6
CHAPTER

ANALYZING NETWORK DATA TRAFFIC

Chapter Outline

Objectives

- Review the TCP/IP suite of protocols
- Introduce the use of netstat for troubleshooting TCP and UDP connections
- Introduce the use of the Wireshark network protocol analyzer
- The use of SNMP for the gathering of the statistical information from network devices
- Introduce the use of NetFlow for acquiring IP traffic operational data
- Introduce filtering techniques for analyzing network data traffic

Key Terms

network forensics
Internet sockets
well-known ports
ICANN
registered ports
transport layer protocols
TCP
connection-oriented protocol
SYN
SYN ACK
ACK
UDP
netstat
ARP
arp –a
show arp
ARP Reply
Echo Request
SNMP (SNMPv1)
management information base (MIB)
snmp community [*community string*]
SNMPv2
SNMPv3
NetFlow
Jflow
Sflow
collector

INTRODUCTION

This chapter looks at the use of a network protocol analyzer to examine data packets. Section 6-1 introduces the concept of protocol analysis/forensics. This section reviews the TCP/IP suite of protocols, the TCP connection states, and the use of the **netstat** command. Section 6-2 introduces the use of the Wireshark protocol analyzer. This section introduces the techniques for using a protocol analyzer to examine how networking packets are exchanged in a TCP/IP network. Section 6-3 examines analyzing network data traffic. The first part of the chapter examines SNMP (Simple Network Management Protocol). The section concludes with a look at NetFlow, which is used for acquiring IP traffic operational data in order to provide network and security monitoring, traffic analysis, and IP accounting. This chapter concludes with Section 6-4, which looks at filtering the captured data packets. Data capture files can be quite large and it often requires that the network administrator search the capture files to find specific information. This section examines techniques to filter the captured data packets using Wireshark.

Network Forensics
The steps required for monitoring and analyzing computer network data traffic.

Internet Sockets
An endpoint across a computer network.

Well-Known Ports
Ports reserved by ICANN.

ICANN
Internet Corporation for Assigned Names and Numbers. ICANN is responsible for IP address space allocation, domain name system management, and root server system management functions.

Registered Ports
Ports registered with ICANN—ports 1024–49151.

6-1 PROTOCOL ANALYSIS/FORENSICS

This section examines the process of protocol analysis and **network forensics**. Network forensics is basically the steps required for monitoring and analyzing computer network data traffic. A solid foundation in the underlying protocols comprising the TCP/IP suite, namely TCP and UDP, is required to be able to analyze network traffic or to be able to gather information for network forensics or intrusion detection.

TCP/IP applications process requests from/to hosts via specific TCP or UDP ports. These ports are called **Internet sockets**, and each has a unique number. An Internet socket is an endpoint across a computer network. There are 65,536 possible TCP/UDP ports. Ports 1–1023 are called **well-known ports** or *reserved* ports. These ports are reserved by Internet Corporation for Assigned Names and Numbers (**ICANN**), and they represent some well-known network services. Ports 1024–49151 are called **registered ports** and are registered with ICANN. Ports 49152–65535 are called *dynamic* or *private* ports.

Examples of well-known ports include HTTP (TCP port 80) for web service and HTTPS (TCP port 443) for secure web service. Applications use these port numbers when communicating with another application as illustrated in Figure 6-1. Host B is passing to Host A data that is destined for TCP port 80 (HTTP). The HyperText Transfer Protocol (HTTP) is used for transferring non-secure web-based documents to a web browser, such as Internet Explorer or Mozilla Firefox. Host A receives the packet and passes the application up to the port 80 application. Table 6-1 lists some

popular applications and their port numbers for TCP/IP. This list includes FTP, SSH, SMTP, DNS, DHCP, HTTP, and HTTPS.

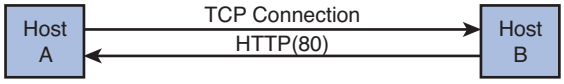

FIGURE 6-1 An example of two hosts connected for a TCP transmission

TABLE 6-1 **Common Applications and Their Port Numbers**

Transport Protocol	Port Number	Application	Description
TCP	20 (data port) 21 (command/ control port)	FTP	File Transfer Protocol
TCP	22	SSH	Secure Shell
TCP	23	Telnet	Virtual terminal connection
TCP	25	SMTP	Simple Mail Transfer Protocol
UDP	53	DNS	Domain name server
UDP	67, 68	DHCP (BOOTP-Client) (BOOTP-Server)	Dynamic Host Control Protocol
UDP	69	TFTP	Trivial File Transfer Protocol
TCP and UDP	80	HTTP	Hypertext Transfer Protocol
UDP	110	POP3	Post Office Protocol
TCP	143	IMAP	Internet Message Access Protocol
UDP	161	SNMP	Simple Network Management Protocol
TCP	443	HTTPS	Secure HTTP
TCP	445	SMB	Server message block
UDP	1701	L2TP	Layer 2 Tunneling Protocol
TCP	1720	H.323/Q.931	Voice over IP
TCP/UDP	1723	PPTP	Point-to-point Tunneling Protocol
TCP/UDP	3389	RDP	Remote Desktop Protocol

You can find a complete list of ports at http://www.iana.org/assignments/port-numbers.

Transport Layer Protocols
Define the type of connection established between hosts and how acknowledgements are sent.

TCP
Transport Control Protocol.

Connection-Oriented Protocol
Establishes a network connection, manages the delivery of data, and terminates the connection.

The **transport layer protocols** in TCP/IP are important in establishing a network connection, managing the delivery of data between a source and destination host, and terminating the data connection. There are two transport protocols within the TCP/IP transport layer: TCP and UDP. **TCP**, the Transport Control Protocol, is a **connection-oriented protocol**, which means it establishes the network connection, manages the data transfer, and terminates the connection. The TCP protocol establishes a set of rules or guidelines for establishing the connection. TCP verifies the delivery of the data packets through the network and includes support for error checking and recovering lost data. TCP then specifies a procedure for terminating the network connection. Figure 6-2 illustrates the TCP datagram. The first two fields are the Internet socket numbers mentioned earlier. The first 16 bits represent the source port and the second 16 bits represent the destination port.

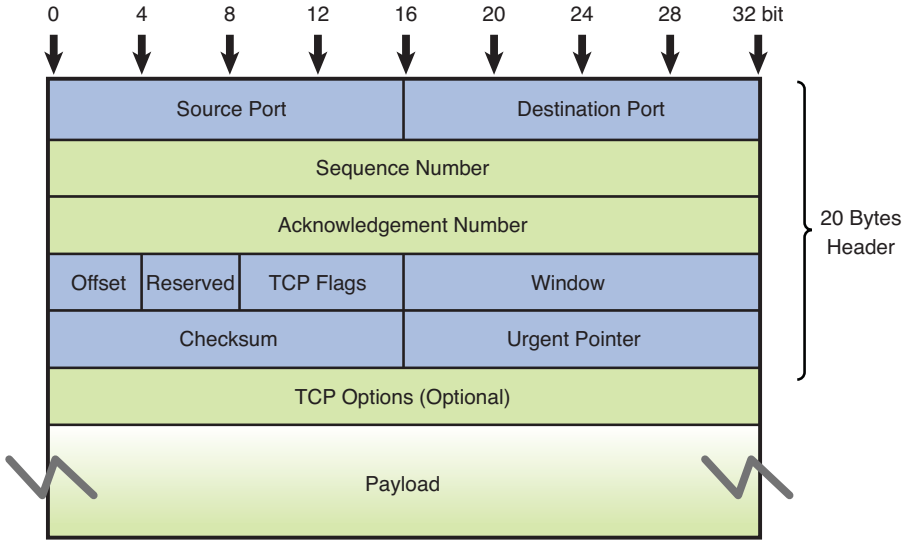

FIGURE 6-2 TCP datagram

SYN
Synchronizing packet.

SYN+ACK
Synchronizing Acknowledgment packet.

ACK
Acknowledgment packet.

Before any TCP connection is made, a TCP three-way handshake must happen to initiate the connection. A TCP three-way handshake is a unique sequence of three data packets exchanged at the beginning of a TCP connection between two hosts, as shown in Figure 6-3. This sequence is as follows:

1. The **SYN** (Synchronizing) packet
2. The **SYN+ACK** (Synchronizing Acknowledgment) packet
3. The **ACK** (Acknowledgment) packet

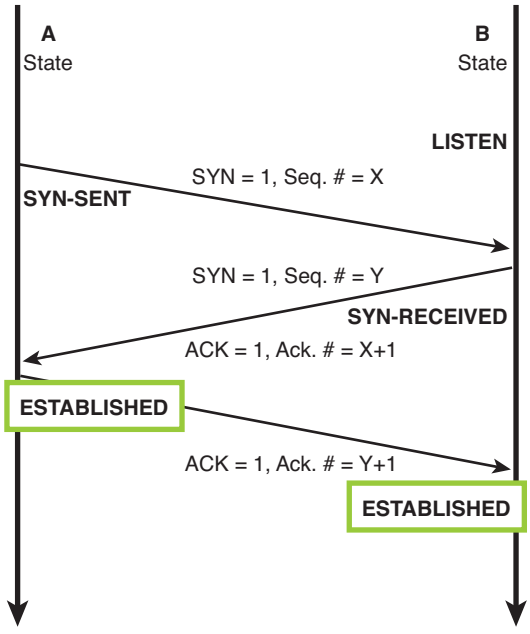

FIGURE 6-3 The TCP three-way handshake

The host initiating the connection will send a synchronizing packet (SYN). The SYN flag is set in the TCP flags field. In this example, Host A issues a SYN packet to initiate the TCP handshake. The SYN will have a sequence number (SEQ) associated with it. In the example shown in Figure 6-3, the sequence number is *x*. The sequence number is used to keep track of the data packets being transferred from Host A to Host B. The length of the packet being sent by Host A is 0 (LEN 0), which indicates that the packet contains no data. At this point, Host A changes its TCP state to SYN-SENT.

In packet 2, Host B replies with a SYN+ACK packet. Both SYN flag and ACK flag are set in the TCP flags field. The ACK is an acknowledgment that Host B received the packet from Host A. A number is attached to the ACK with a value of $(x + 1)$ that should be the sum of the SEQ# from packet 1 plus the length (LEN) of packet 1. Recall that the length of packet 1 is 0 (LEN 0), but packet 1 counts as one packet; therefore, Host B replies with an acknowledgment of packet 1 sequence number plus 1 $(x + 1)$. This acknowledgment notifies Host A that the packet (packet 1) was received. Packet 2 from Host B will also have a sequence number issued by Host B. In this packet, the sequence number has a value of *y*. This sequence number is used to keep track of packets transferred by Host B. When this happens, Host B changes its TCP state from LISTEN to SYN-RECEIVED.

In packet 3, Host A acknowledges the reception of Host B's packet. The ACK number is an increment of one higher than the SEQ# sent by Host B in packet 2 $(y + 1)$. Host A also sends an updated SEQ# that is one larger than the SEQ# Host A sent in packet 1 $(x + 1)$. Remember, Host A and Host B each have their own sequence numbers. Host A changes its TCP state to ESTABLISHED and, upon receiving the ACK packet from Host A, Host B changes its TCP state to ESTABLISHED as well.

This completes the three-packet handshake that establishes the TCP connection. This handshake appears at the beginning of all TCP data transfers.

The last part of the TCP connection is terminating the session for each host. A TCP connection can be terminated gracefully or abruptly due to loss of the network. Typically, TCP has a graceful way of terminating its connection. The first thing that happens is when a network application is about to close, it signals a host to send a FIN (finish) packet to the other connected host, as illustrated in Figure 6-4.

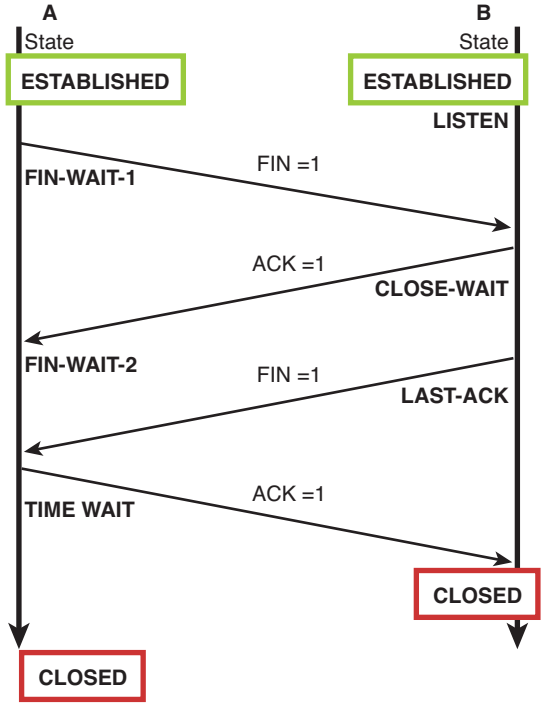

FIGURE 6-4 Terminating the TCP connection

Both Host A and B are in the TCP ESTABLISHED state. Host A sends a FIN packet to Host B indicating the data transmission is complete. This puts Host A in a FIN-WAIT-1 TCP state. Host B responds with an ACK packet acknowledging the reception of the FIN packet. It signals its application to close the connection and put itself in a CLOSE-WAIT state. Host A receives an ACK from Host B and for another FIN packet, and it changes its state to FIN-WAIT-2. Host B then sends Host A a FIN packet when its application is closed, indicating the connection is being terminated. At this point, Host B is in the LAST-ACK state. Host A replies with an ACK packet and changes its state to TIME-WAIT. Upon receiving an ACK, Host B's TCP state becomes CLOSED. Table 6-2 summarizes and briefly explains the TCP connection state.

