CS 640, Homework Assignment # 1
Assigned: April 22, 2008, Due April 29, 2008

(adapted almost in entirety from Wireshark TCP Lab of Kurose, Ross textbook 4th edition)

In this lab, we'll investigate the behavior of TCP 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; we'll see TCP's congestion control algorithm – slow start and congestion avoidance – in action; 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.

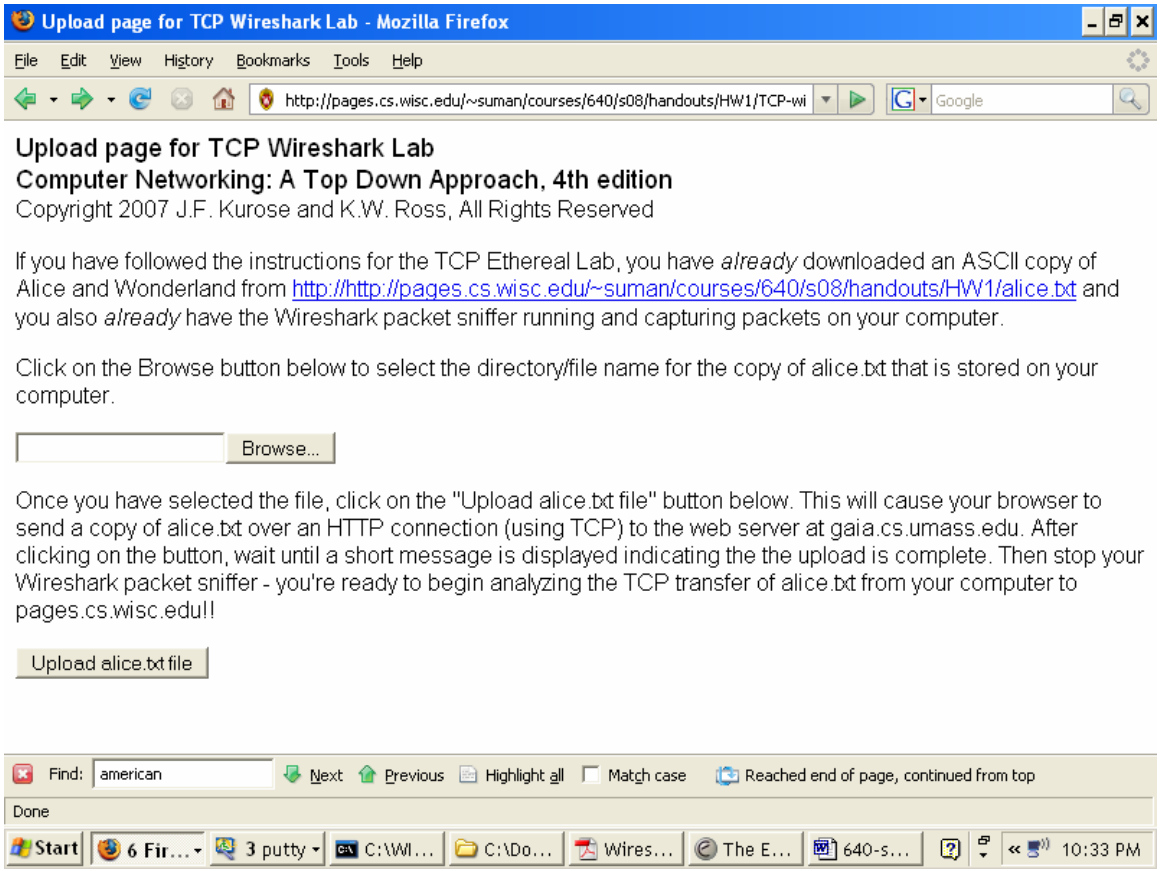
Before beginning this lab, you'll probably want to review sections 3.5 and 3.7 in the text.

