Homework+Lab 6: Due at start of class on 29-Apr.

Please complete all eight homework problems, and the seventeen lab problems.

Homework Problems

1. Stallings Problem 7.3.
2. Stallings Problem 7.4.
3. Stallings Problem 16.11.
4. Stallings Problem 16.12.
5. Suppose that

   $$t_{\text{frame}} = \text{1.000 millisecond (time to transmit one frame)}$$
   $$t_{\text{prop}} = \text{1.500 milliseconds (propagation time of medium, same in both directions)}$$

   with ACK and processing times is assumed to be negligible. The total time to deliver n frames starts at the beginning of the transmission of the first frame and finishes at the end of the receipt of the acknowledgement.

   (a) Suppose that no frames or acknowledgments are lost or damaged. Compute the amount of time required to deliver 10 frames to the receiver if the sender and receiver use **stop-and-wait** error control.

   (b) Suppose that no frames or acknowledgments are lost or damaged. Compute the amount of time required to deliver 10 frames to the receiver if the sender and receiver use **Go-Back-N** error control with a window size of 4 frames (the sender will wait to transmit more frames if there are 4 unacknowledged frames outstanding). Assume that the receiver sends an RR (receive ready) acknowledgment for every correctly received frame and that the sender/receiver have a full-duplex link.

   (c) Now we consider the case where an acknowledgment (ACK or RR) for a frame has been lost. The sender's timeout period is given as 6 milliseconds and begins at the completion of the transmitted frame. Compute the amount of time required to deliver 10 frames to the receiver, given that the acknowledgment (ACK) for frame 5 is lost (recall the first frame is labeled frame 0), if the sender and receiver use **stop-and-wait** error control.

   (d) Under the same assumptions, compute the amount of time required to deliver 10 frames to the receiver, given that the acknowledgment (RR) for frame 5 is lost (recall the first frame is labeled frame 0), if the sender and receiver use **Go-Back-N** error control with a window size of 4 frames (the sender will wait to transmit more frames if there are 4 unacknowledged frames outstanding). Assume that the receiver sends an RR (receive ready) acknowledgment for every correctly received frame and that the sender/receiver have a full-duplex link.

6. Stallings Problem 8.1.
7. Stallings Problem 8.12.
8. Stallings Problem 8.13.

Lab 6: Network Access Layer – Ethernet [adapted from J. Kurose and K.W. Ross]

In this lab, we'll investigate the Ethernet protocol. Before beginning this lab, please skim section 11.2 (link layer addressing), 12.1 (Ethernet), and pages 449-450 (ARP) in the text. Note that the ARP protocol is used by an IP device to determine the IP address of a remote interface whose Ethernet MAC address is known.

Part I: Capturing and analyzing Ethernet frames

Let's begin by capturing a set of Ethernet frames to study. Do the following:

- First, make sure to the http protocol is enabled by going into Analyze→Enabled Protocols menu. Next, make sure your browser's cache is empty. To do this under Internet Explorer, select Tools→Internet Options→Delete Files. For Firefox select Tools→Clear Private Data.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
  `http://spinlab.wpi.edu/wireshark/lab6.html`
  Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture. First, find the packet numbers (the leftmost column in the upper Wireshark window) of the HTTP GET message that was sent from your computer to `spinlab.wpi.edu`, as well as the beginning of the HTTP response message sent to your computer by `spinlab.wpi.edu`. You should see a screen that looks something like Fig. 1 (where packet 4 in the screen shot below contains the HTTP GET message).

