New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Course Title: Data Communication & Networks **Course Number:** g22.2662-001
Instructor: Jean-Claude Franchitti **Session:** 6

Assignment #6

I. Due

Monday November 18, 2013, at the beginning of class.

II. Objectives

1. See protocols in action.

III. References

1. Slides and handouts posted on the course Web site
2. Textbook chapters as applicable

IV. Software Required

1. Wireshark Packet Sniffer and Packet Capture Library (see section V below).
2. Microsoft Word.
3. Win Zip as necessary.

V. Assignment

Preamble and Disclaimer:

> As noted on the corresponding SourceForge site, the Ethereal development team switched names from Ethereal to Wireshark in May 2006 due to trademark issues (see http://www.wireshark.org/faq.html#q1.2 for more details on this). Incidentally, some people pronounce the name Ethereal as "ether-real," while others pronounce it "e-thir-E-al," as in the English word ethereal, which means ghostly or insubstantial. The Ethereal name's origin comes from the Ethernet protocol, a link-level protocol that is studied extensively in Chapter 5 of the textbook, and in the class labs.

1. Wireshark Lab - 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. This can be done in simulated scenarios or in a "real" network environment such as the Internet. The Java applets that accompany the textbook take the first approach. In the 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 "live" labs in this class. 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 class discussions (and corresponding material covered in the textbook used for the class) 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 Figure 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.

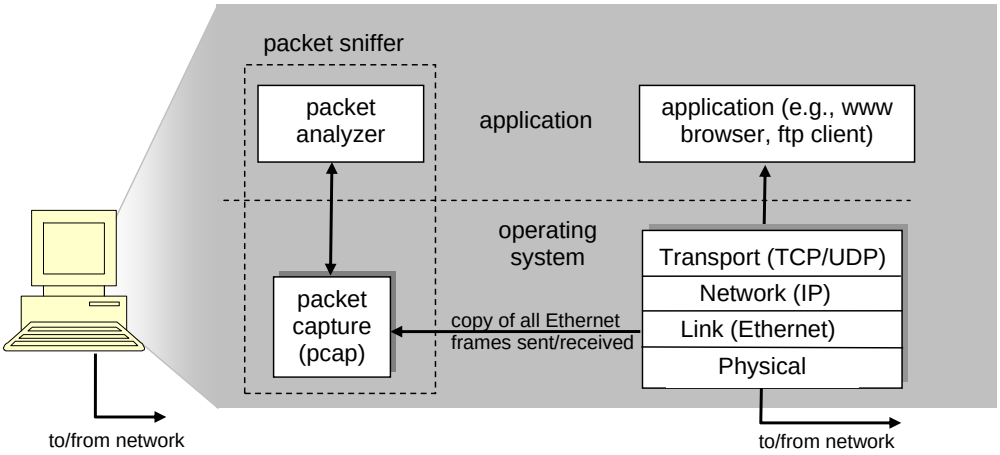

Figure 1: Packet sniffer structure

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 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 (i.e., 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). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac 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) A user-guide (i.e., http://www.wireshark.org/docs/wsug_html_chunked/)
(b) Man pages (i.e., http://www.wireshark.org/docs/man-pages/)
(c) A detailed FAQ (i.e., http://www.wireshark.org/faq.html)
(d) Rich functionality that includes the capability to analyze more than 500 protocols
(e) A well-designed user interface

The Wireshark packet sniffer operates in computers using Ethernet to connect to the Internet, as well as so-called point-to-point protocols such as PPP.

2. Wireshark Lab – Getting Wireshark

In order to run Wireshark, you will need to have access to a computer that supports both Wireshark and the *libpcap* packet capture library. If the *libpcap* software is not installed within your operating system, you will need to install *libpcap* or have it installed for you in order to use Wireshark. See http://www.wireshark.org/download.html for a list of supported operating systems and download sites.

