TCPIPLAB:

--- Adapted from the original lab by J.F. Kurose and K.W. Ross

NOTE:
✓ **[COURSE_URL] in following text is specified on the course web page.**

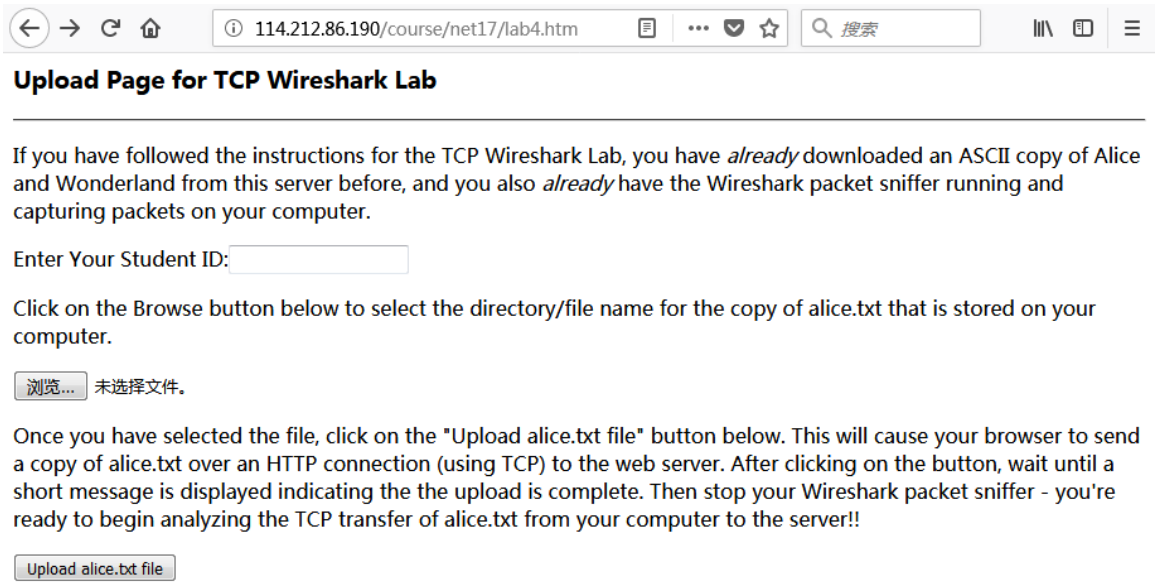
In this lab, we'll investigate the behavior of the TCP protocol in detail. We'll do so by analyzing a trace of the TCP segments sent and received in transferring a 150KB file (containing the text of Lewis Carrol's *Alice's Adventures in Wonderland*) from your computer to a remote server. We'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer; and we'll look at TCP's receiver-advertised flow control mechanism. We'll also briefly consider TCP connection setup and we'll investigate the performance (throughput and round-trip time) of the TCP connection between your computer and the server.

1. Capturing a bulk TCP transfer from your computer to a remote server

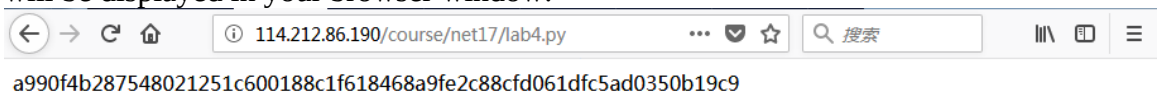
Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method. We're using the POST method rather than the GET method as we'd like to transfer a large amount of data *from* your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

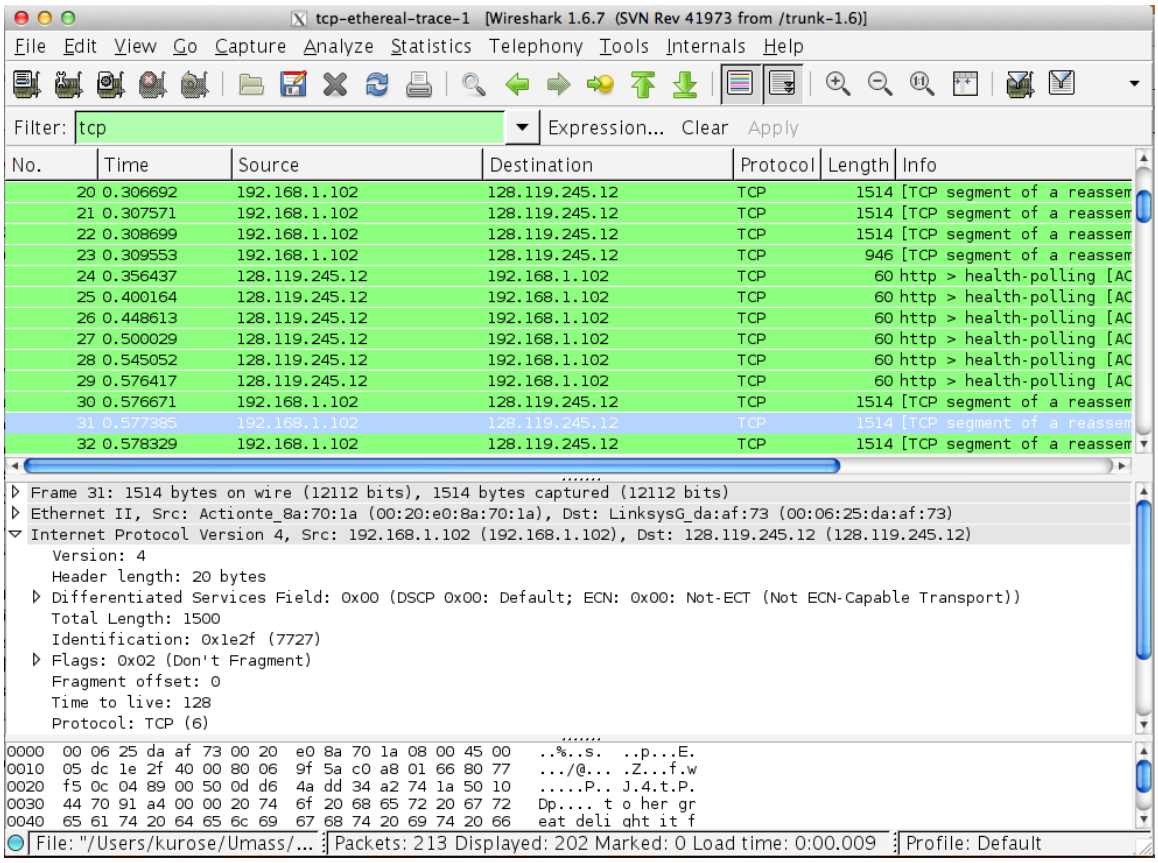
- Start up your web browser. Go the http://[COURSE_URL]/alice.txt and retrieve an ASCII copy of *Alice in Wonderland.* Store this file somewhere on your computer.
- Next go to http://[COURSE_URL]/tcpiplab.htm
- You should see a screen that looks like:



- Use the *Browse* button in this form to enter the name of the file (full path name) on your computer containing *Alice in Wonderland* (or do so manually). Don't yet press the "*Upload alice.txt file*" button.
- Now start up Wireshark and begin packet capture *(Capture->Start)* and then press *OK* on the Wireshark Packet Capture Options screen (we'll not need to select any options here).
- Returning to your browser, press the "*Upload alice.txt file*" button to upload the file to the server. Once the file has been uploaded, a short message like below will be displayed in your browser window.



- Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown below. – **NOTE: The IPs and Hostnames in the screen shots below are not actual and are for illustrative purpose only.**



2. A first look at the captured trace

Before analyzing the behavior of the TCP connection in detail, let's take a high level view of the trace.

- First, filter the packets displayed in the Wireshark window by entering "tcp" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.

What you should see is series of TCP and HTTP messages between your computer and the course server. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of Wireshark you are using, you might see a series of "HTTP Continuation" messages being sent from your computer to the server. Note that there is actually no such thing as an HTTP Continuation message – this is Wireshark's way of indicating that there are multiple TCP segments being used to carry **a single HTTP message**. In more recent versions of Wireshark, you'll see "[TCP segment of a reassembled PDU]" in the Info column of the Wireshark display to indicate that this TCP segment contained data that belonged to an upper layer protocol message (in our case here, HTTP). You should also see TCP ACK segments being returned from the server to your computer.

