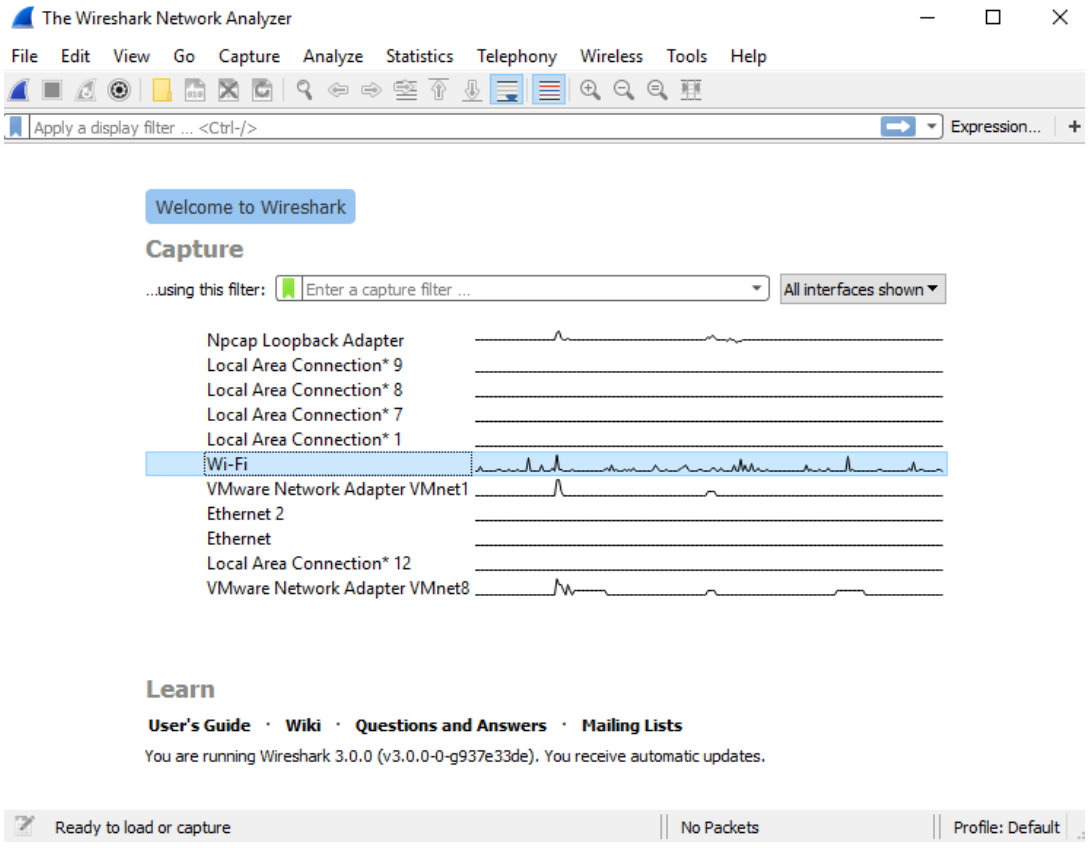
Computer Networks

Assignment IV

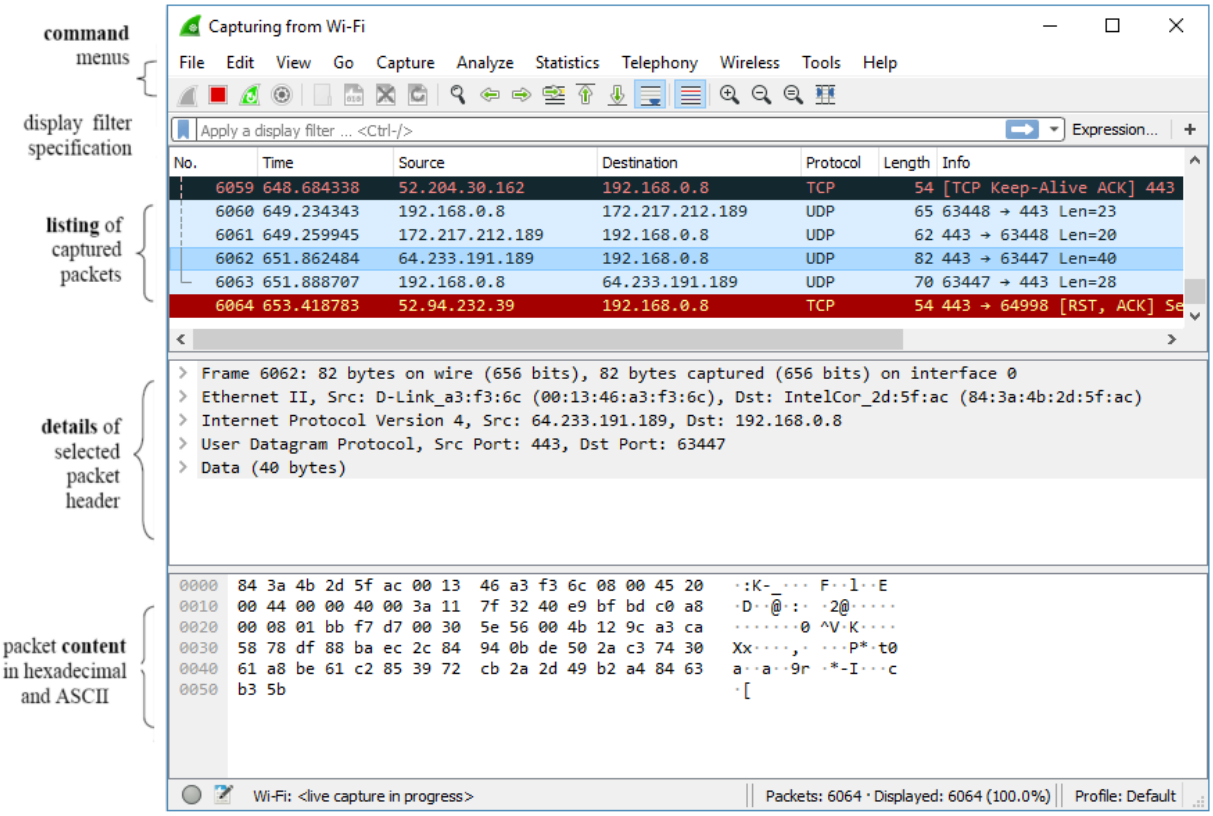
Task 1 (6 points): Wireshark is an open source tool allows you to capture, parse, and analyze the content of real packets sent/received by your computer and to troubleshoot the network.

Step 1 (Prepare the environment):

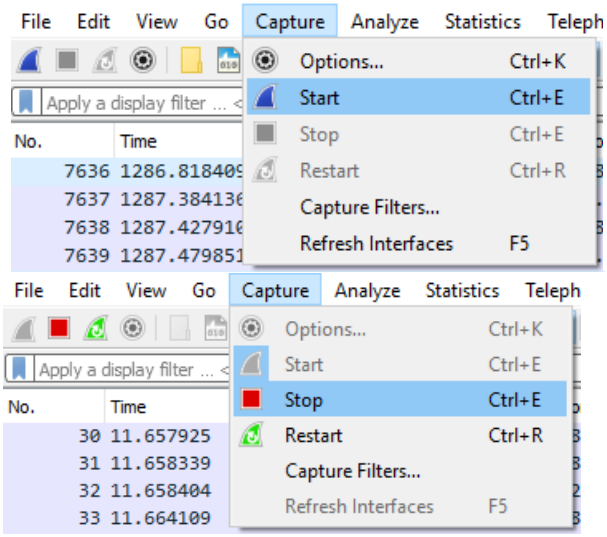
- Visit https://www.wireshark.org/#download and select the version that matches your operating system (Window, Mac). If you are using Linux-based system (e.g., Ubuntu), you may run the root command (sudo apt-get install wireshark) through Linux terminal.
- Install the tool where you find appropriate on your device, then run the tool and the main graphical interface -similar to the figure blew- should pop up.
- From the interface of the previous step, you need to choose the network interface (the physical wireless/wired connection) you will use to capture packets (if you are using your laptop, double click on the WiFi interface as shown below; if you are using a desktop with an Ethernet cable, double click on the Ethernet interface instead).



- Double clicking on the interface as you did in the previous point will open the following window (ignore the content).



- You will use the *display filter specification field* to specify a protocol name or other meta-data to filter the packets displayed in the *listing of captured packets* (and the packet-header and packet-contents windows accordingly).
- To start capturing packets moving around your network from your computer or to your computer, click *Start* under the *Capture* from the command menu. After a while, you can *Stop* capturing packets or even restart capturing again using the same *Capture* menu.



Step 2 (Running some experiments):

The following Lab is taken from the Wireshark Lab - the supplement material to "Computer Networking: A Top-Down Approach", 6th ed., J.F. Kurose and K.W. Ross.

HTTP, as we discussed in class, stands for *HyperText Transfer Protocol*, a protocol used to transfer data (e.g., web pages) on the world-wide-web using the *TCP transport protocol*. Let's suppose that the NEIU server holds *an HTML file* (e.g., the home page):

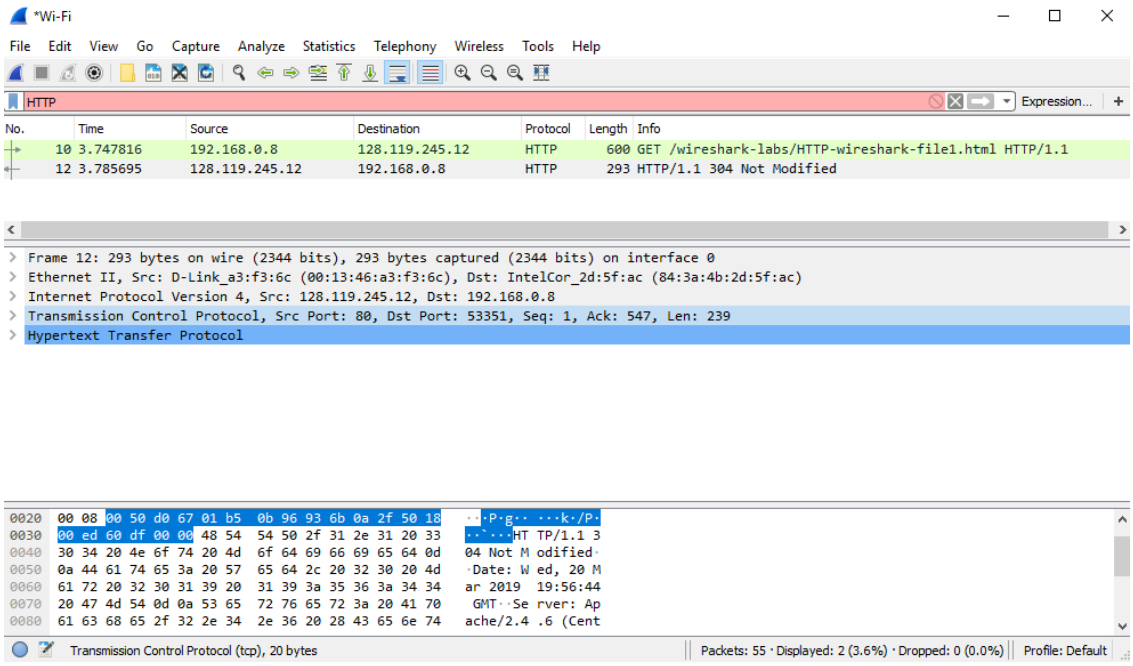
1. Your laptop running Firefox initiates a *TCP connection* with the server and send an *HTTP request*.
2. The server receives the request and retrieves the required page (e.g., homePage.html).
3. The server places the result in an *HTTP response* and sends the response back to the client.

The Basic HTTP GET/response interaction.