Download and install the Wireshark and (if needed) the *libpcap* software:

- If needed, download and install the *libpcap* software. Pointers to the *libpcap* software are provided from the Wireshark download pages.
  For Windows machines, the *libpcap* software is known as *WinPCap*, and can be found at http://www.winpcap.org/install/default.htm.
- Go to www.wireshark.org and download and install the Wireshark binary for your computer.
  It is recommended to download from http://sourceforge.net/project/showfiles.php?group_id=255 which includes a WinPCap bundle option.
- Download the Wireshark user guide. You will most likely only need Chapters 1 and 3.

The Wireshark FAQ has a number of helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

3. Wireshark Lab – Running Wireshark

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

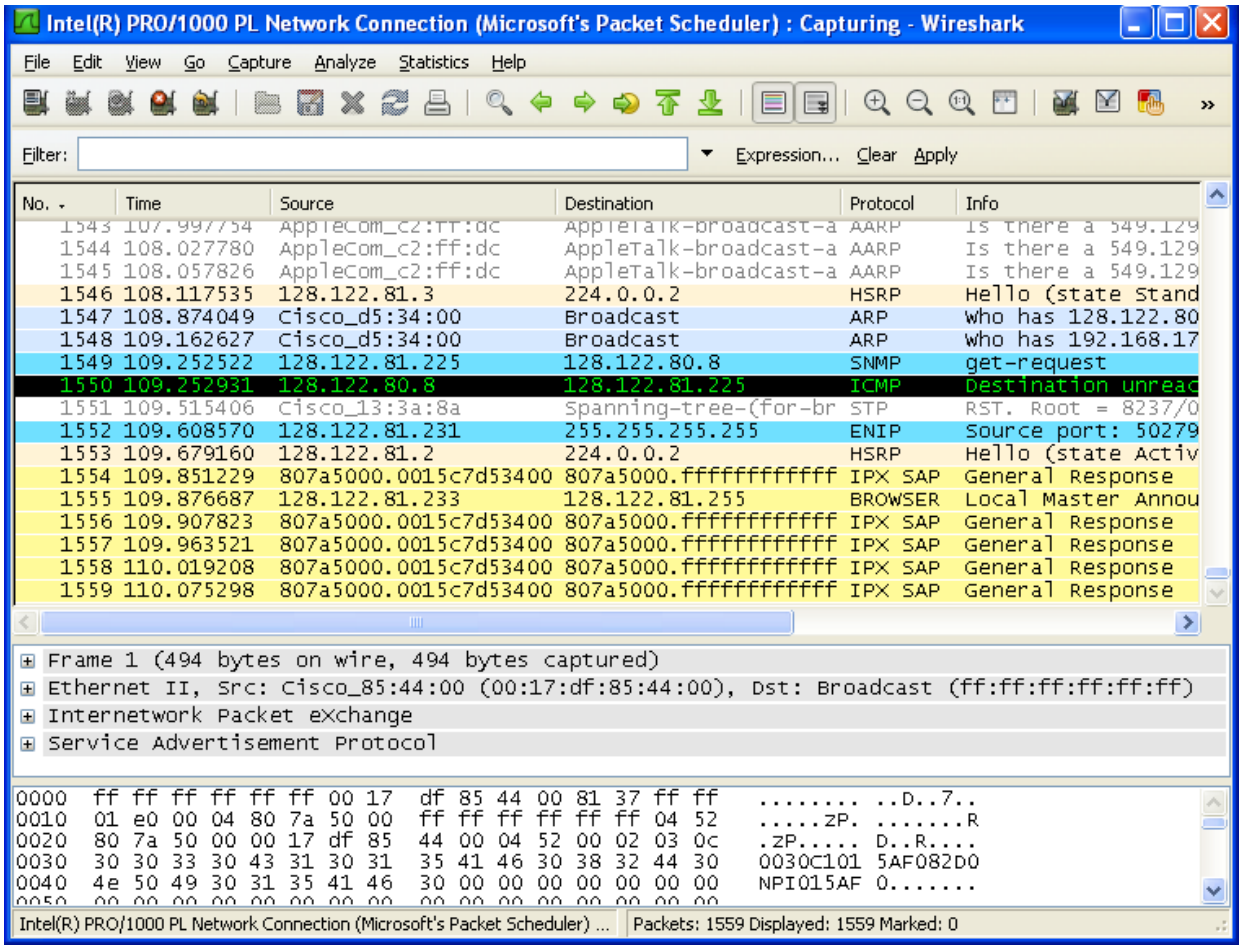

Figure 2a: Wireshark Capture Options Window

Figure 2 below shows the original Ethereal graphical user interface along with an explanation of the various areas which applies to both Ethereal and Wireshark.

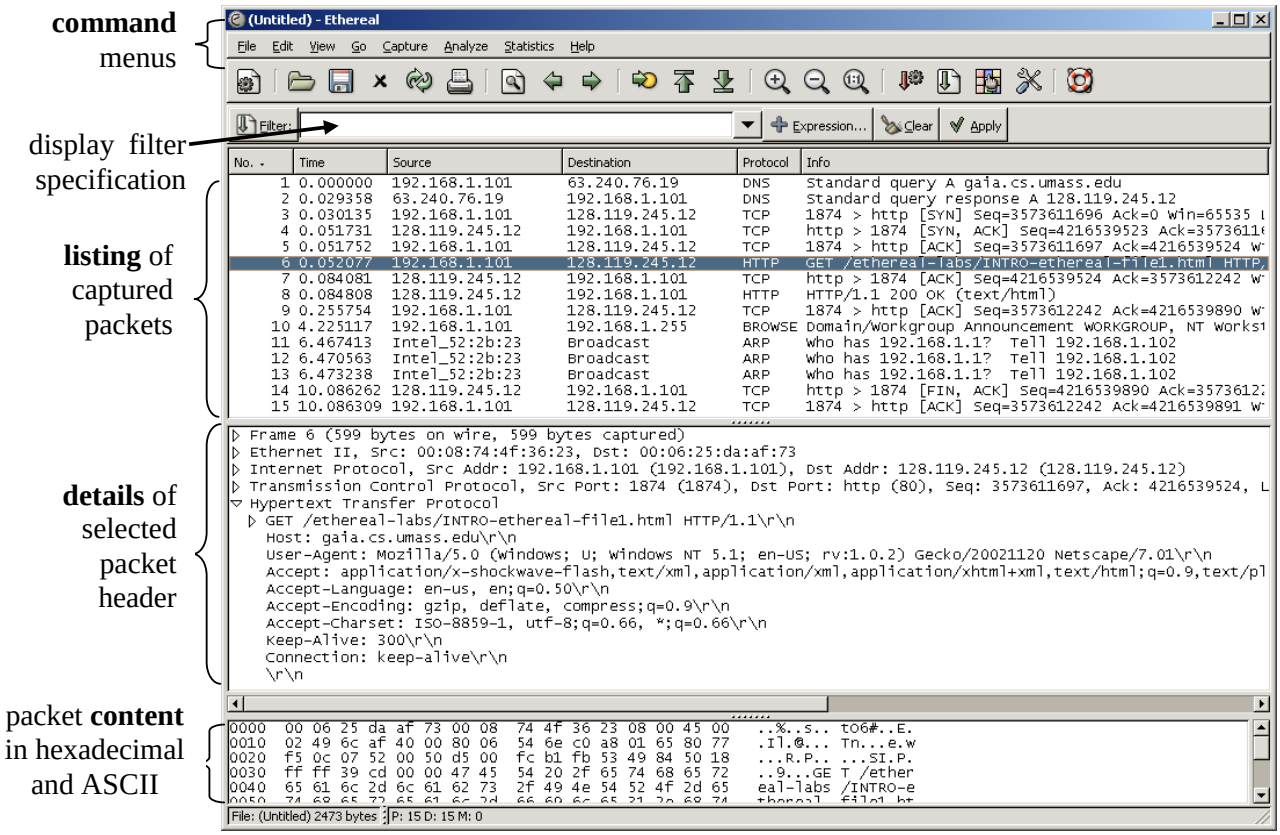

Figure 2: Ethereal Graphical User Interface

The Wireshark 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.

