Lab Assignment 1

Application Layer (Introduction to Wireshark)

Due Date: Friday, January 31st, 2025, by 11:59 PM

Instructors: Dr. Sandra Céspedes, Dr. Abdelhak Bentaleb & Dr. Aiman Hanna

Submission Deadline

31 January 2025 at 11:59 pm. A 5-mark penalty will be imposed on late submission (late submission refers to submission or re-submission after the deadline). The submission link will close on **2 February 2025 (Sunday) at 11:59 pm**, and no late submissions will be accepted afterwards.

Objectives

At the end of this lab, students will work **individually or in groups of up to two** to demonstrate the following:

1. Installing Wireshark and learning how it operates.
2. Understanding packet sniffers and how they capture and analyze network traffic.

This lab is worth **50 marks** in total. **You need at least 35 marks to pass this lab.**

Setup

You should use your local machine for all tasks in this assignment. The tools you need (on your local machine or lab machine) are:

1. Google Chrome or any other modern browser
2. Wireshark

Submission

Create a **single PDF file** that contains your answers and submit it to the corresponding assignment folder in Moodle. Name your file as ≪Student number≫.pdf, where ≪Student number≫ refers to your student ID number. Include both IDs if working in a group.

Plagiarism Warning

You are free to discuss this assignment with your friends. **However, you should refrain from sharing your answers**. We highly recommend that you attempt this assignment on your own and figure things out along the way as many resources are available online.

We employ a zero-tolerance policy against plagiarism. If a suspicious case is found, the student would be asked to explain his/her answers to the evaluator in person. The confirmed breach may result in a zero mark for the assignment and further disciplinary action from the department.

Question & Answer

If you have any doubts about this assignment, please consult **directly** with the lab instructors. However, the lab instructors will NOT debug issues for students. **Please note that your professor will not answer any lab-related questions during lectures or office hours. Therefore, make sure to attend the lab sessions if you have any questions.**.

Theory: Packet Sniffers

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 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). As you know, 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.

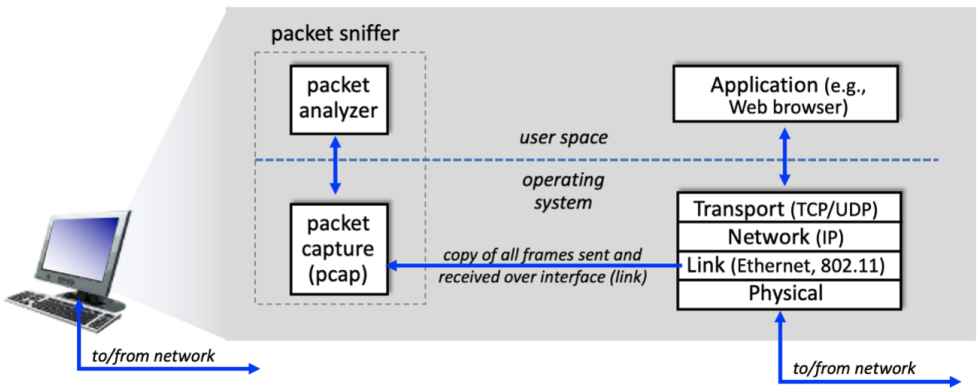

Figure 1: packet sniffer structure.

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 [`http://www.wireshark.org/`] for this lab, 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 on your computer. Also, technically speaking, Wireshark captures link-layer frames as shown in Figur 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 (`http://www.wireshark.org/docs/wsug_html_chunked/`), man pages (`http://www.wireshark.org/docs/man-pages/`), and a detailed FAQ (`http://www.wireshark.org/faq.html`), 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.

Procedures (Getting Started)

Get 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 `http://www.wireshark.org/download.html` for a list of supported operating systems and download sites.

Download and install the Wireshark software:

- Go to `http://www.wireshark.org/download.html` 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.
- You may need to disable anti-virus protection software before your own IP address will show up in captured data.
- You should be connected to an Ethernet connection. If you only have WiFi, you'll need to figure out how to set your WiFi physical layer into monitor mode. Failure to follow this instruction will mean you only see traffic originating or being sent to your own computer, which is sub-optimal for these labs.