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

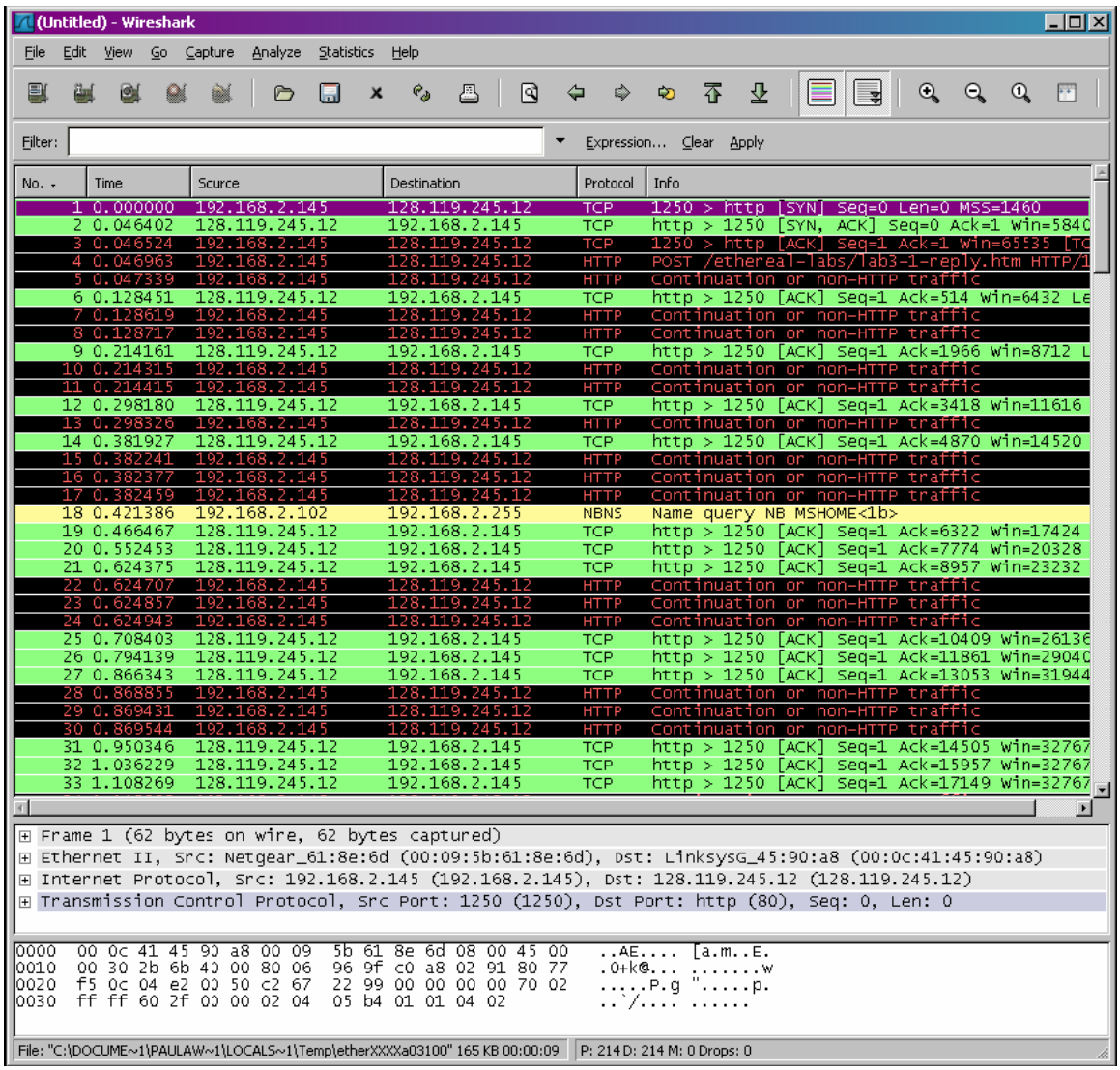
Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method (see section 2.2.3 in the text). 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 to collect your own trace.

- Start up your web browser. Go the http://pages.cs.wisc.edu/~suman/courses/640/s08/handouts/HW1/alice.txt and retrieve an ASCII copy of *Alice in Wonderland.* Store this file somewhere on your computer.
- Next go to http://pages.cs.wisc.edu/~suman/courses/640/s08/handouts/HW1/TCP-wireshark-file1.html.
- 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->Options)* 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 pages.cs.wisc.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
• Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown below.



If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above, with the destination being a host named gaia.cs.umass.edu (IP address 128.119.245.12). This trace file is available at http://pages.cs.wisc.edu/~suman/courses/640/s08/handouts/HW1/tcp-ethereal-trace-1.pcap. (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 tcp-ethereal-trace-1.pcap trace file.)

In this assignment you will use this provided trace to answer all the following posed questions. But for practice, you should try to verify the answers for your own trace as well.

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.