TABLE 6-2 **TCP Connection State**

State	Description
LISTEN	The host is listening and ready to accept connections.
SYN-SENT	The first SYN sent to establish the connection indicates active open.
SYN-RECEIVED	The receiving host receives and acknowledges the SYN.
ESTABLISHED	The connection is fully established.
FIN-WAIT-1	The terminating host sends the FIN to terminate the connection indicates active close.
CLOSE-WAIT	The receiving host acknowledges the FIN.
FIN-WAIT-2	The terminating host receives the acknowledgment from the receiving host.
LAST_ACK	The receiving host sends its own FIN to signal the end and wait for the acknowledgment.
TIME-WAIT	The terminating host acknowledges the last FIN and waits for the connection to close.
CLOSED	Connection is closed.

The User Datagram Protocol (**UDP**) is a *connectionless* protocol. This means that UDP packets are transported over the network without a connection being established and without any acknowledgment that the data packets arrived at the destination. UDP is useful in applications such as videoconferencing and audio feeds, where such acknowledgments are not necessary.

UDP
User Datagram Protocol. A connectionless protocol that transports data packets to a connection being established and without any acknowledgment that the data packets arrived at the destination.

Figure 6-5 shows the UDP datagram. It is much simpler than TCP datagram with only 8 Bytes of header. Often times, UDP and TCP co-exist in the same application. Many of these applications use TCP to initiate the connection and then use UDP to deliver connectionless packets. No acknowledgments are sent back from the client. UDP does not have a procedure for terminating the data transfer; either the source stops delivery of the data packets or the client terminates the connection.

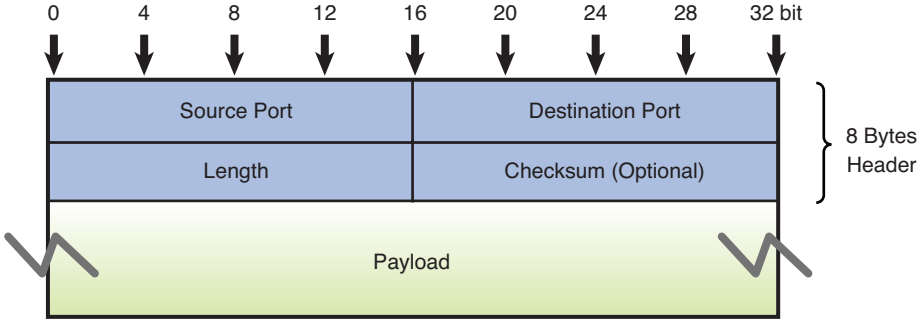

FIGURE 6-5 UPD datagram

Basic TCP/UDP Forensics

netstat
Used to display information such as network statistics, routing table, and TCP/UDP connections.

A useful tool for troubleshooting TCP and UDP connections is netstat. **netstat** (Network Statistics) is a command-line program readily available in multiple platforms (such as Windows, Mac, and Linux). The **netstat** utility has many options, and it can display information such as network statistics, routing tables, and TCP/UDP connections. Some options might vary depending on the operating system. To view all the network connections of a host, the command **netstat –an** is generally used. The option **–a** is for displaying all connections, including listening ports. The option **–n** is used to display the addresses and ports in numerical format. These options are standard on all OS platforms. Here is the example of the **netstat –an** output on Windows:

```
C:\Users\admin>netstat -an
Active Connections
  Proto  Local Address      Foreign Address        State
  TCP    0.0.0.0:22         0.0.0.0:0              LISTENING
  TCP    172.16.101.7:139   0.0.0.0:0              LISTENING
  TCP    172.16.101.7:445   0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1900     0.0.0.0:0              LISTENING
  TCP    172.16.101.7:49188 96.17.159.58:80        ESTABLISHED
  TCP    172.16.101.7:49189 173.194.79.95:443      ESTABLISHED
  TCP    172.16.101.7:49190 199.7.71.190:80        TIME_WAIT
  TCP    172.16.101.7:49191 74.125.224.48:443      ESTABLISHED
  TCP    172.16.101.7:49192 199.7.59.72:80         TIME_WAIT
  TCP    172.16.101.7:49193 74.125.224.48:443      ESTABLISHED
  TCP    172.16.101.7:49195 192.168.3.4:80         ESTABLISHED
  TCP    172.16.101.7:49196 192.168.3.4:80         ESTABLISHED
  TCP    172.16.101.7:49197 74.125.224.47:443      ESTABLISHED
  TCP    172.16.101.7:49198 192.168.3.4:80         ESTABLISHED
  TCP    172.16.101.7:49199 192.168.3.4:80         ESTABLISHED
  TCP    172.16.101.7:49200 192.168.3.4:80         ESTABLISHED
  TCP    172.16.101.7:49201 192.168.3.4:80         ESTABLISHED
  TCP    172.16.101.7:49202 74.125.224.36:80       CLOSE_WAIT
  UDP    172.16.101.7:139       *:*
  UDP    172.16.101.7:445       *:*
```

The preceding output shows all the TCP and UDP connections associated with the host (172.16.101.7). The Proto column is the connection protocol (TCP or UDP). The local address is the IP address followed by the port number of the local host connection immediately following the colon (:) . The Foreign Address is the remote host and its port, of which the connection belongs. The State is the TCP state of the connection as previously discussed in the TCP handshakes. Note that UDP does not have any state. When the connection is in the LISTENING state, the local host is listening on that port and ready to receive connections. In this state, the local address will display either 0.0.0.0, which means it is listening on all network interfaces (second NIC, modem, tunnel), or 127.0.0.1, which means it is only listening for connections from the local host itself, or its IP address (172.16.101.7), which means it is listening for connections from the network.

The output shows that the local host is listening on port 22 (SSH) on all of its interfaces. This is indicated by 0.0.0.0:22. It is also listening on TCP port 139 and 445 on the network. These ports are for Microsoft NetBIOS and Microsoft Directory Service, which are used for file and printer sharing. The last listening port is the TCP 1900 for only the local host itself to connect. For the port 1900, this means that there is some program running or listening on TCP port 1900, and it is only expecting the local host itself to use it. Some programs create a network socket port just for itself to connect. The rest of the TCP connections are made from the host to several IP addresses on TCP port 80 (http) and 443 (https). Judging from those well-known ports, these probably are the web servers on the Internet. Most of these TCP connections to port 80 and 443 are in the ESTABLISHED state. This means that these connections were successfully initiated after the TCP three-way handshake, and they are active. One TCP connection, 74.125.224.36, is in a CLOSE_WAIT state, which means the local host has just acknowledged the FIN packet to terminate the connection from the remote host. A couple of connections are in the TIME_WAIT state, which is the last state before local host closes the TCP connection.

netstat also can provide information about the executable program that created or is involved in the TCP connection. The option **-b** in Windows or the option **–p** in Linux is used to find out what programs are running and what network resources are being used. This is a powerful function that can be used to track down open network ports or exposed network security holes. Sometimes, people do not know what programs are running in the background. They also might not know what programs are communicating over the network, and some of these programs might be unwanted. The ability to associate a program to the network connection and port is extremely useful. The following is the output of the command **netstat -abn** on Windows:

```
C:\Users\admin>netstat -bn

Active Connections

  Proto  Local Address                  Foreign Address          State
  TCP    127.0.0.1:49318             127.0.0.1:49320           TIME_
WAIT
  TCP    128.123.101.13:49289   96.17.159.58:80           ESTABLISHED
[jucheck.exe]
  TCP    128.123.101.13:49315   74.125.224.128:80       ESTABLISHED
[chrome.exe]
  TCP    128.123.101.13:49316   74.125.224.143:80       ESTABLISHED
[chrome.exe]
  TCP    128.123.101.13:49317   128.123.128.102:443   CLOSE_WAIT
[asdm-launcher.exe]
  TCP    128.123.101.13:49335   128.123.128.102:443   CLOSE_WAIT
[javaw.exe]
```

ARP and ICMP

ARP and ICMP are two of the most common network layer protocols seen on any network. Along with TCP or UDP, these protocols show up all the time when scanning the network. ARP is used to discover the neighboring devices, and ICMP is used to report back information and errors. These protocols are examined next.

ARP
Address Resolution Protocol, used to map an IP address to its MAC address.

ARP The Address Resolution Protocol (**ARP**) resolves an IP address to a hardware address for final delivery of data packets to the destination. ARP issues a query in a network called an *ARP request*, asking which network interface has this IP address. The host assigned the IP address replies with an *ARP reply*, the protocol that contains the hardware address for the destination host. Figure 6-6 shows the ARP packet. When an ARP is issued, the requester fills out the source hardware address with its MAC address and the source protocol address with its IP address. It also puts the querying IP address in the destination protocol address field. The destination hardware address is left blank to be filled out by the destination host.

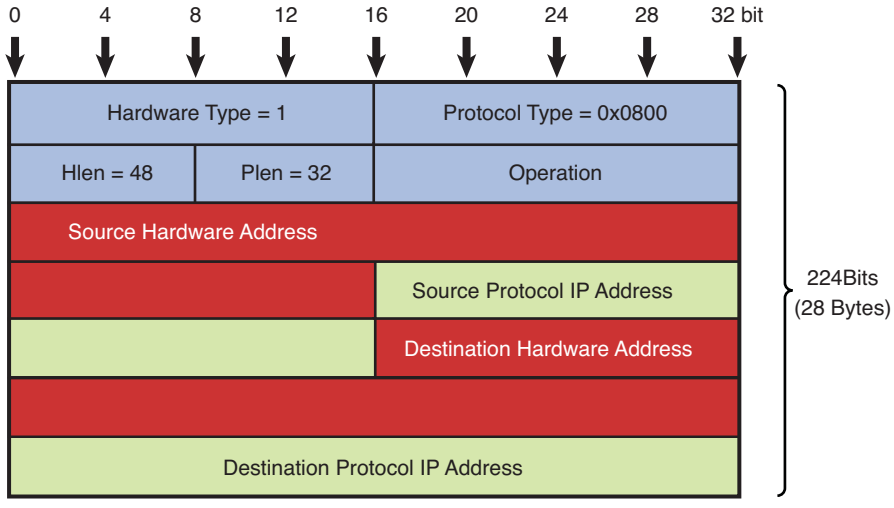

FIGURE 6-6 ARP packet

An ARP request is issued on the LAN as a broadcast, which means this message is being sent to all computers in the local-area network. Additionally, an ARP request is generated for every network device if the MAC address is not known. This is why ARP is one of the most seen protocols on a local network. The destination computer sends an ARP reply back to the source with its MAC address. The ARP request is sent back as a unicast packet. The ARP conversations are illustrated in the next section with the Wireshark protocol analyzer. In typical cases, the owner of the IP address replies to the message, but this is not always the case. Sometimes, another networking device, such as a router, can provide the MAC address information. In that case, the MAC address being returned is for the next networking device in the route to the destination.

To help reduce the amount of ARP broadcast traffic on the network, a network host or device is generally equipped with an ARP cache. When a destination host

receives an ARP request packet and when a source host receives an ARP reply packet, it updates its ARP cache with the IP address and the MAC address that it receives. This way, it does not need to generate an ARP broadcast every time it needs to communicate with its neighbors. The ARP cache can be viewed on most computers using the command **arp –a**. This command works on Windows, Mac, and Linux. The output of the **arp –a** command issued on a Linux host is as follows:

arp –a
The command used to view the ARP cache.

```
 [root@noc-server ~]# arp -a
? (10.20.101.146) at 0:15:e9:1f:2:c6 on en1 [ethernet]
server1.et477.local (10.10.101.1) at 0:25:b4:cf:57:80 on en0
[ethernet]
server2.et477.local(10.10.101.4) at 0:21:70:5b:f7:ea on en0 [ethernet]
```

The output may contain different information and may display in different formats depending on the operating system. The ARP output previously listed shows the IP addresses and their associated MAC addresses. Along with these, the hostname will be displayed if the IP address has a DNS hostname. The ARP cache can also be viewed on network devices, such as routers and switches. On Cisco routers, the command to display the ARP cache entries is **show ip arp** or **show arp**, as shown here:

show arp
The command to view the ARP cache on Cisco switches and routers.

```
et477-router#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.168.247.10           0   a4ba.db1d.190e  ARPA   Vlan100
Internet  192.168.246.95           3   0018.8b22.9141  ARPA   Vlan100
Internet  192.168.245.135          0   0023.ae8a.0831  ARPA   Vlan100
Internet  192.168.245.246        216   001e.7aa3.0980  ARPA   Vlan30
Internet  192.168.245.242        236   001e.7aa3.0980  ARPA   Vlan30
Internet  192.168.245.226         54   000f.8f5d.86e0  ARPA   Vlan10
Internet  192.168.245.227         96   000f.8f5d.86e0  ARPA   Vlan10
```

Similar to the ARP output from Linux, the ARP output from the router shows the IP address, MAC address, and the interface where an ARP entry is learned. Additional information associated with the **show arp** command is the age of each ARP entry. This information is displayed in minutes under the Age (min) column.

ICMP The Internet Control Message Protocol (ICMP) is used to control the flow of data in the network, to report errors, and to perform diagnostics. Figure 6-7 illustrates the ICMP packet. Depending on the type, each ICMP packet serves different functions. For example, a networking device, such as a router, can send an ICMP *source-quench* packet to a host that requests a slowdown in the data transfer. The ICMP packet then will have the type of 4.

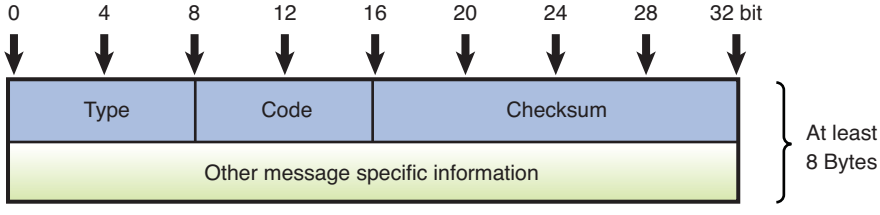

FIGURE 6-7 ICMP packet

An important troubleshooting tool within the ICMP protocol is **ping**, which was named after the "ping" sound the sonar makes. The **ping** command is used to verify connectivity with another host in the network. The destination host could be in a LAN, in a campus LAN, or on the Internet. The **ping** command uses a series of echo requests, and the networking device receiving the echo requests responds with a series of echo replies to test a network connection. The **ping** request is an ICMP type 8 and the **ping** reply is the ICMP type 0. These ICMP types are used to exchange message information. Some other types, like ICMP type 3 and ICMP type 11, are used to report error messages. ICMP type 3 packet reports destination unreachable message to the originating host. ICMP type 11 is the error message for time to live (TTL) exceeded, which indicates that the packet life has expired. Some of the common ICMP messages are as follows:

Type	Name	Reference
0	Echo Reply	[RFC792]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Selection	[RFC1256]
11	Time Exceeded	[RFC792]

ping is the most common tool used in troubleshooting connectivity on the network. Ping is available on every operating system and on most network devices by default. A general **ping** command format is **ping** *hostname/ip address*. There are options that can be used with **ping**, and these can vary depending on the OS. For example, there is an option to change the size of the default ICMP packet. This option is **–l** for Windows and is **–s** for the Mac and Linux machines. This option is useful when trying to simulate bigger size packets and test respond time. Another useful option is **–t**, which will allow a continuous ping to the target in a Windows environment. Typically, Windows sends only four ICMP packets whereas Linux and Mac machines will send continuous ICMP packets until it is stopped.

Ping is also available on routers and switches. On Cisco routers, one can get more use of a ping command by using the extended **ping** option. Typically, a router has multiple network interfaces. When a **ping** command is executed, it will use the interface where the packet exits as its source IP address. The extended **ping** on Cisco gives users the option to choose a different source IP address. This comes in handy when testing out connectivity from a different network. Extended **ping** provides more ways to perform advanced check of host reach ability and network connectivity. To do this, simply enter **ping** at the router prompt and press return. By doing this, step-by-step extended options will be provided to users. When the extended **ping** command is used, the source IP address can be changed to any IP address on the router. Also, the extended **ping** command works only at the privileged EXEC command line. The following is an example of how the extended **ping** is executed on a Cisco router:

```
et477-gate#ping
Protocol [ip]:
```

```
Target IP address: 128.123.2.19
Repeat count [5]: 200
Datagram size [100]: 200
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 128.123.247.44      (This features let's
you specify the source address)
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 200, 200-byte ICMP Echos to 128.123.2.19, timeout is 2 sec-
onds:
Packet sent with a source address of 128.123.247.44
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (200/200), round-trip min/avg/max = 1/1/4
ms
```

6-2 WIRESHARK PROTOCOL ANALYZER

This section introduces the techniques for using a protocol analyzer to examine how networking packets are exchanged in a TCP/IP network. By using a protocol analyzer, such as Wireshark, you will actually be able to develop a better understanding of the protocols being used and how the data packets are being transferred.

The Wireshark software includes many advanced features for packet capture and analysis. The capabilities of this software will help you gain a thorough understanding of packet transfers and networking protocols. In this chapter, you will gain an introductory understanding of the capabilities and techniques for using a sophisticated software protocol analyzer. The protocol analyzer has the capability to capture and decode data packets and allows the user to inspect the packet contents. This enables the user to investigate how information is being transferred in the network. Additionally, the information provided by the protocol analyzer enables the user to detect, identify, and correct network problems. In this section, you are guided through the steps of using the Wireshark Network Analyzer.

The following steps guide you through installing and using the Wireshark software. To download the latest version of the software, visit **www.WireShark.org**. At the WireShark.org home page, select **Download Wireshark**. Once completed, select your corresponding operating system. Click **Run** when the dialog box appears to initiate the download process. At the prompt of the setup wizard, select **Next** and

agree to the license agreement. Choose the components you want to install and click **Next** to continue. At the next screen, choose your program shortcuts and click **Nex**t to continue. Use the default directory paths specified in the setup menu. To complete setup, click **Install** to start the installation process. After installation, you are ready to begin using the software.

In the first exercise, the Wireshark software is used to examine the packets transferred in the process of pinging a computer. This exercise uses the IP and MAC addresses specified in Table 6-1 and the CD provided with this book includes the .cap files used in the following exercises:

1. In Windows, Click **Start** > ALL **Programs** > **WireShark** to start the analyzer program. The procedure for starting the WireShark Network Analyzer is the same for a MAC OS X operating in the dualboot mode with XP.
2. Once WireShark is open, click **File** > **Open**, select your CD-ROM drive, and select the *WireShark* file folder. Double-click the **Ch6-1.cap** file to open the file.

Once you open the Ch6-1.cap capture in WireShark, you should see the captured packets displayed on the detail view screen, as shown in Figure 6-8.

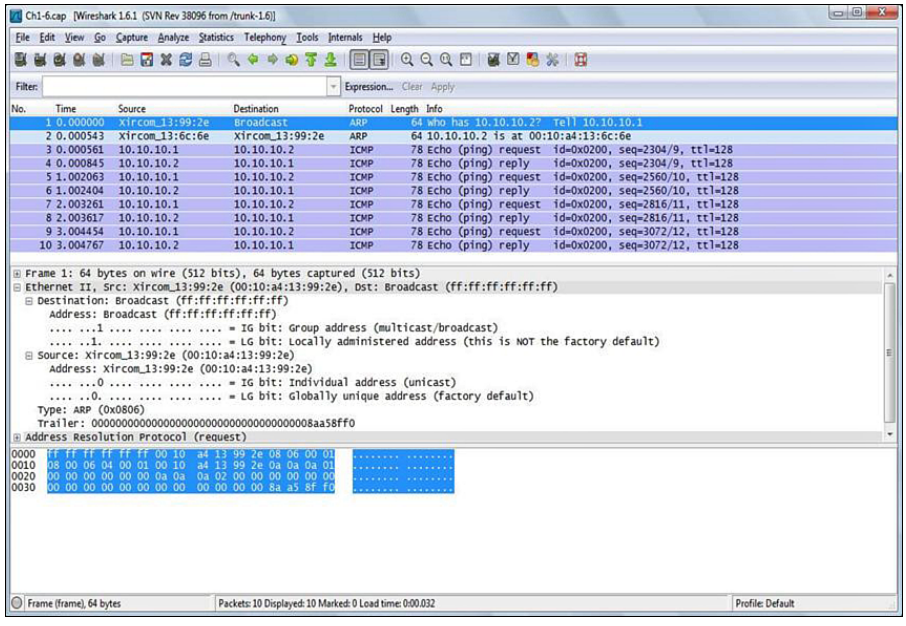

FIGURE 6-8 The captured packets showing the ping from computer 1 to computer 2

In this example, the information on the screen is showing the transfer of packets that occurs when one computer pings another. In this case, computer 1 pinged computer 2. The MAC and IP addresses are listed for your reference in Table 6-3.

TABLE 6-3 **The MAC and Assigned IP Addresses for Computer 1 and Computer 2**

Name (Hostname)	MAC Address	IP Address
Computer 1	00-10-A4-13-99-2E	10.10.10.1
Computer 2	00-10-A4-13-6C-6E	10.10.10.2

In this example, a **ping** command is issued from computer 1 to computer 2. The structure of the command issued by computer 1 at the command prompt is

```
ping 10.10.10.2
```

Shown in packet number 1 in Figure 6-8, computer 1 issues an **ARP** request on the LAN. The source of the packet is 00-10-A4-13-99-2E (computer 1). The destination address on the local-area network shown is BROADCAST, which means this message is being sent to all computers on the network. A query (Q) being asked is who has the IP address 1 0.10.10.2 (PA). In Figure 6-8, the wording to the right of ARP says, "Who has 10.10.10.2?"

The highlighted area (Number 2) in Figure 6-9 shows computer 2 replying with its MAC address back to computer 1. This is called an **ARP reply**, which is a protocol where the MAC address is returned. The source of the ARP reply is from 00-10-A4-13-6C-6E (computer 2), which is replying that the MAC address for 10.10.10.2 is 00-10-A4-13-6C-6E (HA). In this case, the owner of the IP address replied to the message.

ARP Reply
A network protocol where the MAC address is returned.

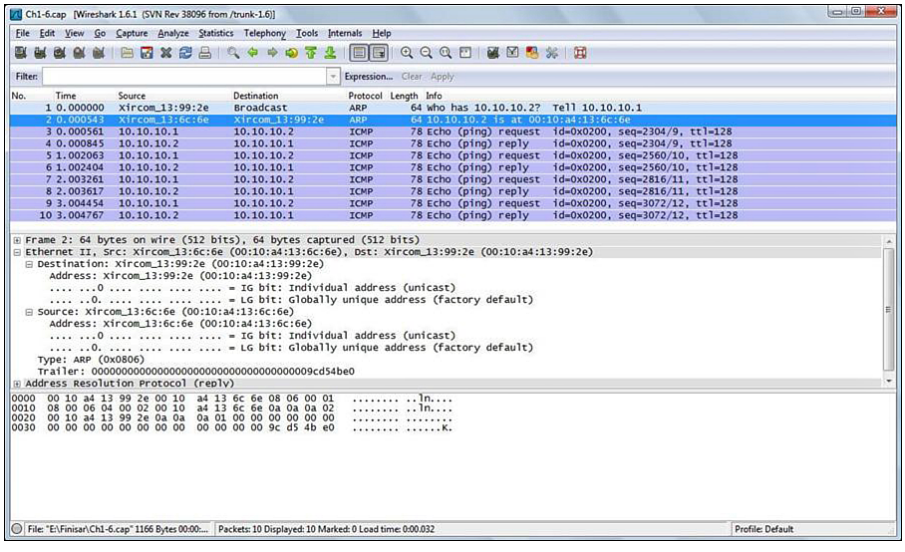

FIGURE 6-9 Computer 2 replying with its MAC address back to computer 1

Figure 6-10 shows computer 1 sending an **echo request** directly to computer 2. An echo request is the part of the ICMP protocol that requests a reply from a computer. Notice in the echo request that the destination address is 00-10-A4-13-6C-6E (computer 2's MAC address), and the source is 00-10-A4-13-99-2E (computer 1's

Echo Request
Part of the ICMP protocol that requests a reply from a computer.

MAC address). Recall that computer 1 now knows the MAC address for IP address 10.10.10.2, so the **ping** request can be sent directly. In this step, computer 1 uses the ICMP **ping** command to verify network connectivity. The highlighted area in Figure 6-11 (number 4) shows computer 2's echo reply. This series of echo requests and replies repeats three more times for a total of four cycles.

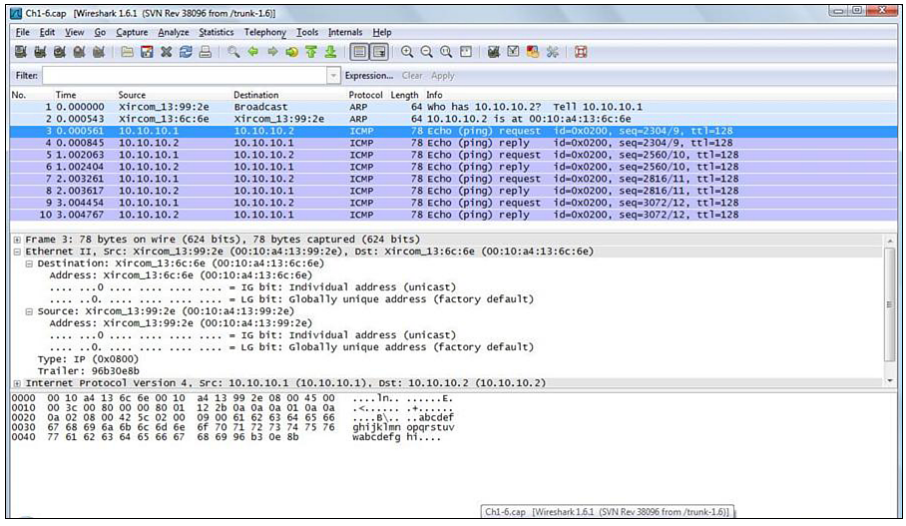

FIGURE 6-10 Computer 1 is sending an echo request to computer 2

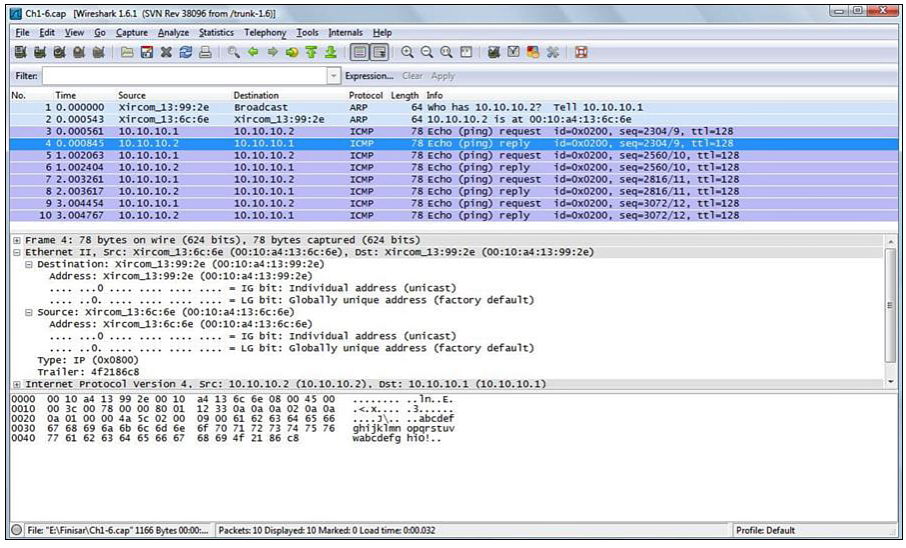

FIGURE 6-11 The echo reply from computer 2

Using Wireshark to Capture Packets

The first exercise with the WireShark software demonstrated how to use the protocol analyzer to inspect captured packets. In most cases, the user will want to capture data packets from their own network. The following steps describe how to use the software to capture packets:

1. In Windows, click **Start** > **Programs** > **WireShark** > and select **WireShark** to start the program.
2. To capture packets on an operating network, you first need to select the interfaces in which you would like to obtain the capture (see Figure 6-12). You can do this by going to **Capture** > **Interfaces**. After selecting your interfaces, click **Start** to start capturing, as shown in Figure 6-13. You can also get to the interface list by clicking on **Interface List** from the WireShark home screen.

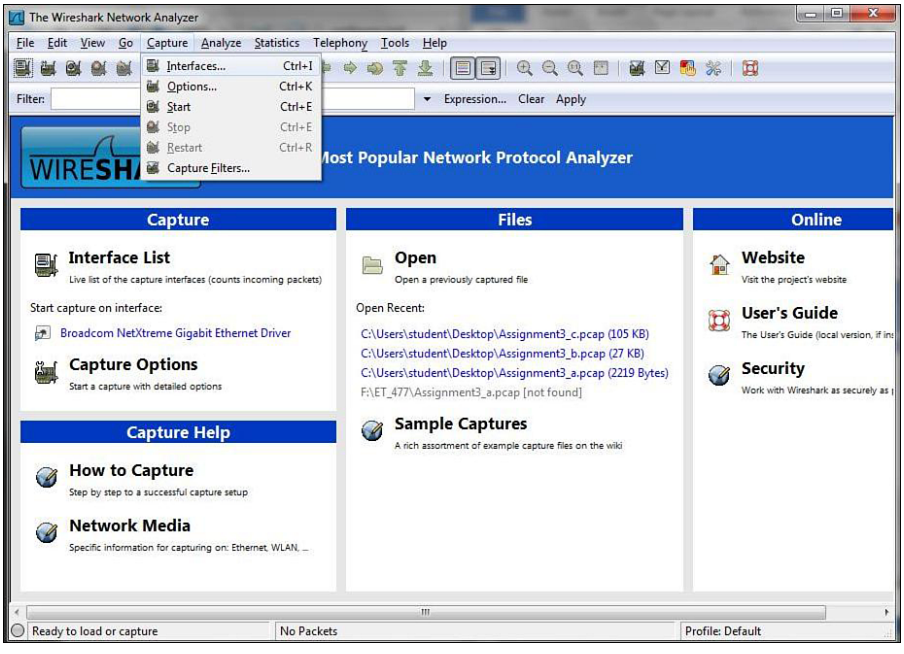

FIGURE 6-12 Initializing Wireshark to capture data packets from your network

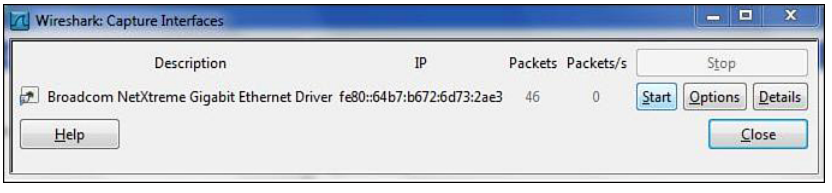

FIGURE 6-13 Starting the capture

3. To examine the packets, stop the simulation by clicking **Capture > Stop**. Remember, there must be some activity on your network for packets to be transferred. You may see little traffic activity if your network is in the lab, and there is limited network activity. You can always use the **ping** command to generate some network data activity, if needed.

To open a saved capture file, click **File > Open** or click **Open** from the Wireshark home screen.

To change capture options, click **Capture > Options** to change the options to your preferred settings.

6-3 ANALYZING NETWORK DATA TRAFFIC

A network of moderate size has a tremendous number of data packets entering and leaving. The number of routers, switches, hubs, servers, and host computers can become staggering. Proper network management requires that all network resources be managed. This requires that proper management tools be in place.

A fundamental network management tool is **SNMP (SNMPv1)**, the Simple Network Management Protocol. SNMPv1, developed in 1988, is widely supported in most modern network hardware. SNMP is a connectionless protocol using the User Datagram Protocol (UDP) for the transmission of data to and from UDP port 161.

SNMP (SNMPv1)
Simple Network Management Protocol.

Management Information Base (MIB)
A collection of standard objects that are used to obtain configuration parameters and performance data on a networking device.

SNMP uses a **management information base (MIB)**, which is a collection of standard objects that are used to obtain configuration parameters and performance data on a networking device, such as a router. MIBs describe the structure of the management data of a device subsystem using a hierarchical namespace that contains object identifiers or OID. Each OID identifies a variable that can be read or set via SNMP. For example, the MIB's object, ifDescr, has an OID of 1.3.6.1.2.1.2.2.1.2. This particular object or OID is used to return a description of the router's interfaces as demonstrated in Figure 6-14. An SNMP software tool was used to collect the interface description information. The IP address of the router is 10.10.10.1, and a get request ifDescr was sent to port 161, the UDP port for SNMP. The descriptions of the interfaces were returned as shown.

Obtaining the SNMP data requires that SNMP be configured on the router. The following discussion demonstrates how to configure SNMP on a Cisco router.

Configuring SNMP

The first step for configuring SNMP on a Cisco router is to enter the router's configuration mode using the **conf t** command:

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

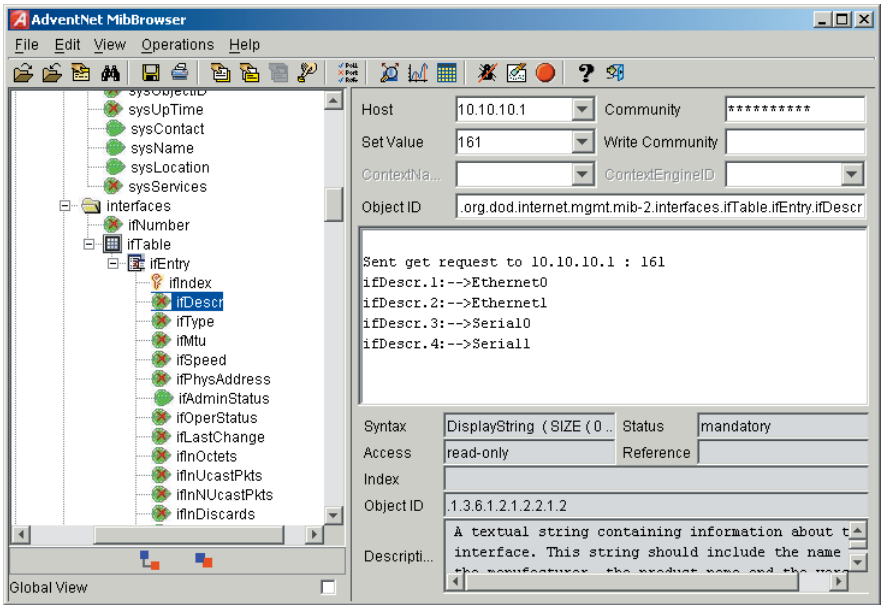

FIGURE 6-14 An example of using an SNMP software management tool to obtain descriptions of a router's interfaces using the MIB (ifDescr)

From the router's (config)# prompt, enter the command **snmp community** [*community string*] [*permissions*]. The community string can be any word. The permissions field is used to establish whether the user can read only (**ro**), or read and write (**rw**). The options for configuring SNMP on the router are shown here:

snmp community [*community string*]
SNMP Community string is a user ID or password that allows access to a network device's statistics.

```
RouterB(config)#snmp community ?
WORD SNMP community string
```

The router was connected to the computer running the SNMP management software, as shown in Figure 6-15. The router's configuration mode was entered, and the **snmp community public ro** command was issued. The word **public** is used as the community string. The community string is the password used by the SNMP software to access SNMP (port 161) on the router. The **ro** sets the permission to read only:

```
RouterB(config)#snmp community public ro
```

In the next example, the community string password is set to **makesecret**, and the permission is set to read write (**rw**). Once again, the router's (config)# mode is entered and the command **snmp community makesecret rw** is entered:

```
RouterB(config)#snmp community makesecret rw
```

The configuration for SNMP can be verified using the **show run** command from the router's privileged mode prompt. A portion of the configuration file that lists the SNMP configuration for the router is shown here:

```
RouterB#sh run
.
.
```

```
snmp-server community makesecret RW
.
.
```

Figure 6-15 shows the setup of the configured router and the computer running the SNMP management software. The SNMP management software issues the SNMP message to the router at port 161, and the router returns the response.

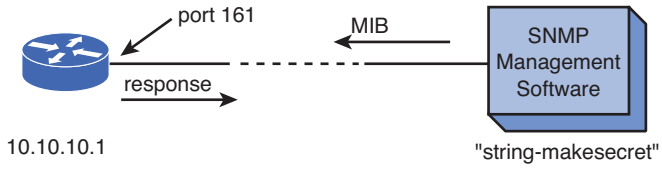

FIGURE 6-15 The setup for connecting the SNMP management software tool to the router

Figure 6-16 shows another example of using SNMP to obtain interface information about a router. The SNMP manager was configured with the host IP address of 10.10.10.1, a set value (port #) of 161, and the 10 character community string of **makesecret** shown as * * * * * * * * * *. The MIB's object (ifspeed) was sent to the router and a status for each of the interfaces was provided. The data displayed shows the speed settings for the router's interfaces.

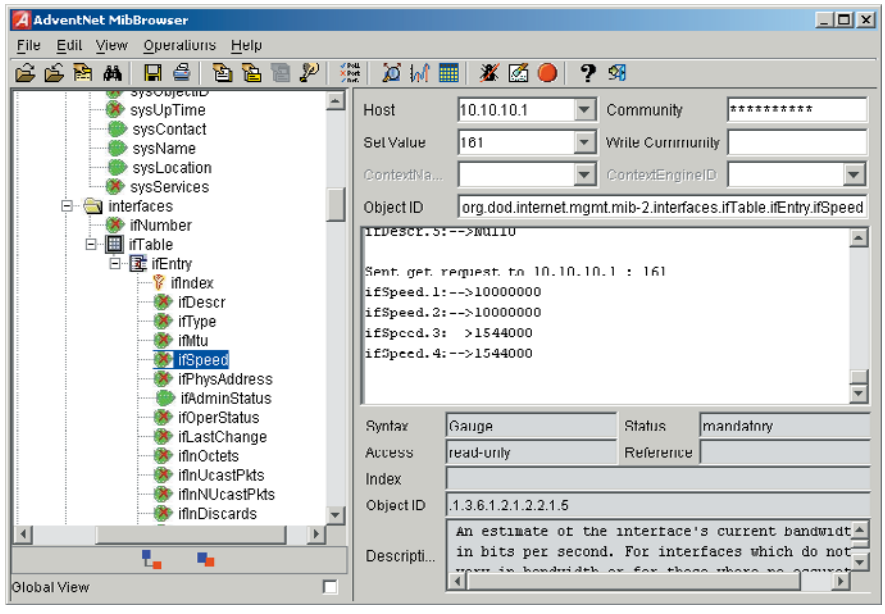

FIGURE 6-16 Using an SNMP software management tool to obtain interface speed settings

Another important application of SNMP is for obtaining traffic data statistics. An example of this is shown in Figure 6-17. The SNMP management program issued an SNMP message with a MIB's object (ifOutOctets), which returns the number of octets of data that have left the router. (The router has a counter that keeps track.) The first result shows ifOutOctets 7002270. The next result display shows that the ifOutOctets returns a value of 7002361.

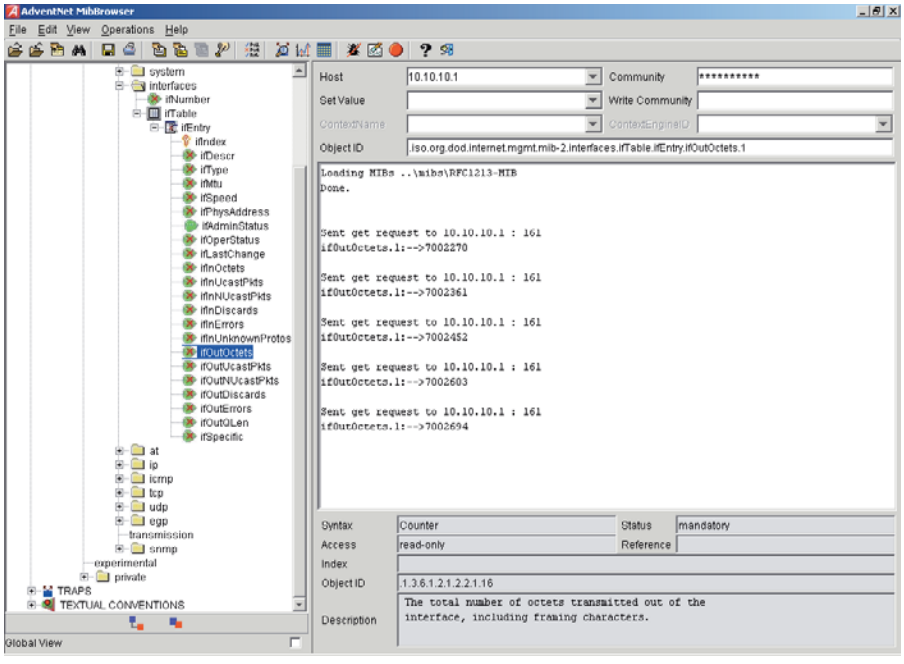

FIGURE 6-17 An example of using SNMP to collect data traffic statistics

The SNMP management program collecting the statistics keeps track of the time interval between measurements and the number of octets that have passed. This information can be used to calculate the average traffic flow by hour, day, week, or month, depending on the information needed. A final note about the router's counter: The counter does not reset unless the router is rebooted.

Two other versions of SNMP have been developed for network management. These versions are **SNMPv2** and **SNMPv3**. SNMPv2 was developed in 1993; however, this version was not directly compatible with SNMPv1. SNMPv2 attempted to address security issues, but this led to the development of many variants and SNMPv2 was never fully accepted by the networking industry. One of the variants called SNMPv2c (Community-based SNMP version 2) was adopted more widely than the others. SNMPv3 was developed in 1998 and achieved the important goal of maintaining compatibility with SNMPv1 and adding security to SNMP. The security features of SNMPv3 include confidentiality, integrity, and authentication. *Confidentiality* means the packets are encrypted to prevent snooping; *integrity* ensures the data being transferred has not been tampered with; and *authentication* means

SNMPv2
Simple Network Management Protocol version 2.

SNMPv3
Simple Network Management Protocol version 3.

the data is from a known source. The steps of how to configure SNMPv3 on Cisco routers are as follows.

```
RouterA(config)#snmp-server group groupv3 v3 priv
RouterA(config)#snmp-server user userv3 groupv3 v3 auth md5 v3password
```

Configuring SNMPv3 on Cisco routers and switches consists of two steps:

1. **Configure an SNMP group**. For example, the command **snmp-server group groupv3 v3 priv** creates a group called **groupv3** with SNMPv3 capability and it will use the version 3's authentication method and privacy (encryption).
2. **Configure SNMP user**. For example, the command **snmp-server user userv3 groupv3 v3 auth md5 v3password** creates a user named **userv3** that belongs to the group **groupv3** created previously. This user will use the HMAC MD5 algorithm for authentication with the password **v3password**.

An example was presented that shows how to obtain the number of octets leaving a router. This type of information can be used in a campus network to monitor the flow of data for many points in the network. Statistics can be obtained for hourly, daily, weekly, and monthly data traffic. This section discusses plots of network router utilization obtained via the router's SNMP port.

Figure 6-18 is a plot of a router's hourly data traffic. The plot shows the average number of bits coming into the router and the average number of bits out. The network administrator should become familiar with the typical hourly data traffic pattern for their network. Notice the decrease in data traffic in the early morning and the dramatic increase in data traffic around 12:00. The traffic clearly shows some type of disturbance around 12:00. The plot is showing that the bit rate significantly increases for a few minutes. This is not necessarily a problem, but it is something that a network administrator will want to watch.

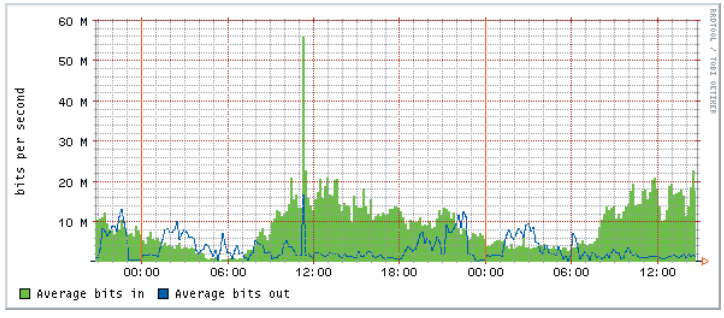

FIGURE 6-18 The hourly plot of a router's data traffic

In this case, the network administrator looked at the daily log of network activity for the same router. This plot is shown in Figure 6-19. The cycle of the data traffic from morning to night is as expected, heavy data traffic about noon and very low data traffic in the mornings. An interesting note is the noon data traffic spikes on the first Wednesday and then repeats the following Wednesday. Whatever is causing the

change in traffic appears to happen on Wednesdays. If this sudden change in data traffic turned out to be something of concern, a protocol analyzer could be set up to capture the data traffic on Wednesdays around noon so that the traffic pattern could be explained.

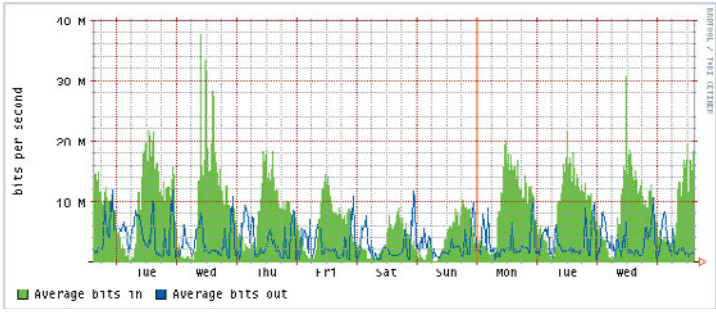

FIGURE 6-19 The daily plot of a router's data traffic

Sometimes, the graph of the network traffic over a longer period of time is needed. Figure 6-20 shows the data traffic through the router over a six-week period. The traffic shows some consistency except for a change from week 11 to week 12. Most likely, this can be explained by examining the network trouble reports and maintenance logs to see if this router was briefly out of service.

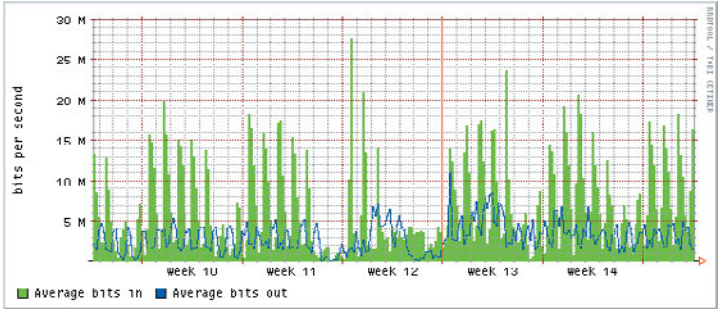

FIGURE 6-20 The weekly plot of a router's data traffic

Justifying the expansion of a network's capability (for example, higher data rate or better core or distribution service) requires showing the manager data traffic statistics. Figure 6-21 is a plot of the router's monthly data traffic. The summer shows a significant decrease in data traffic. The plot also shows that the network was down once in the June–July period and again in January. The manager wants to know if there is justification to increase the data rate of the router to 1 gigabit (1 GB). (The router's current data rate is 100 Mbps.) Is there justification to upgrade the router to 1 GB? Probably not, at least not immediately. The maximum measured average data rate is about 16 Mbps. The router's 100 Mbps data rate does not seem to be causing any traffic congestion problems.

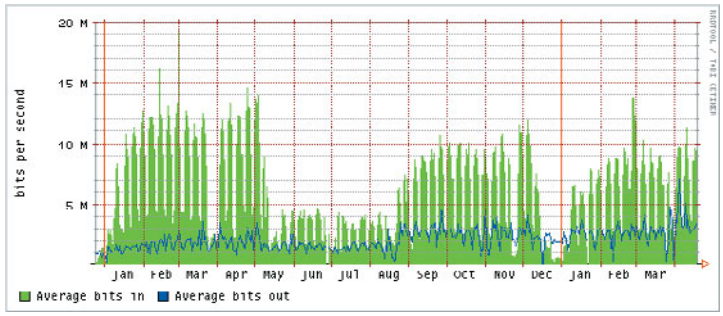

FIGURE 6-21 The monthly plot of a router's data traffic

This section showed how keeping logs of data traffic can be used to spot potential network problems and help plan for possible future expansion of the network.

NetFlow
Used to acquire IP traffic operational data in order to provide network and security monitoring, traffic analysis, and IP accounting.

NetFlow

SNMP allows for the gathering of the statistical information from network devices; however, it does not dive deep into IP information, such as source, destination, or protocol of each data packet. **NetFlow** allows for such data collection. Contrasted to SNMP, NetFlow is a push technology where NetFlow data is pushed from a network device to a collector. NetFlow was created by Cisco in 1996 for acquiring IP traffic operational data in order to provide network and security monitoring, traffic analysis, and IP accounting. Currently, there are 10 versions of NetFlow. NetFlow version 5 is the most common version deployed on many network devices from different vendors. NetFlow version 9 is the first version to support IPv6 and that version is now standardized by the IETF to NetFlow version 10 or Internet Protocol Flow Information Exchange (IPFIX).

Jflow
Juniper's IP traffic flow technology.

Sflow
A traffic flow technology developed by InMon.

Even though NetFlow was developed by Cisco, it is not Cisco proprietary protocol. Many network vendors have adopted NetFlow to collect their IP traffic flow statistics. Nonetheless, there are still variants of the NetFlow protocol available to the public. For example, **Jflow** is Juniper's IP traffic flow technology. It is similar to NetFlow version 5; however, it is a flow sampler technology, which samples the number of packets as defined in the router configuration. Created by InMon, **Sflow** (Sampled Flow) is another traffic flow technology. Similar to Jflow, Sflow is also a sampling technology that is designed to collect a large scale of statistical network information. It has many performance counters that it is collecting, which is different than information collected from NetFlow and Jflow. It can be thought of as SNMP on steroids. Its main deployment is in high-speed switched networks with big support from HP, Extreme, and Alcatel. Sflow is not compatible with NetFlow or Jflow.

Collector
Stores and analyzes the flow information.

There is one thing that all these flow technologies have in common: All the flow information has to be exported or sent to the collector. The **collector** stores and analyzes the flow information. There are many flavors of Flow collector software available. Some of them can even collect all different type of flows (NetFlow, Jflow, and Sflow) and are able to correlate information among them. A final note on any flow technologies is that, because it is a push technology from a network device itself, enabling flow could increase the CPU utilization of the device. This is true especially if the device is a busy router with heavy load of network traffic. One should

constantly monitor the CPU health of the device when turning flow on. The following examples demonstrate how to configure NetFlow on Cisco routers.

The first step is to define the version of the NetFlow, the source of the export, and the destination and its listening UDP port where the flows will be exported as follows:

```
RouterA (config)# ip flow-export source Loopback0
RouterA (config)# ip flow-export version 5
RouterA (config)# ip flow-export destination 10.10.101.19 5000
```

The second step is to enable NetFlow on an interface by using the command **ip route-cache flow**. This command enables NetFlow on the physical interface and its associated subinterfaces, if there are any. The command **ip flow ingress** is used to enable NetFlow on particular subinterfaces. In this example, the Gigabit 1/0 interface is enabled with NetFlow monitoring, as shown:

```
RouterA (config)# int GigabitEthernet1/0
RouterA (config-if)# ip route-cache flow
```

The NetFlow information can be verified with a **show** command. The command **show ip flow export** shows the NetFlow configuration and its basic statistics. Note that all the flow information has to be sent to the collector. The collector stores and analyzes the flow information. There are many flavors of Flow collector software available. Some of them can even collect all information.

```
RouterA#sh ip flow export
Flow export v5 is enabled for main cache
  Exporting flows to 10.10.101.19 (5000)
  Exporting using source interface Loopback0
  Version 5 flow records
  4196949232 flows exported in 139898308 udp datagrams
  138 flows failed due to lack of export packet
  178 export packets were sent up to process level
  0 export packets were dropped due to no fib
  0 export packets were dropped due to adjacency issues
  0 export packets were dropped due to fragmentation failures
  0 export packets were dropped due to encapsulation fixup failures
```

6-4 FILTERING

Data capture files can be quite large, and it often requires that the network administrator search the capture files to find specific information. This could require searching for a specific IP address, or possibly the contents of a file transfer, or searching for a network problem. This section examines several filtering techniques that are available with Wireshark, which include the following:

1. Typing in the display filter
2. Apply saved display filters
3. Right-click filtering
4. Apply conversation filters

Typing in the display filter is a technique that allows you to create your own filter syntax that enables more complex filtering. The following in an exercise that demonstrates how to filter out any occurrences of a specified IP address from a saved capture file. This exercise requires that you first open Wireshark. Select the 6.2.cap file that is provided in the Wireshark folder that is provided in the CD that comes with the text. Figure 6-22 provides a screenshot of the 6-2.cap file. In the upper-left corner of the screen, you will see the word filter. This indicates the location of the filter button. Click the filter button, and this will open the saved display filters option box. This is shown in Figure 6-23. This is showing that the display filters box is empty.

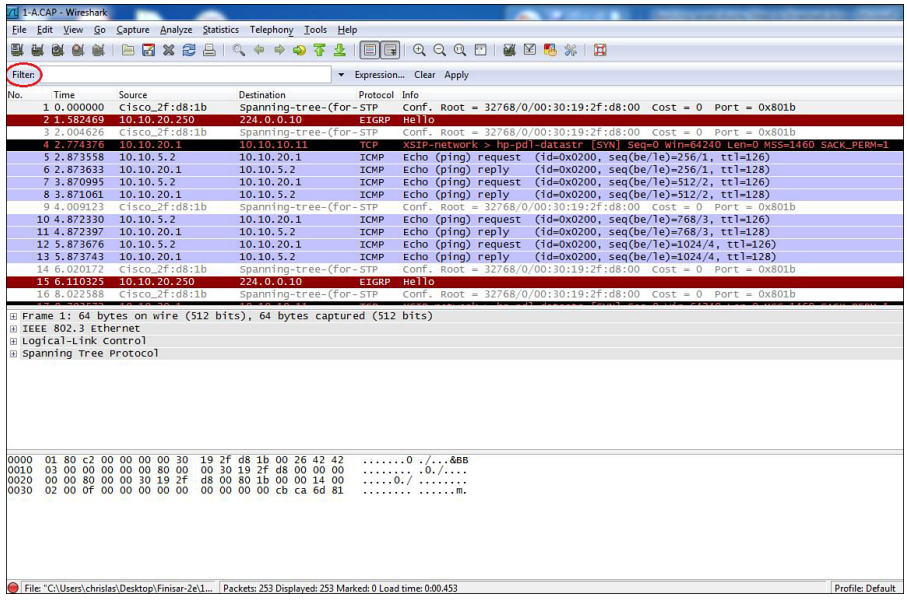

FIGURE 6-22 Screenshot of 6-2.cap file showing the location of the filter button

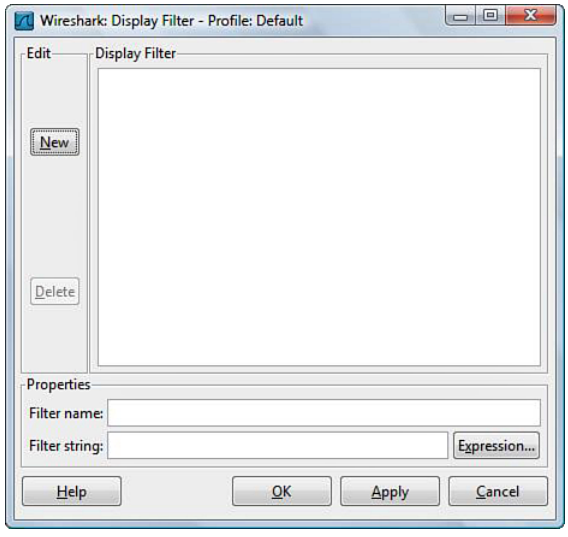

FIGURE 6-23 The Wireshark: Display Filter – Profile menu

In this example, you will first select a filter that only displays IP data packets. To add a filter, click **Expression** at the bottom right of the Wireshark: Display Filter menu. This will open the Wireshark: Filter Expressions – Profile menu. Scroll down until you see IPv4 and click **OK**. You will be returned to the Wireshark: Display Filter menu. The text "ip" is now placed in the Filter String. Next, enter the filter name of **ip only**, as shown in Figure 6-24. Click **New** and the filter for selecting only IP data traffic is entered, as shown in Figure 6-25.

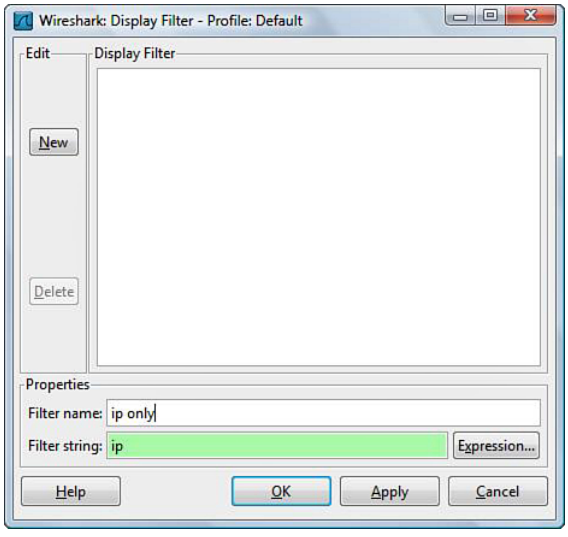

FIGURE 6-24 The creation of the IP only filter

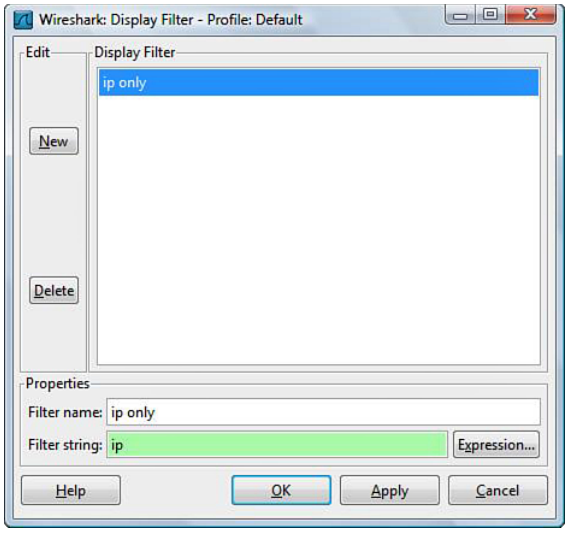

FIGURE 6-25 The addition of the IP only filter to the Wireshark: Display Filter menu

Figure 6-25 shows that the **ip only** filter has been added to the Filter – Profile. Apply this filter by clicking the **OK** button. Notice that the Wireshark screen changes and the non-IP related traffic is filtered out and the Wireshark screen is only displaying Internet Protocol–based traffic. You can also limit this filter down further to show only a specific IP address. For example, search for the IP address 10.10.5.2 by clicking the filter button, which displays the Display Filter – Profile menu.

In the Filter string box, enter **ip == 10.10.5.2**, add the Filter name of **ip address equals 10.10.5.2**, and click **New**. This adds the IP address 10.10.5.2 filter to the Display Filter menu, as shown in Figure 6-26.

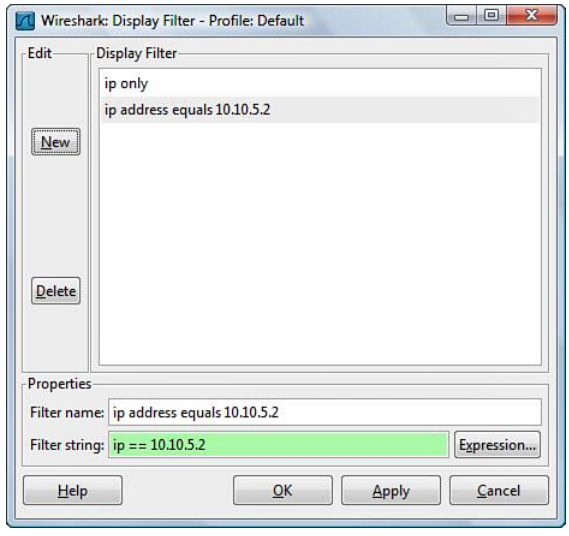

FIGURE 6-26 The addition of the **ip = = 10.10.5.2** filter to the Wireshark: Display Filter menu

Click the **Apply** button and only data packets with the IP address 10.10.5.2 are displayed. The result of applying the **ip.addr = = 10.10.5.2** filter is shown in Figure 6–27. Notice that only data packets containing the IP address 10.10.5.2 are displayed.

Notice that the Filter String box will change color according to the syntax error checker. The text box is highlighted green, meaning that the entered filter is the correct syntax and will produce an output. If an error exists then the text box turns from green to red. This is a built in error checker that Wireshark uses on its filters. The red indicates that incorrect syntax is being input. Any enabled display filters can be cleared at any time using the Clear button on the filter bar.

Another technique for filtering is by right-mouse button clicking the packet you are interested in. For example, right-mouse button click packet 5 from 6-2.cap. Select **Conversation Filter** and **IP**, as shown in Figure 6-28. This produces the same results as generated using the **ip.addr = = 10.10.5.2** filter. The result is the filtered output showing any traffic that is sent in or out of the machine with the IP of 10.10.5.2. This result is shown in Figure 6-29.

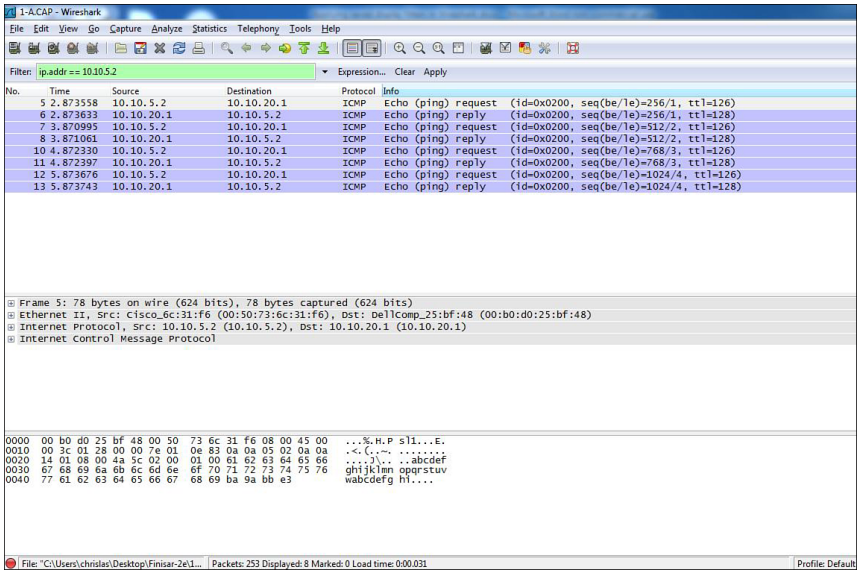

FIGURE 6-27 The results of applying the **ip.addr = = 10.10.5.2** filter

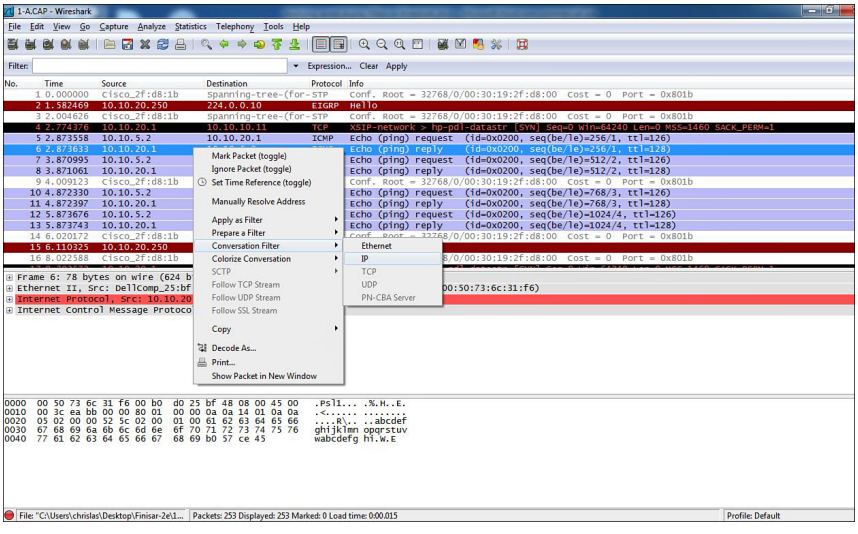

FIGURE 6-28 An example of using the right-mouse button click feature

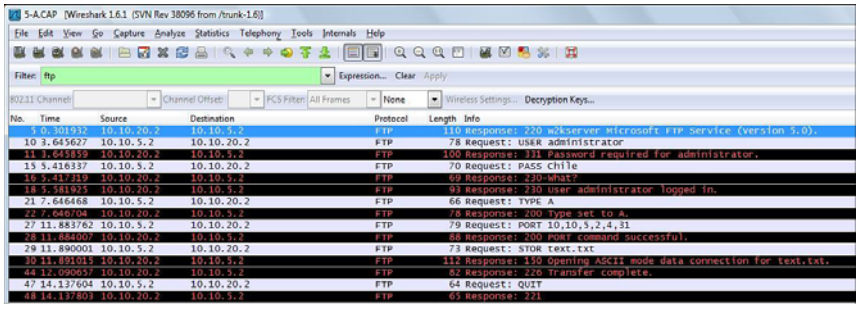

FIGURE 6-29 The result of applying the **ip.addr = = 10.10.5.2** filter

FTP Filtering

The following example demonstrates the process by which Wireshark filtering can be used to isolate File Transfer Protocol (FTP) out of a large list of packets. This can be useful for several reasons. You can use filtering rules to help us find usernames and passwords being used to connect to the FTP servers as well as get an idea of the kind of data that is being transferred.

Start this exercise by opening the capture file 5-A.cap in Wireshark. This is not a huge file, but it's a little difficult to sort through all of it just by looking. Click **Expression** and scroll down until you reach FTP—File Transfer Protocol (FTP). Click **OK** and the Filter for FTP is now displayed, as shown in Figure 6-30.

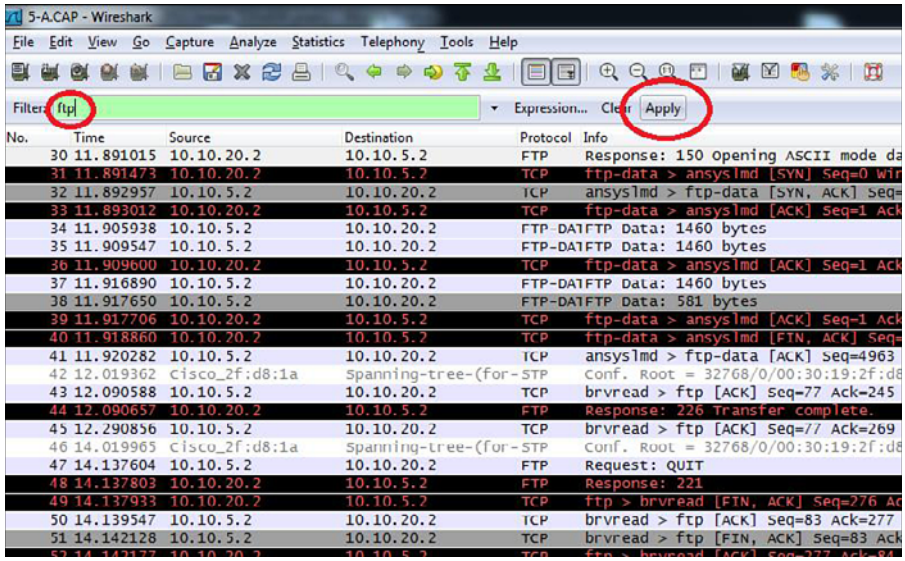

FIGURE 6-30 Adding the FTP filter

Click **Apply**, and the packet list is thinned out to 15 total packets relating to the FTP protocol, as shown in Figure 6-31. From this, we are able to view the username and password used to establish the FTP connection. In this case, the username and passwords are listed in plaintext, as well as the file that was accessed. Most times, a secure version of FTP (SFTP) will be used and this information will be encrypted.

This same rule can also be applied by using the right-click method as previously shown.

Find a packet that is using the FTP protocol (for example, packet 44). Navigate to the datagram field and select the FTP row. Right click -> **Apply as Filter** -> **Selected**. This will generate the same results provided in Figure 6-32 that are used for the FTP filter.

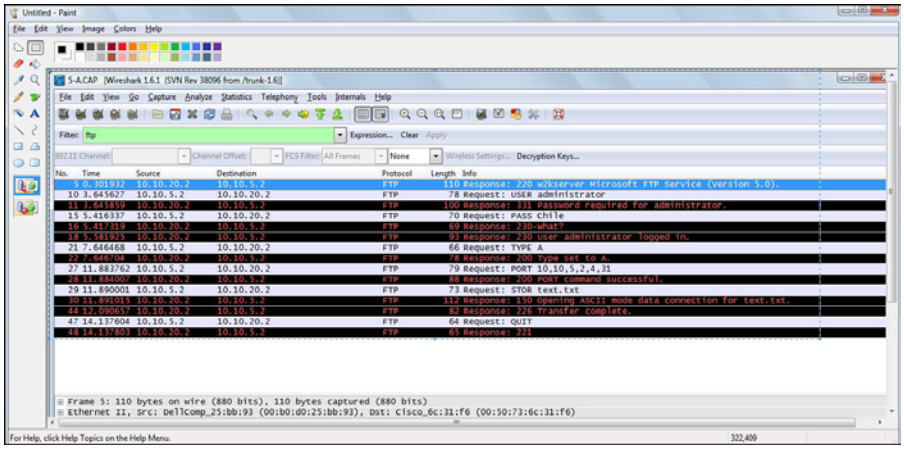

FIGURE 6-31 The result of applying the FTP filter

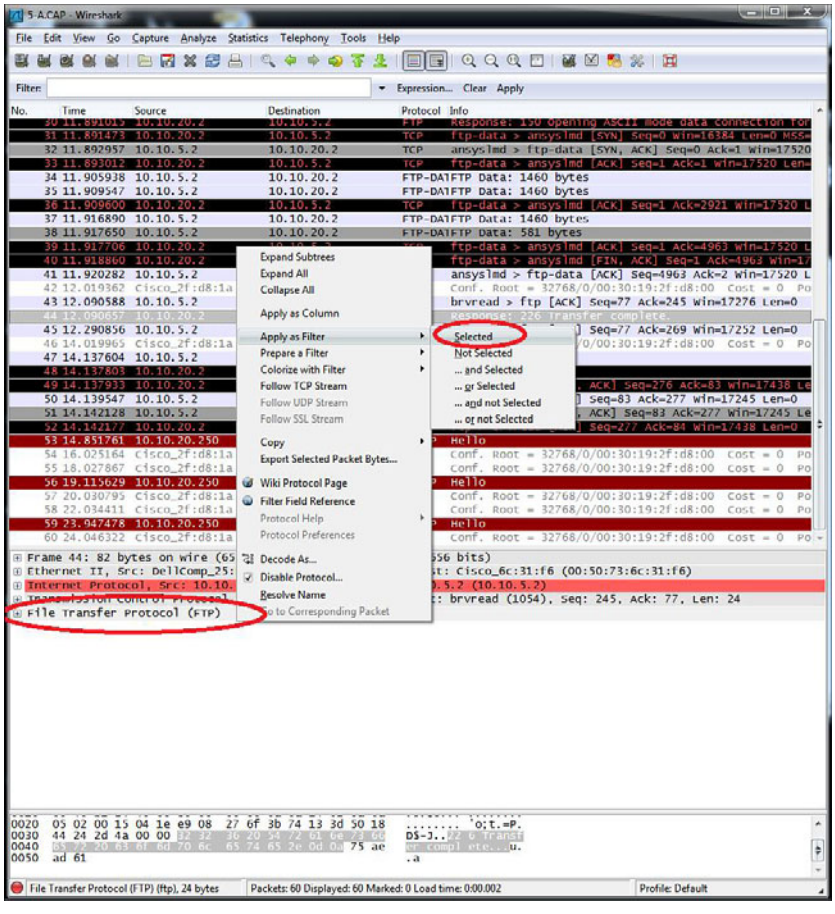

FIGURE 6-32 An example of using the right-mouse button click to filter the FTP data

Another important task is to extract the actual data packets from the capture file. This can be done by using the FTP-DATA protocol filter. To select this filter, click **Expressions** and scroll down to FTP Data. Click **OK** and the FTP Data filter is added to the menu. This filter can be applied by using FTP-DATA in place of FTP. Once filtered, Wireshark can be used to read the plaintext of the file. An example of this is provided in Figure 6-33.

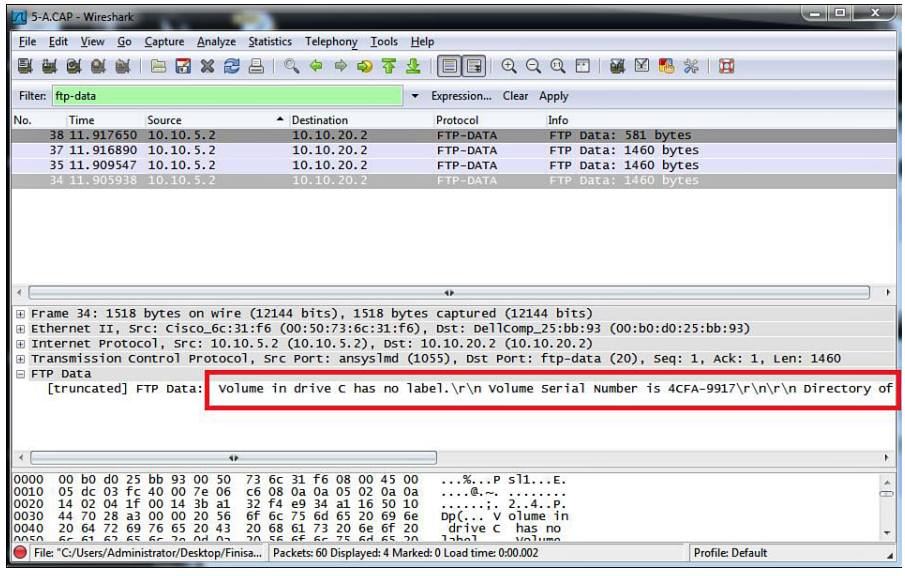

FIGURE 6-33 An example of extracting the data packets from a capture file

Figure 6-33 shows that packet 34 is selected and the FTP data field is expanded. The boxed area shows the first part of the txt file that was transferred. The user can scroll right and read more. Each packet is only 1460 bytes so a text file, even a relatively short one, will be split up in multiple packets. This example text file is only split into four components, but it is a very small text file. When transferring video or audio files over FTP, the files can get large, and there will be many more packets to be examined.

Right-Click Filtering Logic Rules

Within the Apply as Filter and Prepare as Filter menu options are some logical operators. We have already gone over the use of Selected and what it does, but there are other useful operators that can be used. For the purposes of this example, we will also use the Prepare as Filter method. This allows us to verify our filter before applying it. The available logical operators are provided in Figure 6-34.

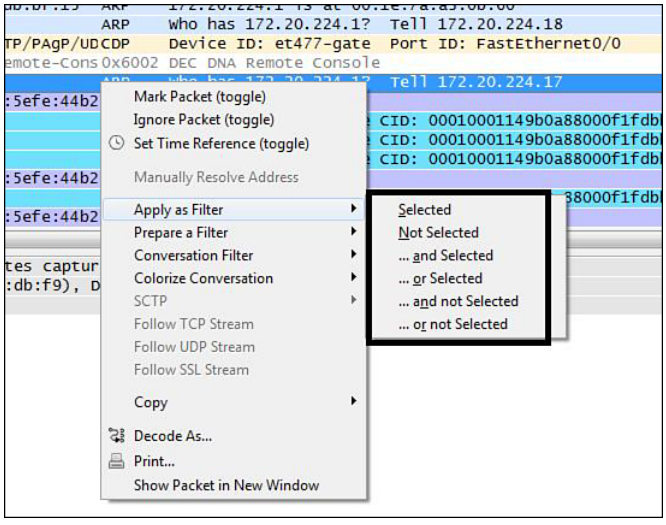

FIGURE 6-34 Available logical operators

In some cases, the network administrator might find it useful to filter out one or more sets of packets at the same time. For this, the **...or Selected** option will be used. In this example, we want to filter the ARP and DHCP data traffic pertaining to releasing and renewing an IP address. To start the exercise, open the 6-DHCP.cap file located in the Chapter 6_Wireshark folder. Both ARP data and the DHCP data will need to be analyzed, so we need to prepare a filter that will show us both of these types of data packets. Begin by selecting an ARP packet and, in the datagram field, right-click **-> Prepare a filter -> ...or Selected**. This puts ARP in the filter string box, but we also need DHCP information. Highlight a DHCP packet and do the same as before. In the datagram field, right-click **-> Prepare a filter -> ...or Selected**. Notice the filter string box now and its inclusion of both sets of filter strings and the newly added or logical operator:

```
(arp) || (dhcpv6)
```

This filter string is used to search for both ARP or DHCPv6 packets. Figure 6-35 shows the newly sorted packet list with only ARP and DHCPv6 Solicit information.

Another possibility is to use the **Not Selected** operator. The objective in this case is to remove protocols one by one that do not pertain to what we need to analyze. For this, you would use the **Not Selected** operator. Start by opening the 6-2.cap file. Next, remove all occurrences of the NBNS protocol. Select an NBNS packet and in the datagram field right-click **-> Prepare a filter -> ...or not Selected**. Next, find and select an EIGRP data packet and in the datagram field and right-click **-> Prepare a filter -> ...and not Selected**. The resulting filter string will list **(!(stp)) && !(eigrp)**. Apply the filter. All occurrences of STP or EIGRP data packets are removed.

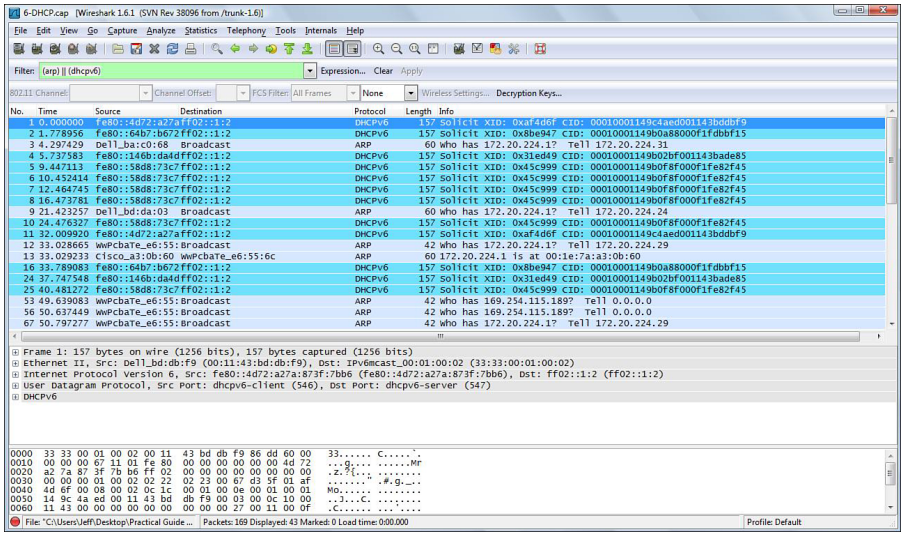

FIGURE 6-35 The results of applying the (arp) || (dhcpv6) filter to the 6-DHCP.cap file

Filtering DHCP

In this exercise, you learn how to filter DHCP packets from a CAP file in Wireshark. You will be using the DHCP.cap file found in the Wireshark folder provided in the companion CD. The filtering method used for this exercise will be typing in the display filter. Open Wireshark and open the DHCP.cap file. As you can see there is a lot of data traffic that was captured in this CAP file. To filter out DHCP, you must type in **bootp** in the filter textbox, as shown in Figure 6-36. (Note: This is case sensitive; do not use BOOTP). The field name for BOOTP can be found by clicking **Expression** and scrolling until you see the **BOOTP/DHCP** filter option.

To apply the **bootp** filter to the DHCP.cap file, click **Apply**. What should be displayed in the packet pane list? You should only see the DHCP protocol data packets displayed, as shown in Figure 6-37.

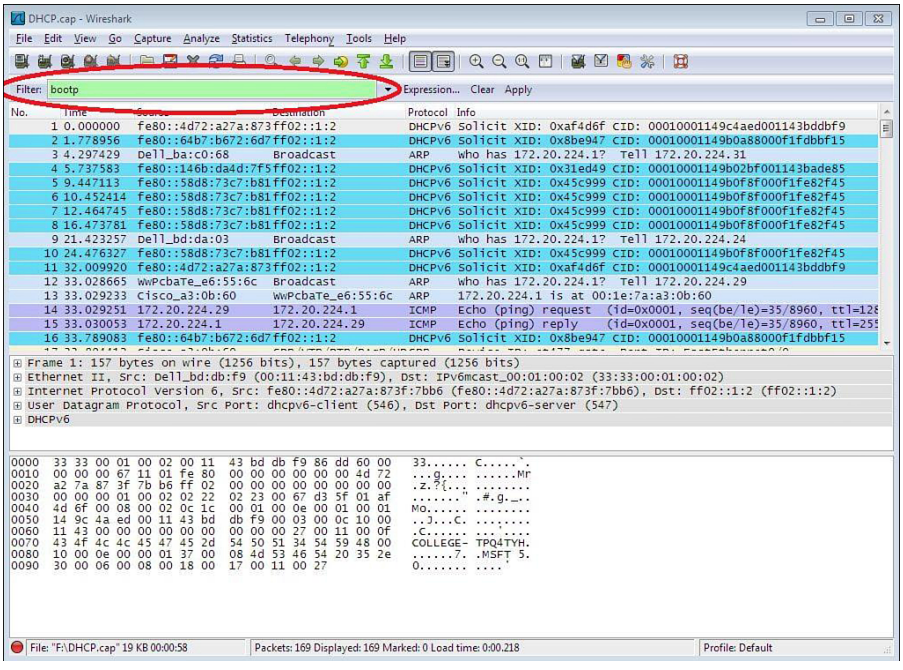

FIGURE 6-36 A screenshot of the DHCP.cap file with "bootp" filter syntax displayed in the filter textbox

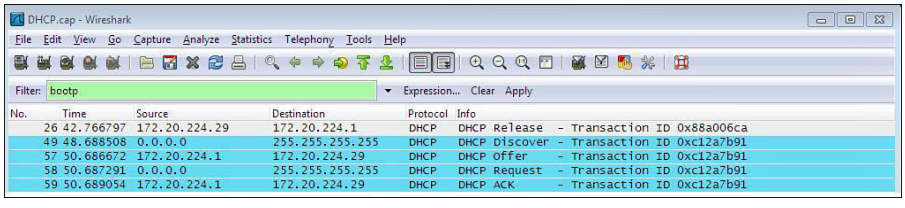

FIGURE 6-37 A screenshot of the DHCP.cap file with the DHCP protocol filtered in the packet list pane

SUMMARY

This chapter looked at the following networking topics. Section 6-1 reviewed the TCP/IP suite of protocols. The use of **netstat** was presented for troubleshooting TCP and UDP connections. Section 6-2 introduced the use of the Wireshark network protocol analyzer. This section is used to get the student started using Wireshark. Section 6-4 introduced filter techniques using Wireshark. Section 6-3 introduced the use of SNMP for the gathering of the statistical information from network devices. Also introduced was the use of NetFlow for acquiring IP traffic operational data.

QUESTIONS AND PROBLEMS

Section 6-1

1. What is an Internet socket?
2. What is network forensics?
3. What are well-known ports?
4. Identify the port numbers for the following applications:
    a. Telnet
    b. HTTP
    c. FTP
    d. DNS
    e. DHCP
5. Define the purpose of a *connection-oriented protocol.* Give an example.
6. What three packets are exchanged between two hosts when establishing a TCP connection?
7. What is the purpose of a sequence number (SEQ=) in TCP data packets?
8. Explain how a host knows whether a data packet was not received.
9. Describe how a TCP connection is terminated.
10. What is a *connectionless protocol*? Give an example.
11. What is the SYN-SENT state?
12. What is the SYN-RECEIVED state?
13. What is the purpose of an ARP request?
14. This state indicates that the three-packet handshake established a TCP connection.
15. What is the FIN-WAIT-1 state, and where is it used?

16. In this state, the terminating host acknowledges the last FIN and waits for the connection to close.
17. In this TCP Connection state, the host is listening and ready to accept connections.
18. In this TCP Connection state, the receiving host acknowledges the FIN.
19. In this TCP Connection State, the terminating host receives the acknowledgment from the receiving host.
20. The **netstat –an** command is issued. What does the following indicate?

```
TCP    0.0.0.0:22             0.0.0.0:0              LISTENING
```

21. The **netstat –an** command is issued. What does the following indicate?

```
TCP    172.16.101.7:49192  199.7.59.72:80        TIME_WAIT
```

22. What is the purpose of an ARP reply?
23. What command is used to view the ARP cache?
24. What command can be used to display the age of each ARP entry?
25. What important networking-troubleshooting tool is part of ICMP, and how does it test a network connection?
26. What is the purpose of the ICMP message type 0?
27. What ICMP message type is Time Exceeded?
28. What is the purpose of the ICMP message type 8?

Section 6-2

29. Expand the acronym *ARP.*
30. What is the purpose of an ARP request?
31. Expand the acronym *ICMP.*
32. What is an *echo request*?
33. What is the purpose of a protocol analyzer?

    Included on the companion CD-ROM in the Wireshark capture file folder is a network packet capture file called *Packet11a.cap***.** Open this file using Wireshark. The following five questions refer to this file.

34. What are the MAC addresses of the computers involved?
35. Which IP addresses correspond to each MAC address?
36. Which packet IDs correspond to ARP requests?
37. Which packet IDs correspond to ARP replies?
38. Which computers are pinging which computers?
39. In terms of computer security, a switch offers better security than a hub. Why is this?

Section 6-3

40. What is the management information base?
41. What port does SNMP use and what transport protocol?
42. The SNMP MIB get request ***ifDescr*** returns what information from a router?
43. What is the purpose of the MIB?
44. Write the Cisco router command for configuring SNMP on a Cisco router. Assume a community string of networking and set the permissions to read-only. Show the router prompt.
45. The command **show run** is entered on a Cisco router. Describe what the output "SNMP-server test RO" means.
46. What SNMP MIBs were most likely issued to the router discussed in Section 6-4?

Use Figure 6-38 to answer questions 47 to 51.

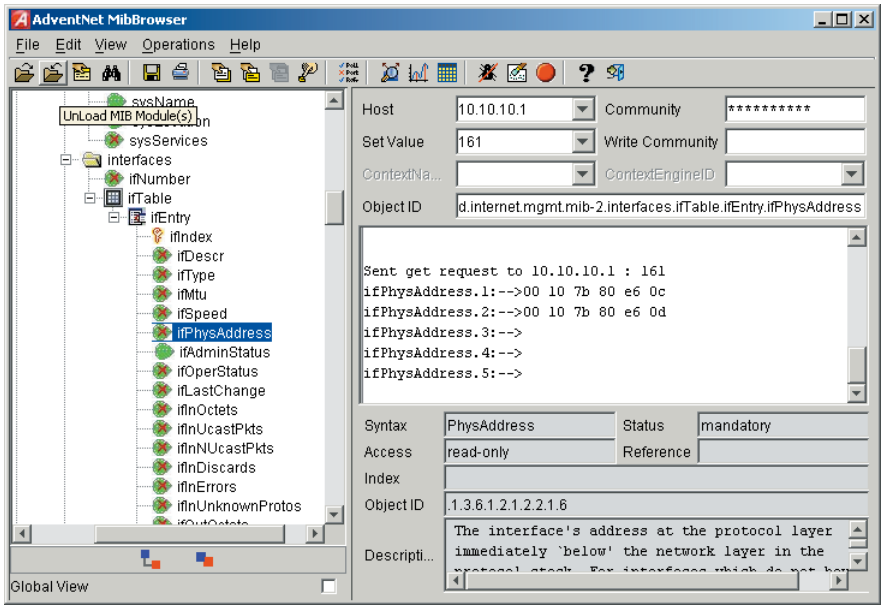

FIGURE 6-38 For problems 47–51

47. What MIB was issued?
48. What information was returned?
49. What port number was used?
50. What protocol is being used? How do you know?
51. Who is the manufacturer of this networking device?
52. What is the advantage of SNMPv3?

53. What security features are provided with SNMPv3?
54. What is confidentiality?
55. What is integrity relative to security?
56. What is authentication relative to security?
57. What is the purpose of NetFlow?
58. What is the purpose of the collector when used with "flow" technologies.
59. What is the following command doing?

```
RouterA (config)# ip flow-export source Loopback0
```

60. What is the purpose of the command **ip route-cache flow**?
61. What command is used to display the NetFlow information?
62. What does the following command do?

```
RouterA (config)# ip flow-export version 5
```

63. What is the purpose of the following command?

```
RouterA (config)# ip flow-export destination 10.10.101.19 5000
```

Section 6-4

64. A filter with the **ip.addr = = 10.10.10.1** filter is applied to captured network data traffic. What happens?
65. What filter could be used to display on data packets containing the IP address 192.168.12.5?
66. What filter could be used to only display data files containing the FTP protocol?
67. What is the purpose of the FTP-DATA filter?
68. What is the purpose of applying the (arp) || (dhcpv6) filter?
69. List a filter to remove all occurrences of STP or EIGRP.
70. List a filter to remove all occurrences of ARP or ICMP.
71. List a filter that can be used to display only data packets containing the IP address 208.76.11.230?

Critical Thinking

72. Use the Wireshark protocol analyzer to capture a file transfer to a TFTP server. Prepare a report on your findings. Identify the port used to establish the TFTP transfer and the source and destination ports used for the TFTP file transfer.
73. Repeat problem 72 for loading a file from a TFTP server.
74. Open the sample **wireless capture.pcap** file provided in the Chapter 6 Wireshark folder in the textbook CD. Search for the 74.125.239.27 IP address. Describe what is happening at this address.
75. When issuing a command **netstat -an** on a server, there are a lot of TCP state of SYN-RECEIVED and ESTABLISHED showing. Should there be any concerns with what netstat is reporting?