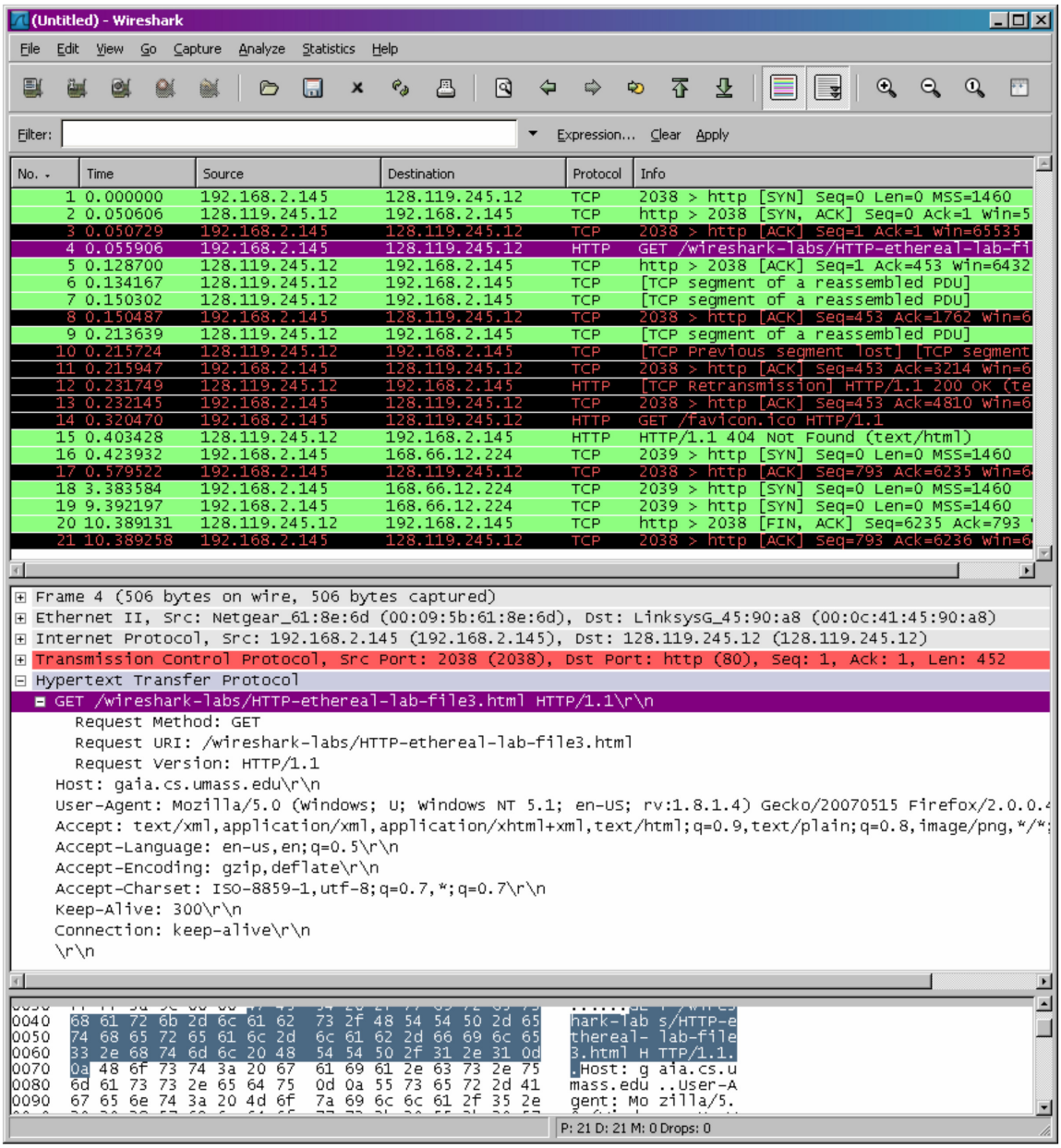

Figure 1: Example Wireshark trace

- Since this lab is about Ethernet and ARP, we're not interested in IP or higher layer protocols. So let's change Wireshark's "listing of captured packets" window so that it shows information only about protocols below IP. To have Wireshark do this, select Analyze→Enabled Protocols. Then uncheck the IP box (or the IPv4 box) and select OK. You should now see a Wireshark window that looks like Fig. 2.

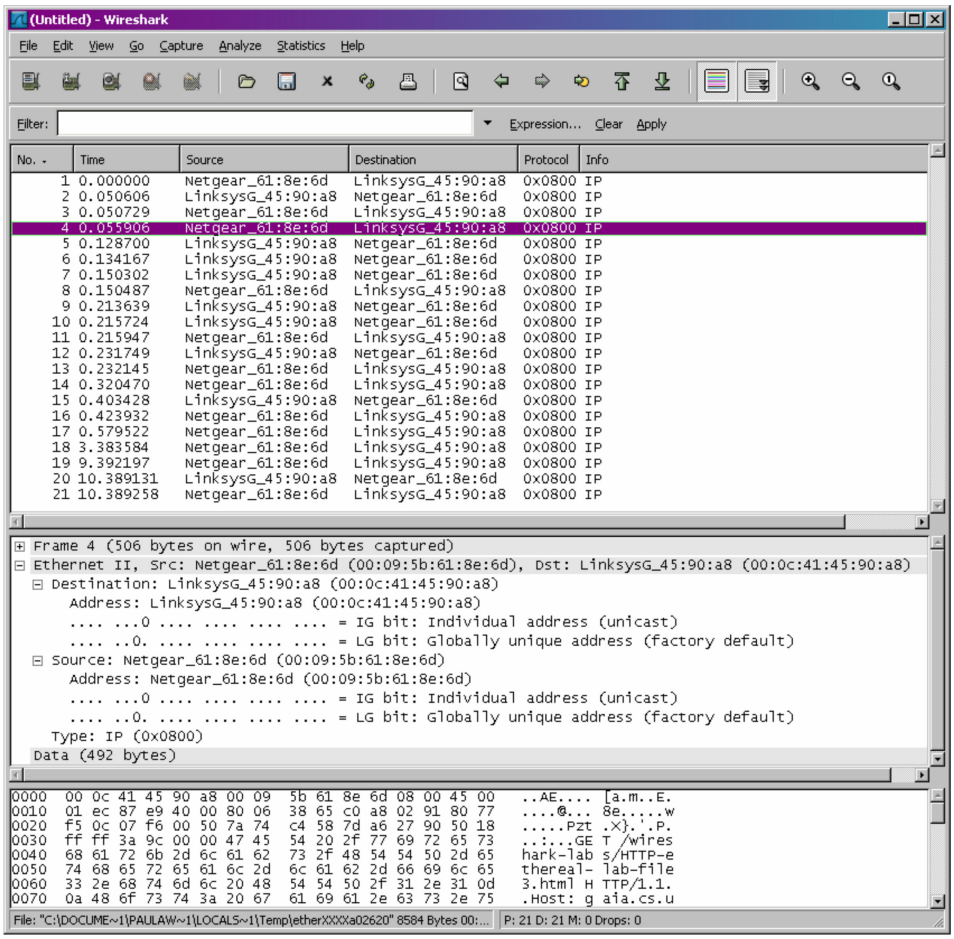

Figure 2: Example Wireshark trace (IP and higher layers omitted)

In order to answer the following questions, you'll need to look into the packet details and packet contents windows (the middle and lower display windows in Wireshark). Select the Ethernet frame containing the HTTP GET message. (Recall that the HTTP GET message is carried inside of a TCP segment, which is carried inside of an IP datagram, which is carried inside of an Ethernet frame; review Figure 2.5 in the text if you still find this nesting a bit confusing). Expand the Ethernet information in the packet details window. Note that the contents of the Ethernet frame (header as well as payload) are displayed in the packet contents window.

Answer the following questions, based on the contents of the Ethernet frame containing the HTTP GET message. Whenever possible, when answering a question you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout 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.

1. What is the 48-bit Ethernet address of your computer?

2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of `spinlab.wpi.edu`? What device has this as its Ethernet address?

3. What is the hexadecimal value for the two-byte Frame type field? This field is also sometimes called the EtherType, and specifies what higher layer protocol is contained within this

Ethernet frame. Using the table in the Wikipedia article on "EtherType" (or, just using the middle window on Wireshark), what is the higher layer protocol that is encapsulated in this Ethernet frame?

4. There should be between 54 to 66 bytes from the very start of the Ethernet frame to the ASCII "G" in the http "GET". How many bytes are there, and what three things do you think these bytes contain? Hint: Think of which layers are below http.

Next, answer the following questions (which are very similar to the above questions), based on the contents of the Ethernet frame containing the first byte of the HTTP *response* message.

5. What is the value of the Ethernet source address? Is this the address of your computer, or of `spinlab.wpi.edu`? What device has this as its Ethernet address?

6. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?

7. What is the hexadecimal value for the two-byte Frame type field?

8. How many bytes from the very start of the Ethernet frame does the ASCII "O" in "OK" (i.e., the HTTP response code) appear in the Ethernet frame?

Part II: The Address Resolution Protocol

In this section, we'll observe the ARP protocol in action. It may be helpful to also review the Wikipedia article on ARP which can be found here:
`http://en.wikipedia.org/wiki/Address_Resolution_Protocol`.

ARP Caching
The ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your computer. The `arp` command (in both Windows and Linux/OSX) is used to view and manipulate the contents of this cache. Since the `arp` command and the ARP protocol have the same name, it's understandably easy to confuse them. But keep in mind that they are different — the `arp` command is used to view and manipulate the ARP cache contents, while the ARP protocol defines the format and meaning of the messages sent and received, and defines the actions taken on message transmission and receipt. Let's take a look at the contents of the ARP cache on your computer:

- Windows. While it may depend on your version of Windows, the `arp` command is generally found in `c:/windows/system32`, so type either "`arp`" or "`c:/windows/system32/arp`" in the Windows command line (without quotation marks)[1].

- OSX. The executable for the `arp` command can be in various places. Popular locations are `/usr/sbin/arp` and `/usr/etc/arp`.

The `arp` command with no arguments will display the contents of the ARP cache on your computer (on OS X you will need to type `arp -a`). Run the `arp` command.

[1]If you have difficulty running the `arp` command in Windows, you may need to run it as administrator. In most versions of Windows, this can be accomplished by right-clicking on the Command Prompt icon (under Accessories), and selecting "Run as Administrator".

9. Write down the contents of your computer's ARP cache. What is the meaning of each column value?

In order to observe your computer sending and receiving ARP messages, we'll need to clear the ARP cache, since otherwise your computer is likely to find a needed IP-Ethernet address translation pair in its cache and consequently not need to send out an ARP message.

- Windows. The Windows `arp -d *` command will clear your ARP cache. The -d flag indicates a deletion operation, and the * is the wildcard that says to delete all table entries.
- OSX. The `sudo arp -a -d` command will clear your ARP cache. In order to run this command you'll need root privileges.
- Linux. Unfortunately, you may have to delete all ARP entries manually on Linux. The command `arp -d [IP address]` will delete an entry in your ARP cache. Please delete all entries. In order to run this command you'll need root privileges.

Observing ARP in action
Do the following:

- Close *all* running programs, even those in the background.
- Open your browser, make sure your browser's cache is empty. (To do this under Internet Explorer, select Tools→Internet Options→Delete Files.)
- Enter the following URL into your browser `http://spinlab.wpi.edu/wireshark/lab6.html`, but *don't* press Enter (i.e. don't load the page yet – just have the address typed in and ready to go).
- Load the Wireshark packet sniffer, but don't start collecting packets yet.
- Clear your ARP cache, as described above.
- Start collecting packets in Wireshark.
- Load the webpage on your browser by pressing the "Enter" key while in the browser address bar. The browser should again display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture. Again, we're not interested in IP or higher-layer protocols, so change Wireshark's "listing of captured packets" window so that it shows information only about protocols below IP. To have Wireshark do this, select Analyze→Enabled Protocols. Then uncheck the IP box and select OK. You should now see a Wireshark window that looks that shown in Fig. 3.

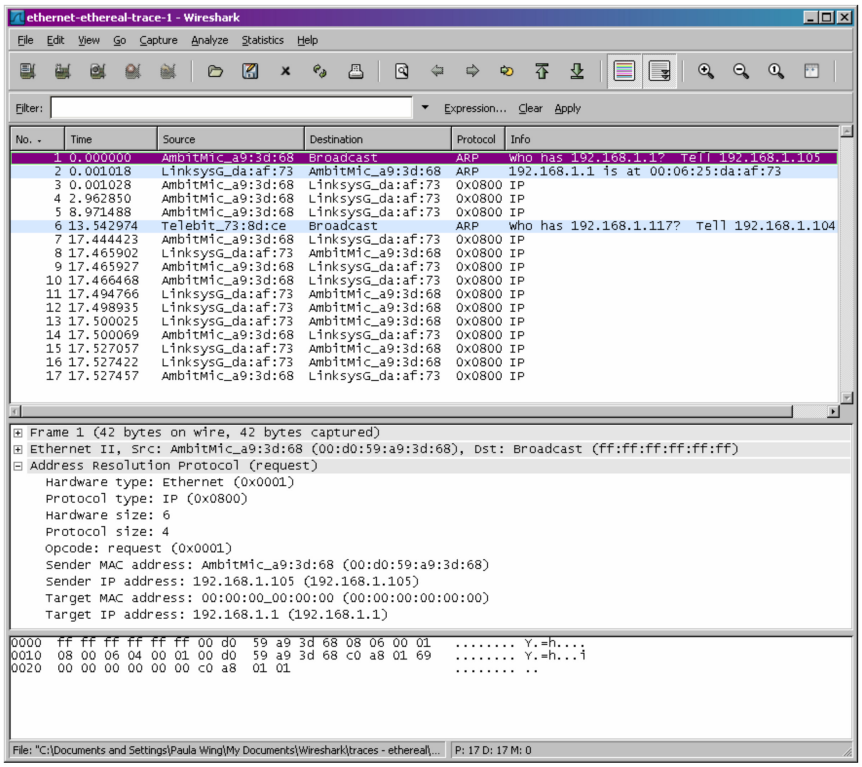

Figure 3: Example trace for observing ARP

In the example above, the first two frames in the trace contain ARP messages (as does the 6th message)[2]. First, click on the ARP *request* message and answer the following questions:

10. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message? Explain what these values mean.

11. What is the hexadecimal value for the two-byte Ethernet Frame type field. Again, you may want to consult the Wikipedia article on EtherType.

Review the ARP article on Wikipedia at `http://en.wikipedia.org/wiki/Address_Resolution_Protocol`.

12. As you can see from the ARP header format in the Wikipedia article, the ARP opcode field begins 6 bytes (48 bits) from the beginning of the ARP frame. Since the Ethernet frame (consisting of 6-byte source and 6-byte destination MAC addresses, as well as 2-byte Frame type) is 14 bytes long, the opcode appears 20 bytes from the start of the packet. What is the value of the opcode field within the ARP payload, and what does it mean?

13. Does the ARP message contain the IP address of the sender?

14. Where in the ARP request does the "question" appear, i.e. the Ethernet address of the machine whose corresponding IP address is being queried?

[2]If you don't see any ARP packets, it is probably because you didn't clear your ARP cache sucessfully. Or, maybe you had some other program running that was attempting to use the Ethernet, which consequently repopulated your ARP cache. Try the last 4 steps above again in very quick succession.

Now find the ARP *reply* that was sent in response to the ARP request.

15. What is the value of the opcode field within the ARP-payload part of the Ethernet frame in which an ARP response is made?

16. Where in the ARP message does the "answer" to the earlier ARP request appear — the IP address of the machine having the Ethernet address whose corresponding IP address is being queried?

17. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP reply message?