Chapter 7

Configuring Network Connections

- ✓ **Objective 1.5: Given a scenario, use the appropriate networking tools or configuration files**
- ✓ **Objective 4.2: Given a scenario, analyze and troubleshoot network resource issues**

These days it's almost a necessity to have your Linux system connected to some type of network. Whether it's because of the need to share files and printers on a local network or the need to connect to the Internet to download updates and security patches, most Linux systems have some type of network connection.

This chapter looks at how to configure your Linux system to connect to a network as well as how to troubleshoot network connections if things go wrong. There are a few different methods for configuring network settings in Linux, and you'll need to know them all for the Linux+ exam. First, we'll cover the common locations for the configuration files in Linux distributions. Next, we'll examine the different tools you have at your disposal that help make configuring the network settings easier. After that, the chapter discusses some simple network troubleshooting techniques you can use to help find the problem if anything goes wrong.

Configuring Network Features

There are five main pieces of information you need to configure in your Linux system to interact on a network:

- The host address
- The network subnet address
- The default router (sometimes called gateway)
- The system hostname
- A DNS server address for resolving hostnames

There are three different ways to configure this information in Linux systems:

- Manually editing network configuration files
- Using a graphical tool included with your Linux distribution
- Using command-line tools

The following sections walk through each of these methods.

Network Configuration Files

Every Linux distribution uses network configuration files to define the network settings required to communicate on the network. However, there's not a single standard configuration file that all distributions use.

Instead, different distributions use different configuration files to define the network settings. Table 7.1 shows the most common network configuration files that you'll run into.

TABLE 7.1 Linux network configuration files

Distribution	Network Configuration Location
Debian based	`/etc/network/interfaces` file
Red Hat based	`/etc/sysconfig/network-scripts` directory
openSUSE	`/etc/sysconfig/network` file

While each of the Linux distributions uses a different method of defining the network settings, they all have similar features. Most configuration files define each of the required network settings as separate values in the configuration file. Listing 7.1 shows an example from a Debian-based Linux system.

Listing 7.1: Sample Debian network static configuration settings

```
auto eth0
iface eth0 inet static
   address 192.168.1.77
   netmask 255.255.255.0
   gateway 192.168.1.254
iface eth0 inet6 static
   address 2003:aef0::23d1::0a10:00a1
   netmask 64
   gateway 2003:aef0::23d1::0a10:0001
```

The example shown in Listing 7.1 assigns both an IP and an IPv6 address to the wired network interface designated as `eth0`.

Listing 7.2 shows how to define the IP network settings automatically using a DHCP server on the network.

Listing 7.2: Sample Debian network DHCP configuration settings

```
auto eth0
iface eth0 inet dhcp
iface eth0 inet6 dhcp
```

If you just want to assign an IPv6 *link local address*, which uniquely identifies the device on the local network, but not retrieve an IPv6 address from a DHCP server, replace the `inet6` line with this:

```
iface eth0 inet6 auto
```

The `auto` attribute tells Linux to assign the link local address, which allows the Linux system to communicate with any other IPv6 device on the local network but not a global address.

Since version 17.04, the Ubuntu distribution has deviated from the standard Debian method and utilizes the Netplan tool to manage network settings. Netplan uses simple YAML text files in the `/etc/netplan` folder to define the network settings for each network interface installed on the system. By default, Netplan just passes the network settings off to the Network Manager tool, so you don't need to worry about how the Netplan configuration files are set.

For Red Hat–based systems, you'll need to define the network settings in multiple files, one for each network interface. The format of each file is:

```
ifcfg-interface
```

where *`interface`* is the device name for the network adapter, such as `ifcfg-enp0s3`. Listing 7.3 shows an example from a Rocky Linux system.

Listing 7.3: Sample Rocky network interface configuration settings

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s3
UUID=c8752366-3e1e-47e3-8162-c0435ec6d451
```

```
DEVICE=enp0s3
ONBOOT=yes
IPV6_PRIVACY=no
```

This configuration indicates that the workstation is using the DHCP process to automatically retrieve network information from a network server. For static IP addresses, you can set the IP address, default gateway, and subnet mask in the configuration file.

Most Linux distributions use the `/etc/hostname` file to store the local hostname of the system; however, some use `/etc/HOSTNAME` instead. You will also need to define a DNS server so that the system can use DNS hostnames. Fortunately, this is a standard that all Linux systems follow and is handled in the `/etc/resolv.conf` configuration file:

```
domain mydomain.com
search mytest.com
nameserver 192.168.1.1
```

The `domain` entry defines the domain name assigned to the network. By default the system will append this domain name to any hostnames you specify. The `search` entry defines any additional domains used to search for hostnames. The `nameserver` entry is where you specify the DNS server assigned to your network. Some networks can have more than one DNS server; just add multiple `nameserver` entries in the file.

For systems using the systemd startup method, you can use the `hostnamectl` command to view or change the hostname information. Also, to help speed up connections to commonly used hosts, you can manually enter their hostnames and IP addresses into the `/etc/hosts` file on your Linux system. The `/etc/nsswitch.conf` file defines whether the Linux system checks this file before or after using DNS to look up the hostname.

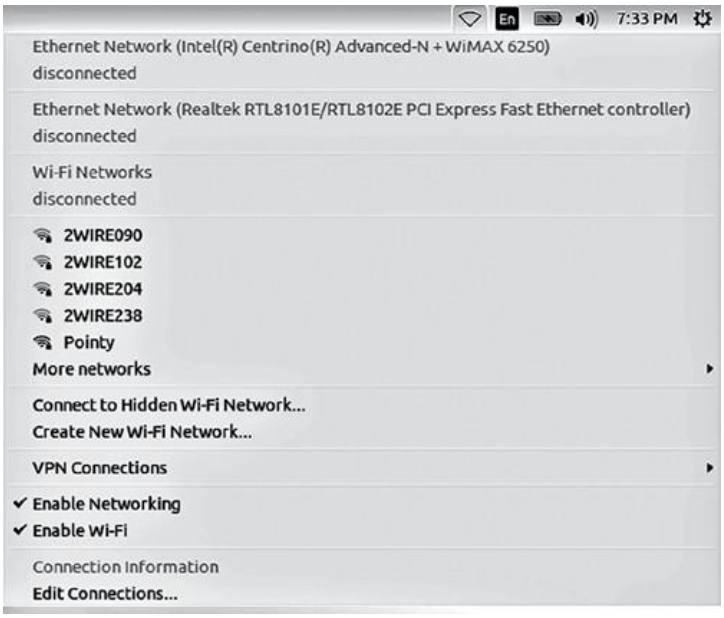
Graphical Tools

The *Network Manager* tool is a popular program used by many Linux distributions to provide a graphical interface for defining network connections. The Network Manager tool starts automatically at boot time and appears in the system tray area of the desktop as an icon.

If your system detects a wired network connection, the icon appears as a mini-network with blocks connected together. If your system detects a wireless network connection, the icon appears as an empty radio signal. When you click the icon, you'll see a list of the available wireless networks detected by the network card (as shown in Figure 7.1).

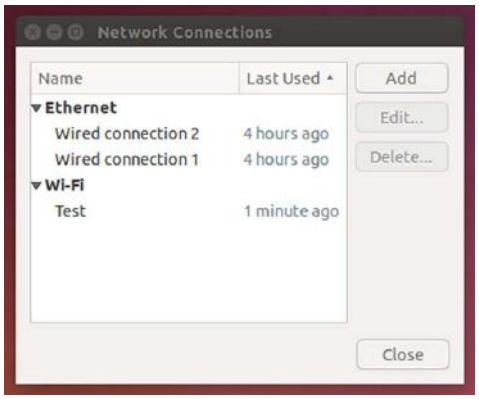
Click your access point to select it from the list. If your access point is encrypted, you'll be prompted to enter the password to gain access to the network.

FIGURE 7.1 Network Manager showing a wireless network connection



Once your system is connected to a wireless access point, the icon appears as a radio signal. Click the icon, and then select Edit Connections to edit the network connection settings for the system, shown in Figure 7.2.

FIGURE 7.2 The Network Connections window



You can select the network connection to configure (either wireless or wired) and then click the Edit button to change the current configuration.

The Network Manager tool allows you to specify all four of the network configuration values by using the manual configuration option or to set the configuration to use DHCP to determine the settings. The Network Manager tool automatically updates the appropriate network configuration files with the updated settings.

Command-Line Tools

If you're not working with a graphical desktop client environment, you'll need to use the Linux command-line tools to set the network configuration information. Quite a few command-line tools are at your disposal. The following sections cover the ones you're most likely to run into (and that you'll likely see on the Linux+ exam).

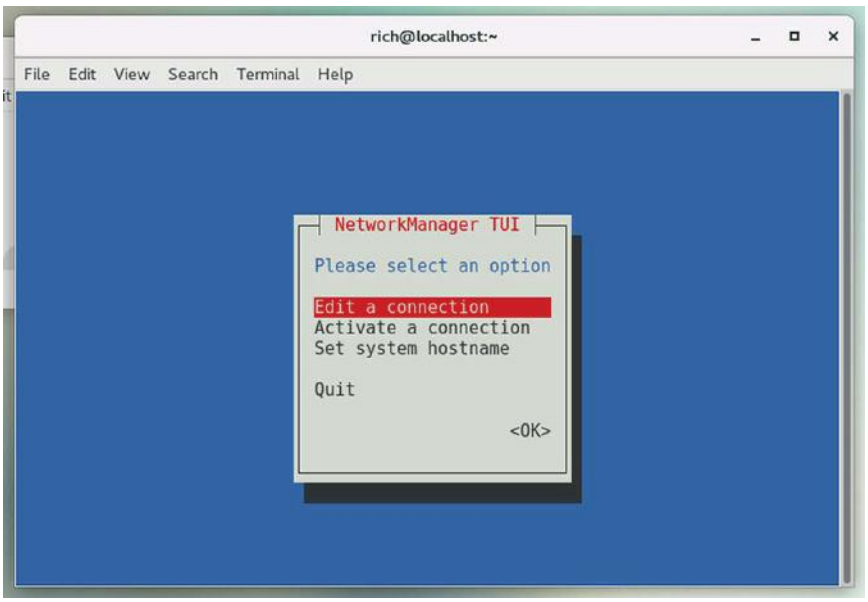
Network Manager Command-Line Tools

The Network Manager tool also provides two different types of command-line tools:

- `nmtui` provides a simple text-based menu tool.
- `nmcli` provides a text-only command-line tool.

Both tools help guide you through the process of setting the required network information for your Linux system. The *nmtui* tool displays a stripped-down version of the graphical tool where you can select a network interface and assign network properties to it, as shown in Figure 7.3.

FIGURE 7.3 The Network Manager `nmtui` command-line tool



The *nmcli* tool doesn't attempt to use any type of graphics capabilities; it just provides a command-line interface where you can view and change the network settings. By default, the command displays the current network devices and their settings, as shown in Listing 7.4.

Listing 7.4: The default output of the `nmcli` command

```
$ nmcli
enp0s3: connected to enp0s3
        "Intel 82540EM Gigabit Ethernet Controller (PRO/1000 MT Desktop
 Adapter)
        ethernet (e1000), 08:00:27:73:1C:6D, hw, mtu 1500
        ip4 default
        inet4 10.0.2.15/24
        route4 0.0.0.0/0
        route4 10.0.2.0/24
        inet6 fe80::5432:eddb:51ea:fb44/64
        route6 ff00::/8
        route6 fe80::/64
        route6 fe80::/64
```

The `nmcli` command uses command-line options to allow you to set the network settings:

```
# nmcli con add type ethernet con-name eth1 ifname enp0s3 ip4
10.0.2.10/24 gw4 192.168.1.254
```

This way, you can set all of the necessary network configuration features in a single `nmcli` command.

The iproute2 Utilities

The *iproute2 package* is a newer open source project that contains a set of command-line utilities for managing network connections. While the package contains several different programs, the `ip` program is the most used.

The `ip` command is the Swiss army knife of network programs, and it's becoming a popular method for defining network settings from the command line. It uses several command options to display the current network settings or define new network settings. Table 7.2 shows these commands.

TABLE 7.2 The `ip` utility command options

Parameter	Description
`address`	Display or set the IPv4 or IPv6 address on the device.
`addrlabel`	Define configuration labels.
`l2tp`	Tunnel Ethernet over IP.
`link`	Define a network device.

Parameter	Description
`maddress`	Define a multicast address for the system to listen to.
`monitor`	Watch for netlink messages.
`mroute`	Define an entry in the multicast routing cache.
`mrule`	Define a rule in the multicast routing policy database.
`neighbor`	Manage ARP or NDISC cache entries.
`netns`	Manage network namespaces.
`ntable`	Manage the neighbor cache operation.
`route`	Manage the routing table.
`rule`	Manage entries in the routing policy database.
`tcpmetrics`	Manage TCP metrics on the interface.
`token`	Manage tokenized interface identifiers.
`tunnel`	Tunnel over IP.
`tuntap`	Manage TUN/TAP devices.
`xfrm`	Manage IPSec policies for secure connections.

Each command option utilizes parameters to define what to do, such as display network settings or modify existing network settings. Listing 7.5 demonstrates how to display the current network settings using the `show` parameter.

Listing 7.5: The `ip address` output

```
$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 08:00:27:73:1c:6d brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global noprefixroute dynamic
enp0s3
       valid_lft 84411sec preferred_lft 84411sec
    inet6 fe80::5432:eddb:51ea:fb44/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
$
```

This example shows two network interfaces on the Linux system:

- `lo` is the local loopback interface.
- `enp0s3` is a wired network interface.

The *local loopback interface* is a special virtual network interface. Any local program can use it to communicate with other programs just as if they were across a network. That can simplify transferring data between programs.

The `enp0s3` network interface is the wired network connection for the Linux system. The `ip` command shows the IP address assigned to the interface (there's both an IP and an IPv6 link local address assigned), the netmask value, and some basic statistics about the packets on the interface.

If the output doesn't show a network address assigned to the interface, you can use the `ip` command to specify the host address and netmask values for the interface:

