Student Name: ______________________

ECCS 3631 – Networks & Data Communications

LAB 10: HTTP PROTOCOL ANALYSIS

OBJECTIVES

After performing this lab, students should be able to:

- Analyze application layer protocol, HTTP.
- Understand HTTP message formats
- Analyze the basic GET/Response interaction of HTTP protocol.
- Analyze the Conditional GET/Response interaction of HTTP protocol

DELIVERABLES

Take screenshots from every step of the lab. Just attach screenshots in a file (convert it to PDF) and upload it. It is an informal lab report.

TAKING WIRESHARK FOR A TEST RUN

Install Wireshark from Canvas. If you have installed a latest version from the Internet, your output screenshot may be different, however, it is fine to use any version of Wireshark.

It is assumed that your computer is connected to the Internet via a Wi-Fi. Do the following:

Open Command Prompt and run ipconfig. Attach the screenshot of the ipconfig that shows your computer IP address.

Write your PC IP Address: __________________________________

Enable HTTP Filter on your Wireshark. Open Wireshark, you do not need to start packet capturing, Click Analyze → Click Enable Protocols → Scroll down to HTTP and check it and then click OK.

1. Start up **CHROME browser**, which will display your selected homepage.

2. Start up the Wireshark software.

3. To begin packet capture, select the Capture pull-down menu and select *Interfaces.*

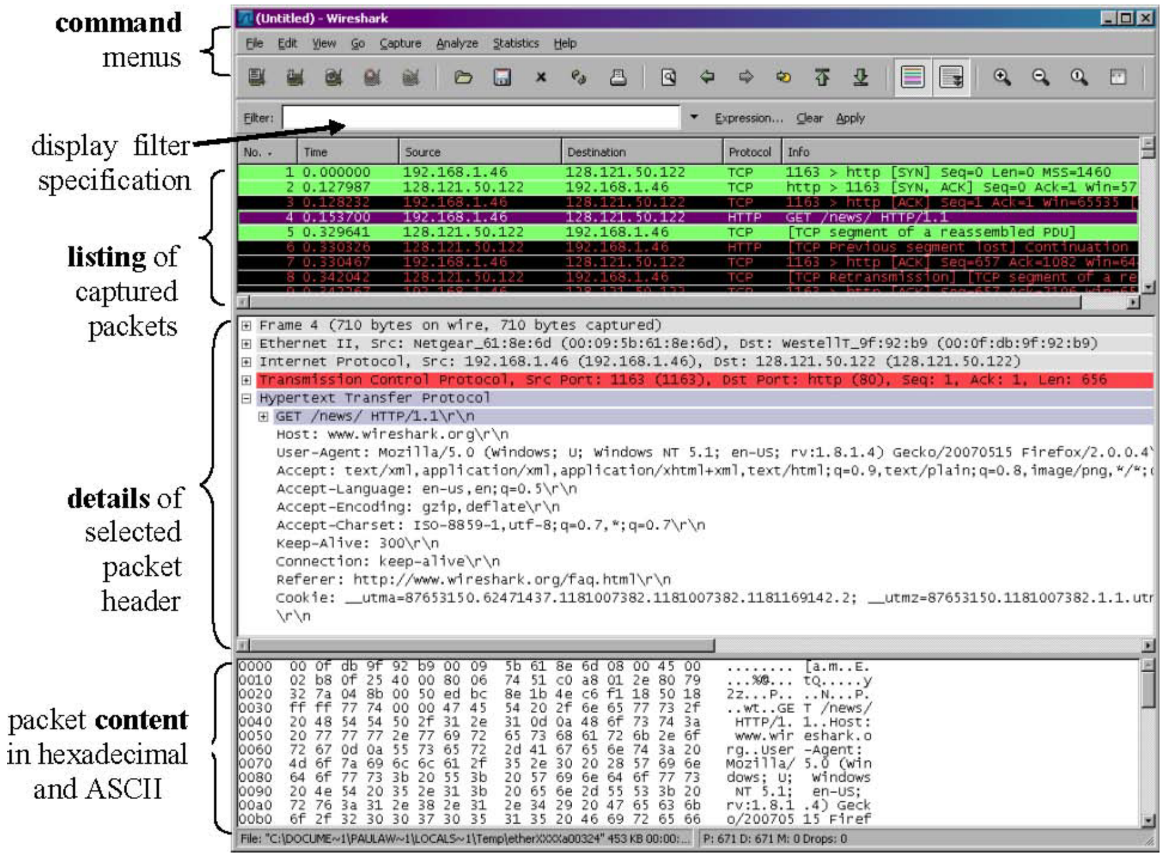
4. You will see a list of the interfaces on your computer as well as a count of the packets that have been observed on that interface so far. Click on Start for the interface on which you want to begin packet capture. Packet capture will now begin -Wireshark is now capturing all packets being sent/received from/by your computer!

