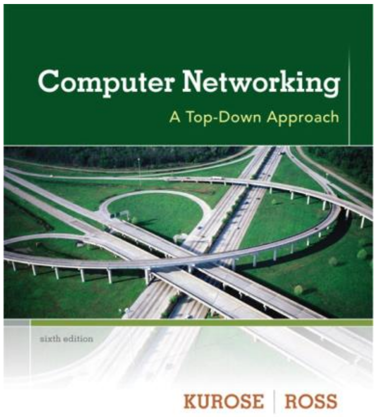
Assigned: March 1, 2013
Lab2 has two parts. Both parts are due on March 15 at 11:59pm (digital Dropbox)
Late submissions are accepted for two days only with a maximum penalty of 15% per day. For each day, submissions between 12 am- 1am: 3%, 1am-2am: 7%, 2-3am: 12%, and after 3am: 15%.

LAB2 – PART I

Wireshark Lab: DHCP



Supplement to *Computer Networking: A Top-Down Approach, 6th ed.,* J.F. Kurose and K.W. Ross

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb

In this lab, we'll take a quick look at DHCP. DHCP is covered in Section 4.4.2 of the text[1]. Recall that DHCP is used extensively in corporate, university and home-network wired and wireless LANs to dynamically assign IP addresses to hosts (as well as to configure other network configuration information).

This lab is brief, as we'll only examine the DHCP packets captured by a host. If you also have administrative access to your DHCP server, you may want to repeat this lab after making some configuration changes (such as the lease time). If you have a router at home, you most likely can configure your DHCP server. Because many linux/Unix machines (especially those that serve many users) have a static IP address and because manipulating DHCP on such machines typically requires super-user privileges, we'll only present a Windows version of this lab below.

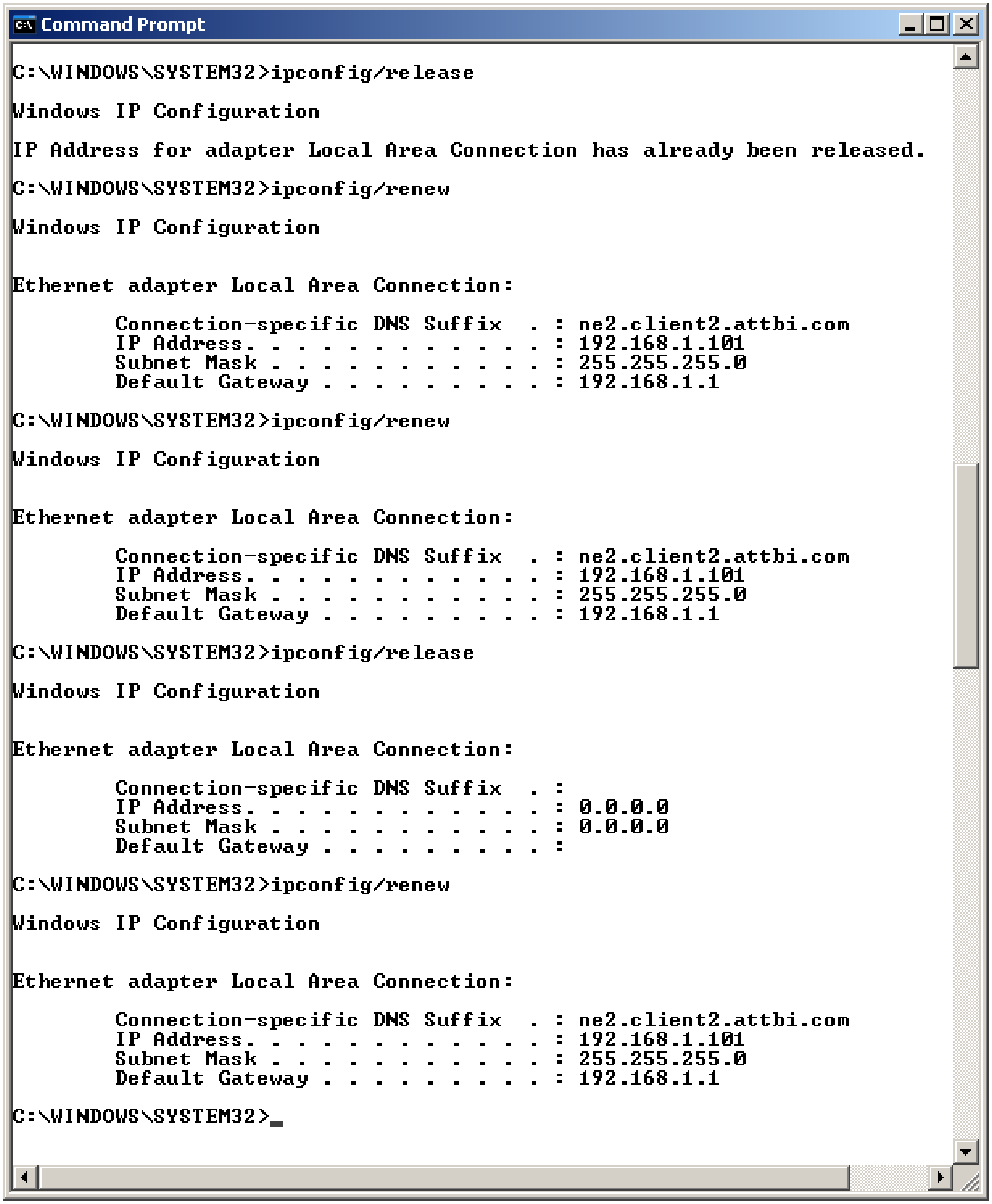
DHCP Experiment

[1] References to figures and sections are for the 6th edition of our text, *Computer Networks, A Top-down Approach, 6th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2012.*

In order to observe DHCP in action, we'll perform several DHCP-related commands and capture the DHCP messages exchanged as a result of executing these commands. Do the following[2]:

1. Begin by opening the Windows Command Prompt application (which can be found in your Accessories folder). As shown in Figure 1, enter "*ipconfig /release*". The executable for *ipconfig* is in C:\windows\system32. This command releases your current IP address, so that your host's IP address becomes 0.0.0.0.
2. Start up the Wireshark packet sniffer, as described in the introductory Wireshark lab and begin Wireshark packet capture.
3. Now go back to the Windows Command Prompt and enter "*ipconfig /renew*". This instructs your host to obtain a network configuration, including a new IP address. In Figure 1, the host obtains the IP address 192.168.1.108
4. Wait until the "*ipconfig /renew*" has terminated. Then enter the same command "*ipconfig /renew*" again.
5. When the second *"ipconfig /renew"* terminates, enter the command "ipconfig/release" to release the previously-allocated IP address to your computer.
6. Finally, enter "*ipconfig /renew*" to again be allocated an IP address for your computer.
7. Stop Wireshark packet capture.

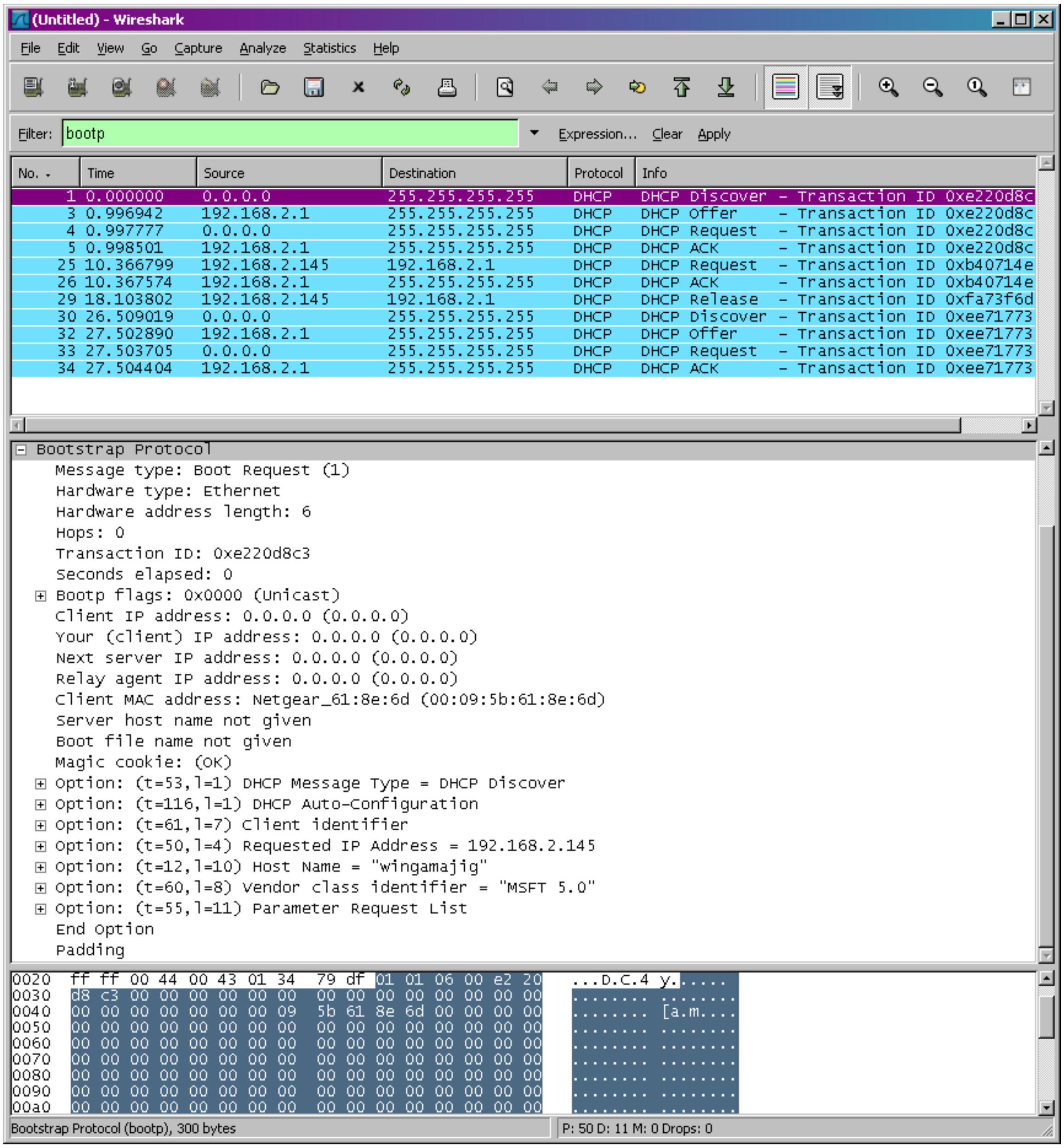
[2] If you are unable to run Wireshark live on a computer, you can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file *dhcp-ethereal-trace-1*. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the dhcp-ethereal-trace-1 trace file. You can then use this trace file to answer the questions below.



```
C:\WINDOWS\SYSTEM32>ipconfig/release

Windows IP Configuration

IP Address for adapter Local Area Connection has already been released.

C:\WINDOWS\SYSTEM32>ipconfig/renew

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : ne2.client2.attbi.com
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\WINDOWS\SYSTEM32>ipconfig/renew

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : ne2.client2.attbi.com
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\WINDOWS\SYSTEM32>ipconfig/release

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 0.0.0.0
        Subnet Mask . . . . . . . . . . . : 0.0.0.0
        Default Gateway . . . . . . . . . :

C:\WINDOWS\SYSTEM32>ipconfig/renew

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : ne2.client2.attbi.com
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\WINDOWS\SYSTEM32>_
```

Figure 1 Command Prompt window showing sequence of *ipconfig* commands that you should enter.

Now let's take a look at the resulting Wireshark window. To see only the DHCP packets, enter into the filter field "bootp". (DHCP derives from an older protocol called BOOTP. Both BOOTP and DHCP use the same port numbers, 67 and 68. To see DHCP packets in the current version of Wireshark, you need to enter "bootp" and not "dhcp" in the filter.) We see from Figure 2 that the first *ipconfig* renew command caused four DHCP packets to be generated: a DHCP Discover packet, a DHCP Offer packet, a DHCP Request packet, and a DHCP ACK packet.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: bootp
Expression... Clear Apply

```
No. -  Time       Source         Destination      Protocol Info
  1 0.000000   0.0.0.0        255.255.255.255  DHCP  DHCP Discover - Transaction ID 0xe220d8c
  3 0.996942   192.168.2.1    255.255.255.255  DHCP  DHCP Offer    - Transaction ID 0xe220d8c
  4 0.997777   0.0.0.0        255.255.255.255  DHCP  DHCP Request  - Transaction ID 0xe220d8c
  5 0.998501   192.168.2.1    255.255.255.255  DHCP  DHCP ACK      - Transaction ID 0xe220d8c
 25 10.366799  192.168.2.145  192.168.2.1      DHCP  DHCP Request  - Transaction ID 0xb40714e
 26 10.367574  192.168.2.1    255.255.255.255  DHCP  DHCP ACK      - Transaction ID 0xb40714e
 29 18.103802  192.168.2.145  192.168.2.1      DHCP  DHCP Release  - Transaction ID 0xfa73f6d
 30 26.509019  0.0.0.0        255.255.255.255  DHCP  DHCP Discover - Transaction ID 0xee71773
 32 27.502890  192.168.2.1    255.255.255.255  DHCP  DHCP Offer    - Transaction ID 0xee71773
 33 27.503705  0.0.0.0        255.255.255.255  DHCP  DHCP Request  - Transaction ID 0xee71773
 34 27.504404  192.168.2.1    255.255.255.255  DHCP  DHCP ACK      - Transaction ID 0xee71773
```

```
⊟ Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xe220d8c3
    Seconds elapsed: 0
  ⊞ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: Netgear_61:8e:6d (00:09:5b:61:8e:6d)
    Server host name not given
    Boot file name not given
    Magic cookie: (OK)
  ⊞ Option: (t=53,l=1) DHCP Message Type = DHCP Discover
  ⊞ Option: (t=116,l=1) DHCP Auto-Configuration
  ⊞ Option: (t=61,l=7) Client identifier
  ⊞ Option: (t=50,l=4) Requested IP Address = 192.168.2.145
  ⊞ Option: (t=12,l=10) Host Name = "wingamajig"
  ⊞ Option: (t=60,l=8) Vendor class identifier = "MSFT 5.0"
  ⊞ Option: (t=55,l=11) Parameter Request List
    End Option
    Padding
```

```
0020  ff ff 00 44 00 43 01 34  79 df 01 01 06 00 e2 20   ...D.C.4 y.......
0030  d8 c3 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  00 00 00 00 00 00 00 09  5b 61 8e 6d 00 00 00 00   ........ [a.m....
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

Bootstrap Protocol (bootp), 300 bytes   P: 50 D: 11 M: 0 Drops: 0

Figure 2 Wireshark window with first DHCP packet – the DHCP Discover packet – expanded.

What to Hand In:

You should hand in a screen shot of the Command Prompt window similar to Figure 1 above. Whenever possible, when answering a question below, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout[3] to explain your answer. To print a packet, use *File->Print*, choose *Selected packet only,* choose *Packet summary line,* and select the minimum amount of packet detail that you need to answer the question.

Answer the following questions:

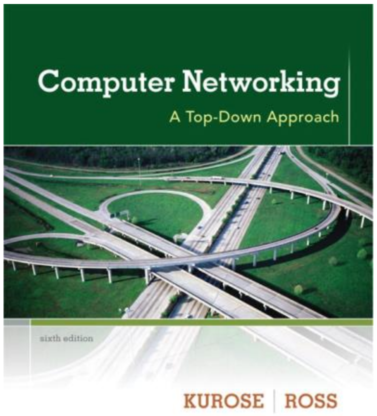
1. Are DHCP messages sent over TCP? If not then how are DHCP messages sent over?
2. Draw a timing datagram illustrating the sequence of the first four-packet Discover/Offer/Request/ACK DHCP exchange between the client and server. For each packet, indicated the source and destination port numbers and IP addresses. Are the port numbers the same as in the example given in this lab assignment?
3. What is the link-layer (e.g., Ethernet) address of your host in both numeric and hex format?
4. What values in the DHCP Offer message differentiate this message from the DHCP ACK message?
5. What is the value of the Transaction-ID in the first and second DHCP messages? Why do we need the Transaction-ID field?
6. A host uses DHCP to obtain an IP address, among other things. But a host's IP address is not confirmed until the end of the four-message exchange! If the IP address is not set until the end of the four-message exchange, then what values are used in the IP datagrams in the four-message exchange? For each of the four DHCP messages (Discover/Offer/Request/ACK DHCP), indicate the source and destination IP addresses that are carried in the encapsulating IP datagram. Also indicate what other information does the user get form the DHCP server?
7. What is the IP address of your DHCP server?
8. What IP address is the DHCP server offering to your host and by which DHCP message.
9. In the example screenshot in this assignment, there is no relay agent between the host and the DHCP server. What values in the trace indicate the absence of a relay agent? Is there a relay agent in your experiment? If so what is the IP address of the agent?
10. Explain the purpose of the router and subnet mask lines in the DHCP offer message and indicate the IP address of the default gateway (router) and its subnet mask.

[3] What do we mean by "annotate"? If you hand in a paper copy, please highlight where in the printout you've found the answer and add some text (preferably with a colored pen) noting what you found in what you 've highlight. If you hand in an electronic copy, it would be great if you could also highlight and annotate.

11. In the DHCP trace file noted in footnote 2, the DHCP server offers a specific IP address to the client (see also question 8. above). In the client's response to the first server OFFER message, does the client accept this IP address? Where in the client's RESPONSE is the client's requested address?
12. Explain the purpose of the lease time. How long is the lease time in your experiment?
13. What is the purpose of the DHCP release message? Does the DHCP server issue an acknowledgment of receipt of the client's DHCP request? What would happen if the client's DHCP release message is lost?
14. Clear the *bootp* filter from your Wireshark window. Were any ARP packets sent or received during the DHCP packet-exchange period? If so, explain the purpose of those ARP packets.

LAB2 – PART II

Wireshark Lab: HTTP



Supplement to *Computer Networking: A Top-Down Approach, 6th ed.*, J.F. Kurose and K.W. Ross

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb

Having gotten our feet wet with the Wireshark packet sniffer in the introductory lab, we're now ready to use Wireshark to investigate protocols in operation. In this lab, 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 Section 2.2 of the text.[4]

[4] References to figures and sections are for the 6th edition of our text, *Computer Networks, A Top-down Approach, 6th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2012.*

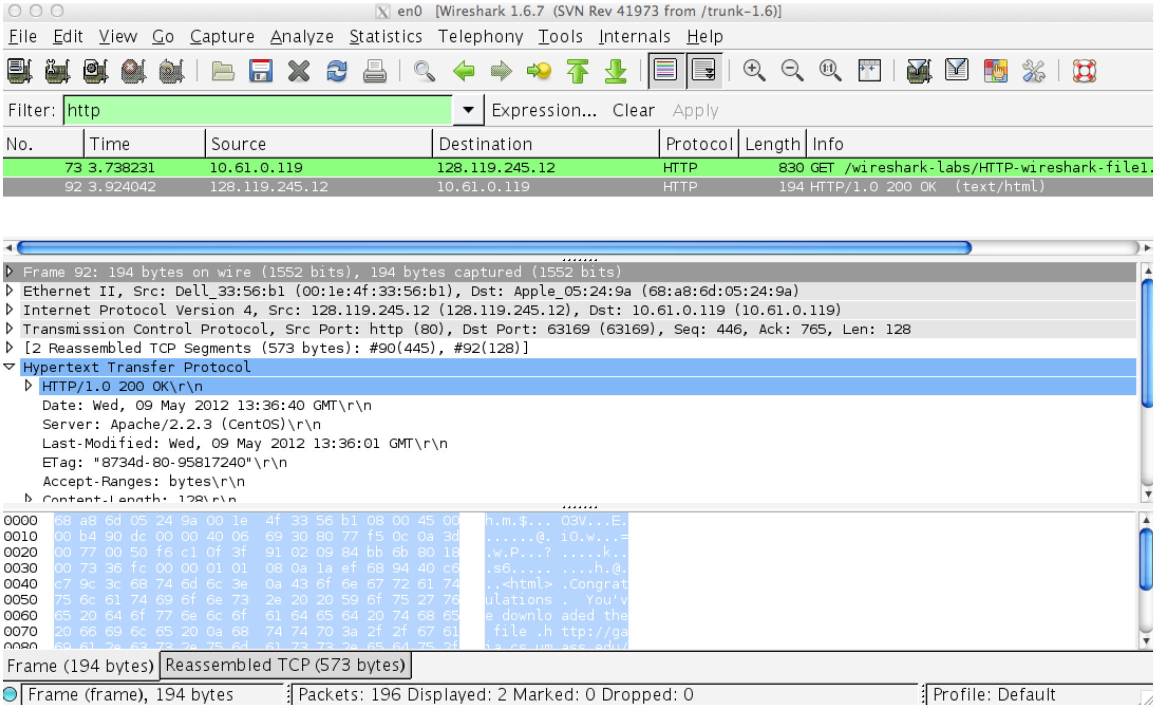
1. The Basic HTTP GET/response interaction

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 up your web browser.
2. Start up the Wireshark packet sniffer, as described in the Introductory lab (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're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).
3. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
4. Enter the following to your browser
   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html
   Your browser should display the very simple, one-line HTML file.
5. Stop Wireshark packet capture.

Your Wireshark window should look similar to the window shown in Figure 1. 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.[5]

[5] Download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file http-ethereal-trace-1. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the http-ethereal-trace-1 trace file. The resulting display should look similar to Figure 1. (The Wireshark user interface displays just a bit differently on different operating systems, and in different versions of Wireshark).


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  69 61 2e 63 73 2e 75 6d  61 73 73 2e 65 64 75 2f   ia.cs.um ass.edu/
```

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

Figure 1: Wireshark Display after http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-file1.html has been retrieved by your browser

The example in Figure 1 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 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).

> (*Note:* You should ignore any HTTP GET and response for favicon.ico. If you see a reference to this file, it is your browser automatically asking the server if it (the server) has a small icon file that should be displayed next to the displayed URL in your browser. We'll ignore references to this pesky file in this lab.).

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 (see the introductory 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. When you hand in your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our classes, we ask that students markup paper copies with a pen, or annotate electronic copies with text in a colored font).

1. What version of HTTP are the server and your browser running?
2. What languages (if any) does your browser indicate that it can accept to the server?
3. What are the IP addresses of your computer and the gaia.cs.umass.edu server?
4. What is the status code sent from your browser to the server and returned from the server to your browser?
5. Which flags are set in the GET HTTP message? (You can find these under "Transmission Control Protocol" -> "flags")
6. How many bytes of content are being returned to your browser?
7. 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.

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.

2. The HTTP CONDITIONAL GET/response interaction

Recall from Section 2.2.6 of the text, 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 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-listing window.
- (*Note:* If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-2 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author's computers.)

Answer the following questions:

8. 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? If so, what information follows the "IF-MODIFIED-SINCE:" header?
9. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell? If so, what was the content?
10. 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? What does that information refer to?
11. Was the HTTP status code and phrase returned from the server in response to this second HTTP GET the same as the first? Did the server explicitly return the contents of the file? Why?

3. 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:

- 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-file3.html
  Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.
- (*Note:* If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-3 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author's computers.)

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 from Section 2.2 (see Figure 2.9 in the text) 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 (see Figure 1.24 in the text). 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:

12. 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?
13. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
14. What is the status code and phrase in the response?
15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

4. 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:

- 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-file4.html Your browser should display a short HTML file with two images. 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. Our publisher's logo is retrieved from the www.aw-bc.com web site. The image of the cover for our 5th edition (one of our favorite covers) is stored at the manic.cs.umass.edu server.
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.
- (*Note:* If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-4 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author's computers.)

Answer the following questions:

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

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

5 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 http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html is password protected. The username is "wireshark-students" (without the quotes), and the password is "network" (again, without the quotes). So let's access this "secure" password-protected site. Do the following:

- Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html
  Type the requested user name and password into the pop up box.
- 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.
- (*Note:* If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-5 packet trace to answer the questions below; see footnote 2. This trace file was gathered while performing the steps above on one of the author's computers.)

Now let's examine the Wireshark output. You might want to first read up on HTTP authentication by reviewing the easy-to-read material on "HTTP Access Authentication Framework" at http://frontier.userland.com/stories/storyReader$2159

Answer the following questions:

18. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser? Explain?
19. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message? Why?

The username (wireshark-students) and password (network) that you entered are encoded in the string of characters (d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=) following the "`Authorization: Basic`" header in the client's HTTP GET message. While it may appear that your username and password 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 http://www.motobit.com/util/base64-decoder-encoder.asp and enter the base64-encoded string d2lyZXNoYXJrLXN0dWRlbnRz 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 Om5ldHdvcms= 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.

Fear not! As we will see in Chapter 8, there are ways to make WWW access more secure. However, we'll clearly need something that goes beyond the basic HTTP authentication framework!