```
# ip address add 10.0.2.15/24 dev enp0s3
```

Then use the `ip` command with the `route` option to set the default router for the network interface:

```
# ip route add default via 192.168.1.254 dev enp0s3
```

Then make the network interface active by using the `link` option:

```
# ip link set enp0s3 up
```

With the single `ip` command, you can manage just about everything you need for your network connections.

The net-tools Legacy Tool

If you need to work on an older Linux distribution, the *net-tools package* may be all you have to work with. The net-tools package was the original method in Linux for managing individual aspects of the network configuration. There are four main command-line tools that you need to use:

- `ethtool` displays Ethernet settings for a network interface.
- `ifconfig` displays or sets the IP address and netmask values for a network interface.
- `iwconfig` sets the SSID and encryption key for a wireless interface.
- `route` sets the default router address.

The *ethtool command* allows you to peek inside the network interface card Ethernet settings and change any properties that you may need to communicate with a network device, such as a switch.

By default, the `ethtool` command displays the current configuration settings for the network interface, as shown in Listing 7.6.

Listing 7.6: Output from the `ethtool` command

```
$ ethtool enp0s3
Settings for enp0s3:
       Supported ports: [ TP ]
       Supported link modes:   10baseT/Half 10baseT/Full
                               100baseT/Half 100baseT/Full
                               1000baseT/Full
       Supported pause frame use: No
       Supports auto-negotiation: Yes
       Supported FEC modes: Not reported
       Advertised link modes:  10baseT/Half 10baseT/Full
                               100baseT/Half 100baseT/Full
                               1000baseT/Full
       Advertised pause frame use: No
       Advertised auto-negotiation: Yes
       Advertised FEC modes: Not reported
       Speed: 1000Mb/s
       Duplex: Full
       Port: Twisted Pair
       PHYAD: 0
       Transceiver: internal
       Auto-negotiation: on
       MDI-X: off (auto)
Cannot get wake-on-lan settings: Operation not permitted
       Current message level: 0x00000007 (7)
                              drv probe link
       Link detected: yes
$
```

You can change features such as speed, duplex, and whether or not the network interface attempts to auto-negotiate features with the switch.

The *ifconfig command* is a legacy command that allows you to set the network address and subnet mask for a network interface:

```
$ sudo ifconfig enp0s3 down 10.0.2.10 netmask 255.255.255.0
```

You can also use the `ifconfig` command to view the current statistics for a network interface, as shown in Listing 7.7.

Listing 7.7: The network interface stats from the `ifconfig` command

```
$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe55:dfbd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:55:df:bd  txqueuelen 1000  (Ethernet)
        RX packets 19067  bytes 28092762 (26.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6431  bytes 414153 (404.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:10:7a:b8  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

$
```

Using the `ifconfig` command, you can see the link status of a network interface, whether it is receiving or transmitting packets, and whether there were any dropped packets or collisions. This can be a handy network troubleshooting tool.

Each command option utilizes parameters to define what to do, such as display network settings or modify existing network settings. Listing 7.7 demonstrates how to display the current network settings by using the `ifconfig` command without specifying a command-line parameter.

With the net-tools package you must also set the default router using the separate `route` command:

```
# route add default gw 192.168.1.254
```

You can also use the `route` command by itself to view the current default router configured for the system:

```
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
default         192.168.1.254   0.0.0.0         UG    0      0   0   enp0s3
192.168.1.0     *               255.255.255.0   U     1      0   0   enp0s3
$
```

The default router defined for this Linux system is 192.168.1.254 and is available from the `enp0s3` network interface. The output also shows that to get to the 192.168.1.0 network, you don't need a gateway because that's the local network the Linux system is connected to.

If your network is connected to multiple networks via multiple routers, you can manually create the routing table in the system by using the `add` or `del` command-line option for the `route` command. The format for that is:

```
route [add] [del] target gw gateway
```

where *`target`* is the target host or network and *`gateway`* is the router address.

If you're working with a wireless network card, you must assign the wireless SSID and encryption key values using the `iwconfig` command:

```
# iwconfig wlp6s0 essid "MyNetwork" key s:mypassword
```

The `essid` parameter specifies the access point SSID name, and the `key` parameter specifies the encryption key required to connect to it. Notice that the encryption key is preceded by an `s:`. That allows you to specify the encryption key in ASCII text characters; otherwise you'll need to specify the key using hexadecimal values.

If you don't know the name of a local wireless connection, you can use the `iwlist` command to display all of the wireless signals your wireless card detects. Just specify the name of the wireless device, and use the `scan` option:

```
$ sudo iwlist wlp6s0 scan
```

Once you've set the wireless network card configuration, you can proceed to assign it an IP address and default route the same as you would a wired network card.

You can fine-tune networking parameters for a network interface using the `/etc/sysctl.conf` configuration file, or files stored in the `/etc/sysctl.d` or `/usr/lib/sysctl.d` directories. This file defines kernel parameters that the Linux system uses when interacting with the network interface. This has become a popular method to use for setting advanced security features, such as to disable responding to ICMP messages by setting the `icmp_echo_ignore_broadcasts` value to 1, or if your system has multiple network interface cards, to disable packet forwarding by setting the `ip_forward` value to 0.

Additional Network Features

If your network uses DHCP, you'll need to ensure that a proper DHCP client program is running on your Linux system. The DHCP client program communicates with the network DHCP server in the background and assigns the necessary IP address settings as directed by the DHCP server. There are three common DHCP programs available for Linux systems:

- `dhcpcd`
- `dhclient`
- `pump`

The dhcpcd program is becoming the most popular of the three, but you'll still see the other two used in some Linux distributions.

When you use your Linux system's software package manager utility to install the DHCP client program, it sets the program to automatically launch at boot time and handle the IP address configuration needed to interact on the network.

If you're working with a Linux server that acts as a DHCP server, the `/etc/dhcpd.conf` file contains the IP address settings that the server offers to DHCP clients. The file contains a section for each subnet the DHCP server services:

```
subnet 10.0.2.0 netmask 255.255.255.0 {
        option routers                  192.168.1.254;
        option subnet-mask              255.255.255.0;

        option domain-name              "mynetwork.com";
        option domain-name-servers       192.168.1.254;

        option time-offset              -18000;
# Eastern Standard Time

    range 10.0.2.1 10.0.2.100;
}
```

One final network configuration setting you may run into has to do with network interface *bonding*. Bonding allows you to aggregate multiple interfaces into one virtual network device.

You can then tell the Linux system how to treat the virtual network device using three different bonding types:

- *Load balancing*: Network traffic is shared between two or more network interfaces.
- *Aggregation*: Two or more network interfaces are combined to create one larger network pipe.

- *Active/passive*: One network interface is live while the other is used as a backup for fault tolerance.

There are seven different bonding modes you can choose from, as shown in Table 7.3.

TABLE 7.3 Network interface bonding modes

Mode	Name	Description
0	balance-rr	Provides load balancing and fault tolerance using interfaces in a round-robin approach
1	active-backup	Provides fault tolerance using one interface as the primary and the other as a backup
2	balance-xor	Provides load balancing and fault tolerance by transmitting on one interface and receiving on the second
3	broadcast	Transmits all packets on both interfaces
4	802.3ad	Aggregates the interfaces to create one connection combining the interface bandwidths
5	balance-tlb	Provides load balancing and fault tolerance based on the current transmit load on each interface
6	balance-alb	Provides load balancing and fault tolerance based on the current receive load on each interface

To initialize network interface bonding, you must first load the bonding module in the Linux kernel:

```
$ sudo modprobe bonding
```

This creates a `bond0` network interface, which you can then define using the `ip` utility:

```
$ sudo ip link add bond0 type bond mode 4
```

Once you've defined the bond type, you can add the appropriate network interfaces to the bond using the `ip` utility:

```
$ sudo ip link set eth0 master bond0
$ sudo ip link set eth1 master bond0
```

The Linux system will then treat the bond0 device as a single network interface using the load balancing or aggregation method you defined.

If you have multiple network interface cards on your Linux system and choose to connect them to separate networks, you can configure your Linux system to act as a bridge between the two networks. The `brctl` command allows you to control how the bridging behaves. To do this, though, you must set the `ip_forward` kernel parameter in the `/etc/sysctl.conf` file to 1 to enable bridging.

Command-Line Networking Tool

Linux provides a wealth of networking tools for connecting to remote hosts, but none is more versatile than the *netcat* program. The netcat program can act as either a network server or network client, sending and receiving data packets using either TCP or UDP. This section provides some examples of the versatility of the netcat program.

Depending on your Linux distribution, the netcat program may be available as either `netcat`, or just `nc`. The format of the command is simply:

```
nc host port
```

where *host* is the IP address or hostname of the remote server and *port* is the port number for the connection. By default netcat will attempt to establish a TCP connection with the remote server. To establish a UDP connection, add the `-u` option.

There are lots of different options available to customize the connection. Table 7.4 lists the netcat options.

TABLE 7.4 The `netcat` command options

Option	Description
`-4`	Use only IPv4 addresses
`-6`	Use only IPv6 addresses
`-C`	Use a carriage return/linefeed combination at the end of each line
`-D`	Enable socket debugging
`-d`	Do not read from STDIN
`-h`	Displays the netcat help document
`-i`	Specify a delay interval between text sent and received
`-k`	Continuing listing for an incoming connection after the current connection terminates

Option	Description
`-l`	Listing for an incoming connection instead of initializing a new connection
`-n`	Do not use DNS lookups for hostnames
`-p`	Specifies the port used for the connection
`-r`	Use a random source and/or destination port
`-S`	Enables the MD5 signature option
`-s`	Specify the IP address of the network interface used for sending packets
`-T`	Specify the IP Type of Service (ToS) used for the connection
`-t`	Reply to Telnet protocol options send from servers
`-U`	Uses Unix domain sockets instead of network sockets
`-u`	Use UDP instead of TCP
`-v`	Enable verbose mode to display more information
`-w`	Specify a timeout value for inactivity disconnections
`-X`	Use SOCK or HTTP proxy server protocols
`-x`	Specify the proxy server to use for the connection
`-z`	Scan for listening applications rather than attempting to connect

A great troubleshooting feature of `nc` is the ability to send HTTP requests directly to servers and see the HTTP response as well as the HTML code returned. Listing 7.8 shows an example of the output you would see.

Listing 7.8: Using netcat to retrieve HTTP data

```
$ printf "GET / HTTP/1.0\r\n\r\n" | nc richblum.com 80
HTTP/1.1 200 OK
Date: Mon, 04 Dec 2021 16:14:35 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close
```

```
Content-Type: text/html
,,,

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<link rel="stylesheet" type="text/css" href="main/mystyle.css" />
<link rel="stylesheet" media="print" type="text/css"
href="main/print.css" />
<title>Rich Blum's Blog</title>
</head>
,,,
$
```

The output in Listing 7.8 shows the HTTP options sent by the server, followed by the HTML code for the web page.

You can also create a simple chat dialogue between two systems from the command-line. On one system, just start `nc` in listen mode using the `-l` option:

```
$ nc -l 8000
```

Then on the other system, connect to that port on the remote system:

```
$ nc hostname 8000
```

Now any text you type in one side displays on the other side! To break the connection, just press `Ctrl+C` on either side of the connection.

Finally, another great use of the netcat program is as a quick way to transfer a file from one system to another. Just redirect the output of the listening host to a file:

```
$ nc -l 8000 > filename.txt
```

Then on the sending host, redirect the file as input to the sending `nc` command:

```
$ nc hostname 8000 < myfile.txt
```

When the file transfer completes, both sides of the connection will automatically terminate, and the new file will be available on the receiving host. This makes moving files between systems a breeze!

If you need to test secure SSL connections with a network server, the `s_client` package allows you to do that. It can utilize certificates to establish connections with secure servers.

Basic Network Troubleshooting

Once you have a Linux kernel installed, there are a few things you can do to check to make sure things are operating properly. The following sections walk through the commands you should know to monitor the network activity, including watching what processes are listening on the network and what connections are active from your system.

Sending Test Packets

One way to test network connectivity is to send test packets to known hosts. Linux provides the `ping` and `ping6` commands to do that. The `ping` and `ping6` commands send *Internet Control Message Protocol (ICMP)* packets to remote hosts using either the IP (*ping*) or IPv6 (*ping6*) protocol. ICMP packets work behind the scenes to track connectivity and provide control messages between systems. If the remote host supports ICMP, it will send a reply packet back when it receives a ping packet.

The basic format for the `ping` command is to specify the IP address of the remote host:

```
$ ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=63 time=14.6 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=63 time=3.82 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=63 time=2.05 ms
64 bytes from 10.0.2.2: icmp_seq=4 ttl=63 time=0.088 ms
64 bytes from 10.0.2.2: icmp_seq=5 ttl=63 time=3.54 ms
64 bytes from 10.0.2.2: icmp_seq=6 ttl=63 time=3.97 ms
64 bytes from 10.0.2.2: icmp_seq=7 ttl=63 time=0.040 ms
^C
--- 10.0.2.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6020ms
rtt min/avg/max/mdev = 0.040/4.030/14.696/4.620 ms
$
```

The `ping` command continues sending packets until you press Ctrl+C. You can also use the `-c` command-line option to specify a set number of packets to send and then stop.

For the `ping6` command, things get a little more complicated. If you're using an IPv6 link local address, you also need to tell the command which interface to send the packets out on:

```
$ ping6 -c 4 fe80::c418:2ed0:aead:cbce%enp0s3
PING fe80::c418:2ed0:aead:cbce%enp0s3(fe80::c418:2ed0:aead:cbce) 56 data
bytes
```

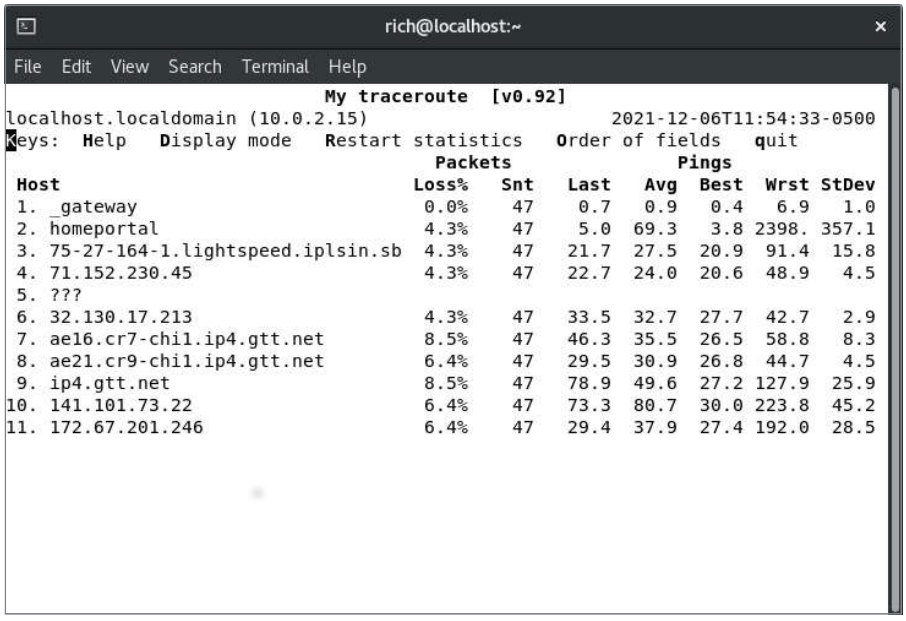
```
64 bytes from fe80::c418:2ed0:aead:cbce: icmp_seq=1 ttl=128 time=1.47 ms
64 bytes from fe80::c418:2ed0:aead:cbce: icmp_seq=2 ttl=128 time=0.478 ms
64 bytes from fe80::c418:2ed0:aead:cbce: icmp_seq=3 ttl=128 time=0.777 ms
64 bytes from fe80::c418:2ed0:aead:cbce: icmp_seq=4 ttl=128 time=0.659 ms

--- fe80::c418:2ed0:aead:cbce%enp0s3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.478/0.847/1.475/0.378 ms
$
```

The `%enp0s3` part tells the system to send the ping packets out the enp0s3 network interface for the link local address.

Yet another useful tool is the `traceroute` command. The `traceroute` command uses a feature of ICMP packets that restrict the number of network "hops" they can make. By manipulating that value in the packet, the `traceroute` command allows you to see the network routers used to get the packets from the client to the server.

Finally, the `mtr` program is a package that utilizes data retrieved from `ping` and `traceroute` commands to document network availability and latency in a real-time chart. Figure 7.4 shows the output of the `mtr` command tracing the connectivity to the `linux.org` server.

FIGURE 7.4 Using `mtr` to monitor network connectivity to a server



Unfortunately, these days many hosts don't support ICMP packets because they can be used to create a denial-of-service (DOS) attack against the host. Don't be surprised if you try to ping a remote host and don't get any responses.

Finding Host Information

Sometimes the problem isn't with network connectivity but with the DNS hostname system. You can test a hostname using the *host* command:

```
$ host www.linux.org
www.linux.org is an alias for linux.org.
linux.org has address 107.170.40.56
linux.org mail is handled by 20 mx.iqemail.net.
$
```

The `host` command queries the DNS server to determine the IP addresses assigned to the specified hostname. By default it returns all IP addresses associated with the hostname. Some hosts are supported by multiple servers in a load balancing configuration. The `host` command will show all of the IP addresses associated with those servers:

```
$ host www.yahoo.com
www.yahoo.com is an alias for atsv2-fp-shed.wg1.b.yahoo.com.
atsv2-fp-shed.wg1.b.yahoo.com has address 98.138.219.231
atsv2-fp-shed.wg1.b.yahoo.com has address 72.30.35.9
atsv2-fp-shed.wg1.b.yahoo.com has address 72.30.35.10
atsv2-fp-shed.wg1.b.yahoo.com has address 98.138.219.232
atsv2-fp-shed.wg1.b.yahoo.com has IPv6 address 2001:4998:58:1836::10
atsv2-fp-shed.wg1.b.yahoo.com has IPv6 address 2001:4998:58:1836::11
atsv2-fp-shed.wg1.b.yahoo.com has IPv6 address 2001:4998:44:41d::3
atsv2-fp-shed.wg1.b.yahoo.com has IPv6 address 2001:4998:44:41d::4
$
```

You can also specify an IP address for the `host` command, and it will attempt to find the hostname associated with it:

```
$ host 98.138.219.231
231.219.138.98.in-addr.arpa domain name pointer media-router-
fp1.prod1.media.vip.ne1.yahoo.com.
$
```

Notice, though, that often an IP address will resolve to a generic server hostname that hosts the website and not the website alias, as is the case here with the `www.linux.org` IP address.

Another great tool to use is the *dig* command. The `dig` command displays all of the DNS data records associated with a specific host or network. For example, you can look up the information for a specific hostname:

```
$ dig www.linux.org

; <<>> DiG 9.9.4-RedHat-9.9.4-18.el7_1.5 <<>> www.linux.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45314
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.linux.org.         IN    A

;; ANSWER SECTION:
www.linux.org.         14400   IN    CNAME  linux.org.
linux.org.         3600    IN    A    107.170.40.56

;; Query time: 75 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Sat Feb 06 17:44:29 EST 2016
;; MSG SIZE  rcvd: 72

$
```

Or you can look up DNS data records associated with a specific network service, such as a mail server:

```
$ dig linux.org MX

; <<>> DiG 9.9.5-3ubuntu0.5-Ubuntu <<>> linux.org MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16202
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;linux.org.            IN    MX

;; ANSWER SECTION:
linux.org.        3600    IN    MX    20  mx.iqemail.net.

;; Query time: 75 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Tue Feb 09 12:35:43 EST 2016
;; MSG SIZE  rcvd: 68

$
```

If you need to look up DNS information for multiple servers or domains, the *nslookup* command provides an interactive interface where you can enter commands:

```
$ nslookup
> www.google.com
Server:      192.168.1.254
Address:     192.168.1.254#53

Non-authoritative answer:
Name:   www.google.com
Address: 172.217.2.228
> www.wikipedia.org
Server:      192.168.1.254
Address:     192.168.1.254#53

Non-authoritative answer:
Name:   www.wikipedia.org
Address: 208.80.153.224
> exit

$
```

You can also dynamically specify the address of another DNS server to use for the name lookups, which is a handy way to determine if your default DNS server is at fault if a name resolution fails.

One final tool that can be useful is the `whois` command. The `whois` command attempts to connect to the centralized Internet domain registry at `http://whois.networksolutions.com` and retrieve information about who registered the requested domain name. Listing 7.9 shows a partial output from the `whois` command.

Listing 7.9: Partial output from the `whois` command

```
$ whois linux.com
   Domain Name: LINUX.COM
   Registry Domain ID: 4245540_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.1api.net
   Registrar URL: http://www.1api.net
   Updated Date: 2021-03-18T15:40:08Z
   Creation Date: 1994-06-02T04:00:00Z
   Registry Expiry Date: 2022-06-01T04:00:00Z
   Registrar: 1API GmbH
   Registrar IANA ID: 1387
   Registrar Abuse Contact Email: abuse@1api.net
   Registrar Abuse Contact Phone: +49.68949396850
   Domain Status: clientTransferProhibited
https://icann.org/epp#clientTransferProhibited
   Name Server: NS1.DNSIMPLE.COM
   Name Server: NS2.DNSIMPLE.COM
   Name Server: NS3.DNSIMPLE.COM
   Name Server: NS4.DNSIMPLE.COM
...
```

Theoretically the registry contains complete contact information for the owner of the domain, but these days due to privacy concerns that information is usually blocked. But there is usually a contact email address for the domain in case you need to report suspected abuse from the domain.

Advanced Network Troubleshooting

Besides the simple network tests shown in the previous section, Linux has some more advanced programs that can provide more detailed information about the network environment. Sometimes it helps to be able to see just what network connections are active on a Linux system. There are two ways to troubleshoot that issue: the `netstat` command and the `ss` command.

The *netstat* Command

The *netstat* command is part of the net-tools package and can provide a wealth of network information for you. By default, it lists all of the open network connections on the system:

```
# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
```

```
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    10825
 @/org/freedesktop/systemd1/notify
unix  2      [ ]         DGRAM                    10933
 /run/systemd/shutdownd
unix  6      [ ]         DGRAM                    6609
 /run/systemd/journal/socket
unix  25     [ ]         DGRAM                    6611     /dev/log
unix  3      [ ]         STREAM     CONNECTED     25693
unix  3      [ ]         STREAM     CONNECTED     20770
 /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     19556
unix  3      [ ]         STREAM     CONNECTED     19511
unix  2      [ ]         DGRAM                    24125
unix  3      [ ]         STREAM     CONNECTED     19535
unix  3      [ ]         STREAM     CONNECTED     18067
 /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     32358
unix  3      [ ]         STREAM     CONNECTED     24818
 /var/run/dbus/system_bus_socket
...
```

The `netstat` command produces lots of output because there are normally lots of programs that use network services on Linux systems. You can limit the output to just TCP or UDP connections by using the `-t` command-line option for TCP connections or `-u` for UDP connections:

```
$ netstat -t
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address      Foreign Address         State
tcp   1      0 10.0.2.15:58630         productsearch.ubu:https CLOSE_WAIT
tcp6  1      0 ip6-localhost:57782     ip6-localhost:ipp       CLOSE_WAIT
$
```

You can also get a list of what applications are listening on which network ports by using the `-l` option:

```
$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 ubuntu02:domain         *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
```

```
udp        0      0 *:ipp                   *:*
udp        0      0 *:mdns                  *:*
udp        0      0 *:36355                 *:*
udp        0      0 ubuntu02:domain         *:*
udp        0      0 *:bootpc                *:*
udp        0      0 *:12461                 *:*
udp6       0      0 [::]:64294              [::]:*
udp6       0      0 [::]:60259              [::]:*
udp6       0      0 [::]:mdns               [::]:*
...
```

As you can see, just a standard Linux workstation still has lots of things happening in the background, waiting for connections.

Yet another great feature of the `netstat` command is that the `-s` option displays statistics for the different types of packets the system has used on the network:

```
# netstat -s
Ip:
    240762 total packets received
    0 forwarded
    0 incoming packets discarded
    240747 incoming packets delivered
    206940 requests sent out
    32 dropped because of missing route
Icmp:
    57 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 12
        timeout in transit: 38
        echo replies: 7
    7 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        echo request: 7
IcmpMsg:
        InType0: 7
        InType3: 12
        InType11: 38
```

```
        OutType8: 7
Tcp:
    286 active connections openings
    0 passive connection openings
    0 failed connection attempts
    0 connection resets received
    0 connections established
    239933 segments received
    206091 segments send out
    0 segments retransmited
    0 bad segments received.
    0 resets sent
Udp:
    757 packets received
    0 packets to unknown port received.
    0 packet receive errors
    840 packets sent
    0 receive buffer errors
    0 send buffer errors
UdpLite:
TcpExt:
    219 TCP sockets finished time wait in fast timer
    15 delayed acks sent
    26 delayed acks further delayed because of locked socket
    Quick ack mode was activated 1 times
    229343 packet headers predicted
    289 acknowledgments not containing data payload received
    301 predicted acknowledgments
    TCPRcvCoalesce: 72755
IpExt:
    InNoRoutes: 2
    InMcastPkts: 13
    OutMcastPkts: 15
    InOctets: 410722578
    OutOctets: 8363083
    InMcastOctets: 2746
    OutMcastOctets: 2826
#
```

The `netstat` statistics output can give you a rough idea of how busy your Linux system is on the network or if there's a specific issue with one of the protocols installed.

Examining Sockets

The `netstat` tool provides a wealth of network information, but it can often be hard to determine just which program is listening on which open port. The *ss* command can come to your rescue for that.

A program connection to a port is called a *socket*. The `ss` command can link which system processes are using which network sockets that are active:

```
$ ss -anpt
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN     0      100    127.0.0.1:25                      *:*
LISTEN     0      128          *:111                      *:*
LISTEN     0      5      192.168.122.1:53                       *:*
LISTEN     0      128          *:22                       *:*
LISTEN     0      128    127.0.0.1:631                      *:*
LISTEN     0      100        ::1:25                      :::*
LISTEN     0      128         :::111                     :::*
LISTEN     0      128         :::22                      :::*
LISTEN     0      128        ::1:631                     :::*
ESTAB      0      0          ::1:22                    ::1:40490
ESTAB      0      0          ::1:40490                 ::1:22
users:(("ssh",pid=15176,fd=3))
$
```

The `-anpt` option displays both listening and established TCP connections and the process they're associated with. This output shows that the ssh port (port 22) has an established connection and is controlled by process ID 15176, the `ssh` program.

Monitoring the Network

Often when troubleshooting network applications it helps to see what's going on "behind the scenes" in the network. Knowing what TCP or UDP packets are being sent between the client and server can be crucial in determining what's going wrong. Fortunately, Linux has a few different tools that can help with that:

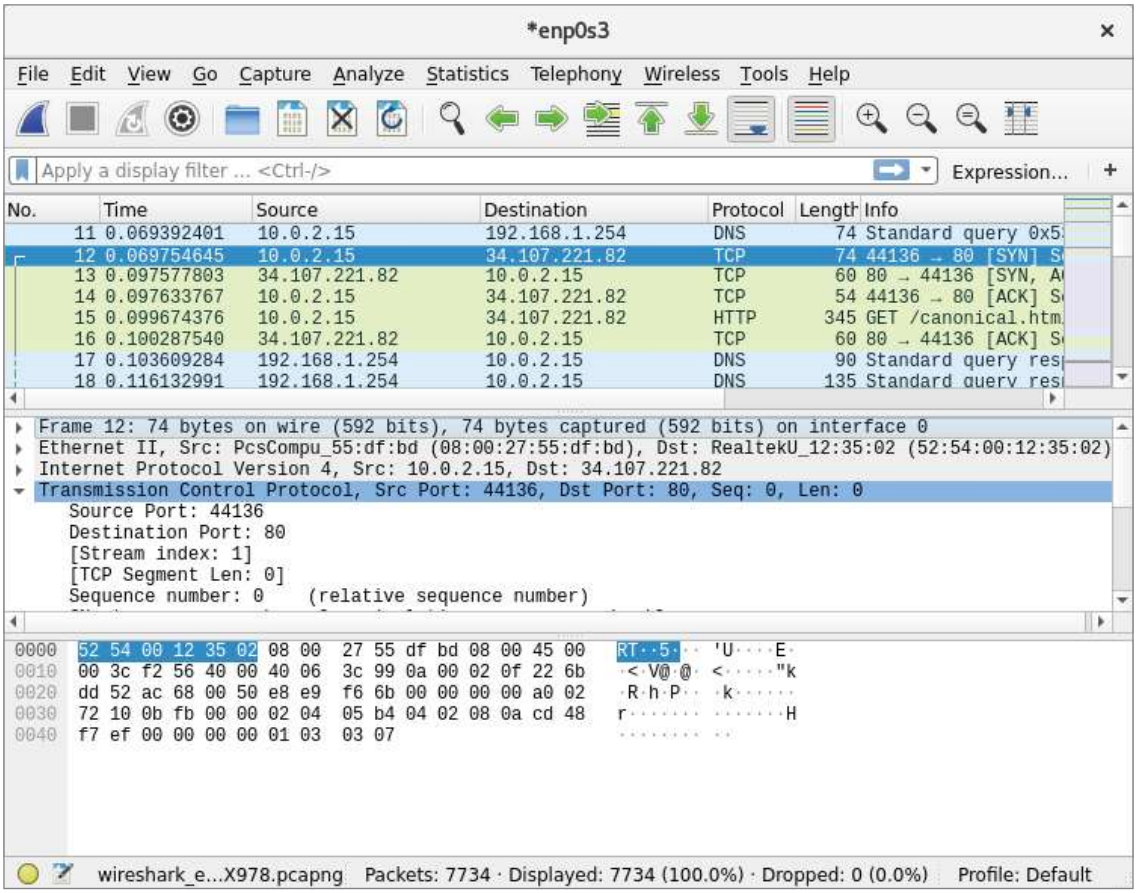
- `tcpdump`—the legacy command-line tool for watching network packets
- `wireshark`—a graphical tool for watching network packets and performing advanced network analysis
- `tshark`—the command-line version of Wireshark

The `tcpdump` program is a legacy tool that's been around for a long time, but it can still be useful if that's all you have to work with. It provides simple capturing of network data on the system and can do rudimentary decoding of the packets to break out the different data contained within the network packet.

The `tcpdump` program also provides basic filtering capabilities so that you can limit the capture to a single host, client, or even network session.

The `wireshark` package is an open source graphical tool for performing advanced network analysis of packets. Not only will `wireshark` capture and decode network packets, but it also provides color coding of traffic types and can display groups of packets based on applications. Figure 7.5 shows a sample `wireshark` display of simple network traffic.

FIGURE 7.5 The `wireshark` network analysis window



If you don't have a graphical desktop environment on your Linux system, you can still use the power of `wireshark` from the command line with `tshark`. The `tshark` program provides many of the same network analysis tools as `wireshark`, but in a more rudimentary display format on the command line.

To see all network traffic on a network interface, you must have administrator privileges on your system. That usually means either logging in with the root user account or using the `sudo` command from a normal account to gain root privileges.

Determining the Network Environment

This exercise will demonstrate how to quickly assess the network configuration and programs for your Linux system without you having to dig through lots of configuration files. To document your system network information, follow these steps (depending on your distribution you may need to first install the `netstat` and `iwlist` programs from the software repository):

1. Log in as root, or acquire root privileges by using `su` or by using `sudo` with each of the following commands.
2. Type **`ip address show`** to display the current network interfaces on your system. You will most likely see a loopback interface (named l0) and one or more network interfaces. Write down the IP (called inet) and IPv6 (called inet6) addresses assigned to each network interface along with the hardware address and the network mask address.
3. If your system has a wireless network card, type **`iwlist wlan0 scan`** to view the wireless access points in your area.
4. If your system has a wireless network card, type **`iwconfig`** to display the current wireless settings for your network interface.
5. Type **`route`** to display the routes defined on your system. Note the default gateway address assigned to your system. It should be on the same network as the IP address assigned to the system.
6. Type **`cat /etc/resolv.conf`** to display the DNS settings for your system.
7. Type **`netstat -l`** to display the programs listening for incoming network connections. The entries marked as `unix` are using the loopback address to communicate with other programs internally on your system.
8. Type **`ss -anpt`** to display the processes that have active network ports open on your system.

Summary

Connecting Linux systems to networks can be painless if you have the correct tools. To connect the Linux system, you'll need an IP address, a netmask address, a default router, a hostname, and a DNS server. If you don't care what IP address is assigned to your Linux system, you can obtain those values automatically using DHCP. However, if you are running a Linux server that requires a static IP address, you may need to configure these values manually.

Linux stores network connection information in configuration files. You can either manually modify the files to store the appropriate network information or use a graphical or

command-line tool to do that. The Network Manager tool is the most popular graphical tool used by Linux distributions. It allows you to configure both wired and wireless network settings from a graphical window. The Network Manager icon in the system tray area shows network connectivity as well as basic wireless information for wireless network cards.

If you must configure your network settings from the command line, you'll need a few different tools. For wireless connections, use the `iwconfig` command to set the wireless access point and SSID key. For both wireless and wired connections, use the `ifconfig` or `ip` command to set the IP address and netmask values for the interface. You may also need to use the `route` command to define the default router for the local network.

To use hostnames instead of IP addresses, you must define a DNS server for your network. You do that in the `/etc/resolv.conf` configuration file. You will also need to define the hostname for your Linux system in either the `/etc/hostname` or the `/etc/HOSTNAME` file.

Once your network configuration is complete, you may have to do some additional troubleshooting for network problems. The `ping` and `ping6` commands allow you to send ICMP packets to remote hosts to test basic connectivity. If you suspect issues with hostnames, you can use the `host` and `dig` commands to query the DNS server for hostnames.

For more advanced network troubleshooting, you can use the `netstat` and `ss` commands to display what applications are using which network ports on the system.

Exam Essentials

Describe the command-line utilities required to configure and manipulate Ethernet network interfaces. To set the network address on a network interface you can use the `nmtui`, `nmcli`, `ip`, or `ifconfig` commands. The `nmtui` and `nmcli` commands are available on systems that utilize the Network Manager tool for managing network interfaces. The `ip` command is from the iproute2 package, and the `ifconfig` command is from the legacy net-tools package. If you use the `ifconfig` command you'll also need to use the `route` command to set the default router (or gateway) for the network.

Explain how to configure basic access to a wireless network. Linux uses the `iwlist` command to list all wireless access points detected by the wireless network card. You can configure the settings required to connect to a specific wireless network using the `iwconfig` command. At a minimum, you'll need to configure the access point SSID value and most likely specify the encryption key value to connect to the access point.

Describe how to manipulate the routing table on a Linux system. For legacy systems use the `route` command to display the existing router table used by the Linux system. You can add a new route by using the `add` option or remove an existing route by using the `del` option. You can specify the default router (gateway) used by the network by adding the `default` keyword to the command. For systems that utilize the iproute2 package, you use the `ip route` command to display and manipulate the routing table.

Summarize the tools you would need to analyze the status of network devices. The `nmtui`, `nmcli`, `ifconfig` and `ip` commands display the current status of all network interfaces on the system. You can also use the `netstat` or `ss` command to display statistics for all listening network ports.

Describe how Linux initializes the network interfaces. Debian-based Linux systems use the `/etc/network/interfaces` file to configure the IP address, netmask, and default router. Red Hat–based Linux systems use files in the `/etc/sysconfig/network-scripts` folder. The `ifcfg-emp0s3` file contains the IP address and netmask settings, while the `network` file contains the default router settings. These files are examined at bootup to determine the network interface configuration. Newer versions of Ubuntu use the Netplan tool, which stores the network configuration in the `/etc/netplan` folder.

Explain how to test network connectivity. The `ping` and `ping6` commands allow you to send ICMP messages to remote hosts and display the response received. The `traceroute` command allows you to view the network path used to reach a specific remote host. The `mtr` command provides real-time connectivity and response statistics for a specific remote host.

Describe one graphical tool used to configure network settings in Linux. The Network Manager tool provides a graphical interface for changing settings on the network interfaces. The Network Manager appears as an icon in the desktop system tray area. If your Linux system uses a wireless network card, the icon appears as a radio signal, while for wired network connections it appears as a mini-network. When you click the icon, it shows the current network status, and for wireless interfaces, it shows a list of the access points detected. When you open the Network Manager interface, it allows you to either set static IP address information or configure the network to use a DHCP server to dynamically set the network configuration.

Review Questions

1. Which two commands can be used to set the IP address, subnet mask, and default router information on an interface using the command line?
   - **A.** `netstat`
   - **B.** `ping`
   - **C.** `nmtui`
   - **D.** `ip`
   - **E.** `route`
2. Which tool does newer versions of Ubuntu use to set network address information?
   - **A.** `netstat`
   - **B.** Netplan
   - **C.** `iwconfig`
   - **D.** `route`
   - **E.** `ifconfig`
3. Which command displays the duplex settings for an Ethernet card?
   - **A.** `ethtool`
   - **B.** `netstat`
   - **C.** `iwconfig`
   - **D.** `iwlist`
   - **E.** `route`
4. Which command displays what processes are using which ports on a Linux system?
   - **A.** `iwconfig`
   - **B.** `ip`
   - **C.** `ping`
   - **D.** `nmtui`
   - **E.** `ss`
5. If your Linux server doesn't have a graphical desktop installed, what two tools could you use to configure network settings on a wired network card from the command line?
   - **A.** `nmcli`
   - **B.** `iwconfig`
   - **C.** `ip`
   - **D.** `netstat`
   - **E.** `ping`

6. What network setting defines the network device that routes packets intended for hosts on remote networks?
   - **A.** Default router
   - **B.** Netmask
   - **C.** Hostname
   - **D.** IP address
   - **E.** DNS server

7. What device setting defines a host that maps a host name to an IP address?
   - **A.** Default router
   - **B.** Netmask
   - **C.** Hostname
   - **D.** IP address
   - **E.** DNS server

8. What is used to automatically assign an IP address to a client?
   - **A.** Default router
   - **B.** DHCP
   - **C.** ARP table
   - **D.** Netmask
   - **E.** `ifconfig`

9. What type of address is used so local applications can use network protocols to communicate with each other?
   - **A.** Dynamic address
   - **B.** Loopback address
   - **C.** Static address
   - **D.** Hostname
   - **E.** MAC address

10. Which command would you use to find the mail server for a domain?
    - **A.** `dig`
    - **B.** `netstat`
    - **C.** `ping6`
    - **D.** `host`
    - **E.** `ss`

11. What command would you use to find out what application was using a specific TCP port on the system?
    A. `ip`
    B. `ss`
    C. `host`
    D. `dig`
    E. `ifconfig`

12. What directory do Red Hat–based systems use to store network configuration files?
    A. `/etc/sysconfig/network-scripts`
    B. `/etc/network`
    C. `/etc/ifcfg-eth0`
    D. `/etc/ifconfig`
    E. `/etc/iwconfig`

13. Which configuration line sets a dynamic IP address for a Debian system?
    A. `iface eth0 inet static`
    B. `iface eth0 inet dhcp`
    C. `auto eth0`
    D. `iface eth0 inet6 auto`
    E. `BOOTPROTO=dynamic`

14. Which file contains a list of DNS servers the Linux system can use to resolve hostnames?
    A. `/etc/dhcpd.conf`
    B. `/etc/resolv.conf`
    C. `/etc/nsswitch.conf`
    D. `/etc/network/interfaces`
    E. `/etc/sysctl.conf`

15. Which `ifconfig` format correctly assigns an IP address and netmask to the eth0 interface?
    A. `ifconfig eth0 up 192.168.1.50 netmask 255.255.255.0`
    B. `ifconfig eth0 255.255.255.0 192.168.1.50`
    C. `ifconfig up 192.168.1.50 netmask 255.255.255.0`
    D. `ifconfig up`
    E. `ifconfig down`

16. What command displays all of the available wireless networks in your area?
    A. `iwlist`
    B. `iwconfig`
    C. `ifconfig`
    D. `ip`
    E. `arp`

17. What option sets the wireless access point name in the `iwconfig` command?
    A. `key`
    B. `netmask`
    C. `address`
    D. `essid`
    E. `channel`

18. What command can you use to both display and set the IP address, netmask, and default router values?
    A. `ifconfig`
    B. `iwconfig`
    C. `router`
    D. `ifup`
    E. `ip`

19. What tool allows you to send ICMP messages to a remote host to test network connectivity?
    A. `netstat`
    B. `ifconfig`
    C. `ping`
    D. `iwconfig`
    E. `ss`

20. You have a network application that fails to connect to a remote server. What command-line tool should you use to watch the network packets that leave your system to ensure that they use the correct network port?
    A. `nc`
    B. `tcpdump`
    C. `ping`
    D. `traceroute`
    E. `mtr`