5. Once you begin packet capture, a window will appear that shows the packets being captured. By selecting Capture pulldown menu and selecting Stop, 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.

6. 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. The Ethernet frames containing these HTTP messages (as well as all other frames passing through your Ethernet adapter) will be captured by Wireshark.

7. After your browser has displayed the INTRO-wireshark-file1.html page, stop Wireshark packet capture by selecting stop in the Wireshark capture window. 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 1). 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"). This will cause only HTTP message to be displayed in the packet-listing window.


command menus
display filter specification
listing of captured packets
details of selected packet header
packet **content** in hexadecimal and ASCII
(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. - Time      Source          Destination     Protocol Info
    1 0.000000  192.168.1.46    128.121.50.122  TCP      1163 > http [SYN] Seq=0 Len=0 MSS=1460
    2 0.127987  128.121.50.122  192.168.1.46    TCP      http > 1163 [SYN, ACK] Seq=0 Ack=1 Win=57
    3 0.128232  192.168.1.46    128.121.50.122  TCP      1163 > http [ACK] Seq=1 Ack=1 Win=65535 [
    4 0.153700  192.168.1.46    128.121.50.122  HTTP     GET /news/ HTTP/1.1
    5 0.329641  128.121.50.122  192.168.1.46    TCP      [TCP segment of a reassembled PDU]
    6 0.330326  128.121.50.122  192.168.1.46    HTTP     [TCP Previous segment lost] Continuation
    7 0.330467  192.168.1.46    128.121.50.122  TCP      1163 > http [ACK] Seq=657 Ack=1082 Win=64
    8 0.342042  128.121.50.122  192.168.1.46    TCP      [TCP Retransmission] [TCP segment of a re
```

