**Data Comm/Networking**

**Lab 1: Introduction**

This lab has two parts. Both parts must be done individually so at the end of the lab you have the WiresShark software installed on your PC, and you know how to capture packets.

**Part I. Introduction**

*(Adapted from material by Johann Sarmiento)*

In this first Part, you will run some basic networking tools and try to make sense of the results you obtain. In addition, I would like you to describe your prior experience with networks.

1. **How many network interfaces (i.e., the physical network connections) does your computer have? Describe them briefly. They could be wired and/or wireless.**

Running nmcli device in a UNIX terminal lists 5 devices. 1 wifi device, one wifi peer to peer device, 2 ethernet devices, and a virtual loopback device.

1. **Bring up a command window (Start menu > Run… > cmd).   
   Using the ipconfig /allcommand on Windows and answer the following questions:**
   1. **What is the MAC address (physical address) you are using?**

58:6c:25:4d:fb:0c

* 1. **What is its IP address?**

150.250.210.143

* 1. **What is the default gateway?**

150.250.208.1

* 1. **What other information that ipconfig provides do you understand?**

Since I am using a Linux-based OS, I’m using ifconfig instead. It shows IPv4, IPv6, MAC address, and some other values I am not familiar with such as netmask and broadcast IPs.

1. **Try to pingyour default gateway by tying the command: ping XXX.XXX.XXX.XXX where the XXX is the IP address from Q2C.What information do you get? What does it mean?**

I receive some bytes, a icmp\_seq value, a ttl value, and a ping time in milliseconds. After ending the ping command, I receive some analytical information about the number of packets transmitted, received, the percent of packets lost, the total time, and the length of the shortest, longest, and mean ping times. The icmp\_seq increments by 1 after each successive ping, so I believe it is just a sequential identifier, ttl is the maximum time the packet has to ping, and the ping time is the time between request and response.

1. **Try to ping a public web page using the following command: ping**[**www.google.com**](http://www.google.com)**What information do you get? What does it mean?**

I get 64 bytes of packet data from 142.250.81.228. I receive the same values as before, but the ttl and destination IP have different values. This time ttl is 116 rather than 255.

1. **Try to execute a tracert (traceroute) to www.rowan.edu web site. How do you interpret the results you got?**

I get a list of hops with [www.rowan.edu](http://www.rowan.edu) being the first followed by 29 blank hops. I also receive 3 time values. This is likely because I am on Rowan’s network already.

**Try a tracerouteto www.google.com. Why are the results so different (assuming you run both commands from campus)?**

**(Note: At home you could also web-based network tools such as those in** [**http://www.dnsstuff.com/**](http://www.dnsstuff.com/)**or**[**http://www.all-nettools.com/toolbox**](http://www.all-nettools.com/toolbox)**if you can’t execute the commands on your machine.)**

This time I get a list of 10 hops: starting at Rowan’s IP address and ending at Google’s.

1. How would you rate your knowledge in the following areas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Topic | No clue  [1] | Vaguely  familiar  [2] | Somewhat  familiar  [3] | Somewhat  knowledgeable  [4] | Very knowledgeable  [5] |
| a. Configuring a home network |  |  |  | X |  |
| b. TCP/IP protocols |  | X |  |  |  |
| c. Routing protocols and concepts |  |  | X |  |  |
| d. Cisco IOS | X |  |  |  |  |

|  |
| --- |
| Part II.  Based on Version: July 2005  © 2005 J.F. Kurose, K.W. Ross. All Rights Reserved |

*“Tell me and I forget. Show me and I remember. Involve me and I understand.”*

Chinese proverb

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. This can be done in simulated scenarios or in a “real” network environment such as the Internet. The Java applets that accompany this text take the first approach. In these Wireshark labs, we’ll take the latter approach. 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; 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.

Figure 1 shows the structure of a packet sniffer. At the right of Figure 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 Figure 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 from the discussion from section 1.7.2 in the text 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. 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.



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 the HTTP protocol in Figure 1. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example,knows that the first bytes of an HTTP message will contain the string “GET,” “POST,” or “HEAD,” as shown in Figure 2.8 in the text.

We will be using the Wiresharkpacket sniffer [<http://www.wireshark.org/>] for these labs, allowing us to display the contents of messages being sent and received from and by protocols at different levels of the protocol stack. (Technically speaking, Wiresharkis a packet analyzer that uses a packet capture library in your computer). Wiresharkis a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers. It’s stable, has a large user base and well-documented support that includes a user-guideand man pages (<http://www.wireshark.org/docs/>), a detailed FAQ (<http://www.wireshark.org/faq.html>), includes the capability to analyze more than 500 protocols, and a well-designed user interface. It operates in computers using Ethernet to connect to the Internet, as well as point-to-point protocols such as PPP.

**Wireshark was known for years as Ethereal, which is why many resources still refer to Ethereal. Some screen caps in the lab instructions will show Ethereal instead of Wireshark, because they’re the same program except for the name!**

Getting Wireshark(for home use)

Wireshark can be installed on almost any home computer – Windows, Mac OS X, Linux, Solaris, etc. See <http://www.wireshark.org/download.html>for a list of supported operating systems and download sites. The Wireshark installer handles installing WinPcap on Windows machines.

Running Wireshark

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

TIP: Screen captures will be used a lot in labs to show your output. Recall in Windows that pressing the ‘Print scr’ key captures the entire display. **To capture just the active window, use ‘Alt-Print scr’ (alt – print screen)** instead. Then use Edit > Paste Special to paste the image as a Device-Independent Bitmap.



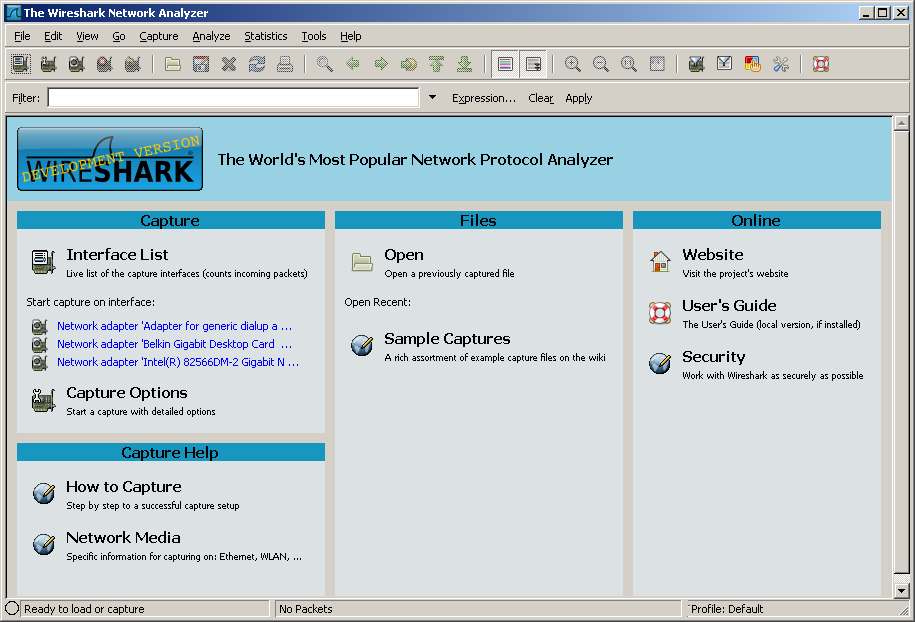
The Wireshark (formerly Ethereal)interface has five major components:

* The **command menus** are standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. 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, i.e., the protocol that is the source or ultimate sink for 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 right-pointing or down-pointing arrowhead 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, TCP or UDP details will also be displayed, which 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 entire contents of the captured frame, in both ASCII and hexadecimal format.
* 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

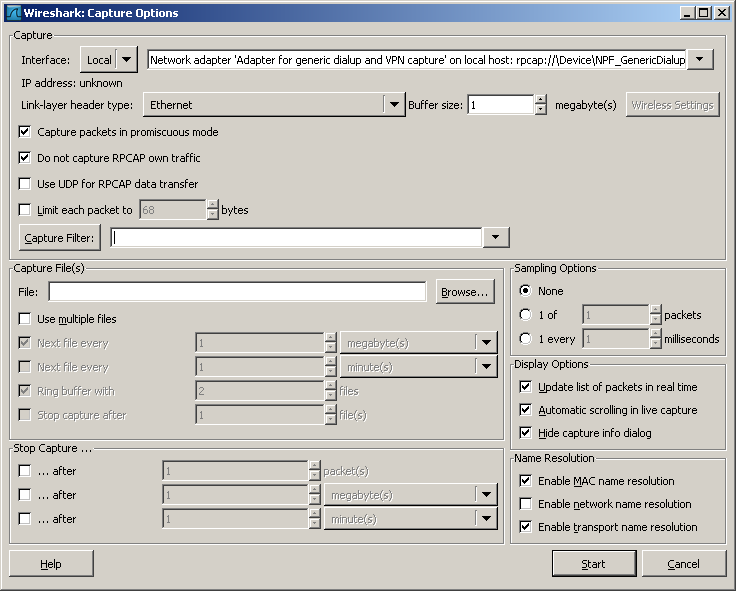
The best way to learn about any new piece of software is to try it out! Do the following

1. Start up your favorite web browser, which will display the default homepage.
2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 3.



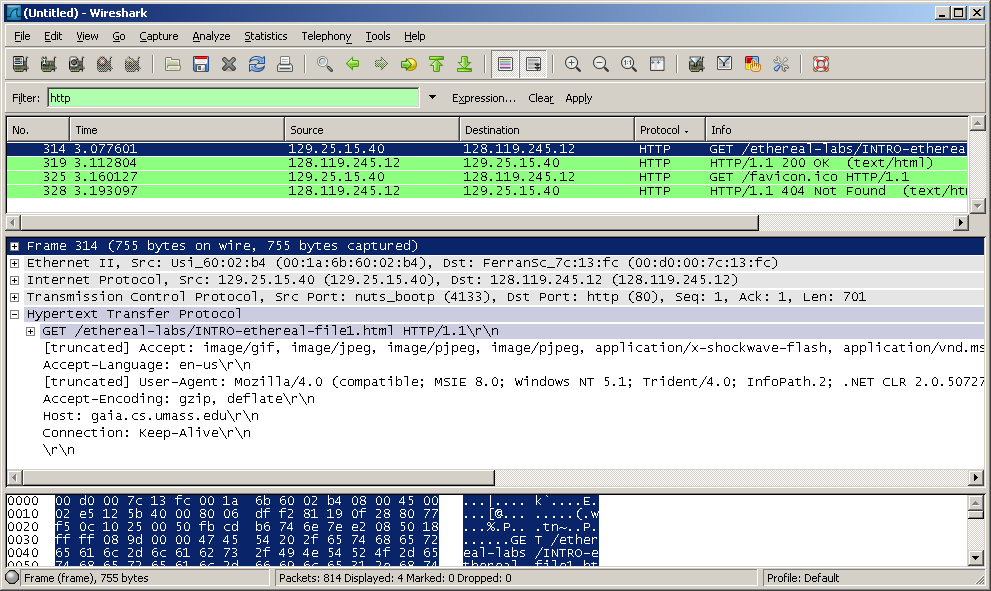
**Figure 3:** Wireshark Startup screen.

1. To begin packet capture, first select Capture Options. This will cause the “Wireshark: Capture Options” window to be displayed, as shown in Figure 4.



**Figure 4:**WiresharkCapture Options Window

1. You can use all of the default values in this window, except **deselect “Capture packets in promiscuous mode**.”The network interfaces (i.e., the physical connections)that your computer has to the network will be shown in the Interface pull down menu at the top of the Capture Options window. (You need to select the network interface that you have on your device, and this may be wired or wireless).In case your computer has more than one active network interface (e.g., if you have both a wireless and a wired Ethernet connection), you will need to select an interface that is being used to send and receive packets (mostly likely the wired interface).   
   After selecting the network interface (or using the default interface chosen by Wireshark), click OK. Packet capture will now begin - all packets being sent from or received by your computer are now being captured by Wireshark!
2. Once you begin packet capture, the data window (like in Figure 2) will start to fill with captured packets.
3. While Wiresharkis running, enter the URL:   
   <http://gaia.cs.umass.edu/ethereal-labs/INTRO-ethereal-file1.html>  
   and have that page displayed in your browser. In order to display this page, your browser will contact the HTTP server at gaia.cs.umass.edu and exchange HTTP messages with the server in order to download this page, as discussed in section 2.2 of the text. The Ethernet frames containing these HTTP messages will be captured by Wireshark.
4. **After your browser has displayed the INTRO-ethereal-file1.html page,STOP PACKET CAPTURE using the icon with a red X on it. It should be the fourth one from the left.**The main Wiresharkwindow should now look similar to Figure 2. You now have packet data that contains all protocol messages exchanged between your computer and other network entities! The HTTP message exchanges with the gaia.cs.umass.edu web server should appear somewhere in the listing of packets captured. But there will be many other types of packets displayed as well (see, e.g., the many different protocol types shown in the *Protocol* column in Figure 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. We’ll learn much more about these protocols as we progress through the text! For now, you should just be aware that there is often much more going on than “meets the eye”!
5. 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 Wiresharkwindow. 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.   
   [This may not work – recent versions of Wiresharkare weak on filtering. Instead, click on the Protocol column heading, to sort frames by protocol, and look for the ones which are HTTP.]
6. 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 gaia.cs.umass.edu 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[[1]](#footnote-2). By clicking on right-pointing and down-pointing arrowsheads to the left side of the packet details window, *minimize* the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed. *Maximize*(expand) the amount information displayed about the HTTP protocol. Your Wiresharkdisplay should now look roughly as shown in Figure 5 (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).



**Figure 5:** HTTP packets from the first run of Wireshark.   
Notice the Filter of ‘http’ is used, and the HTTP GET for the favicon.ico is ignored.

1. You can save the Wireshark packet capture results (File > Save As). This is handy if you want to double check your results later, or want to run the capture on campus, and email it to yourself to finish the lab writeup later.
2. Exit Wireshark.

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:

Questions

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 Wiresharkup and running, and have explored some of its capabilities. Answer the following questions, based on your Wiresharkexperimentation:

1. **List the different protocols that appear in the protocol column in the unfiltered packet-listing window in step 7 above.**

I see 4 protocols: TCP, UDP, HTTP, & NTP

1. **How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received?**

[Time since request 0.017851215 seconds]

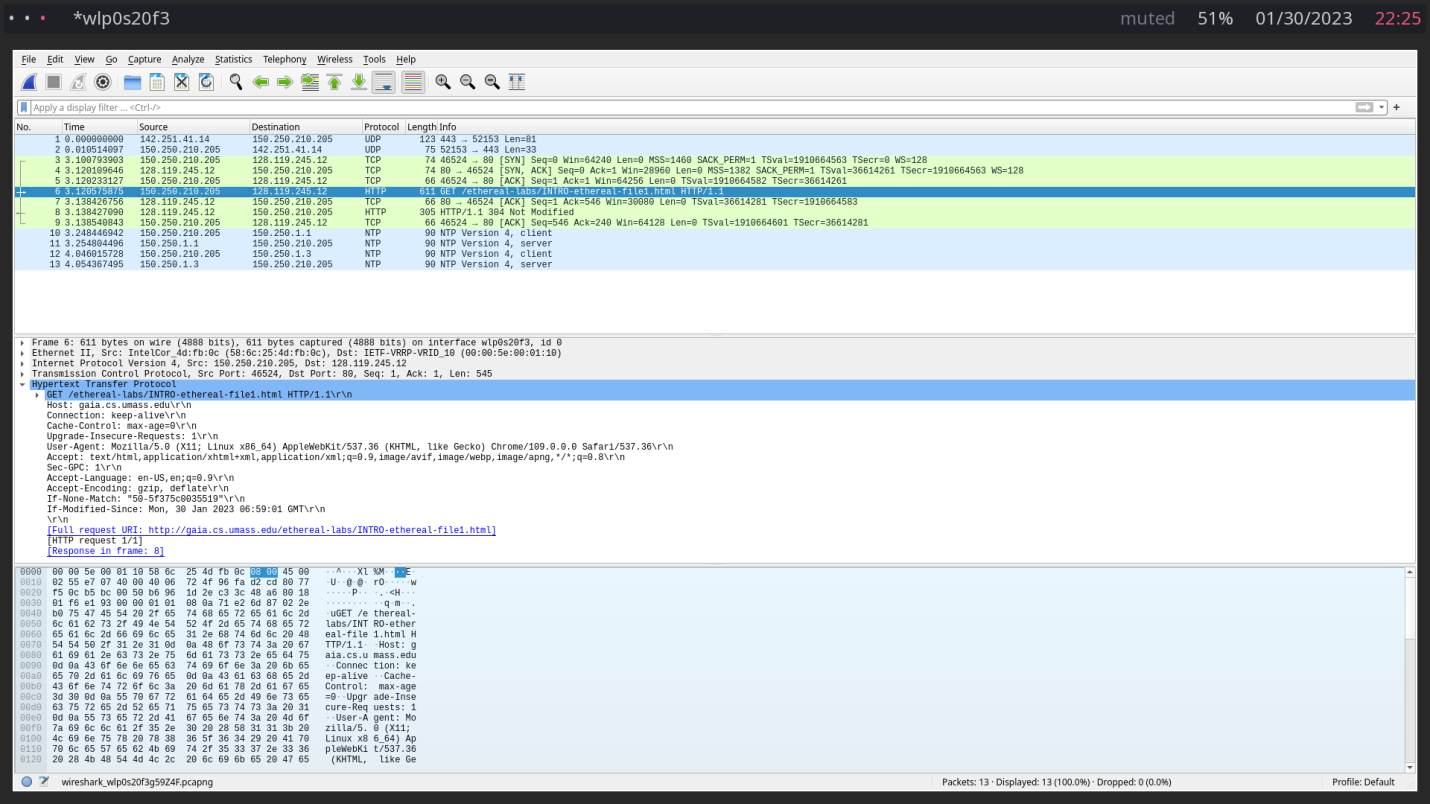
TIP for time of day display in Wireshark: By default, the value of the Time column in the packet-listingwindow is the amount of time, in seconds, since Wiresharktracing began. To display the Time field in time-of-day format, select the Wireshark*View* pull down menu, then select Time *Display Format*, then select *Time-of-day*.

1. **What is the Internet address (IP address) of the gaia.cs.umass.edu (also known as www-net.cs.umass.edu)? What is the Internet address of your computer?**

Source IP for HTTP GET: 150.250.210.205

Destination IP for HTTP GET: 128.119.245.12

1. **Show a screen capture of the HTTP GET command expanded in step 9.**

****

1. **Examine the HTTP portion of the GET command by double clicking on the entry to bring up a sub-window. Expand the HTTP section and find the version of HTTP that the GET command used from your PC. What version are you running?**

HTTP version 1.1

What To Hand In

From this lab you should be turning in:

* From Part 1, the answers to the five questions. Be sure to answer all parts of each question.
* From Part 2, the answers to the five questions. One question includes a screen capture.

Turn in you lab work (MS Word) online via our course shell link by the due date. You are welcome to include screenshot in your answer.

1. Recall that the HTTP GET message that is sent to the gaia.cs.umass.edu web server is contained within a TCP segment, which is contained (encapsulated) in an IP datagram, which is encapsulated in an Ethernet frame. If this process of encapsulation isn’t quite clear yet, review section 1.7 in the text [↑](#footnote-ref-2)