4. Wireshark Lab – 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.

2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2, except that no packet data will be displayed in the packet-listing, packet-header, or packet-contents window, since Wireshark has not yet begun capturing packets.

3. To begin packet capture, select the Capture pull down menu and select *Options*. This will cause the "Wireshark: Capture Options" window to be displayed, as shown in Figure 3.

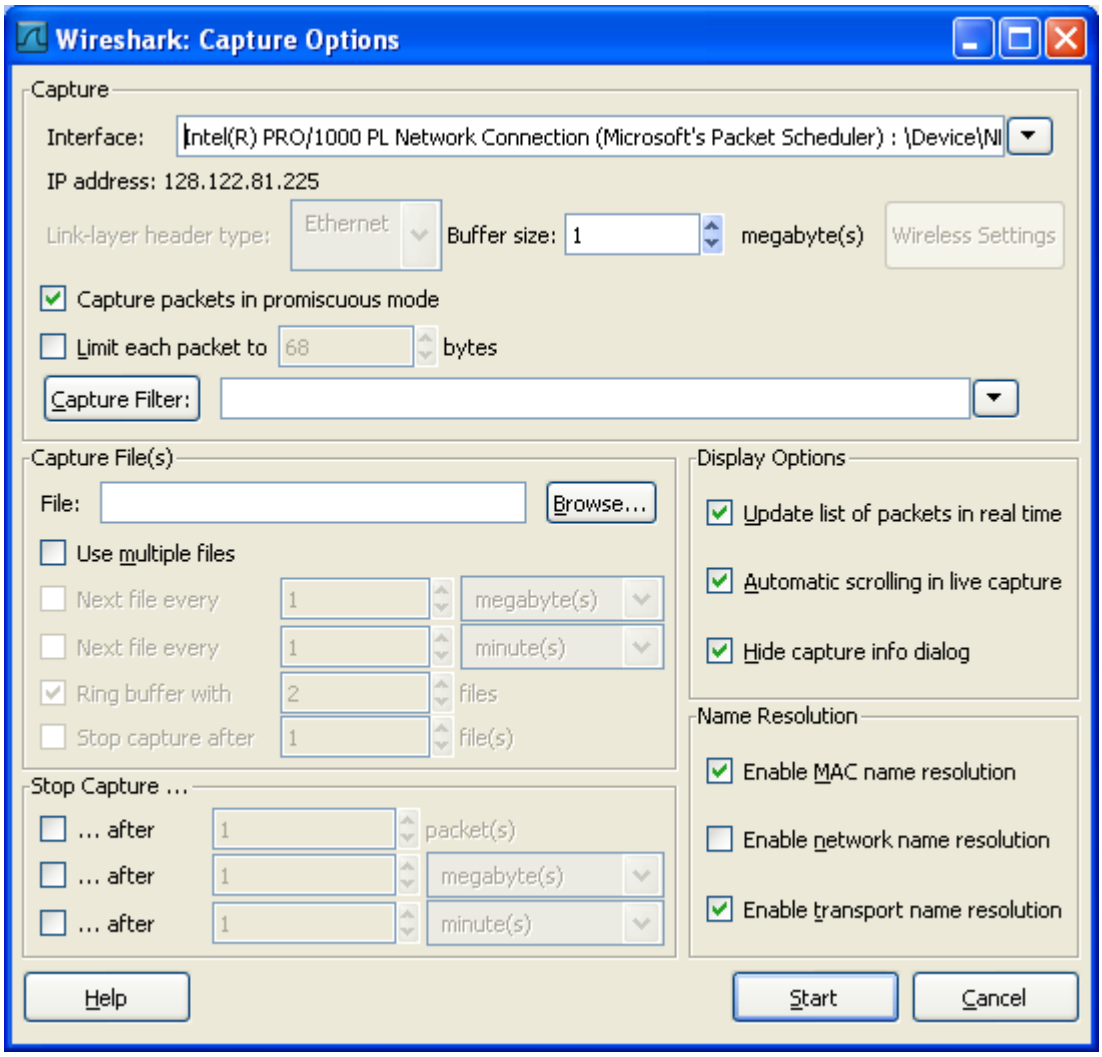

Figure 3: Wireshark Capture Options Window

4. You can use all of the default values in this window. 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. 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 *Start*. Packet capture will now begin - all packets being sent/received from/by your computer are now being captured by Wireshark!

5. After you begin packet capture, you can select Statistics > Protocol Hierarchy from the command menus to obtain a summary of the number of packets of various types that are being captured as shown in Figure 4.