Run Wireshark

When you run the Wireshark program, you'll get a startup screen that looks something like the screen below. Different versions of Wireshark will have different startup screens – so don't panic if yours doesn't look exactly like the screen below! 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 the startup page through which you are getting Internet connectivity, e.g., most likely a WiFi or Ethernet interface, and select that interface).

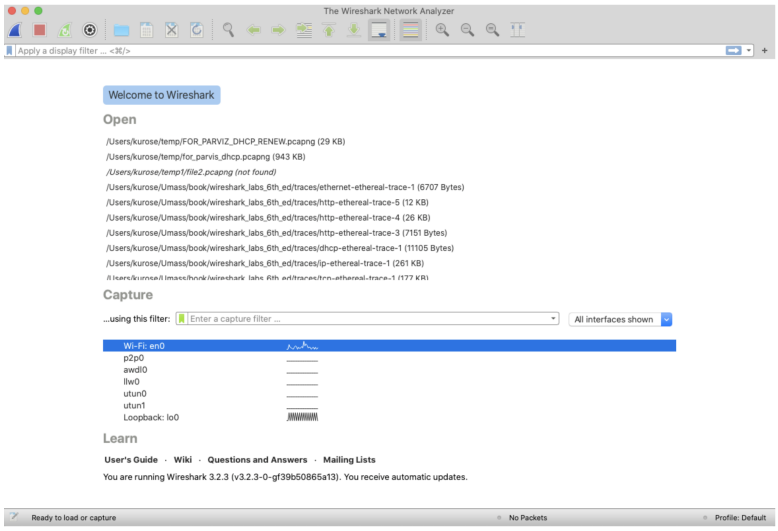

Figure 2: Initial Wireshark Screen.

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 in Figure 3 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)[1].

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

[1]If you are unable to run Wireshark, you can still look at packet traces that were captured offline. You can download the zip file from `https://comp445.github.io/wireshark-labs/wireshark-traces.zip` and extract the trace file intro-wireshark-trace-1.pcap. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the File pull down menu, choosing Open, and then and then selecting the intro-wireshark-trace trace file. The resulting display should look similar to Figure 3 and 5. (The Wireshark user interface displays just a bit differently on different operating systems, and in different versions of Wireshark).