```
⊞ Frame 4 (710 bytes on wire, 710 bytes captured)
⊞ Ethernet II, Src: Netgear_61:8e:6d (00:09:5b:61:8e:6d), Dst: WestellT_9f:92:b9 (00:0f:db:9f:92:b9)
⊞ Internet Protocol, Src: 192.168.1.46 (192.168.1.46), Dst: 128.121.50.122 (128.121.50.122)
⊞ Transmission Control Protocol, Src Port: 1163 (1163), Dst Port: http (80), Seq: 1, Ack: 1, Len: 656
⊟ Hypertext Transfer Protocol
  ⊞ GET /news/ HTTP/1.1\r\n
    Host: www.wireshark.org\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Referer: http://www.wireshark.org/faq.html\r\n
    Cookie: __utma=87653150.62471437.1181007382.1181007382.1181169142.2; __utmz=87653150.1181007382.1.1.utr
    \r\n
```

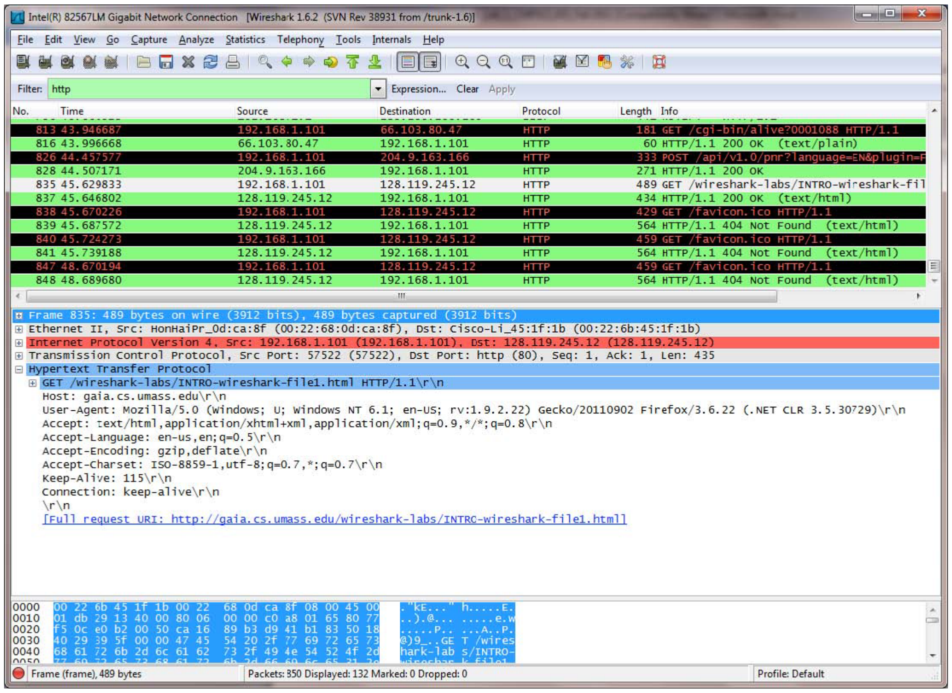
```
0000  00 0f db 9f 92 b9 00 09  5b 61 8e 6d 08 00 45 00   ........ [a.m..E.
0010  02 b8 0f 25 40 00 80 06  74 51 c0 a8 01 2e 80 79   ...%@... tQ.....y
0020  32 7a 04 8b 00 50 ed bc  8e 1b 4e c6 f1 18 50 18   2z...P.. ..N...P.
0030  ff ff 77 74 00 00 47 45  54 20 2f 6e 65 77 73 2f   ..wt..GE T /news/
0040  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1. 1..Host:
0050  20 77 77 77 2e 77 69 72  65 73 68 61 72 6b 2e 6f    www.wir eshark.o
0060  72 67 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20   rg..User -Agent:
0070  4d 6f 7a 69 6c 6c 61 2f  35 2e 30 20 28 57 69 6e   Mozilla/ 5.0 (Win
0080  64 6f 77 73 3b 20 55 3b  20 57 69 6e 64 6f 77 73   dows; U;  Windows
0090  20 4e 54 20 35 2e 31 3b  20 65 6e 2d 55 53 3b 20    NT 5.1;  en-US;
00a0  72 76 3a 31 2e 38 2e 31  2e 34 29 20 47 65 63 6b   rv:1.8.1 .4) Geck
00b0  6f 2f 32 30 30 37 30 35  31 35 20 46 69 72 65 66   o/200705 15 Firef
```

```
File: "C:\DOCUME~1\PAULAW~1\LOCALS~1\Temp\etherXXXXa00324" 453 KB 00:00:...   P: 671 D: 671 M: 0 Drops: 0
```

Figure 1: Wireshark Graphical User Interface, during packet capture and analysis

9. 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 Figure 1) that shows "GET" followed by the gaia.cs.umass.edu 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 '-' 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 Control Protocol information displayed. *Maximize* the amount information displayed about the HTTP protocol. Your Wireshark display should now look roughly as shown in Figure 2. (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).


Intel(R) 82567LM Gigabit Network Connection [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply

Filter: http

```
No.   Time       Source           Destination      Protocol   Length Info
  813 43.946687  192.168.1.101    66.103.80.47     HTTP       181 GET /cgi-bin/alive?0001088 HTTP/1.1
  816 43.996668  66.103.80.47     192.168.1.101    HTTP        60 HTTP/1.1 200 OK  (text/plain)
  826 44.457577  192.168.1.101    204.9.163.166    HTTP       333 POST /api/v1.0/pnr?language=EN&plugin=F
  828 44.507171  204.9.163.166    192.168.1.101    HTTP       271 HTTP/1.1 200 OK
  835 45.629833  192.168.1.101    128.119.245.12   HTTP       489 GET /wireshark-labs/INTRO-wireshark-fil
  837 45.646802  128.119.245.12   192.168.1.101    HTTP       434 HTTP/1.1 200 OK  (text/html)
  838 45.670226  192.168.1.101    128.119.245.12   HTTP       429 GET /favicon.ico HTTP/1.1
  839 45.687572  128.119.245.12   192.168.1.101    HTTP       564 HTTP/1.1 404 Not Found  (text/html)
  840 45.724273  192.168.1.101    128.119.245.12   HTTP       459 GET /favicon.ico HTTP/1.1
  841 45.739188  128.119.245.12   192.168.1.101    HTTP       564 HTTP/1.1 404 Not Found  (text/html)
  847 48.670194  192.168.1.101    128.119.245.12   HTTP       459 GET /favicon.ico HTTP/1.1
  848 48.689680  128.119.245.12   192.168.1.101    HTTP       564 HTTP/1.1 404 Not Found  (text/html)
```

```
⊞ Frame 835: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
⊞ Ethernet II, Src: HonHaiPr_0d:ca:8f (00:22:68:0d:ca:8f), Dst: Cisco-Li_45:1f:1b (00:22:6b:45:1f:1b)
⊞ Internet Protocol Version 4, Src: 192.168.1.101 (192.168.1.101), Dst: 128.119.245.12 (128.119.245.12)
⊞ Transmission Control Protocol, Src Port: 57522 (57522), Dst Port: http (80), Seq: 1, Ack: 1, Len: 435
⊟ Hypertext Transfer Protocol
  ⊞ GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.22) Gecko/20110902 Firefox/3.6.22 (.NET CLR 3.5.30729)\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 115\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html]
```

```
0000  00 22 6b 45 1f 1b 00 22  68 0d ca 8f 08 00 45 00   ."kE..." h.....E.
0010  01 db 29 13 40 00 80 06  00 00 c0 a8 01 65 80 77   ..).@... .....e.w
0020  f5 0c e0 b2 00 50 ca 16  89 b3 d9 41 b1 83 50 18   .....P.. ...A..P.
0030  40 29 39 5f 00 00 47 45  54 20 2f 77 69 72 65 73   @)9_..GE T /wires
0040  68 61 72 6b 2d 6c 61 62  73 2f 49 4e 54 52 4f 2d   hark-lab s/INTRO-
```

Frame (frame), 489 bytes | Packets: 850 Displayed: 132 Marked: 0 Dropped: 0 | Profile: Default

Figure 2: Wireshark window after step 9

10. Exit Wireshark

You have now completed the first part of this lab.

HTTP PROTOCOL

In this part of the lab, you will 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.

THE HTTP GET/RESPONSE INTERACTION

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

- Start up your web browser.

- Start up the Wireshark packet sniffer, but don't yet begin packet capture. Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We are only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).

- Wait a bit more than one minute, and then begin Wireshark packet capture.

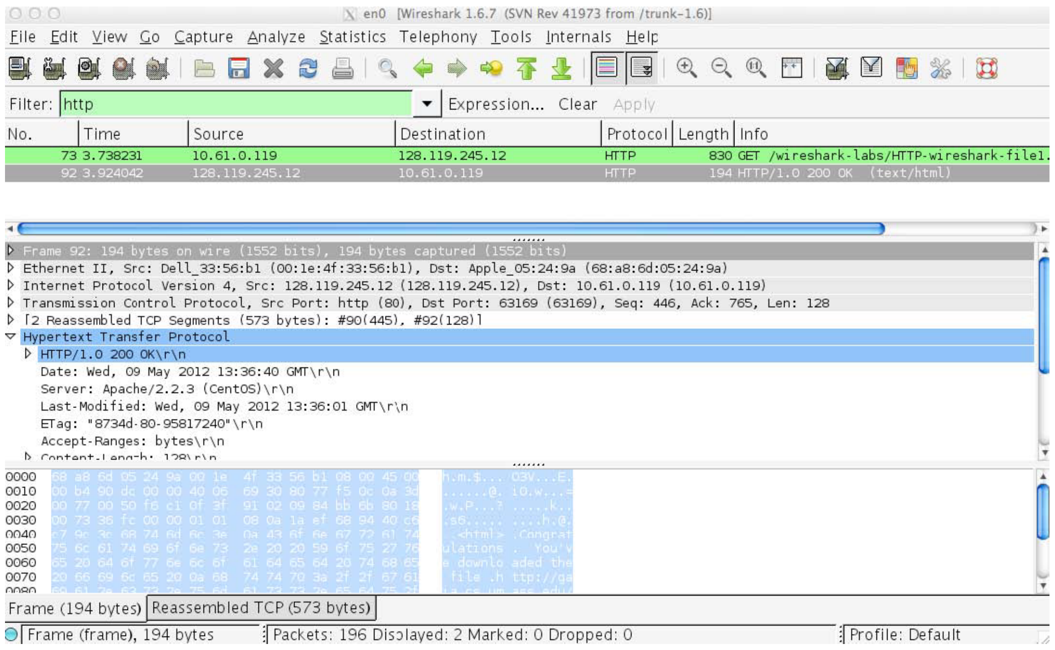
- Enter the following to your browser
  - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html
  - Your browser should display a very simple, one-line HTML file.

- Stop Wireshark packet capture.

Your Wireshark window should look similar to the window shown in Figure 3.

Attach the screenshots of the HTTP GET and response messages:

The example in Figure 3 shows in the packet-listing window that 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. The packet-contents window shows details of the selected message (in this case the HTTP OK message, which is highlighted


en0 [Wireshark 1.6.7 (SVN Rev 41973 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply

```
No.   Time        Source          Destination     Protocol Length Info
   73 3.738231    10.61.0.119     128.119.245.12  HTTP        830 GET /wireshark-labs/HTTP-wireshark-file1.
   92 3.924042    128.119.245.12  10.61.0.119     HTTP        194 HTTP/1.0 200 OK  (text/html)
```

```
▷ Frame 92: 194 bytes on wire (1552 bits), 194 bytes captured (1552 bits)
▷ Ethernet II, Src: Dell_33:56:b1 (00:1e:4f:33:56:b1), Dst: Apple_05:24:9a (68:a8:6d:05:24:9a)
▷ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.61.0.119 (10.61.0.119)
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 63169 (63169), Seq: 446, Ack: 765, Len: 128
▷ [2 Reassembled TCP Segments (573 bytes): #90(445), #92(128)]
▽ Hypertext Transfer Protocol
  ▷ HTTP/1.0 200 OK\r\n
    Date: Wed, 09 May 2012 13:36:40 GMT\r\n
    Server: Apache/2.2.3 (CentOS)\r\n
    Last-Modified: Wed, 09 May 2012 13:36:01 GMT\r\n
    ETag: "8734d-80-95817240"\r\n
    Accept-Ranges: bytes\r\n
  ▷ Content-Length: 128\r\n
```

```
0000  68 a8 6d 05 24 9a 00 1e  4f 33 56 b1 08 00 45 00   h.m.$... O3V...E.
0010  00 b4 90 dc 00 00 40 06  69 30 80 77 f5 0c 0a 3d   ......@. i0.w...=
0020  00 77 00 50 f6 c1 0f 3f  91 02 09 84 bb 6b 80 18   .w.P...? .....k..
0030  00 73 36 fc 00 00 01 01  08 0a 1a ef 68 94 40 c6   .s6..... ....h.@.
0040  c7 9c 3c 68 74 6d 6c 3e  0a 43 6f 6e 67 72 61 74   ..<html> .Congrat
0050  75 6c 61 74 69 6f 6e 73  2e 20 20 59 6f 75 27 76   ulations .  You'v
0060  65 20 64 6f 77 6e 6c 6f  61 64 65 64 20 74 68 65   e downlo aded the
0070  20 66 69 6c 65 20 0a 68  74 74 70 3a 2f 2f 67 61    file .h ttp://ga
0080
```

Frame (194 bytes)
Reassembled TCP (573 bytes)
Frame (frame), 194 bytes
Packets: 196 Displayed: 2 Marked: 0 Dropped: 0
Profile: Default

Figure 3: Wireshark Display after http://gaia.cs.umass.edu/wireshark-labs/ HTTPwireshark-file1.html has been retrieved by your browser

in the packet-listing window). 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 is 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 that you captured, answer the following questions:

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

____

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

   ____

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

   ____

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

   ____

5. When was the HTML file that you are retrieving last modified at the server? What is the time zone?

   ____

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

   ____

In your answer to question 5 above, you might have been surprised to find that the document you just retrieved was last modified within a minute before you downloaded the document. That's because (for this particular file), the gaia.cs.umass.edu server is setting the file's last-modified time to be the current time, and is doing so once per minute. Thus, if you wait a minute between accesses, the file will appear to have been recently modified, and hence your browser will download a "new" copy of the document.

THE HTTP CONDITIONAL GET/RESPONSE INTERACTION

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 remove cached files from your browser's cache.) Now do the following:

- Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer.
- Enter the following URL into your browser
  http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html
  Your browser should display a very simple five-line HTML file.
- Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
- 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-lisa ting window.

Attach screenshot after doing the two identical HTTP GETs. Identify the first GET

Answer the following questions based on packets that you have captured:

7. 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?

   ____

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

   ____

9. 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?

   ____

10. 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.

    ____