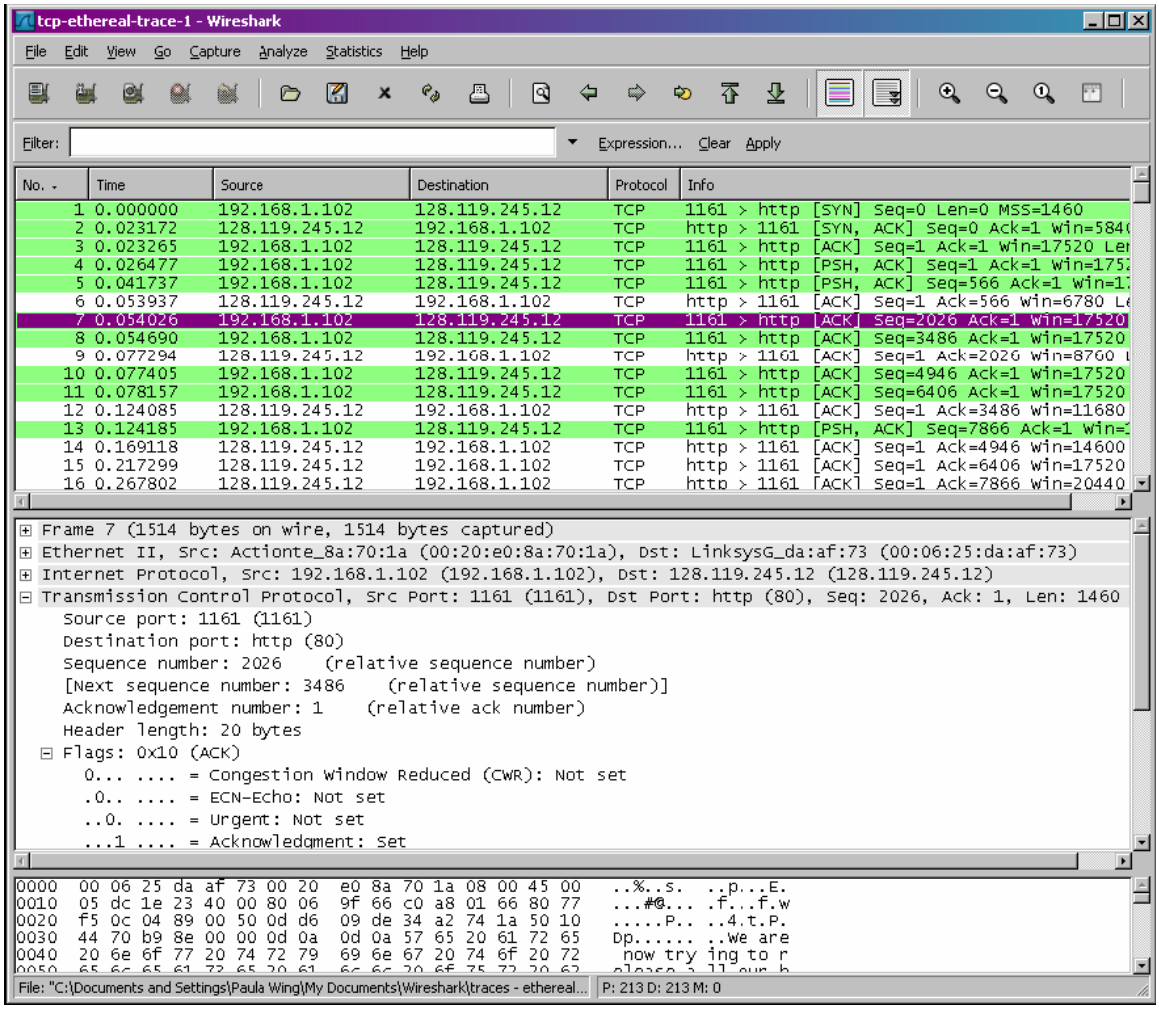
What you should see is series of TCP and HTTP messages between your computer and pages.cs.wisc.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message and a series of “HTTP Continuation” messages being sent from the source computer to gaia.cs.umass.edu. You should also see TCP ACK segments being returned from pages.cs.wisc.edu to your computer.

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 IP address and TCP port number used by the source computer that is transferring the file to gaia.cs.umass.edu? To answer this question, it’s probably easiest to select an HTTP message and 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 destination host (gaia.cs.umass.edu) sending and receiving TCP segments for this connection?

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:



This is what we're looking for - a series of TCP segments sent between the source computer and gaia.cs.umass.edu. We will continue to use this packet trace to study TCP behavior in the rest of this lab.

3. TCP Basics

Answer the following questions for the TCP segments:

3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the source computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

4. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the source computer in reply to the SYN? What is the value of the ACKnowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

5. 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.