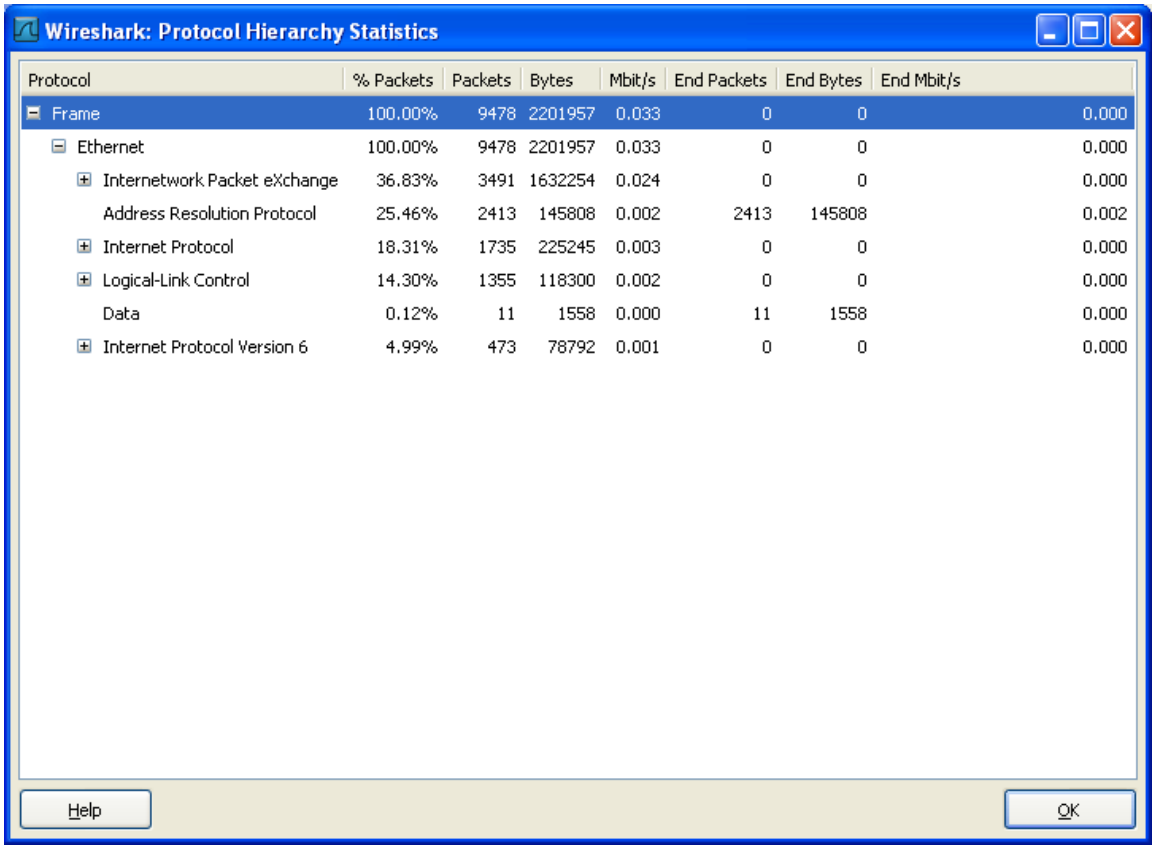

Figure 4: Wireshark Protocol Hierarchy Statistics

6. While Wireshark is 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. The Ethernet frames containing these HTTP messages will be captured by Wireshark.

7. After your browser has displayed the INTRO-ethereal-file1.html page, stop Wireshark packet capture by selecting Capture > Stop in the Wireshark in the command menus. The Wireshark window will display all packets captured since you began packet capture. The Wireshark window should now look similar to Figure 2. 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 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 "meet's the eye"!

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. The HTTP GET message that was sent from your computer to the gaia.cs.umass.edu HTTP server should be shown among the first few http message shown in the packet-listing window. 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. 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 the material covered in class and the corresponding material covered in the textbook. By clicking on the expansion buttons (+ or -) to the left side of the packet details window, you can *minimize* or *maximize* 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 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).

