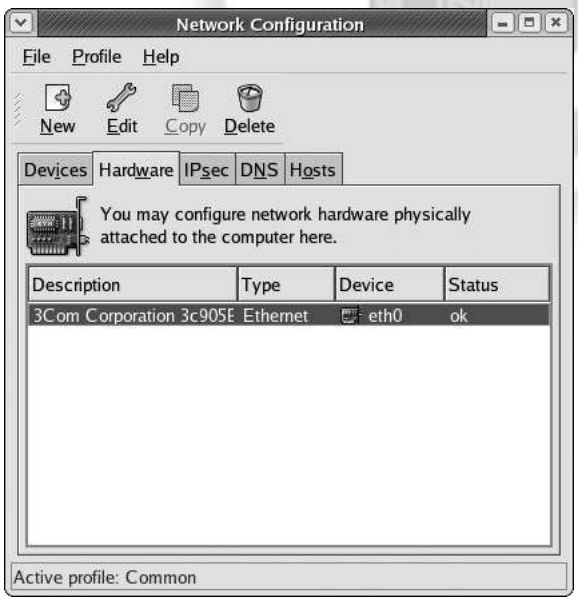

Figure 2-5 The Hardware tab of the Fedora Core Network Configuration Tool

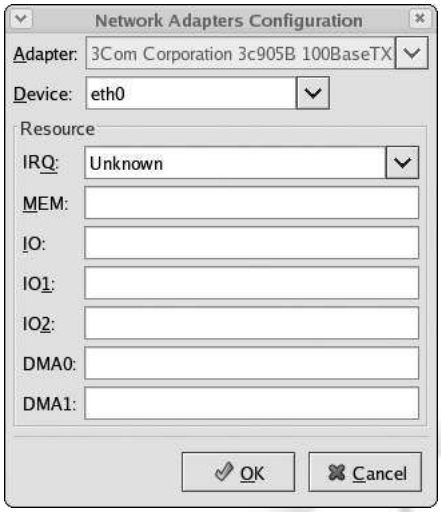

Figure 2-6 The Network Adapters Configuration window

Another popular graphical configuration program is **Webmin**, a Web browser-based program. When you start Webmin, you're starting a tiny Web server that runs on a high-numbered TCP port—usually 10,000. You then point your Web browser at that port and Webmin answers. It first requires that you log on. Thereafter, you can configure most aspects of Linux, including networking.

Webmin works with most versions of Linux as well as most versions of UNIX. If you or your IT staff need to maintain Linux and UNIX computers, Webmin allows different OS platforms to be managed by a common program with a common user interface. Training costs are reduced.

Webmin is extensible with modules. Anyone can write Webmin modules to add needed functionality. For example, if Webmin provides DNS support for BIND but not for another DNS server, someone can write a Webmin module for the other DNS server. Webmin and Webmin modules are written primarily in Perl.

Webmin is a moving target—its user interface changes periodically. Therefore, you should refer to the Webmin Web site (*www.webmin.com*) for more details. You should download Webmin from the Web site, install it, and practice using it.

Using Basic Networking Utilities

With basic networking running smoothly on your Linux system, you're ready to begin exploring the utilities and applications that make it useful. This section provides a description of three popular utilities that you should become familiar with as troubleshooting and informational tools. For each, a brief summary and a few examples are provided. To learn more, you should experiment with these utilities (the projects at the end of this chapter are

a good place to start) and review the online man page documentation when questions arise about how to use them.

The Telnet Remote Logon Utility

Telnet is a terminal emulator program. It allows you to log on to a remote computer as if you were sitting at that computer's terminal (keyboard and screen). After you are logged on, you can view any files that your user account permits you to view. You can also execute any command, including changing or deleting files, as long as you have the permissions to do so. Telnet is also used for troubleshooting network problems.

Telnet is a convenient program, but a warning is in order: Telnet packets are not encrypted or encoded to prevent eavesdropping. Everything you do in Telnet can easily be viewed by anyone connected to the same network. You should use the `ssh` program when eavesdropping could be a problem.

To use Telnet as a terminal emulator program, you must connect to a host running a Telnet server (daemon). The details of setting up clients and servers are discussed in Chapters 4 and 5. Many Linux systems already have a Telnet server running (though this is considered a security risk). All Linux systems have a Telnet client installed, as do most Windows systems.

The `telnet` command in Linux requires you to enter the host name or IP address to which you want to connect. If a connection can be made—that is, if a Telnet server on that host accepts your client's request for a connection—then you are prompted to enter a username and password. After logging on, you see a command prompt as if you were sitting at the remote computer.

If you want to try a Telnet session from Windows, launch the Telnet program included with some Windows versions or obtain a free Telnet program. (You can search on a site such as *www.tucows.com* for free Windows Telnet client software.) Graphical Telnet programs generally ask for the following information:

- The remote host to which you want to connect. For this field, you can enter an IP address or a host name.
- The port number to which you want to connect. By default, the Telnet port—23—is used. You can choose other ports for testing or experimentation as you learn about other services.
- The type of dumb terminal that you want the software to emulate. A standard choice would be a **VT100** terminal. Hundreds of dumb terminal models exist, but VT100 is widely supported.
- You might be asked to enter a username and password, which is sent to the remote host as part of your logon. The Telnet program typically requests that this information be saved, so you don't have to enter it each time you connect to the same remote host. Figure 2-7 shows a Telnet session being started in the Windows Telnet program. (Depending on your version of Windows, the program may differ slightly from that shown in the figure.)

