Computer Network Lab (1)

This lab will go over two parts:

Part1. Wireshark + HTTP Basics

- Install and run Wireshark; capture and filter HTTP traffic.
- Analyze a simple HTTP GET/Response; measure request–response time.
- Perform a conditional GET (cache validation).

Part2. DNS & HTTP Name Resolution

- Capture DNS queries and responses preceding an HTTP fetch.
- Inspect transaction ID, record types (A/AAAA), TTLs, and response times.
- Correlate DNS answer IP with the IP used by the subsequent HTTP connection.

What to turn in for this lab:

1. Each part of the lab has a list of questions regarding your work in that part of the lab. You are required to give a report in **PDF** format including your answer to the questions given in the lab.
2. In addition to your answers, you are required to upload an unlisted video to Youtube (no more than 15 minutes) for each of the 2 parts, of you demoing the lab and demonstrating your knowledge of the material by explaining your answers to the lab questions. You need to provide the Youtube link on your report.

Part 1: Introduction to Wireshark

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. In the Wireshark labs you'll be doing in this course, you'll be running various network applications in different scenarios using your own computer. 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.

In the first part of this lab, you'll get acquainted with Wireshark, and make some simple packet captures and observations.

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

Computer Networking: a Top-Down Approach, 8th Edition, Kurose and Ross.

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 email 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 over a given interface (link layer, such as Ethernet or WiFi). 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 or an 802.11 WiFi radio. Capturing all link-layer frames thus gives you all messages sent/received across the monitored link from/by all protocols and applications executing in your computer.

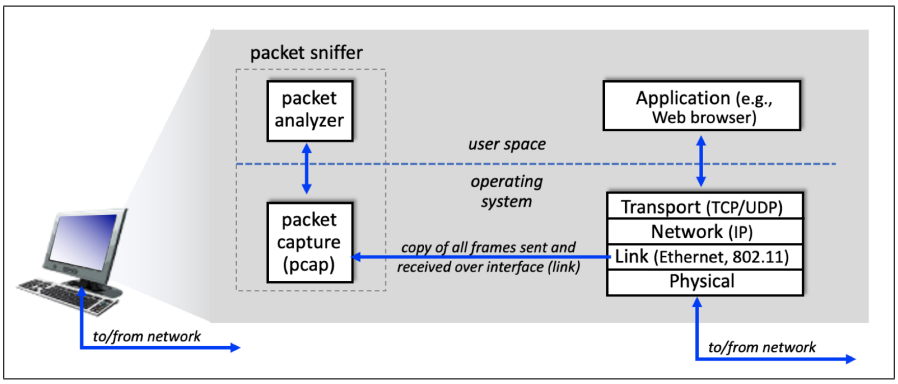

Figure 1: packet sniffer structure

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 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,".

We will be using the Wireshark packet sniffer for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer. Also, technically speaking, Wireshark captures link-layer frames as shown in Figure 1, but uses the generic term "packet" to refer to link-layer frames, network-layer datagrams, transport-layer

segments, and application-layer messages, so we'll use the less-precise "packet" term here to go along with Wireshark convention). Wireshark is a free network protocol analyzer that runs on Windows, Mac, and Linux/Unix computers. It's an ideal packet analyzer for our labs – it is stable, has a large user base and well-documented support that includes a user-guide, man pages, and a detailed FAQ, rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, serial (PPP), 802.11 (WiFi) wireless LANs, and many other link-layer technologies.

Getting Wireshark

In order to run Wireshark, you'll need to have access to a computer that supports both Wireshark and the *libpcap* or *WinPCap* packet capture library. The *libpcap* software will be installed for you, if it is not installed within your operating system, when you install Wireshark. See current stable release of Wireshark for a list of supported operating systems and download sites.

Download and install the Wireshark software:

Go to Wireshark download page and download and install the Wireshark binary for your computer. The Wireshark FAQ has a number of helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

Download tutorial for ubuntu: If you are running Linux you can refer to Install and Use Wireshark on Ubuntu Linux page.

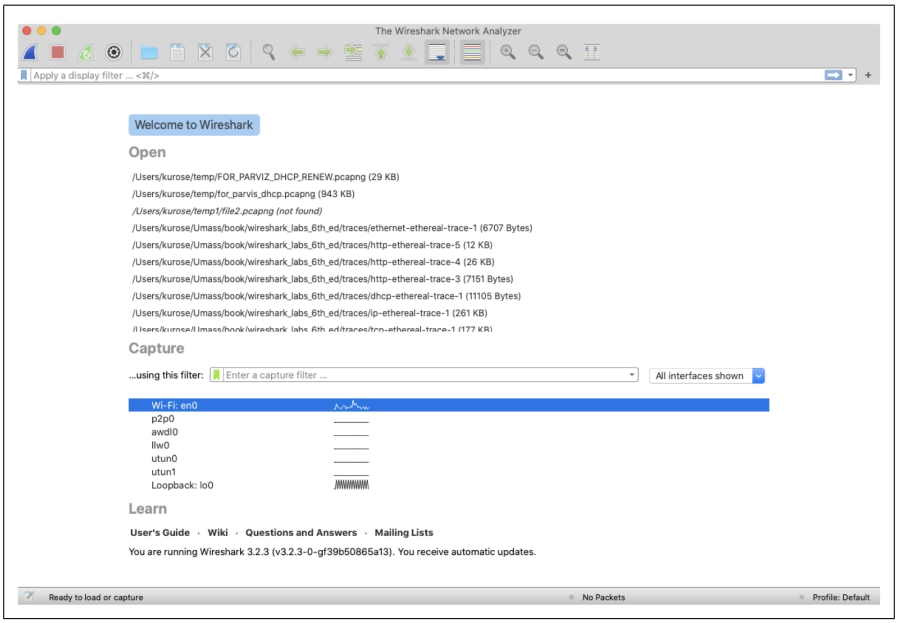

Figure 2: Initial Wireshark Screen

Running Wireshark

When you run the Wireshark program, you'll get a startup screen that looks something like the screen in Figure 2. Different versions of Wireshark will have different startup screens – so don't panic if yours doesn't look exactly like the screen above! The Wireshark documentation states "As Wireshark runs on many different platforms with many different window managers, different styles applied and there are different versions of the underlying GUI toolkit used, your screen might look different from the provided screenshots. But as there are no real differences in functionality these screenshots should still be well understandable." Well said.

There's not much that's very interesting on this screen. But note that under the Capture section, there is a list of so-called interfaces. The Mac computer we're taking these screenshots from has just one interface – "Wi-Fi en0," (shaded in blue in Figure 2) which is the interface for Wi-Fi access. All packets to/from this computer will pass through the Wi-Fi interface, so it's here where we'll want to capture packets. On a Mac, double click on this interface (or on another computer locate the interface on startup page through which you are getting Internet connectivity, e.g., mostly likely a WiFi or Ethernet interface, and select that interface).

Let's take Wireshark out for a spin! If you click on one of these interfaces to start packet capture (i.e., for Wireshark to begin capturing all packets being sent to/from that interface), a screen like the one below will be displayed, showing information about the packets being captured. Once you start packet capture, you can stop it by using the Capture pull down menu and selecting Stop (or by clicking on the red square button next to the Wireshark fin in Figure 2).

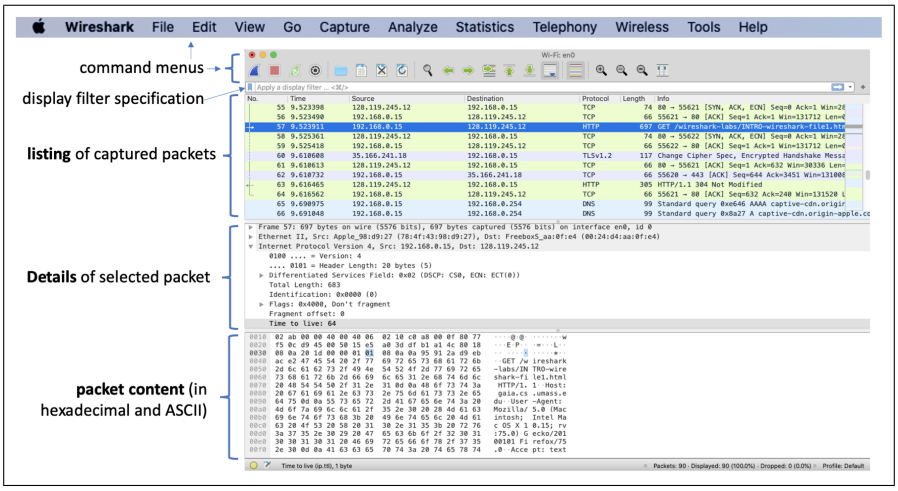

Figure 3: Wireshark window, during and after capture

This looks more interesting! The Wireshark interface has five major components:

- The **command menus** are standard pulldown menus located at the top of the Wireshark window (and on a Mac at the top of the screen as well; the screenshot in Figure 3 is from a Mac). 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; note that 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 (assuming the packet was sent/received over an Ethernet interface) 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 plus/minus boxes or right/downward-pointing triangles 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! We'll assume that your computer is connected to the Internet via a wired Ethernet interface or a wireless 802.11 WiFi interface. Do the following:

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

2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2. Wireshark has not yet begun capturing packets.

3. To begin packet capture, select the Capture pull-down menu and select **Interfaces**. This will cause the "Wireshark: Capture Interfaces" window to be displayed (on a PC) or you can choose **Options** on a Mac. You should see a list of interfaces, as shown in Figures 4 (Windows) and 5 (Mac).

4. You'll see a list of the interfaces on your computer as well as a count of the packets that have been observed on that interface so far. Select the interface on which you want to begin packet capture (typically Wi-Fi or Ethernet) and click **Start**. Packet capture will now begin — Wireshark is now capturing all packets being sent/received from/by your computer!

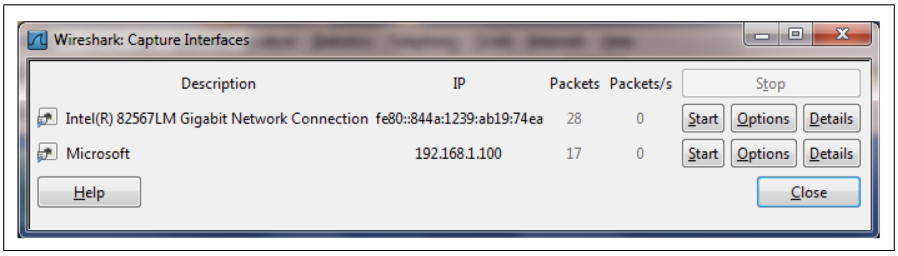

Figure 4: Wireshark Capture interface window, on a Windows computer

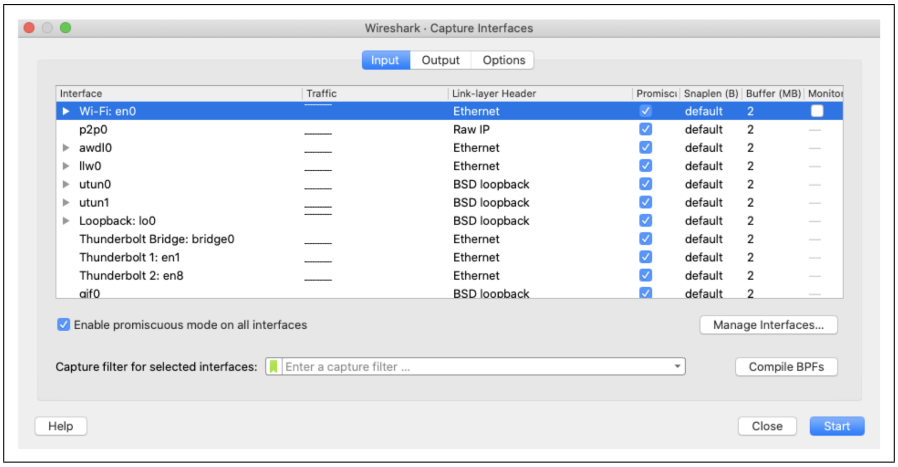

Figure 5: Wireshark Capture interface window, on a Mac computer

5. Once you begin packet capture, a window similar to Figure 3 will appear. This window shows the packets being captured. By selecting the Capture pull-down menu and selecting **Stop**, or by clicking on the red Stop square, you can stop packet capture. But don't stop packet capture yet. Let's capture some interesting packets first. To do so, we'll generate some network traffic using a web browser, which will use the HTTP protocol to download content from a website.

This site intentionally serves *plain HTTP* (not HTTPS), so your browser will exchange HTTP messages with the server.

6. While Wireshark is running, visit http://cybernetlab.org and have that page displayed in your browser. This site intentionally serves *plain HTTP* (not HTTPS), so your browser will exchange HTTP messages with the server.

Tip: If you don't see HTTP packets, try a private/incognito window or a different browser.

7. After your browser has displayed the page, stop Wireshark packet capture. The main Wireshark window should now show live packet data that contains protocol messages exchanged between your computer and other network entities! The HTTP message exchanges with the server (cybernetlab.org) 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). 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`) or just hit Return. This will cause only HTTP messages to be displayed in the packet-listing window. Figure 6 shows a filtered example. Note also that in the Selected packet details window, you can choose to show detailed content for the Hypertext Transfer Protocol application message that was found within the TCP segment, that was inside the IP datagram that was inside the Ethernet/Wi-Fi frame. Focusing on content at a specific message/segment/datagram/frame level lets us focus on just what we want to look at (in this case HTTP messages).

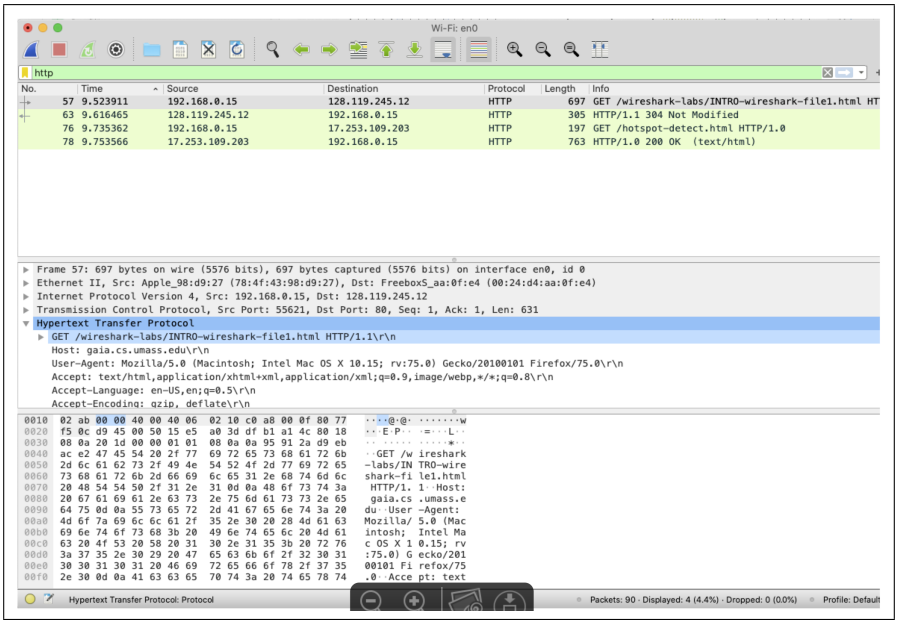

Figure 6: HTTP request and response shown in Wireshark

9. Find the HTTP `GET` message that was sent from your computer to the *server you accessed* (e.g., cybernetlab.org). (Look for an HTTP `GET` message in the listing of captured packets that shows `GET` followed by the URL you visited.) 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 '+'/'–' (or the arrowheads) to the left of each layer, minimize Frame/Ethernet/IP/TCP information and expand HTTP. Your Wireshark display should now look roughly like Figure 6.

10. Exit Wireshark.

Answer the following questions:

The goal of this first part of this lab is to introduce you to Wireshark. The following questions demonstrate that you've been able to get Wireshark up and running and have explored some of its

capabilities.

1. List 3 different protocols that appear in the protocol column in the *unfiltered* packet-listing window.

2. How long did it take from when the HTTP `GET` message was sent until the first `HTTP 200 OK` reply was received? (By default, the Time column shows seconds since capture start. To display time-of-day, use *View* → *Time Display Format* → *Time of Day*.)

3. What is the Internet address of your computer? What is the Internet address of the *server you accessed*?

4. Print the two HTTP messages (GET and OK) referred to above. Select *File* → *Print*, choose "Selected Packet Only" and "Print as displayed," then click OK.

5. What was the HTTP status code in the HTTP response?

6. Refresh the website (http://cybernetlab.org), is the HTTP response code still the same? If not, what is the new HTTP status code and why is there a difference?

7. Which field in the HTTP request indicates the OS used? And what is the value of that field?

8. Which field in the HTTP request indicates the browser used? And what is the value of that field?

For the following questions, continue packet capturing and click on the Login button on the webpage

1. Were there any additional HTTP GET requests? If so, what pages were fetched?
2. Put in your Auburn username, and password **MUST** be **notsafepassword** and click on Submit. After doing this, what type of HTTP request is sent? [GET, POST, PUT, DELETE, PATCH]
3. Do you see the credentials that you have just put in on Wireshark? If so, in which packet and which field do you see them?

Part 2a: DNS & Name Resolution

Before an HTTP connection to a domain, the client typically performs a DNS lookup to resolve the domain name to an IP address.

Do the following

1. Start a fresh capture and set the display filter to `dns`.
2. In the browser, visit http://cybernetlab.org. This triggers at least one DNS query (A/AAAA).
3. Stop the capture. If you see no query due to caching, try a private/incognito window or a hostname you haven't visited recently.

Answer the following (DNS)

Q1. What is the **Transaction ID** of the DNS query and corresponding response?

Q2. Which **record types** are returned (A, AAAA)? List the **answer IP(s)**.

Q3. What is the **TTL** for the returned record(s)?

Q4. Measure the **DNS response time**: response timestamp minus query timestamp.

Q5. Verify that the **HTTP connection** later uses one of the returned IP addresses (clear the display filter and check the IP used by the HTTP packet pair).

Part 2b: HTTP

Having gotten our feet wet with the Wireshark packet sniffer in the first part of the lab, we're now ready to use Wireshark to investigate protocols in operation. In this part, we'll explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message formats, retrieving large HTML files, retrieving HTML files with embedded objects, and HTTP authentication and security. Before beginning these labs, you might want to review HTTP section from the lectures.

The Basic HTTP GET/response interaction

Let's begin our exploration of HTTP by downloading a very simple HTML file - one that is very short, and contains no embedded objects.

Do the following:

1. Start a new capture. Optional display filter: `http`.
2. Visit http://cybernetlab.org and wait for the page to load.

3. Stop the capture.

Answer the following (HTTP Basic):

Q1. What HTTP version does the `GET` use? What version is reported in the server response?

Q2. What is the **request–response latency**? (time from `GET` to first `200 OK` packet)

Q3. What is your host IP and the server IP used for the HTTP exchange?

Q4. What **headers** are present in the request and in the response (list at least 3 from each)?

Your Wireshark window should look similar to the window shown in Figure 7. If you are unable to run Wireshark on a live network connection, you can download a packet trace that was created when the steps above were followed.

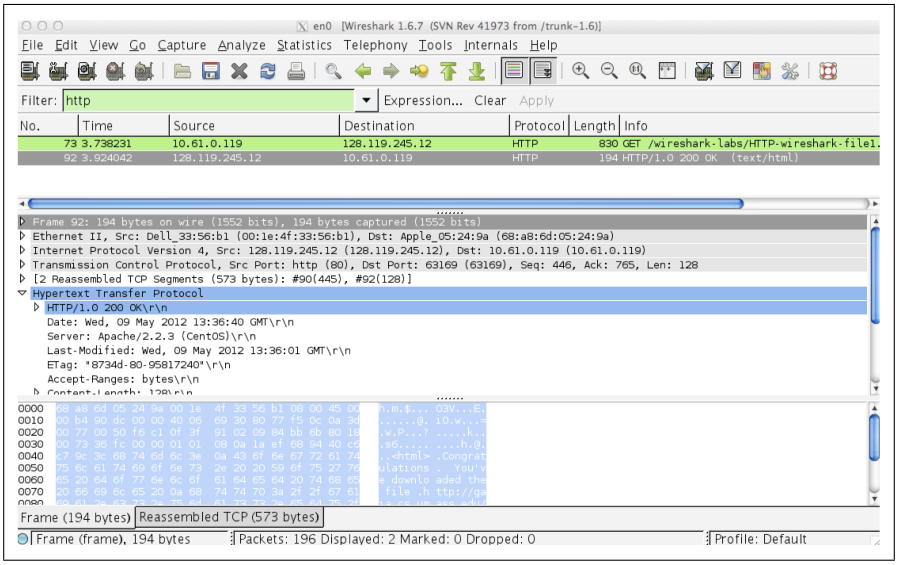

Figure 7: Wireshark Display after html file has been retrieved by your browser

The example in Figure 7 shows two HTTP messages in the packet-listing window: the **GET** request (from your browser to the server) and the **HTTP 200 OK** response. Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well. We want to minimize the amount of non-HTTP data displayed (we're interested in HTTP here, and will be investigating these other protocols in later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle (which means there is hidden, undisplayed information), and the HTTP line has a minus sign or a down-pointing triangle (which means that all information about the HTTP message is displayed).

By looking at the information in the HTTP GET and response messages, answer the following questions. When answering the following questions, you should print out the GET and response messages (refer to the first part of this Wireshark lab for an explanation of how to do this) and indicate where in the message you've found the information that answers the following questions.

Q1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

Q2. What languages (if any) does your browser indicate that it can accept to the server?

Q3. What is the IP address of your computer? What is the IP address of the server you accessed (e.g., http://cybernetlab.org)?

Q4. What is the status code returned from the server to your browser?

Q5. When was the HTML file that you are retrieving last modified at the server?

Q6. How many bytes of content are being returned to your browser?

Q7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

The HTTP CONDITIONAL GET/response interaction

Recall that most web browsers perform object caching and thus perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty. (To do this under Firefox, select Tools → Clear Recent History and check the Cache box, or for Internet Explorer, select Tools → Internet Options → Delete File; these actions will re-move cached files from your browser's cache.) Now do the following:

1. Clear your browser cache (or simply open a new private/incognito window).
2. Start capture and visit http://cybernetlab.org.
3. Without clearing the cache, reload the same page two more times while capture is running.
4. Stop the capture and apply the filter `http`.

Answer the following (Conditional):

Q1. Inspect the headers in the second and third requests. Do you observe `If-Modified-Since` or `If-None-Match`? Copy the line if present.

Q2. What status code is returned to the cached request (`304 Not Modified` vs `200 OK`)? Explain the meaning of the code.

Q3. Compare payload sizes between the first request and the cached request(s). Did the server resend the entire page or did the browser use its cache?

Answer the following questions:

Q4. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

Q5. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

Q6. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?

Q7. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let's next see what happens when we download a long HTML file.

Do the following:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer
3. Click here to display the rather lengthy US Bill of Rights into your browser.
4. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall that the HTTP response message consists of a status line, followed by header lines, followed by a blank line, followed by the entity body. In the case of our HTTP GET, the entity body in the response is the entire requested HTML file. In our case here, the HTML file is rather long, and at 4500 bytes is too large to fit in one TCP packet. The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment. In recent versions of Wireshark, Wireshark indicates each TCP segment as a separate packet, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the "TCP segment of a reassembled PDU" in the Info column of the Wireshark display. Earlier versions of Wireshark used the "Continuation" phrase to indicated that the entire content of an HTTP message was broken across multiple TCP segments. We stress here that there is no "Continuation" message in HTTP!

Answer the following questions:

Q8. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?

Q9. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?

Q10. What is the status code and phrase in the response?

Q11. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

HTML Documents with Embedded Objects

Now that we've seen how Wireshark displays the captured packet traffic for large HTML files, we can look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (in the example below, image files) that are stored on another server(s).

Do the following:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer
3. Click here to display a short HTML file with two images in your browser. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. As discussed in the textbook, your browser will have to retrieve these logos from the indicated web sites.
4. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

Answer the following questions:

Q12. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?

Q13. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

HTTP Authentication

Finally, let's try visiting a web site that is password-protected and examine the sequence of HTTP message exchanged for such a site. The URL here is password protected. The username is "network-labs" (without the quotes), and the password is "comp4320networks" (again, without the quotes). So let's access this "secure" password-protected site.

Do the following:

1. Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
2. Start up the Wireshark packet sniffer
3. Click here to open an html page in your browser and type the requested user name and password into the pop up box.
4. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

   Now let's examine the Wireshark output.

Answer the following questions:

Q14. What is the server's response (status code and phrase) in response to the initial HTTP GET mes-sage from your browser?

Q15. When your browser's sends the HTTP GET message for the second time, what new field is includ-ed in the HTTP GET message?

The username (network-labs) and password (comp4320networks) that you entered are encoded in the string of characters (bmV0d29yay1sYWJzOmNvbXA0MzIwbmV0d29ya3M=) following the "Authorization: Basic" header in the client's HTTP GET message. While it may appear that your username and pass-word are encrypted, they are simply encoded in a format known as Base64 format. The username and password are not encrypted! To see this, go to here and enter the base64-encoded string bmV0d29yay1sYWJz and decode. Voila! You have translated from Base64 encoding to ASCII encoding, and thus should see your username! To view the password, enter the remainder of the string OmNvbXA0MzIwbmV0d29ya3M= and press decode. Since anyone can download a tool like Wireshark and sniff packets (not just their own) passing by their network adaptor, and anyone can translate from Base64 to ASCII (you just did it!), it should be clear to you that simple passwords on WWW sites are not secure unless additional measures are taken.

End of Lab (1)