10. Exit Wireshark

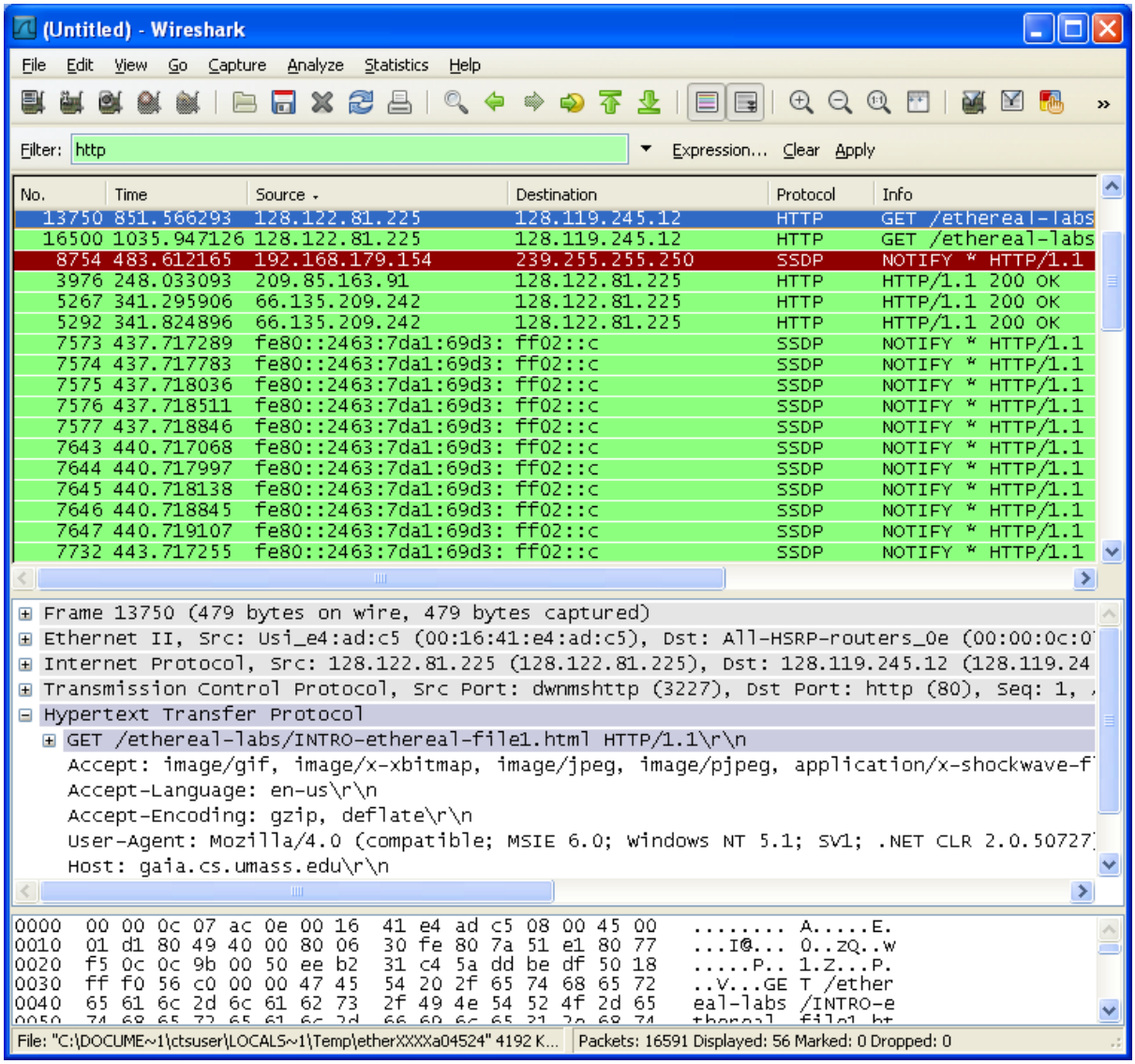

Figure 5: Wireshark Display After Step 9

Congratulations! You've now completed the first lab.

5. Wireshark 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. What is the MAC address of your Host? You can find this in the frame level information.
2. List the different protocols that appear in the protocol column in the unfiltered packet-listing window in step 4.7 above.

3. 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. 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*.)
4. What is the Internet address of the gaia.cs.umass.edu (also known as www-net.cs.umass.edu)? What is the Internet address of your computer?
5. Print the two HTTP messages displayed in step 4.9 above. To do so, select *Print* from the Wireshark *File* command menu, and select "*Selected Packet Only"* under Packet Range and *"As displayed"* under Packet Format and then click OK.

6. Submit your homework solution:

Save your capture in a capture file named Nxxx.cap where Nxxx is your student ID.
Submit this capture file and the answers to the questions above in a separate report (word document.
Email your assignment (archive) file to your grader.
Name the report "firstname_lastname_hw_6.doc" (e.g., "john_doe_hw_6.doc").
Name the archive "firstname_lastname_hw_6.zip" (e.g., "john_doe_hw_6.zip").
Email your assignment files to the course grader, and submit a hard copy of the report to the professor by the due date.
Use the following naming convention in the subject line of the eMail:
"DCN - firstname lastname - homework 6"
(e.g.: "DCN – John Doe - homework 6").
(Note: all files submitted should include your name).

VI. Deliverables

1. Electronic:

Your assignment archive must be emailed to the course grader. The file must be created and sent by the beginning of class. After the class period, the homework is late. The email clock is the official clock.

2. Cover page and other formatting requirements:

The cover page supplied on the next page must be the first page of your report file.

Fill in the blank area for each field.

NOTE:

The sequence of the hardcopy submission of the report is:

1. **Cover sheet**
2. **Assignment Answer Sheet(s)**

VII. Sample Cover Sheet

Name _____ Date: _____
(last name, first name)
Section: _____

Assignment 6

Assignment Layout (25%)

☐ Assignment is neatly assembled on 8 1/2 by 11 paper.
☐ Cover page with your name (last name first followed by a comma then first name), username and section number with a signed statement of independent effort is included.
☐ Answers to Question 5 are correct.
☐ File name is correct.

Answers to Individual Questions:

(100 points total, all questions weighted equally)

☐ Assumptions provided when required.

Total in points (100 points total): _____

Professor's Comments: