Section 1: Getting Started

Introduction

Throughout this book, you will learn about the network protocols on which the Internet is based by examining traces of real network activity. To do this, we will use an open source network protocol analyzer called Wireshark, formerly called Ethereal.* This section is designed to provide you with everything you need to get started.

We begin with an exercise that uses Wireshark/Ethereal to capture and examine a short trace taken on a network with very little activity. This exercise will introduce you to the basic features of Wireshark/Ethereal.

With the basics of Wireshark/Ethereal in place, we will move on to an overview of the layered protocol stack in the Internet. Later sections of the book focus on each layer in detail. However, an overview is essential for understanding the context in which each layer occurs.

The third exercise in this section examines a more complicated network trace containing interleaved threads of activity. It shows you how to use filters to identify and understand each individual thread of network activity.

*Screenshots in the text were taken with Ethereal version 0.10.7. This exact version can be downloaded from the book companion site at www.wiley.com/college/matthews, but we recommend that you install and use the latest version of Wireshark from www.wireshark.org.

Getting Started: Exercise 1.1

Examining a Quiet Network

INTRODUCTION

If you use network software like web browsers or e-mail clients, then you know they require a network connection to work properly. However, do you know what kind of messages they send over the Internet? For example, what does your computer say to a remote web server in order to retrieve a web page and how does your computer direct your e-mail to the person to whom you have addressed it?

You can examine the details of network conversations using a tool called a *network protocol analyzer*. A network protocol analyzer is a piece of software that can record each packet sent over the network and display them in a human-readable format. On a busy network, this can be a lot of information so network protocol analyzers typically provide summary statistics about all packets and allow users to filter out unwanted data or search for specific packets of interest.

Throughout this book, we are going to use an open source network protocol analyzer called Wireshark/Ethereal. This first chapter will give you a basic introduction to Wireshark/Ethereal. Once you have mastered the basics of Wireshark/Ethereal, we will be ready to use it as a tool for exploring the details of network protocols like HTTP, SMTP, TCP, UDP, IP and many more.

You will learn the most from each exercise if you install Wireshark/Ethereal on your local computer and follow along as you read. Wireshark/Ethereal is available for most platforms, including Windows, Mac, and Unix/Linux. You can also install the latest version directly from the Wireshark/Ethereal web site, **http://www.wireshark.org.** Screenshots in the text were taken with Ethereal version 0.10.7. This exact version can be downloaded from the book companion site (**www.wiley.com/college/matthews**). If you upgrade to the most recent version of Wireshark the screenshots will not match perfectly, but you should still be able to follow along.

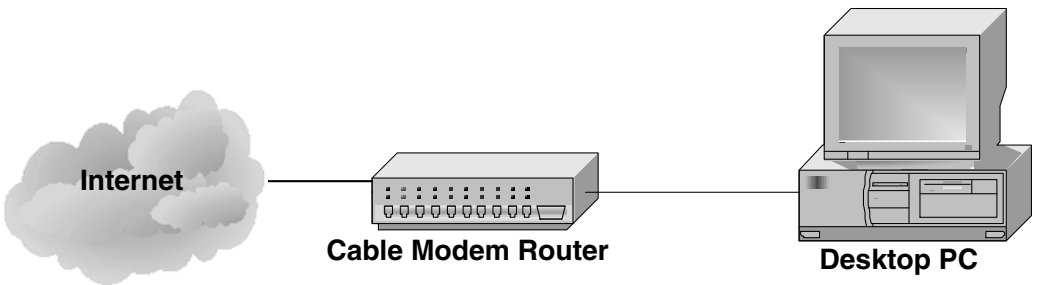
In this first exercise, we will examine a packet trace captured from a "quiet network." We will show you how to capture your own traces (if you have

permission to do so on your local network). We will also show you how to examine the details of each individual packet in the trace and how to view summary statistics about the entire trace.

CONFIGURATION

Each exercise in this book will begin by describing the network configuration in which the experiment is performed. This experiment was performed on a private home network. A single PC running Windows was connected to a cable modem router. Before beginning the experiment, all applications using the network (ex. web browsers, e-mail clients, etc) were closed.