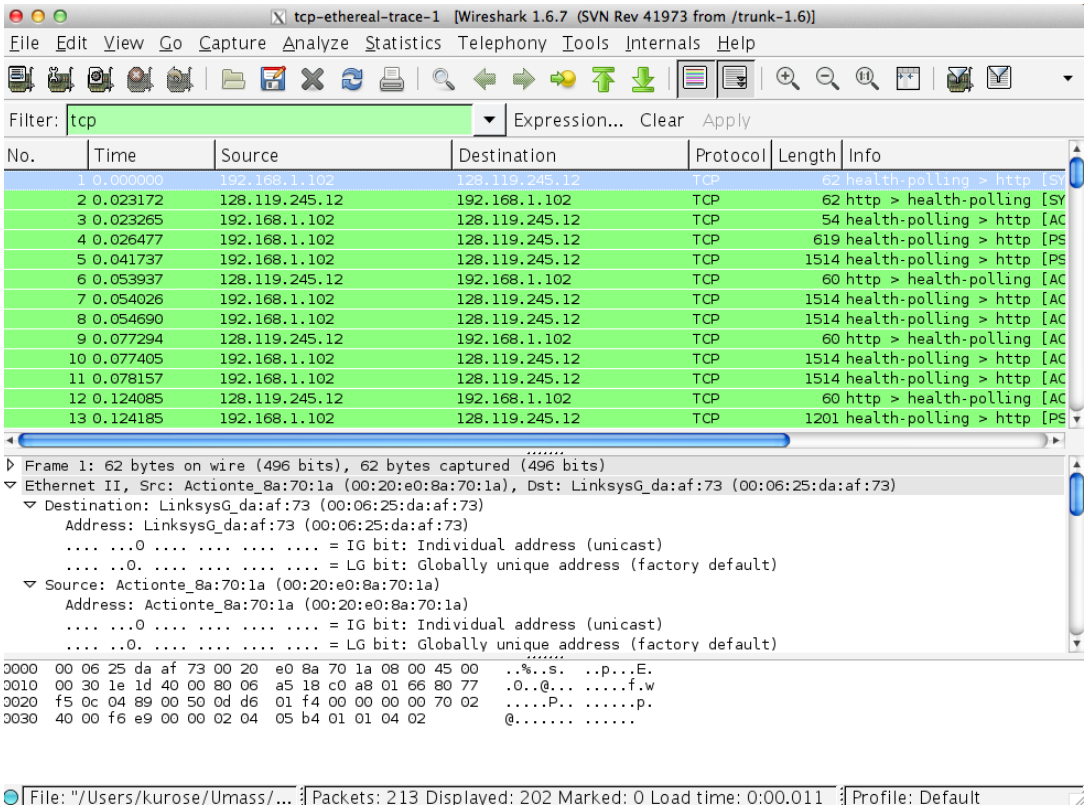
Answer the following questions. **Whenever possible, when answering a question you should hand in a printout of the packet(s) within the trace that you used to answer the question asked.** ***Annotate the printout[1] to explain your answer***. To print a packet, use *File->Print*, choose *Selected packet only*, choose *Packet summary line*, and select the minimum amount of packet detail that you need to answer the question.

Locate the HTTP POST message. Based on it, answer following questions:

1. What is the IP address and TCP port number used by your client computer (source) that is transferring the file to the course server? To answer this question, it's probably easiest to explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window".
2. On what port number is the server sending and receiving TCP segments for this connection?
3. Within the IP packet header, what is the value in the upper layer protocol field?
4. How many bytes are in the IP header? How many bytes are in the payload *of the IP datagram*? Explain how you determined the number of payload bytes.
5. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.
6. What is the value in the Identification field and the TTL field of the IP packet?

Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select *Analyze->Enabled Protocols.* Then uncheck the HTTP box and select *OK*. You should now see a Wireshark window that looks like:

[1] 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.



This is what we're looking for - a series of TCP segments sent between your computer and the Web server. We will use the packet trace that you have captured to study TCP behavior in the rest of this lab.

3. TCP Basics

Answer the following questions for the TCP segments:

7. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and the server? What is it in the segment that identifies the segment as a SYN segment?
8. What is the sequence number of the SYNACK segment sent by the server to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did the server determine that value? What is it in the segment that identifies the segment as a SYNACK segment?
9. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.
10. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received?

Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the `Estimated RTT` value after the receipt of each ACK? Assume that the value of the `Estimated RTT` is equal to the measured RTT for the first segment, and then is computed using the `Estimated RTT` equation for all subsequent segments.

> *Note:* Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the server. Then select: *Statistics->TCP Stream Graph->Round Trip Time Graph.*

11. What is the length of each of the first six TCP segments?
12. What is the minimum amount of available buffer space advertised by the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
13. Are there any retransmitted segments? What did you check for (in the trace) in order to answer this question?
14. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment?
15. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

How to hand in

Put the above answers into a PDF file **named with your student ID and lab name only**, e.g. “123456789_TCPIPLAB.pdf”, then follow the instructions on the course web page to upload your file to the submission server.