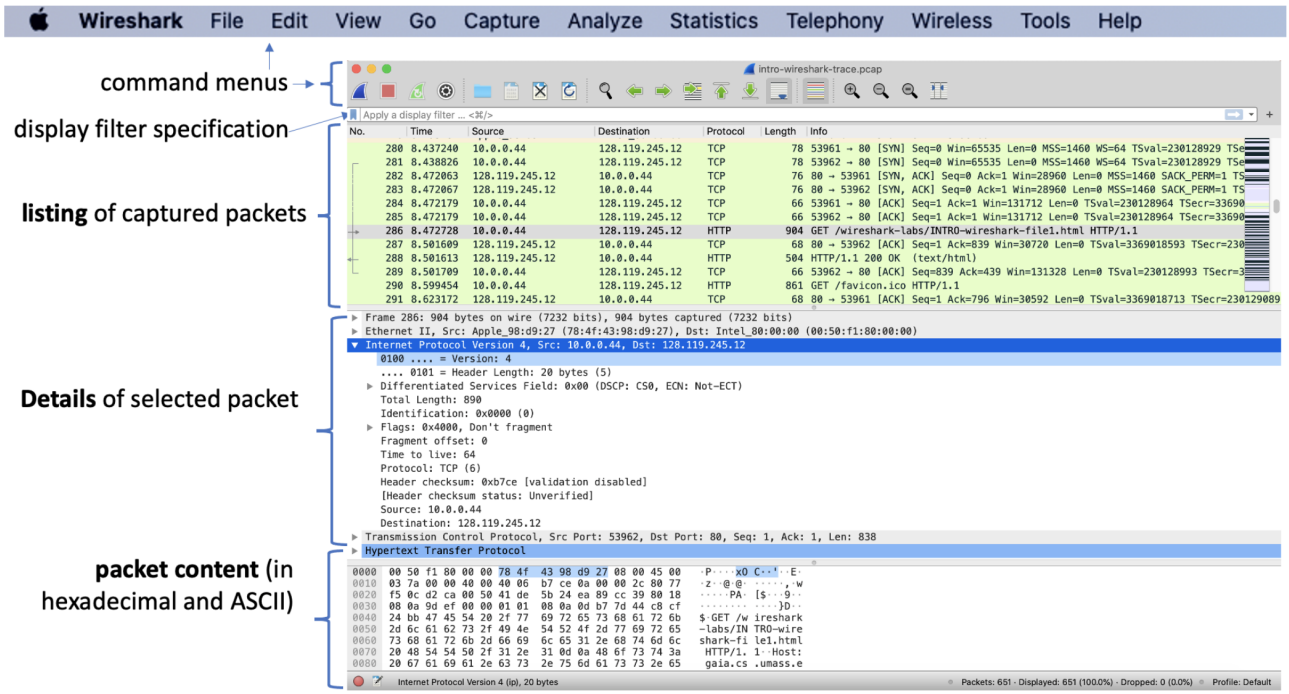

Figure 3: Wireshark window, during and after capture.

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.

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:

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

- 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.

- 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 4a (Windows) and 4b (Mac).

- 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. On a Windows machine, click on *Start* for the interface on which you want to begin packet capture (in the case in Figure 4a, the Gigabit network Connection). On a Windows machine, select the interface and click Start at the bottom of the window). Packet capture will now begin - Wireshark is now capturing all packets being sent/received from/by your computer!

- Once you begin packet capture, a window similar to that shown in Figure 3 will appear. This window shows the packets being captured. By selecting *Capture* pulldown 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 need to generate some network traffic. Let's do so using a web browser, which will use the HTTP protocol that we will study in detail in class to download content from a website.

- While Wireshark is running, enter the URL: `http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-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 class. The Ethernet or WiFi frames containing these HTTP messages (as well as all other frames passing through your Ethernet or WiFi adapter) will be captured by Wireshark.

- After your browser has displayed the INTRO-wireshark-file1.html page (it is a simple one-line of congratulations), stop Wireshark packet capture by selecting stop in the Wireshark capture window. The main Wireshark window should now look similar to Figure 3. You now have live 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 3). 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 "meet's the eye"!

- Type in "http" (without the quotes, and in lower case – all protocol names are in lower case in Wireshark, and make sure to press your enter/return key) 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 message to be displayed in the packet-listing window. Figure 5 below shows a screenshot after the http filter has been applied to the packet capture window shown earlier in Figure 3. Note also that in the Selected packet details window, we've chosen to show detailed content for the Hypertext Transfer Protocol application message that was found within the TCP segment, that was inside the IPv4 datagram that was inside the Ethernet II (WiFi) frame. Focusing on content at a specific message, segment, datagram and frame level lets us focus on just what we want to look at (in this case HTTP messages).

- Find the HTTP GET message that was sent from your computer to the gaia.cs.umass.edu HTTP server. (Look for an HTTP GET message in the "listing of captured packets" portion of the Wireshark window (see Figures 3 and 5) that shows "GET" followed by the server URL that you entered. 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 '+' and '-' and right-pointing and down-pointing arrowheads to the left side of the packet details window, *minimize* the amount of Frame, Ethernet, Internet Protocol, and Transmission

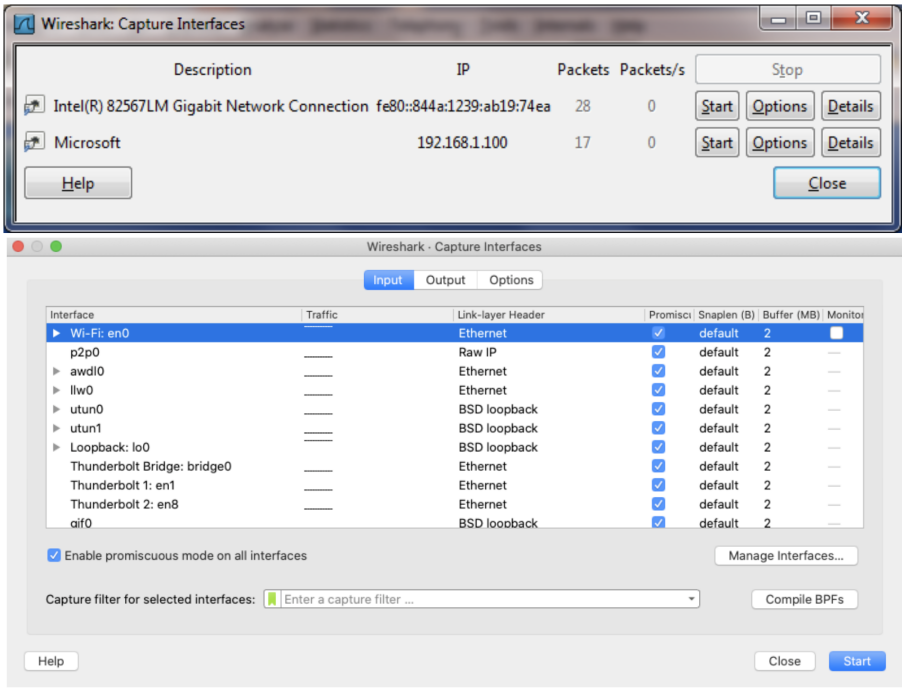

Figure 4: (a) Wireshark Capture interface window, on a Windows computer. (b) Wireshark Capture interface window, on a Mac computer

Control Protocol information displayed. *Maximize* the amount of information displayed about the HTTP protocol. Your Wireshark display 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).

- Exit Wireshark

Congratulations! You've now completed setting up an important network engineering tool and learning a bit about its operation.

Turn-in

The goal of this first lab is primarily to introduce you to Wireshark. The following questions will demonstrate that you have been able to get Wireshark up and running, and have explored some of its capabilities. Answer the following questions, based on your Wireshark experimentation. If you're unable to run Wireshark on a live network connection, you may download a packet trace file that was captured while following the steps above [2]. Make sure you document all your answers since you will need/discuss them throughout the demo with the marker.

1. **[5 marks]** List up to five different protocols that appear in the protocol column in the unfiltered packet-listing window as shown when requesting `http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html`. As I don't have control over the data flowing over your network at the time of your lab, I don't know exactly how many and what protocols those will be. I do expect that you have a bunch (if less than 2, please look harder). Just list out those that you see, but don't bother to list more than 5.

[2] You can download the zip file `https://comp445.github.io/wireshark-labs/wireshark-traces.zip` and extract the trace file intro-wireshark-trace-1.pcap. This trace file can be used to answer these Wireshark lab questions without actually capturing packets on your own. Each trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the File pull-down menu, choosing Open, and then selecting the trace file name.