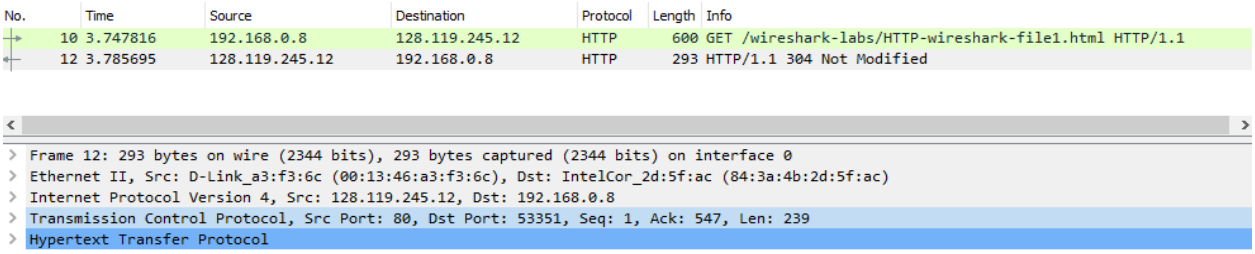
Let's begin our exploration of HTTP by downloading a very simple HTML file -one that is very short and contains no embedded objects- hosted by a remote web server, as follows:

1. Start your web browser (e.g., Chrome, Firefox).
2. Start the Wireshark packet sniffer but don't yet begin packet capture.
3. Enter "http" (just the letters, not the quotation marks) in the display-filter-specification field, so that only captured HTTP messages will be displayed later in the packet-listing window.
4. Wait a bit more than one minute, and then begin Wireshark packet capture (as illustrated in step1).
5. Enter the following to your web browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html Your browser should display the very simple, one-line HTML file.
6. Stop Wireshark packet capture (as illustrated in step1) and your Wireshark window should look similar to the window shown below.

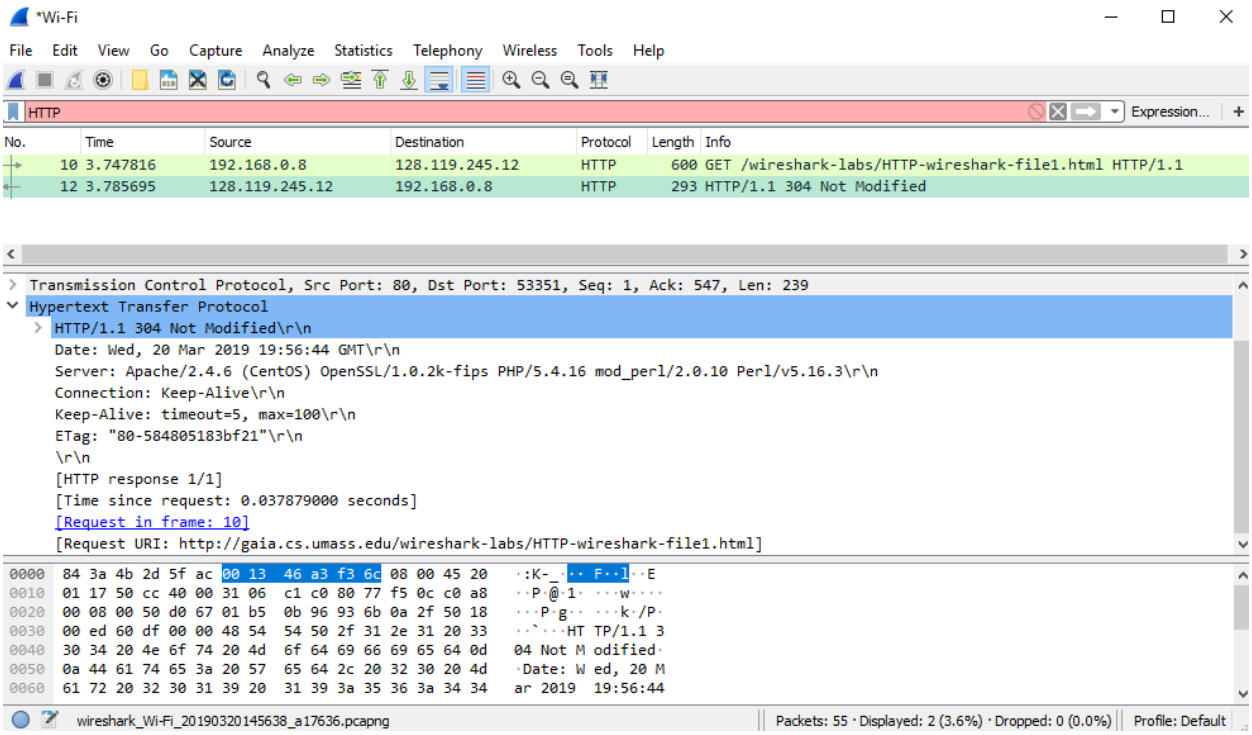
Two HTTP messages were captured: the GET message (from your browser to the gaia.cs.umass.edu web server) and the response message from the server to your browser. When you click on any of the captured messages, the content of the packet will be illustrated in the packet-contents window. If, for any reason, you should repeat these steps again – make sure you clear the Brower's cache (To do this under Firefox, select Tools->Clear Recent History and check the Cache box; or close the browser and open a new incognito window for google chrome).