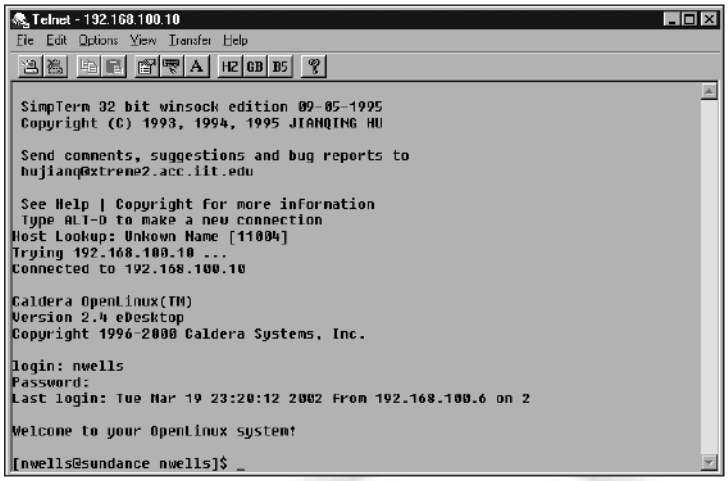

Figure 2-7 The Windows Telnet program with a session in progress

Using `ping` for System Testing

In Chapter 1, you learned about the `echo` message type in the ICMP protocol. ICMP is used to send status or error messages about an IP connection; `echo` is used to send test packets to see whether a remote host can be reached. The utility that you use to send ICMP echo packets is called **`ping`**.

To use `ping`, just include the host name or address of the host that you want to contact. For example:

```
ping 198.60.22.20
```

This command sends a series of ICMP echo packets to the host with the IP address 198.60.22.20. If the host is reachable, the IP stack on that host responds with a message saying "I'm here," and your system displays a message to show that the remote host is "alive." Another packet is sent every second until you stop the `ping` program. The resulting listing looks like this:

```
PING 198.60.22.20 from 64.24.90.213 : 56(84) bytes of data.
64 bytes from 198.60.22.20: icmp_seq=0 ttl=244 time=319.537 msec
64 bytes from 198.60.22.20: icmp_seq=1 ttl=244 time=299.984 msec
64 bytes from 198.60.22.20: icmp_seq=2 ttl=244 time=299.994 msec
64 bytes from 198.60.22.20: icmp_seq=3 ttl=244 time=300.008 msec
64 bytes from 198.60.22.20: icmp_seq=4 ttl=244 time=280.005 msec
64 bytes from 198.60.22.20: icmp_seq=5 ttl=244 time=280.006 msec
64 bytes from 198.60.22.20: icmp_seq=6 ttl=244 time=279.998 msec
-- 198.60.22.20 ping statistics --
7 packets transmitted, 7 packets received, 0% packet loss
  round-trip min/avg/max/mdev =
  279.998/291.298/319.537/10.805 ms
```

When you press Ctrl+C to end the `ping` program, it calculates the statistics that are printed at the end of the listing. It's common to use a series of `ping` commands to test networking. By using these commands in the order given here, you can identify at what point a problem occurs.

The `ping` command on a Windows system simply sends four echo packets and then stops; note that in Linux, `ping` continues until you press Ctrl+C.

- Ping `127.0.0.1` to check that the internal networking stacks are functioning and that networking is enabled in the Linux kernel.
- Ping your own IP address to check that your networking card is configured as expected.
- Ping the IP address of another host on your local network segment to check that your Ethernet card and cable are functioning correctly.
- Ping the IP address of a host on another network segment close by to check that the default gateway or basic routing tables (depending on the complexity of your network) are correctly configured.
- Ping an IP address beyond your organization to check that distant routers are able to work with those in your organization in getting traffic to and from the Internet or other distant networks.

You can also use `ping` with a host name, as you did in the previous example. Ping your own host name to see that it can be resolved (converted into an IP address). Then ping the host name for another host on your local network segment, then of a distant host or Internet site. Chapter 3 explores this further, but the basic idea is the same: Ping to hosts farther and farther away from your system. If something stops working, you know where the error occurred. For example, if you can ping other hosts within your local network segment but not on any other segments, you know that your packet cannot get outside the local segment. This might be caused by bad cabling between segments, a lack of IP forwarding at the router, a bad routing table entry, a firewall entry blocking your traffic, or a few other things. Pinging lets you know where to start looking.

Several types of attacks in the past have relied on the `ping` command. One example is the "flood ping," which sends many `ping` commands in rapid succession, overwhelming the server that tries to respond to them. Another example is the "ping of death," in which a single `ping` command with a very large payload is sent to a server; the payload overflows the memory space allocated to receive ICMP packets and corrupts other parts of the server's memory. Because of attacks like these, many hosts on the Internet completely block pings originating from anywhere outside their organization.

Find a few Internet sites (universities are a good bet) that respond to pings so you can use them for troubleshooting.

Notice the parts of the `ping` output from the listing shown previously. They include the following:

- The number of bytes sent in the packet. You can change this using a command-line option.
- The host name (if you included one) and IP address of the host you are pinging.
- A sequence number, starting with 0. ICMP assigns the number and simply keeps counting up sequentially.
- The Time to Live (TTL) of the ICMP packet. This is the number of routers that the packet can pass through before being discarded. The default here—224—is quite high.
- The time elapsed between sending the ICMP echo packet and when it returns. In this example, you see a time of about 280 milliseconds (shown as msecs). This time gets shorter as the host you're pinging gets closer. Try pinging within your local network to see much shorter times.
- The statistics shown after you press Ctrl+C include minimum, maximum, and average time to get a response, plus the number of packets for which no response was received (typically zero percent if you reached the host at all).

The `ping` command has numerous command-line options that let you set parameters, such as the number of packets to send before automatically stopping, the time to wait between each packet (the default is one second), the size of packet to send (the default is 64 bytes), and other specialized features that are useful for debugging routing problems. For example, the `-R` option lists all of the intervening routers that your ping packet passed through to reach the destination host. (This is similar to the `traceroute` command described next.)

Using `traceroute` to Examine Routing Patterns

When the information provided by `ping` is not enough to help you figure out a routing problem, `traceroute` is the logical next step. The **`traceroute`** command carefully identifies each router (each hop) between you and another host. This lets you see the path your packets are taking and how long each hop takes. Sample output from the command `traceroute 198.60.22.77` is shown here:

```
 1 192.168.100.5 (192.168.100.5) 2.922 ms 9.798 ms 19.928 ms
 2 wdc2-dial7.popsite.net (64.24.80.232) 219.817 ms 199.408
   ms 199.944 ms
 3 wdc2-dial7.popsite.net (64.24.80.232) 199.930 ms 209.430
   ms 199.942 ms
 4 wdc2-core1.popsite.net (64.24.80.225) 199.995 ms 199.461
   ms 199.923 ms
```

```
 5 wdc1-core1-p1-0.starnetusa.net (64.24.80.1) 200.004
   ms 199.437 ms 189.952 ms
 6 wdc3-core1-pos2-0.starnetusa.net (216.126.145.122)
   199.987 ms 199.401 ms 199.952 ms
 7 jsy1-core1-a3-0-2.starnetusa.net (216.126.145.105)
   220.021 ms 949.325 ms 900.048 ms
 8 chi1-core1-s4-0.starnetusa.net (216.126.145.113) 1619.974
   ms 3579.417 ms *
 9 * sjc1-core1-a3-0-2.starnetusa.net (216.126.146.18)
   2045.692 ms 319.424 ms
10 pao1-core1-p6-0.starnetusa.net (216.126.145.97) 360.012
   ms 319.415 ms 339.969 ms
11 xmission-paix.xmission.com (198.32.176.42) 289.961
   ms 1049.435 ms 949.921 ms
12 xmission-paix.xmission.com (204.228.132.29) 819.957
   ms 309.562 ms 309.941 ms
13 core-border.xmission.com (166.70.4.10) 309.973
   ms 309.580 ms 309.887 ms
14 ftp.xmission.com (198.60.22.77) 309.989
   ms 319.627 ms 309.934 ms
```

Each line is numbered. In this example, it took 14 hops to get from our host to the 198.60.22.77 host. The IP address is shown for each router along the way, along with the router's host name if it has one. Three timing values are shown after the IP address for each router. `Traceroute` sends three "probe packets" and shows the length of time that each took to respond. If an asterisk (*) appears in the listing, the router did not respond within the default time limit of five seconds. `Traceroute` tries a maximum of 30 hops to reach a destination, though you can change that number using a command-line parameter.

`Traceroute` relies on the TTL field and ICMP "packet timed out" messages to move step-by-step through the Internet to reach the host in which you are interested. Because not all systems follow standard practices in using these features, some lines of the `traceroute` output may contain nothing but three asterisks. In most cases, however, `traceroute` is a very useful tool for diagnosing problems such as the following:

- Where a packet stops. You try to reach a certain host, but because of bad routing information, it reaches a certain point on the Internet and doesn't go any farther. The output of `traceroute` shows you the last router reached by the packet.
- Where a packet slows down. Your connectivity to a certain site seems unusually slow. `Traceroute` indicates the time to receive a response from each router. The one that takes an inordinate amount of time to respond deserves special attention.

Other problems are possible, of course, but these are two common uses for `traceroute`. Command-line options in `traceroute` include setting the maximum number of routers to try, limiting the time to wait for each response, and indicating that packets cannot be fragmented, which can help you diagnose problems related to packet fragmentation between different types of networks (as discussed in Chapter 1).

Some administrators block `traceroute` packets at their routers, firewalls, and servers because they perceive them as a security threat. It's common for `traceroute` to stop displaying hops when the `traceroute` packet reaches the destination network.

Troubleshooting Network Connections

This may surprise you, but you have learned enough already about Linux networking to troubleshoot a variety of problems. Networking can become very complex, and you have not yet delved into many of the hardware issues that could arise (such as cabling problems). Table 2-2, however, lists some common networking problems that you can diagnose and troubleshoot with the tools covered so far. This list will be expanded with other troubleshooting tables as you learn more about different aspects of networking, such as DNS in Chapter 3.

Table 2-2 Basic networking troubleshooting review

Trouble	Things to Check
Networking doesn't appear to function at all	Use the `ifconfig` command to see whether networking is up and running; if not, try the network script in `/etc/rc.d/init.d`. (Remember to check things that seem obvious: Is the cable plugged into the Ethernet card? Is an Ethernet card installed? Is the cable plugged into the wall or a hub?)
Network script doesn't appear to work	Use the `lsmod` command to see whether any network modules are installed in the kernel. If not, use `modprobe` as described in this chapter or a graphical utility to install the right kernel module for your networking card.
Can't ping any other systems on the local network segment	Check the cables; check with `ifconfig` to see whether a valid IP address has been assigned; check the routing tables to see that a route is listed for the local network.
Can't ping any system on another segment within the organization	Check the cables at the hub or server room; check with `ifconfig` to see whether a valid IP address is assigned and how that address compares with those of the other network segment (is there a conflict?); does the routing table include a route for the other network that refers to an intermediate router if necessary?
Traffic to another segment seems very slow	Check the routing table to see whether the most direct route is defined; check whether another system has the same IP address assigned; review the output of the `ifconfig` command to see whether a large number of collisions are occurring—if so, the network may be overloaded; check `ifconfig` output to see whether a large number of errors are occurring—if so, the NIC may be defective; review whether fragmentation problems between the two segments could be slowing down traffic; use `traceroute` to see at what point the transmission slows down significantly.

Other Networking Protocols

In later chapters, you will learn a lot about many different protocols designed for different types of information sharing. But two lower-level protocols deserve mention in the same context as the discussion of TCP/IP and routing. These are IPX and AppleTalk.

IPX

The **Internetwork Packet Exchange (IPX)** protocol was designed by Novell Inc. in the early 1980s for use in its NetWare server product. IPX was the dominant protocol on local area networks from about 1985 through the mid-1990s because NetWare was the dominant server operating system for businesses using personal computers. People ask why Novell chose to use its own proprietary protocol rather than use the standard TCP/IP protocol suite. IPX is based on the Xerox Network System (XNS) protocol because, in the early 1980s, it was not clear to anyone that TCP/IP would become so popular more than a decade later.

In the early to mid-1990s, Novell added TCP/IP support to NetWare and customers converted from IPX. TCP/IP is the protocol of the Internet and is perceived as a superior protocol. However, many networks still run IPX and you might have to integrate Linux into them.

If you work with NetWare, you can use Linux as a client to communicate with NetWare servers. You can also have Linux route IPX network traffic. For more information on setting up IPX on Linux, review the IPX HOWTO document at *www.tldp.org/HOWTO/IPX-HOWTO.html*.

Apple Networking

AppleTalk is the name of the networking protocol used by Macintosh computers. Anyone who has used the Macintosh might wonder about the complex maneuvers needed to set up a PC network (running Windows or Linux). Macs and AppleTalk provide simple plug-and-play networking. But that simplicity doesn't scale to something like the worldwide Internet, so AppleTalk is used on small local networks.

On Linux, you can install the **Netatalk** package to allow Macintosh computers to have a Linux server show up on the Mac desktop as an available shared resource. Another program called **`afpfs`** lets Linux users access Macintosh resources. However, `afpfs` is not currently being developed and is not supported by the latest Linux kernels.

To get started, carefully review the Netatalk HOWTO document and download the appropriate software as directed. See *www.anders.com/projects/netatalk/* for details.

The latest version of the Macintosh operating system, OS X, supports TCP/IP-based protocols and Microsoft SMB networking as well as AppleTalk. Mac users can choose how they want to network. They can access Linux servers running NFS, Samba, or Netatalk. If

you don't already have an AppleTalk network and your Macs support Ethernet, it's best to use NFS or Samba.

Mac users can also choose from a wide variety of applications. They can run native OS X applications, older applications written for Mac OS, X applications ported from Linux or UNIX, and most Windows applications via Apple's Virtual PC software. Mac OS X users have one of the most flexible and compatible desktop computers on the planet with a user interface more attractive than either Windows or Linux. Mac OS X on desktops and Linux servers in the back room are a very attractive combination.

Chapter Summary

- Modern Linux distributions normally load support for network interfaces automatically without you having to get involved.
- Linux networking devices are created directly in the Linux kernel when a kernel module supporting a type of networking is loaded.
- Many types of networking are supported in Linux, though the most widely used is Ethernet.
- You can change the name of interfaces using the `ip` program.
- The `modprobe` command is used to add a networking module to the Linux kernel. The currently loaded kernel modules can be listed using the `lsmod` command.
- Sometimes, installing a kernel module requires that you include parameters to help the module locate or operate the networking hardware.
- The `ifconfig` command sets up a networking interface in the Linux kernel or displays the current setup for all configured interfaces.
- The `route` command establishes entries in the kernel IP routing table or displays the current routing table entries.
- The `arp` command lets you view the hardware address entries in the system's ARP cache. These entries are used by IP to route a packet to its final destination correctly.
- A number of networking scripts are used to streamline the configuration of Linux networking, making it more flexible and robust. The main networking control script is `/etc/rc.d/init.d/network`, or a script in a similar location.
- Networking configuration parameters are stored in files within the `/etc/sysconfig/network-scripts` subdirectory. These files can be edited directly, but are usually modified using a configuration tool of some sort, such as Webmin.
- IP aliasing occurs when multiple IP addresses are assigned to the same physical network interface (such as a single Ethernet card).
- Fedora Core includes a powerful Network Configuration Tool. Other Linux distributions use a variety of graphical tools to configure networking.

- The Telnet utility lets you connect to a remote host as if you were sitting at that host, but Telnet is not secure or encrypted and must be used with caution.
- `Ping` is a utility that uses the ICMP `echo` command to check whether a remote host is accessible and alive. `Ping` is a widely used tool for testing network connections, though some systems on the Internet do not respond to `ping` because of previous attacks that used this utility.
- The `traceroute` command displays each of the intervening routers between your host and another host you want to contact. `Traceroute` provides information that helps troubleshoot problems connecting to a remote host or with slow connections.
- IPX is a useful protocol that originated with Novell's NetWare operating system. It is supported in Linux, though not widely used.
- AppleTalk is supported in Linux via the Netatalk package, which you can add to Linux so that a Macintosh computer can see and access Linux resources.

Key Terms

`/etc/sysconfig` — The directory in which networking configuration files and scripts are stored, specifically in the subdirectory `network-scripts` and the file `networking`.

`/lib/modules` — The directory in which all of the available kernel modules for Linux are stored.

`afpfs` — A software package that lets Linux users access Macintosh resources. This software is still available for download, but because it is not currently being developed, is not supported by the latest Linux kernels.

AppleTalk — The networking protocol used by Apple Macintosh computers. AppleTalk is supported in Linux via the Netatalk package.

`arp` — A command that displays or alters the contents of the ARP cache. Used mainly for troubleshooting network connectivity.

ARP cache — A list of IP address-to-hardware mappings maintained by the ARP protocol to assist in routing packets.

`eth0` — The device name in Linux for the first Ethernet card installed in a host.

frame relay — A technology used to provide dedicated high-speed Internet connectivity via telephone wires.

`ifconfig` — The interface configuration utility that configures a networking interface within the Linux kernel or that lists all currently configured network interfaces.

`ifdown` — The script used to shut down individual networking interfaces. Generally used by other scripts such as `/etc/rc.d/init.d/network` rather than directly by a user.

`ifup` — The script used to start individual networking interfaces. Generally used by other scripts such as `/etc/rc.d/init.d/network` rather than directly by a user.

`insmod` — The command used to install a module into a running Linux kernel. It can include parameters that provide additional information to the module being installed.

Integrated Services Digital Network (ISDN) — A special type of telephone service, widely used in Europe, that provides 128- or 144-kb/s bandwidth.

Internetwork Packet Exchange (IPX) — A protocol designed by Novell Inc. based on an older protocol called the Xerox Network System (XNS) protocol. IPX was the dominant protocol on local area networks for many years and is fully supported in Linux. It uses the network hardware address as a host ID, thus simplifying network configuration.

`ip` — A program that allows you to change interface names.

IP aliasing — A networking feature that allows a single physical interface to have more than one IP address assigned to it.

`ip link` — A utility that lists currently configured network interfaces.

ISA bus — The Industry Standard Architecture bus that was developed by IBM in the early 1980s and included in early personal computers. It is also called the AT bus. The ISA bus is obsolete.

`iwconfig` — The interface configuration utility specific to wireless interfaces.

`lo` — The device name of the loopback network device.

`lsmod` — The command used to list the modules that are loaded in your kernel at that moment.

metric — A value assigned to a network interface to guide dynamic routing decisions about when to use that interface. A higher metric means an interface is less likely to be used if other interfaces are also available.

`modprobe` — The command to load a kernel module while automatically checking for and loading dependent modules.

`neat` — The command-line invocation of the Network Configuration Tool.

Netatalk — A software package for Linux that allows Macintosh computers to recognize Linux (have it show up on the Mac desktop as an available shared resource).

Network Configuration Tool — A graphical interface provided in Fedora Core that is used to manage network interfaces. It can be started from the command line using `neat`.

Parallel Line Internet Protocol (PLIP) — A protocol that relies on a parallel port to transmit network data, often to connect two computers in a simple, inexpensive network.

`ping` — The utility used to send ICMP echo packets for network testing.

Point-to-Point Protocol (PPP) — A protocol that allows a host to tie directly to a single computer, making a network of two systems. PPP is used most often with a modem providing the underlying physical connection to the second computer.

`proc` filesystem — A virtual filesystem that allows you to view and modify kernel settings.

`route` — A command that displays or configures the routing table within the Linux kernel.

Serial Line Internet Protocol (SLIP) — A protocol that relies on a serial port as the underlying physical connection used to transmit network data, usually over a modem to an ISP.

T-1 — A high-speed transmission format (1.544 Mb/s) available through your local telephone company, usually for a few hundred dollars per month.

T-3 — A high-speed transmission format (45 Mb/s) available through your local telephone company, usually for several thousand dollars per month.

Telnet — A terminal emulator program that allows a user to log on to a remote computer as if sitting at that computer's keyboard.

`traceroute` — A command used to list each router (each hop) that a packet passes through between a source host and a destination host.

VT100 — The most widely supported dumb terminal standard.

Webmin — A browser-based utility that can manage many system functions, including network configuration.

Review Questions

1. A key advantage of IPX over IP is that:
   a. IPX does not require configuration of a host ID because it uses the network hardware address.
   b. Even IPv6 does not provide as large an address space as IPX.
   c. Linux has stronger support for IPX than for ICMP.
   d. IPX routing relies on RIP instead of just the `route` command.
2. Networking devices differ from other Linux devices in that:
   a. Networking devices use a different set of major and minor device numbers than devices that do not rely on kernel modules.
   b. Networking devices can transmit data much faster than other types of devices.
   c. Networking devices are not directly visible in the `/dev` subdirectory, but are created on the fly when a networking device module is loaded into the kernel.
   d. Networking devices can only be accessed via shell scripts.
3. `Modprobe` is preferable to `insmod` for loading modules because:
   a. `Modprobe` allows you to include module parameters on the command line and `insmod` does not.
   b. `Modprobe` loads any dependent modules automatically before the requested module.
   c. `Insmod` is not supported in all versions of Linux, but `modprobe` is.
   d. `Insmod` can interfere with operation of the `ifconfig` command on Token-Ring systems.
4. From the Hardware tab in Webmin, what is the next step to reconfigure the IP addresses of a network interface?
   a. Choose the Network Configuration icon.
   b. Choose the Network Interfaces icon.
   c. Select the hardware interface for which you want to enter new IP addresses.
   d. Select whether you want to modify the active interface parameters or the boot-up interface parameters.

5. Which of the following directories is most likely to contain kernel modules for networking devices?
   a. `/lib/modules/networking`
   b. `/etc/sysconfig/network-scripts`
   c. `/proc/sys/net`
   d. `/lib/modules/2.4.7-10/kernel/drivers/net`
6. When using Telnet, you might refer to VT100 because it is:
   a. a standard protocol designation used by Telnet
   b. the port used by default to connect to the Telnet server
   c. the most commonly used terminal emulation standard
   d. the speed at which terminal emulator connections are typically handled
7. In which circumstance would kernel parameters most likely be needed to make a NIC function correctly in Linux?
   a. You are using an older ISA NIC that has a special feature allowing automatic IRQ mapping in software.
   b. You are using a brand-new PCI NIC that is identical to an established model.
   c. The documentation with your new NIC explains that Windows-based software is provided to help you configure the card.
   d. You want to do IP aliases and IP forwarding on your network.
8. Explain the difference between the TX packets and TX bytes lines in the output of `ifconfig`.
9. Which is not a valid `ifconfig` command?
   a. `ifconfig eth0 up`
   b. `ifconfig eth0 192.168.100.1 netmask 255.255.255.0 broadcast 255.255.255.0`
   c. `ifconfig`
   d. `ifconfig eth0 if-up`
10. Which information field is not part of the output of the `route` command?
    a. the gateway to reach the specified network
    b. the MAC address of the interface used to transfer packets to the specified network
    c. the interface through which the specified network can be reached
    d. the network address to which each routing table entry applies

11. Which statement is true about the following command?

```
route add –net 192.168.20.0 netmask 255.255.255.0 gw
192.168.10.1 dev eth0
```

    a. It is valid and defines the router that can reach network 192.168.20.0.
    b. It is valid and defines a default gateway for all traffic not destined for network 192.168.20.0.
    c. It is invalid because it does not specify the host IP address of the source of the packets to be routed.
    d. It is invalid in format because it lacks needed dashes before command-line options.

12. The PLIP networking device refers to:
    a. Protocol Layer Internet Proxy activity
    b. ProxyARP Layered Internet Protocol transmissions
    c. Parallel Line Internet Protocol support
    d. multiple, parallel SLIP channels

13. Distinguish between RARP and InARP.

14. You would use the `arp` command to:
    a. send a hardware address to a remote host per an ARP protocol request
    b. turn on and off ARP functionality for your network
    c. collect host name-to-IP address mappings for each of the hosts on your local network
    d. view or modify the ARP cache containing hardware address-to-IP address mappings

15. Scripts such as `/etc/rc.d/init.d/network` and their corresponding data files are used to control networking because:
    a. Using `ifconfig` and `route` is too complicated for the average user.
    b. The `ifconfig` and `route` commands alone don't complete the network configuration.
    c. Using scripts allows flexibility and power in managing multiple interfaces, firewall settings, and peripheral requirements that involve many diverse parts of Linux.
    d. By relying on scripts, graphical utilities can create a more user-friendly environment that makes system administrators more efficient.

16. The correct device designation for an IP alias to the first Ethernet card is:
    a. `eth00`
    b. `eth0:0`
    c. `Ethernet II`
    d. `eth1:0`

17. In the Network Configuration Tool (`neat`), the Hardware tab includes:
    a. a list of all networking hardware for which a module is currently installed
    b. a list of all available networking modules in Fedora Core
    c. a list of parameters that are available for currently installed modules
    d. the IP addresses of each installed hardware interface
18. The loopback device is assigned the IP address:
    a. 255.255.255.0
    b. 127.0.0.1
    c. 127.0.0.0
    d. any IP address beginning with 192.168
19. IP forwarding is enabled by which command?
    a. `ifconfig eth0 ip_forward`
    b. `arp ip_forward`
    c. `route add default ip_forward –net 192.168.10.0 netmask 255.255.255.0`
    d. `echo 1 > /proc/sys/net/ipv4/ip_forward`
20. Which utility cannot be used to configure network interfaces?
    a. `ip`
    b. `ifconfig`
    c. `route`
    d. `ping`
21. Telnet is considered dangerous because:
    a. It provides access to networking stacks, to which only the root user should have access.
    b. It transmits data—including passwords—without encrypting them, so anyone on the network can see them by using special software.
    c. It causes an increased number of Ethernet collisions by ramping up network traffic with broadcast messages to the local network segment.
    d. It requires that a user provide a password to gain access to a remote system.
22. `Ping` is used to test networking connections by:
    a. trying to contact systems that are progressively farther from your host to see if any networking problems occur
    b. flooding the network with traffic to see if it has sufficient bandwidth
    c. testing whether remote servers are configured to respond to ICMP echo-request packets
    d. noting the route that packets take when responding that they are "alive"

23. `Traceroute` is a useful troubleshooting tool because:
    a. All data sent by `traceroute` is fully encrypted, so it can be used safely on open networks.
    b. It reports the ARP cache of each system it contacts.
    c. It reports each router that a packet passes through to a destination IP address, along with the time needed to reach that router.
    d. It uses UDP to report errors generated by the IP stack of intermittent routers that may be misconfigured and slow down packets as they traverse large networks.
24. If you cannot ping a host on a different segment of your local network, you probably wouldn't bother checking:
    a. the cable connections at the hub or in the server room
    b. whether `ifconfig` indicates that the host has a valid IP address that does not conflict with any other host on the network
    c. whether the routing tables of the host and the router include the proper entries to tie the segments together
    d. whether the `ping` command was subject to fragmentation by the router in trying to reach the other segment
25. Define what the Netatalk package does for a Linux system.

Hands-On Projects

In the following projects, you experiment with the networking configuration of your host. You must be aware that changing your IP address causes your networking to cease functioning correctly. It may also cause someone else's networking to cease functioning! Check with your instructor or system administrator before completing these projects. In addition, note that standard file locations are used when discussing networking files. Some Linux systems may place these files in slightly different subdirectories. You should be able to locate them by exploring the `/etc` subdirectory or asking someone who is familiar with your Linux distribution.

Project 2-1

In this project, you explore the Address Resolution Protocol on your host. To complete this project, you should have a network connection configured with at least one other host on the network. An Internet connection is used for the last part of the project, but is not strictly necessary.

1. From any Linux command line, enter this command:

```
arp -a
```

This displays the contents of the ARP cache on your Linux system. The result may be nothing—the cache is probably empty at this point if you have not done any networking in the last few minutes. Each entry is saved only for about two minutes.

2. If the ARP cache is not empty, note the contents carefully. See if you can identify a neighbor on the network (in your computer lab, for example) that is not listed in the output of the command. Obtain that neighbor's IP address.
3. Ping your neighbor using a command such as the following (substitute the neighbor's IP address for 192.168.100.45):

```
ping 192.168.100.45
```

You can press Ctrl+C to end the `ping` command.

4. Run the `arp` command a second time:

```
arp -a
```

Do you see your neighbor's system listed (as indicated by an IP address that you recognize)?

5. Ping an Internet site such as 155.99.1.2 or 192.20.4.70.
6. Run **`arp -a`** again. Why is no additional entry included for the new site that you pinged?

Project 2-2

For this project, you need a system running Linux with Internet access on which you can log on as `root`. You should open a command-line window to complete this project.

1. Obtain the IP address of a host (not your own) within your computer lab or your organization, but not on the Internet.
2. Execute the `traceroute` command using this IP address and note the times displayed for reaching each point on that route to that host.
3. Locate a business that is headquartered or located near you, such as a bank in your city or the local government Web site. (Don't choose your own school or organization. You may want to ask your instructor for a suggestion or use an Internet search site to help you find the Internet address for the entity you select.)
4. Use a `traceroute` command to see how packets reach that site from your host. If your system has DNS configured already, feel free to use a command such as this, rather than using only an IP address:

```
traceroute www.uva.edu
```

For some sites, you might see a number of asterisks instead of the times you expect to see. Review the man page for `traceroute` to learn what these indicate.

5. You are likely to see a long list of unusual names for the routers that the packets pass through to reach someone's site close to you. Can you identify where these routers are physically located by their names (that is, which city and state they are located in)?
6. Why might a packet travel so far to get across town? How could the routing be improved?

Project 2-3

For this project, you need a system running Linux with Internet access. You should open a command-line window to complete this project. After each step shown, make a note about what its successful completion indicates about your network. To complete the last few steps (as indicated), you need to have `root` access and your instructor's permission, as these steps might create a heavy load on your network.

1. Run the **`ifconfig`** command to view your networking devices. See that you have both `lo` and `eth0` devices configured.
2. Ping your loopback device using its IP address.
3. Ping your own IP address.
4. If you are working in a classroom or computer lab, ask your neighbor for her IP address, then ping her host.
5. If your instructor gives you permission to continue, make sure you are logged on as `root`.
6. Use the **`ping`** command with the **`-f`** option, for flood. This sends pings (ICMP echo-request packets) as quickly as possible. Each period that is printed on your screen after you execute this command indicates one dropped packet (because the host being pinged cannot respond quickly enough).
7. If possible, have your neighbor try to complete a networking task such as opening a browser window and viewing a graphically intensive page. The flood ping places a heavy load on the network and should be used with caution. This is why some system administrators block pings, and why you will want to learn in Chapter 11 how to do the same yourself.

Project 2-4

To complete this project, you must have a network interface on your computer. It is useful but not necessary to have other computers on the network. You must be logged on as `root` and open a command-line window to complete this project.

1. Using either the `ifconfig` or `ip` program, note the IP address and the broadcast address of your computer's network interface.
2. By default, Linux responds to pings. You can verify this by using the `ping` program to ping your interface. Enter the command **ping –c 1 *1.2.3.4***, but substitute your IP address for 1.2.3.4. The `-c 1` option limits the pings to one—normally it will ping continuously. You see output similar to this:

```
PING 1.2.3.4 (1.2.3.4): 56 data bytes
64 bytes from 1.2.3.4: icmp_seq-0 ttl=64 time=0.1 ms

--- 1.2.3.4 ping Statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
```

3. If there is another computer on your network that you (or a lab partner) can use, try pinging your computer's IP address. The other computer should display output similar to that in Step 2.
4. You can disable your computer from responding to pings without having to configure a firewall. Enter the following command:

```
echo "1">/proc/sys/net/ipv4/icmp_echo_ignore_all
```

5. Enter the command **ping –c 1 *1.2.3.4*** again. This time, your computer does not respond to the ping. The last line of the output will be this:

```
1 packets transmitted, 0 packets received, 100% packet loss
```

6. Enable pings with the following command. You should be able to ping your computer and have it respond.

```
echo "0">/proc/sys/net/ipv4/icmp_echo_ignore_all
```

7. By default, Linux responds to broadcast pings. If you specified the broadcast address in your command, all Linux computers on your network might respond. Enter **ping –c 1 *1.2.3.255***, substituting your computer's broadcast address for 1.2.3.255. Your computer will respond to the ping, but other computers on your network might as well. You'll see output similar to the following if another computer responds:

```
PING 1.2.3.4 (1.2.3.4): 56 data bytes
64 bytes from 1.2.3.4: icmp seq-0 ttl=64 time=0.1 ms
64 bytes from 1.2.3.7: icmp seq-0 ttl=64 time=0.1 ms

--- 1.2.3.4 ping Statistics ---
1 packets transmitted, 1 packets received, +1 duplicates, 0%
packet loss
```

8. You can disable your computer's ability to respond to broadcast pings without disabling its ability to respond to normal (specific) pings. Enter the command **echo "1">/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts**. Your computer will no longer respond to broadcast pings. Enter the command **ping –c 1 *1.2.3.255***, substituting your computer's broadcast address for 1.2.3.255. You won't get a response from your computer, but other computers on your network may respond.
9. You can still ping your computer with a specific ping. Enter the command **ping –c 1 *1.2.3.4***. Your computer will respond.
10. Of course, you can disable both broadcast and specific pings. Enter the command **echo "0">/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts** to enable your computer to once again respond to broadcast pings.

By default, Windows computers do not respond to broadcast pings. If you have Windows computers on your network, they will not respond when you ping the network's broadcast address.

Project 2-5

For this project, you need to be connected to an Ethernet and have the permission of your instructor and a static IP address assignment to use when creating an IP alias. You must be logged on as `root` to complete this project.

1. Run the **`ifconfig`** command to see that you have `eth0` configured.
2. Change to the directory: `/etc/sysconfig/network-scripts`:

   `cd /etc/sysconfig/network-scripts`
3. Copy the configuration file for the `eth0` interface as a basis for creating a new interface called `eth0:0`, which is an IP alias to the same physical Ethernet card.

   `cp ifcfg-eth0 ifcfg-eth0:0`
4. Open the file `ifcfg-eth0:0` in a text editor.
5. Change the `DEVICE` parameter from `eth0` to read **`eth0:0`** instead.
6. Change the `IPADDR` parameter to reflect the new IP address provided by your instructor. Remember, if you simply choose a random IP address, you might cause networking problems for others in your lab or on the Internet.
7. Restart networking by running the network script with `start` and `stop` (or `restart` on some Linux distributions):

   `/etc/rc.d/init.d/network stop`
   `/etc/rc.d/init.d/network start`
8. Run the **`ifconfig`** command. Notice that you now have a new interface, `eth0:0`.

9. Ping the IP address of `eth0`. Ping the IP address of `eth0:0`. In later chapters, you will learn how you can use an IP alias to make management of multiple Web sites easier.
10. Return to the `/etc/sysconfig/network-scripts` directory if you have left it:

    ```
    cd /etc/sysconfig/network-scripts
    ```

11. Delete the file you created, confirming the deletion if prompted:

    ```
    rm ifcfg-eth0:0
    ```

12. Restart networking a second time:

    ```
    /etc/rc.d/init.d/network stop
    /etc/rc.d/init.d/network start
    ```

13. Run the **ifconfig** command to verify that you once again have only two networking devices: `lo` and `eth0`.

Case Projects

Case Project 2-1

You have been hired as a consultant by the law firm of Snow, Sleet, and Hale, based in Fairbanks, Alaska. The firm has been in business for many years, but is just now realizing the need to upgrade its information technology infrastructure. The firm consists of three offices, two in Fairbanks and one in Juneau. The Fairbanks headquarters is the largest office, with 40 attorneys plus a support staff of paralegals, secretaries, librarians, and others. They are divided into two practice groups, one for environmental work and another for energy work relating to oil, natural gas, hydrothermal, and hydroelectric. The work of the two groups doesn't intermingle much, though the attorneys from the two groups occasionally have a common client. All staff members share e-mail, of course, and occasionally need to access the same files. The managing partner tells you that she isn't ready to provide Internet access to the company as a whole. (A few attorneys have a private modem connection for legal research, but you needn't be concerned about that yet.) The second Fairbanks office is located about two miles away from headquarters. It consists of 10 lawyers who also do energy work, almost exclusively for a large company that occupies the same building. The Juneau office is relatively new and focuses on government-related work at the state capitol. It consists of seven lawyers and a few support staff.

Based on the volume of network traffic you anticipate over the next four years or so, you have already decided to use standard 100-Mb/s Ethernet cards in each computer, with new CAT6 cabling throughout the office. You have also planned for a high-speed connection among all three offices (don't worry about the specifics of the connection at this point).

1. Your immediate question is how to set up the three offices to make the best use of the network bandwidth so that everyone has fast access to the information they use most, with perhaps slower access to information they need less often. Diagram the

network segments as you would arrange them, noting your reasons for the arrangement. Within the parameters described previously, you can assume a physical arrangement of personnel within each office so that it suits your plan. Try to plan for the placement of hubs and routers, as well as how segments are arranged. Consider also these questions: What questions would you ask the managing partner about future growth plans before solidifying your network arrangement? What contingency plans could you make based on her responses? This scenario leaves many questions unanswered that will be addressed in future chapters. What additional questions come to mind that you may not have adequate information to answer yet?

2. Snow, Sleet, and Hale has merged with another firm: Sand & Son, located in Amarillo, Texas. Sand & Son is an energy law firm of 10 attorneys serving mainly oil companies. The two firms have several common clients and feel the merger will strengthen their position among clients seeking energy law specialists. Sand & Son has a fairly new office network for its 10 lawyers and staff. It runs on Token-Ring at 4 Mb/s. They have no Internet access. Assuming that you set up a high-speed network connection between Texas and Alaska, how will you incorporate Sand & Son into the existing corporate network? How will the need to integrate this office affect existing network topology? Will you need additional routers? Will you need to reconfigure anyone's IP addresses? What about possible packet fragmentation and slow data transfers between offices? What other problems might the Token-Ring network pose?

3. One year later, the managing partner accepts your recommendation. The entire firm must have Internet access to work effectively for its clients. Make some simple assumptions about the costs of different types of connections such as a modem or a dedicated circuit, then consider questions such as these: How can you add Internet access to your network design? How many connection points do you anticipate? What new problems arise because of how you set up the segments and routing? Would you have been better off setting things up differently if you had known that this new requirement would come up, or would it have made things too inefficient in the intervening year?