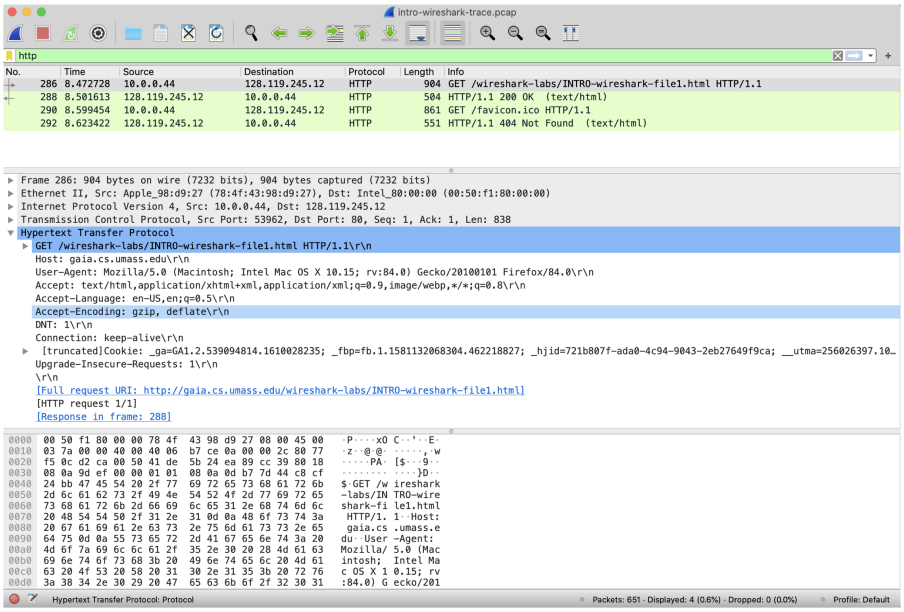

Figure 5: Looking at the details of the HTTP message that contained a GET when requesting INTRO-wireshark-file1.html

2. **[10 marks]** How long did it take from when the HTTP GET message was sent until the HTTP OK reply 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. (If you want 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. **[5 marks]** What is the Internet address of the `gaia.cs.umass.edu`? What is the Internet address of your computer (This might be a private address if you are behind a NAT device. No worries, we'll learn about that later) or (if you are using the trace file) the computer that sent the HTTP GET message? Include a screenshot and describe where you got the data to answer this question.

4. **[5 marks]** Expand the information on the Transmission Control Protocol (TCP) for this packet (from question 3) in the Wireshark "Details of selected packet" window (see Figure 3 in the lab writeup) so you can see the fields in the TCP segment carrying the HTTP message. What is the destination port number (the number following "Dest Port:" for the TCP segment containing the HTTP request) to which this HTTP request is being sent? To answer this question, you'll need to select the TCP packet containing the HTTP GET request (hint: this is packet number 286). The purpose of this question is to familiarize you with using Wireshark's "Details of selected packet window"; see Figure 3. To do this, click on Packet 286 (your screen should look similar to Figure 3). To answer this question, then look in the "Details of selected packet" window toggle the triangle for HTTP (your screen should then look similar to Figure 5).

5. **[20 marks]** How many packets did you capture (total of all protocols, not just HTTP)? Now, use display filters to determine how many packets contain your ip address (hint: Use `ip.addr` instead of the clumsy `ip.src` or `ip.dst` format). What is this filter you used? Now, reverse the filter to determine how many packets don't contain your ip address. See any problems here? If not, you've already figured out the point of this question, so explain how you did so. If so, how can this problem be fixed? What are the appropriate display filters to use? How does Wireshark warn you of such a problem? (This is an important detail to remember about Wireshark. Please ensure you've discussed the problem well enough so that the marker can ensure you explore it thoroughly. If your numbers show you don't have a problem, then figure out how you might reverse the filter in such a way as to cause a problem.)

6. **[5 marks]** Print the two HTTP messages (GET and OK) referred to in question 2 above.

IMPORTANT: Demo

A demo for about 5 to 10 minutes will take place with the marker. THERE WILL BE A DEADLINE TO BOOK YOUR DEMO SLOT, AND YOU MUST BOOK YOUR DEMO PRIOR TO THAT TIME. You (or both members if working in a group) must attend the demo and be able to explain their program to the marker. Different marks may be assigned to teammates based on this demo. The schedule of the demos will be determined and announced by the markers, and students must reserve a time slot for the demo (only one time-slot per group; or zero mark is given).

Now, please read very carefully:

- If you fail to demo, a No Pass mark is assigned regardless of your submission.
- If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty will be applied, which may very well result in you not passing the assignment.
- Failing to demo at the second appointment will result in a No Pass mark, and no more chances will be given under any conditions.

EXTREMELY IMPORTANT: Please notice very carefully the warning/disclaimer at the start of the assignment concerning the publication of the assignment, its text, or its solution (i.e. by having the solution publicly online, on GitHub, etc.). This will immediately constitute an academic misconduct, and such act will not be tolerated even after graduation.