6. 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 `EstimatedRTT` value (see page 249 in text) after the receipt of each ACK? Assume that the value of the `EstimatedRTT` is equal to the measured RTT for the first segment, and then is computed using the `EstimatedRTT` equation on page 249 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 source computer to the server. Then select: *Statistics->TCP Stream Graph-> Round Trip Time Graph.*

7. What is the length of each of the first six TCP segments?

8. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

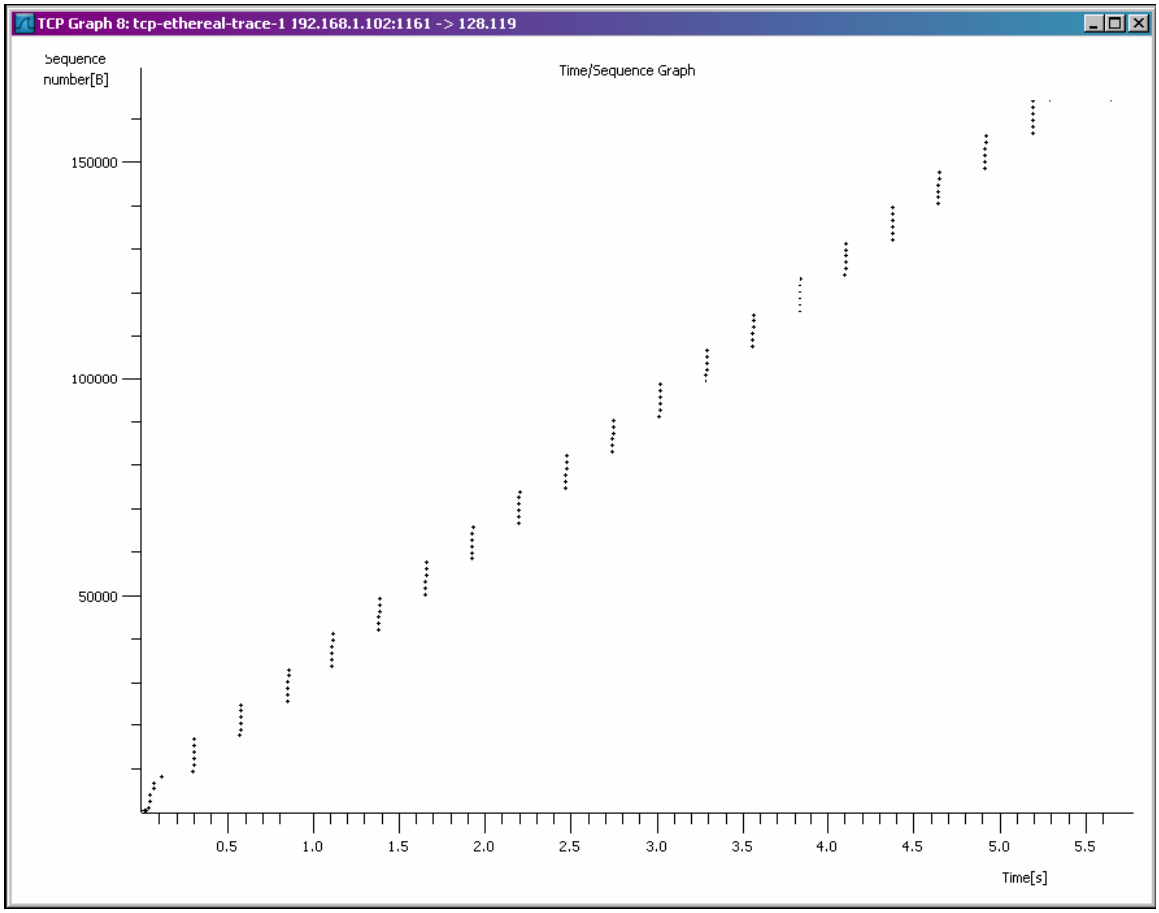
9. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

10. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 257 in the text).

11. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

4. TCP congestion control in action

Let's now examine the amount of data sent per unit time from the source computer to the server.
Rather than (tediously!) calculating this from the raw data in the Wireshark window, we'll use one of Wireshark's TCP graphing utilities - *Time-Sequence-Graph(Stevens*) - to plot out data.
• Select a TCP segment in the Wireshark's "listing of captured-packets" window.
Then select the menu : *Statistics->TCP Stream Graph-> Time-Sequence-Graph(Stevens*).
You should see a plot that looks similar to the following plot, which was created from the captured packets in the provided packet:



Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

Now answer the following question for the trace file.

12. Use the *Time-Sequence-Graph(Stevens*) plotting tool to view the sequence number versus time plot of segments being sent from the source computer to the gaia.cs.umass.edu server. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.