The HTTP message (application layer) was carried inside a TCP segment (transport layer), which was carried inside an IP datagram (network layer), which was carried within an Ethernet frame (datalink layer). The Wireshark displays as shown below the Frame, Ethernet, IP, and TCP packets, in order.



Minimize them all and open (maximize) only the information regarding the HTTP (Hypertext Transfer Protocol) message (as displayed below) - Just click on the > arrow on the left.



For each of the HTTP request and response messages, take a clear screenshot of the whole Wireshark window showing the details of the packet-content window (similar to the figure above). Then by looking at the content, ***search on the internet about the meaning of the different fields*** to answer the following questions.

1. What languages (check the *Accept-Language* tag in the GET request's header) does your browser indicate that it can accept to the server?

2. What is the IP address of your computer? And the IP of the gaia.cs.umass.edu server?

3. When did the web-server modify the content of the requested HTML file? (check the *Last-Modified* tag in the response)?

4. How many bytes of content are being returned to your browser (check the *Content-Length*)?

2. Retrieving Long Documents

The document retrieved in the previous step is a simple and short HTML file. Let's next see what will happen when we download a large HTML file (4500 bytes). First make sure you clear the Brower's cache, then follow these steps:

1. Start up your web browser (e.g., Chrome, Firefox).

2. Start up the Wireshark packet sniffer but don't yet begin packet capture.

3. Enter "http" (just the letters, not the quotation marks) in the display-filter-specification filed, so that only captured HTTP messages will be displayed later in the packet-listing window.

4. Wait a bit more than one minute, and then begin Wireshark packet capture (as illustrated in step1).

5. Enter the following to your web browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html Your browser should display the rather lengthy US Bill of Rights.

6. Stop Wireshark packet capture. In the packet-listing window, you should see your HTTP request message, followed by a HTTP response (4500 bytes) with multiple TCP segments (the response is large to fit one TCP segment). The single HTTP response message is thus broken into several pieces by TCP, with each piece is contained within a separate TCP segment.

For each of the HTTP request and response messages, take a clear screenshot of the whole Wireshark window showing the details of the packet-content window.

Task 2 (4 points): ARP (Address Resolution Protocol) is an essential and important protocol in the OSI model and is used to guide the mapping between the MAC and IP addresses. Do some ***search on the internet***, then use your own words and/or develop figure(s) to "***clearly***" explain the work/algorithm of this protocol in a paragraph or two. You should cite (mention) the online articles/resources you used to answer this question – plagiarism (and copying from resources) will be checked and will receive ***zero***.

Submission:

1. This is an **individual assignment** -- Cheating/plagiarism will be checked and will receive zero.
2. Use the tool you find appropriate to develop your answers (e.g., Microsoft word) – **DON'T submit hand written solutions and don't use camera to take screenshots**.
3. Submit only **ONE PDF file** to the folder titled Assignment 4 under the D2L Assignments tab (*other formats will not be accepted*).
4. The assignment is due **November 17th – 10:00pm**. You can submit your assignment, within 24 hours after this due date, to be graded out of 50% of the assignment's grade. After this grace period your late submission will not be accepted.