WireShark Lab 0 – Getting Started

One's understanding of network protocols can often be greatly deepened by "seeing protocols in action" and by "playing around with protocols" – observing the sequence of messages exchanged between two protocol entities, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. You'll be running various network applications in different scenarios using a computer on your desk, at home, or in a lab. You'll observe the network protocols in your computer "in action," interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you and your computer will be an integral part of these "live" labs. You'll observe, and you'll learn, by doing.

The basic tool for observing the messages exchanged between executing protocol entities is called a **packet sniffer**. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer(s); it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a *copy* of packets that are sent/ received from/by application and protocols executing on your machine.

Fig. 1 shows the structure of a packet sniffer. At the right of Fig. 1 are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in Fig. 1 is an addition to the usual software in your computer, and consists of two parts. The **packet capture library** receives a copy of every link-layer frame that is sent from or received by your computer. Recall that messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Fig. 1, the assumed physical media is an Ethernet, and so all upper layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

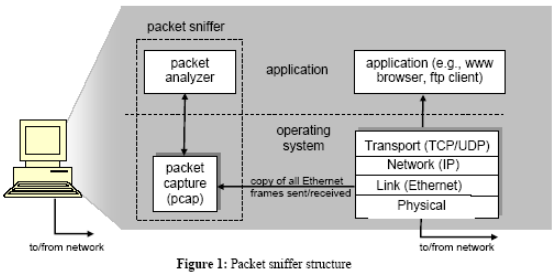

Figure 1: Packet sniffer structure

The second component of a packet sniffer is the **packet analyzer**, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must "understand" the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by HTTP in Fig. 1. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. It understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands HTTP and so, knows that the first bytes of an HTTP message will contain the string "GET," "POST," or "HEAD," headers.

We will use the WireShark packet sniffer (*www.wireshark.org*) for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. WireShark is a free network protocol analyzer that runs on Windows, Linux, and Mac computers. It's an ideal packet analyzer for these labs – it is stable, has a large user base and good support with a user guide (*www.wireshark.org/docs/wsug_html*), man pages (*www.wireshark.org/docs/man-pages/*), FAQ (*www.wireshark.org/faq.html*), rich functionality that includes the capability to analyze over 500 protocols, and a well-designed user interface. It operates in computers using Ethernet to connect to the Internet, as well as so-called point-to-point protocols such as PPP.

Getting WireShark

Download WireShark from http://www.rishiheerasing.net/Network.Tools/WireShark/Wireshark-win64-1.10.5.exe on your computer.
Note: Ensure that you install the WinPcap 4.0 packet capture library if it is not already present on your machine. Please start WinPcap NPF as a service as well.

Running WireShark

When you run the WireShark program, the WireShark graphical user interface shown in Figure 2 will be displayed. Initially, no data will be displayed in the various windows.

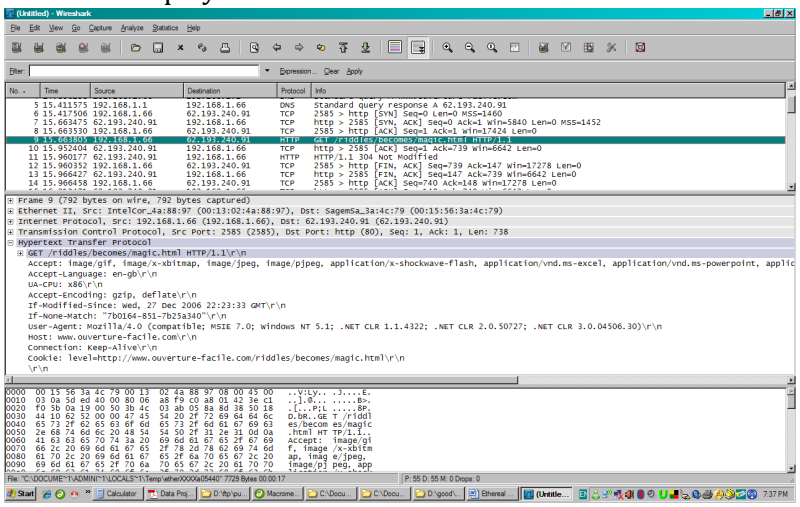

Figure 2

The WireShark interface has 5 major components:

• The **command menus** are standard pull-down menus located at the top of the window. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the WireShark application. The Capture menu allows you to begin packet capture.

• The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by WireShark; this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest level protocol that sent or received this packet.

• The **packet-header details window** provides details about the packet selected (highlighted) in the packet listing window. (To select a packet in the packet listing window, place the cursor over the packet's one-line summary in the packet listing window and click with the left mouse button.). These details include information about the Ethernet frame and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the crosshairs to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, these can similarly be expanded or minimized. Finally, details about the highest level protocol that sent or received this packet are also provided.

• The **packet-contents window** displays the contents of the captured frame in ASCII & Hex.

• Towards the top of the WireShark graphical user interface, is the **packet display filter field,** into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have WireShark hide (not display) packets except those that correspond to HTTP messages.

Taking WireShark for a Test Run

The best way to learn about any new piece of software is to try it out! Do the following:

1. Start up your favorite web browser, which will display your selected homepage.

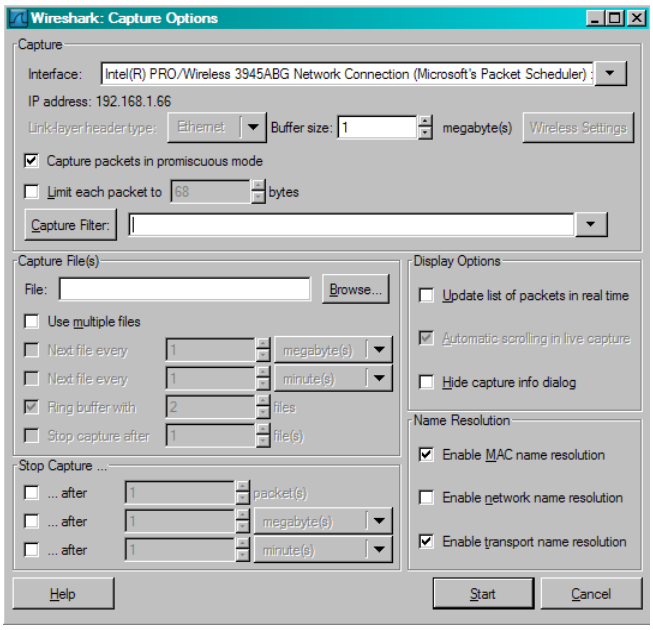

Figure 3

2. Start up the WireShark software.

3. To begin packet capture, select the *List the available capture interfaces…* icon on the far-left menu bar and choose the interface (usually there will only be one interface with an IP address like 172.16.x.x) and Click on *Capture*.

4. You can also the select the second-most icon to the left of the menu bar. This will cause the "WireShark: Capture Options" window to be displayed, as shown in the Fig. 3. You then select the appropriate interface (in case your computer has got 2 network cards) and click on *Start*. Please uncheck the first and last *Display Options* checkboxes.

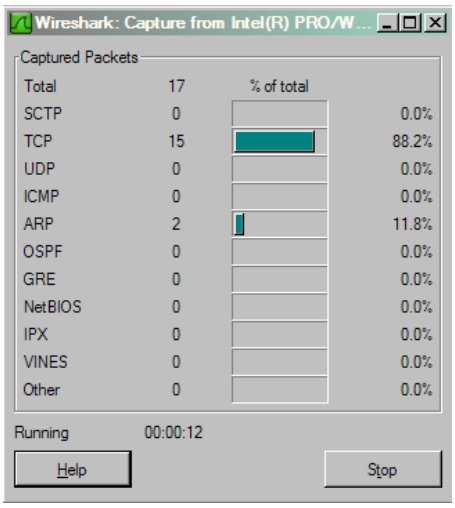

Figure 4

5. Once you begin packet capture, a packet capture summary window will appear, as shown in Figure 4. This window summarizes the number of packets of various types that are being captured, and contains the *Stop* button that will allow you to stop packet capture. Don't stop packet capture yet.

6. While WireShark is running, enter the URL: http://www.fscmauritius.org and have that page displayed in your browser. In order to display this page, your browser will contact the "HTTP" server at www.fscmauritius.org and exchange HTTP messages with the server in order to download this page. The Ethernet frames containing these HTTP messages will be captured by WireShark.

7. After your browser has displayed the FSC homepage, stop WireShark packet capture by selecting ***Stop*** in the WireShark capture window. This will cause the WireShark capture window to disappear and the main WireShark window to display all packets captured since you began packet capture. The main WireShark window should now look similar to Fig. 2. You now have live packet data that contains all protocol messages exchanged between your computer and other network entities! The HTTP message exchanged with the "FSC web server" should appear somewhere in the listing of packets captured. But there will be many other types of packets displayed as well (e.g., the many different protocol types shown in the *Protocol* column in Fig. 2). Even though the only action you took was to download a web page, there were evidently many other protocols running on your computer that are unseen by the user.
8. Type in "http" (without the quotes, and in lower case – all protocol names are in lower case in WireShark) into the display filter specification window at the top of the main WireShark window. Then select *Apply* (to the right of where you entered "http"). This will cause only HTTP message to be displayed in the packet-listing window.

9. Select the first http message shown in the packet-listing window. This should be the HTTP GET message that was sent from your computer to the "www.fscmauritius.org" HTTP server. When you select the HTTP GET message, the Ethernet frame, IP datagram, TCP segment, and HTTP message header information will be displayed in the packet-header window. By clicking on the crosshairs to the left side of the packet details window, *minimize* the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed. *Maximize* the amount information displayed about the HTTP protocol. Your WireShark display should now look roughly as shown in Fig. 2 (Note, in particular, the minimized amount of protocol information for all protocols except HTTP, and the maximized amount of protocol information for HTTP in the packet-header window).

10. Exit WireShark.

Congratulations! You've now completed the first lab.

What to hand in

The goal of this first lab was primarily to introduce you to WireShark. The following questions will demonstrate that you've been able to get WireShark up and running, and have explored some of its capabilities.

Answer the following questions, based on your WireShark experimentation:

1. Screen-capture the HTTP protocols that appear in the protocol column in the filtered packet-listing window in step 8 above.

2. How long did it take from when the first HTTP GET message was sent until the respective HTTP response was received? (By default, the value of the Time column in the packet-listing window is the amount of time, in seconds, since WireShark tracing began. To display the Time field in time-of-day format, select the WireShark *View* pull-down menu, then select Time *Display Format*, then select *Time-of-day)*.

3. Why is the HTTP response code not "200 OK" as you would expect?

4. What is the actual HTTP response code received and what does this code mean?

5. What is therefore the Internet address of the actual server where the FSC homepage resides?

6. What is the Internet address of your computer?

7. What did your browser do before the first HTTP GET message was sent to the web server?

8. Why were more GET messages needed to display the FSC homepage?