Figure 1.1.1: Exercise Configuration



EXPERIMENT

We begin by starting Wireshark/Ethereal. To capture a trace, we choose Start from the Capture menu and use the Capture Options Dialog to specify desired aspects of the trace. In Figures 1.1.2 and 1.1.3 we show both the Capture Menu and the Capture Options Dialog.

Figure 1.1.2: Capture Menu

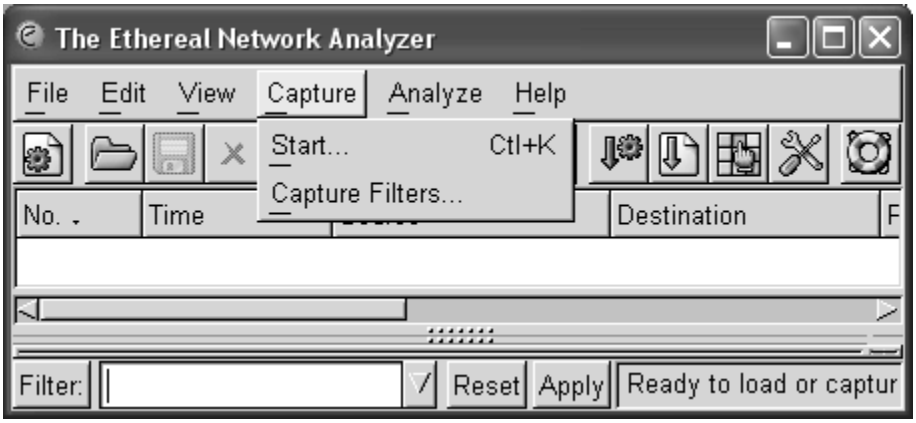
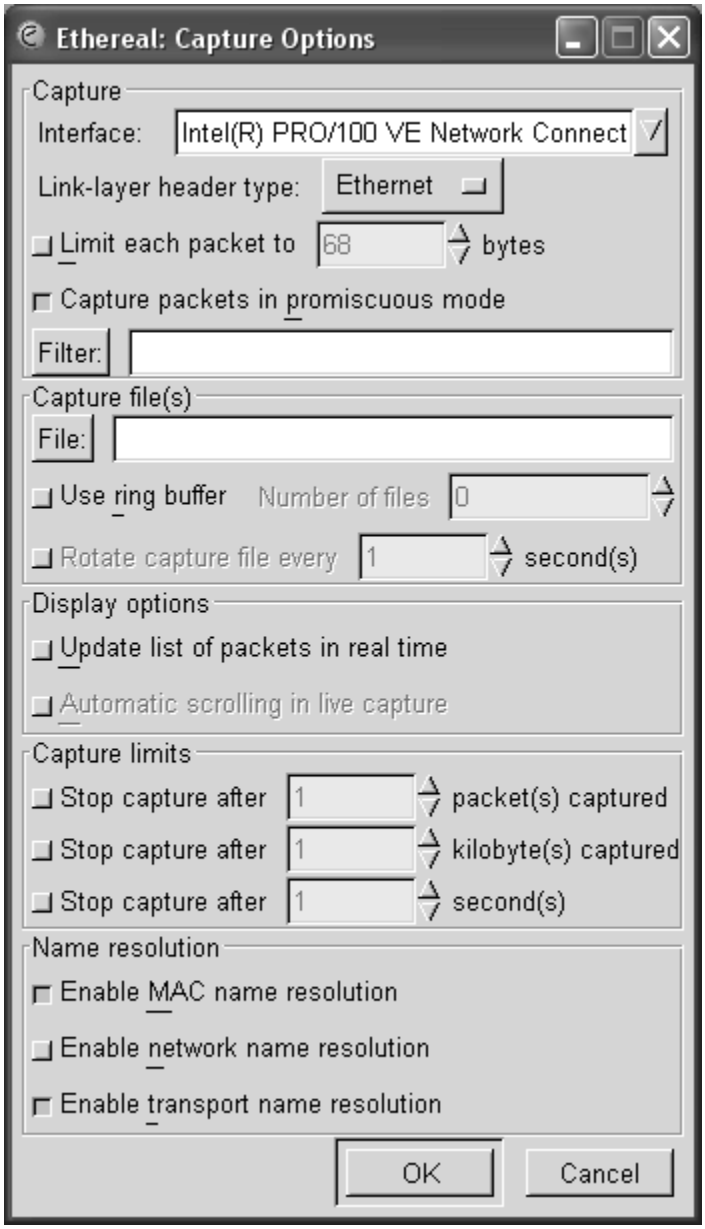


Figure 1.1.3: Capture Options Dialog



USING THE CAPTURE OPTIONS DIALOG

We aren't using all of the options available from the Capture Options Dialog for this exercise, but we provide a brief overview of each one.

1. Interface and Link-layer Header Type

The Interface drop-down list allows you to choose the interface for which you want a packet trace taken. For example, if your machine has both an Ethernet interface and a wireless interface, you must choose which interface to monitor. If you choose the Ethernet interface, only traffic sent over the wired connection will be recorded. If you start a capture and see no traffic or do not see the traffic you expect, then you may need to choose a different network interface.

The Link-layer Header Type Field represents how Wireshark/Ethereal will interpret the link-layer frames. This is also related to the type of interface.

2. Limit Each Packet To N Bytes

Wireshark/Ethereal can capture the entire packet – data and headers and all. Typically, the data takes up the most space, but often, the headers contain the most interesting information (source, destination, type of packets, etc.). If you are not planning to examine the data, you can save space by capturing only the headers. To do this, you compute the maximum length in bytes of the headers you are interested in examining and use this option to capture only that portion of each packet.

3. Capture Packets In Promiscuous Mode

If your computer is on a shared network segment, like a wireless LAN or an Ethernet hub, then your network interface can detect all packets, even those addressed to other computers. Normally, however, packets destined for other computers are simply ignored. If you place your network interface in "promiscuous mode," then it will report all packets even those not addressed to your computer. You need administrative privileges on a machine to put a network interface into promiscuous mode.

When you capture packets in promiscuous mode on a shared network, you may capture sensitive information being transmitted by others. Therefore, it is essential that you have the permission of the network administrator and preferably the consent of network users before capturing packets in promiscuous mode. Many companies and universities expressly prohibit capturing traces on their networks.

For each exercise in this book, we will provide you with captured traces that you can analyze. You are free to open the traces we have provided without any special privileges.

4. Filter

You can limit the amount of data captured by specifying a capture filter. Only packets matching the filter criteria will be recorded. For example, you could capture only packets sent to and from the IP address 192.168.0.1 with the following filter: `host 192.168.0.1`. These filters are quite powerful, but require learning a simple filter language.

In addition to capture filters, you can use a different language to specify display filters that limit the packets displayed while still recording all packets. For example, you can display only packets sent to and from IP address 192.168.0.1 with the following filter: `(ip.dst eq 192.168.0.1) || (ip.src eq 192.168.0.1`. You can use the same language to associate colors with various filters. This technique can be used to highlight some packets while still displaying all of them.

5. Capture Files

You can specify that the captured packets be saved directly to a file rather than saved in memory. If you choose "Use ring buffer," packets will be written into multiple files, switching files when each becomes full or after a specified number of seconds. These controls are important when capturing large or long running traces.

6. Display Options

By default, packets are not displayed as they are captured because when network activity is high it can be difficult for the display to keep up. You can choose to see the packets updated in real time and, if so, you can have the display scroll to the last captured packet automatically.

7. Capture Limits

You can stop the trace manually with the Stop button on the Capture Summary Screen. You can also use these options to request that the trace stop after a certain number of packets are captured, after the trace reaches a certain size, or after a specified amount of time.

8. Name Resolution

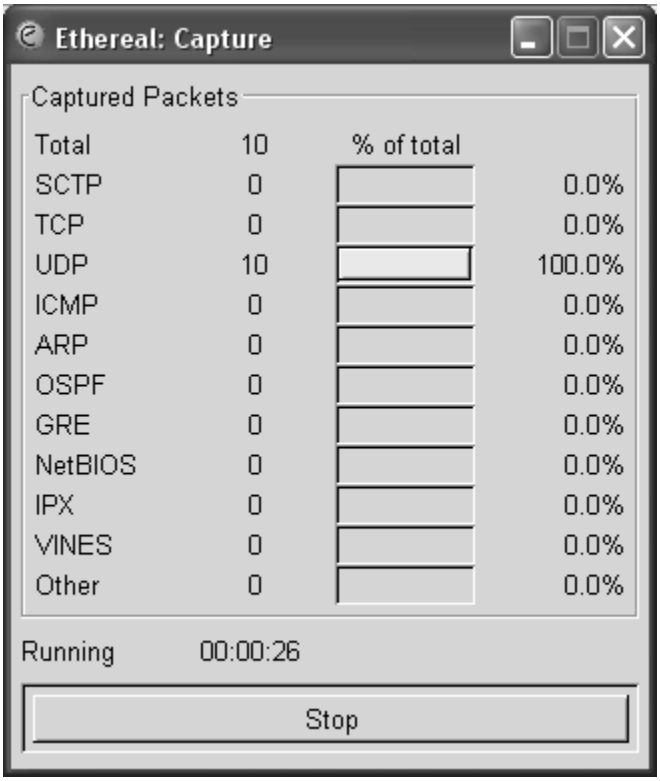
You can request that Wireshark/Ethereal translate various numbers in the packets into human readable names where possible. With MAC addresses translation enabled, Wireshark/Ethereal will translate a portion of this address into the Manufacturer's name. With network address translation enabled, Wireshark/Ethereal will attempt to translate the network address (an IP addresses such as 201.100.0.1) into a host name like www.foo.org. It does this by contacting the local DNS server and requesting a translation. This causes additional network traffic and can result in delays.
With transport name resolution

enabled, Wireshark/Ethereal will translate well-known port numbers into their corresponding protocols. For example, port 80 would be translated into http.

For this exercise, we specify our Ethernet interface and choose "Update list of packets in real time." We leave all other options set to their defaults, as shown in the Capture Option Dialog (Figure 1.1.3).

When we choose Ok, Wireshark/Ethereal begins to capture packets and a capture summary window appears. Each time a packet is transmitted over the Ethernet, it is added to the packet totals in the capture summary window. Since we choose "Update list of packets in real-time," each new packet is also added to the list of packets in the main window. To end the capture, we click stop in the Capture Summary Window. In this experiment, we capture 30 seconds of data.

Figure 1.1.4: Captured Packets Statistics



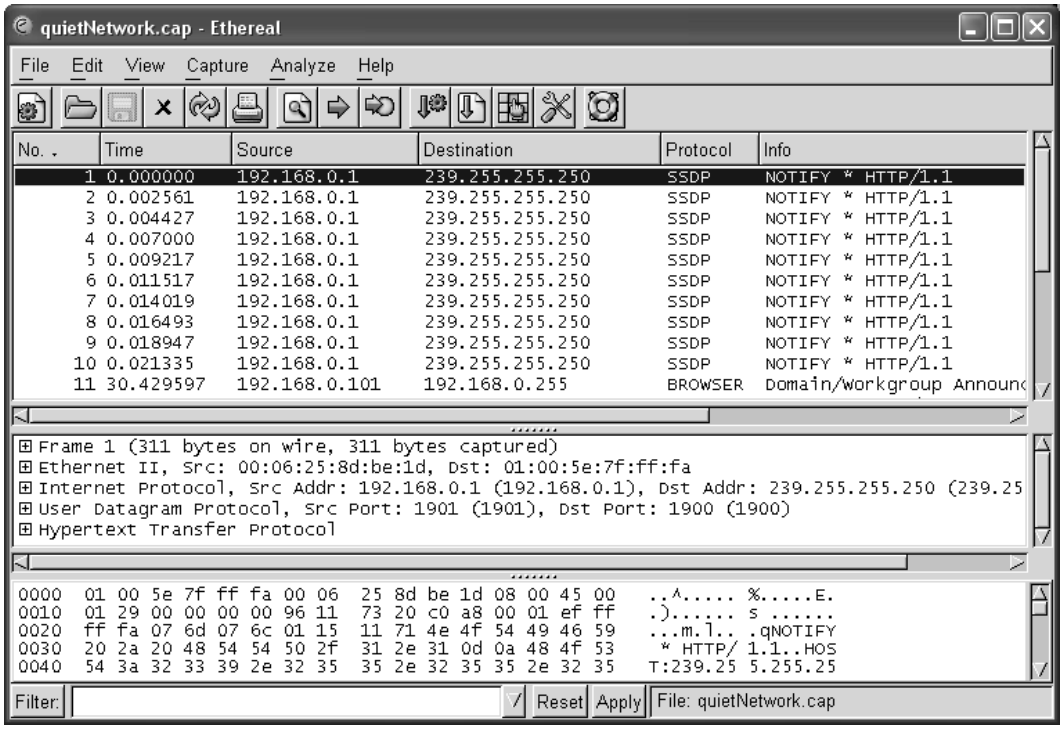
EXAMINING A SHORT TRACE

With our capture complete, we save the resulting trace to the file **`quietNetwork.cap`** (Save from the File menu). You can open this same file and follow along.

In this experiment, we are looking at the network traffic that remains after we close all applications we expect to be using the network. Besides a good exercise for exploring Wireshark/Ethereal, this is an important type of capture for network administrators. An otherwise quiet network is an ideal place to look for suspicious activity. For example, a computer infected with a computer virus may send packets to attack other computers without the knowledge of the computer's owner.

We capture only 21 packets in 30 seconds. A busy network could have hundreds in that same time. In the upper pane of the main window, we see a list of the 21 packets we captured. Each packet is given a number in the trace itself and several important details are displayed including the time the packet was sent (relative to the time of the first packet in the trace), the address of the machine that sent the packet (source), the address to which the packet was sent (destination), the protocol (or language being spoken in the payload of the packet), and finally some information about the contents of the packet.

Figure 1.1.5: Quiet Network Capture



LIST, PROTOCOL, AND RAW PANES

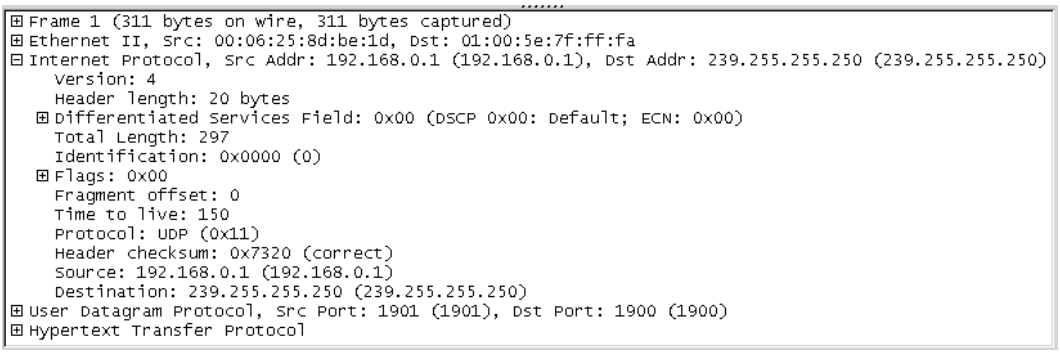
Wireshark/Ethereal can be used to further dissect each packet. When a packet is highlighted in the upper pane of the main window, the two lower panes fill with greater detail about the chosen packet. We will call the top pane the *trace list pane* or simple the *list pane*. We will call the middle pane the *protocol layer pane* or simply the *protocol pane*. We will call the bottom pane the *raw packet pane* or simply the *raw pane*.

To illustrate the function of these three panes, we highlight the first packet in the list pane. As we do so, the protocol pane and the raw pane fill with details of the first packet.

The protocol pane shows each protocol layer of the selected packet: the physical layer frame, the Ethernet frame and its headers, the Internet Protocol (IP) datagram and its headers, the User Datagram Protocol (UDP) datagram and its headers, and finally the Hypertext Transfer Protocol (HTTP) notify message.

For each protocol, you can expand the information even further. For example, if you expand the Internet Protocol Layer, you can see each field in the IP header including the version, the header length, the differentiated services field etc.

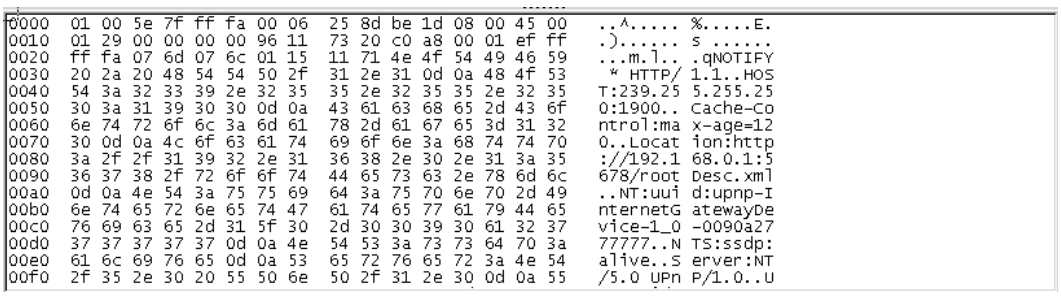
Figure 1.1.6: The Protocol Pane



The raw pane shows each byte of the data contained in the packet. This allows you to look at the data transmitted without interpretation. It is shown in hex on the left and in ASCII on the right.

Notice that when we select the Internet Protocol layer in the protocol pane, the IP header section of the raw packet is highlighted in the raw pane. In this case, "45 00 01 29 00 00 00 00 96 11 73 20 c0 a8 00 01 ef ff ff fa" translates into all the Internet Protocol information displayed in the protocol pane. At the very least, this level of detail can help you appreciate the GUI display capabilities of Wireshark/Ethereal!

Figure 1.1.7: The Raw Pane



TRACE SUMMARY STATISTICS

Now, let's return to the list pane and examine the 21 packets we captured at a higher level.

Twenty of the packets are from source IP address, 192.168.0.1 to destination IP address 239.255.255.250. These 20 packets all list SSDP or the Simple Service Discovery Protocol as their protocol. SSDP is a protocol used for network devices to discover one another. In this case, they are broadcast by the Linksys cable modem router (239.255.255.250 is actually a multicast address). Notice that the cable modem router actually sent 2 groups of 10 identical messages approximately 30 seconds apart (from the timestamps, you can see 10 packets sent from time 0 to time 0.021 and 10 packets sent from time 30.99 to 31.02). If you look inside the Hypertext Transfer Protocol section of one of these packets, you will see that it is announcing it is an available Internet Gateway Device.

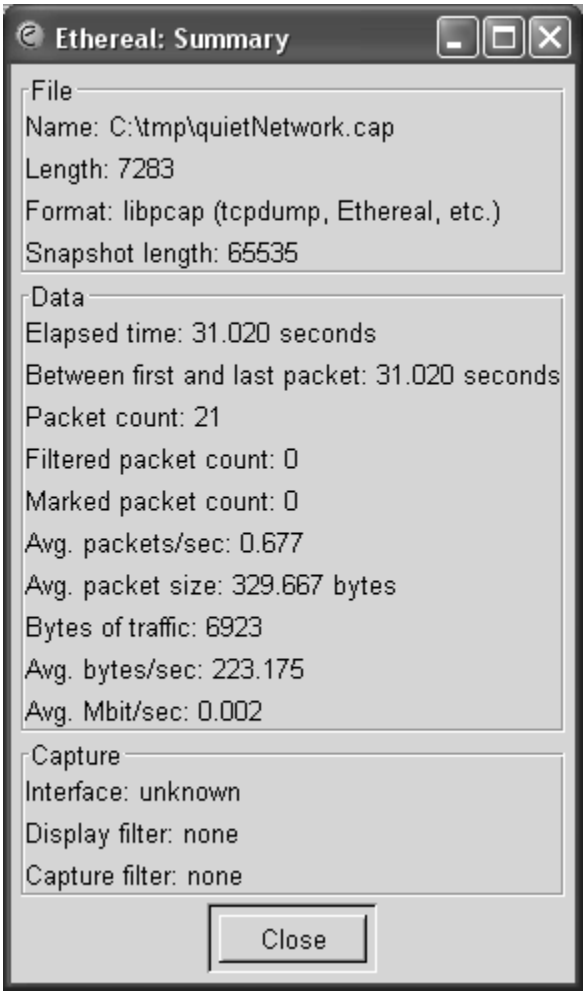
We have discussed the 20 SSDP packets in the **`quietNetwork.cap`** trace. The one remaining packet, packet 11, is sent from IP address 192.168.0.101 to the broadcast address, 192.168.0.255. 192.168.0.101 is the address of the desktop machine. It is announcing its membership in the Domain/workgroup MSHOME.

You can group the 20 packets together by clicking on the "Protocol" heading in the list pane. This sorts the list of packets by protocol. You can click on any of the headings to request a different sort order.

These two types of announcements are just two examples of the type of network activity that occurs even when you are not aware that your computer is using the network. You might be surprised to learn about all the different types of activity on your network that you did not even know existed!

You can view a summary of the trace by choosing Summary from the Analyze menu. It will report a number of useful statistics like the total number of bytes of traffic, the traffic rate, and the average packet size. You may also want to experiment with some of the other analysis tools in the Analyze menu including the Protocol Hierarchy Statistics and the Statistics menu.

Figure 1.1.8: Trace Summary Statistics



QUESTIONS

Answer the following questions about the file **`quietNetwork_15 minutes.cap`**

1. How many seconds does Wireshark/Ethereal report the trace to be? What are two different ways to determine this?
2. When this trace was captured we specified a capture limit of 15 minutes. Give a reasonable explanation for the trace to report less than 15 minutes.
3. How many packets appear in this trace?
4. In **`quietNetwork.cap`**, we saw 2 types of packets SSDP announcements from the cable modem router and Domain/Workgroup announcement from the Windows PC. Do you see these same types of packets in this trace? If so, give the packet number of the first one of each that occurs in the trace.
5. What other protocols appear in the trace? (Hint: try sorting by protocol.)
6. In **`quietNetwork.cap`**, we saw only two different source IP addresses, 192.168.0.1 and 192.168.0.101. Do we see any additional source IP addresses in this trace? If so, give the address and the packet number at which it occurs.

DISCUSSION AND INVESTIGATION

1. What other kinds of traffic might you expect to find on a "quiet network" (i.e. one in which no user is deliberately running an application that is using the network)? Consider both useful background activity and potentially malicious activity?
2. Many system administrators do not allow users to run programs like Wireshark/Ethereal on shared networks. Why do you think this is? Investigate the policies for all the networks that you regularly use. Can you find a network on which you are allowed to take traces?
3. When taking traces, it can be important to understand with whom you are sharing the network. In addition to requesting permission for tracing, knowing who shares your network can help you understand the traffic that you capture and diagnose network performance problems. Make a diagram showing one or more of the local networks you use regularly. How many machines are present on the local network? Are the connections wireless or wired? What type of device do they connect to (hub, switch)?

4. If you have permission, take a trace on a local network. How many source IP addresses and destination IP addresses do you see? Can you identify each machine? Is the network you are tracing "quiet"? Make a list of questions (5–10) you have about the traffic. You could revisit periodically as you complete more exercises in this book.
5. Investigate Snort, an excellent open source network intrusion detection system (**http://www.snort.org**). You can use Snort to monitor your local network for suspicious traffic patterns. Snort users write rules that specify what suspicious traffic is using the same type of information captured by Wireshark/Ethereal (source and destination IP address, protocol, etc.) From what you have learned in this exercise, discuss how Snort might work and propose some simple rules that might detect malicious activity.

RESOURCES

- Wireshark/Ethereal Documentation, **http://www.wireshark.org**
- SSDP, **http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt**
- Snort, **http://www.snort.org**
- Web search: Network Protocol Analyzers