ICSI 516 Fall 17, Homework 3 – Transport Layer
20 points
Due date: Wednesday, 10/11 at 11:59PM as a single PDF file via Blackboard

All parts of the assignment are to be completed independently. Submission of the same text by multiple students will be considered cheating. Students caught cheating will receive 0 points for the assignment and will be reported.

Part 1 [14 points]: Answer the theoretical questions on the following two pages.
Part 2 [6 points]: Complete the Wireshark Lab and answer the questions in the lab. You can download the Wireshark trace for this assignment from
http://www.cs.albany.edu/~mariya/courses/csi516F17/hw/tcp-wireshark-trace-1

[Theoretical questions]

1. [1 point] Describe why an application developer might choose to run an application over UDP rather than TCP.

2. [1 point] Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain.

3. [1 point] Consider the basic operations of TCP. Is TCP a Go-Back-N or a Selective Repeat protocol? Provide a brief justification.

4. [2 points] The pipelined protocol in Figure 1 has a window size of N packets and runs over a physical link with bandwidth R=100Mbit/s. The link has a propagation delay of 300ms and each of the packets sent over this link is of size L=5KBytes. What is the achieved throughput of the above pipelined protocol if N=4, 8 and 16? Write the formula you are using to calculate throughput (using the same notation as in the assignment) along with your result. Comment on the trends you observe with increasing N.
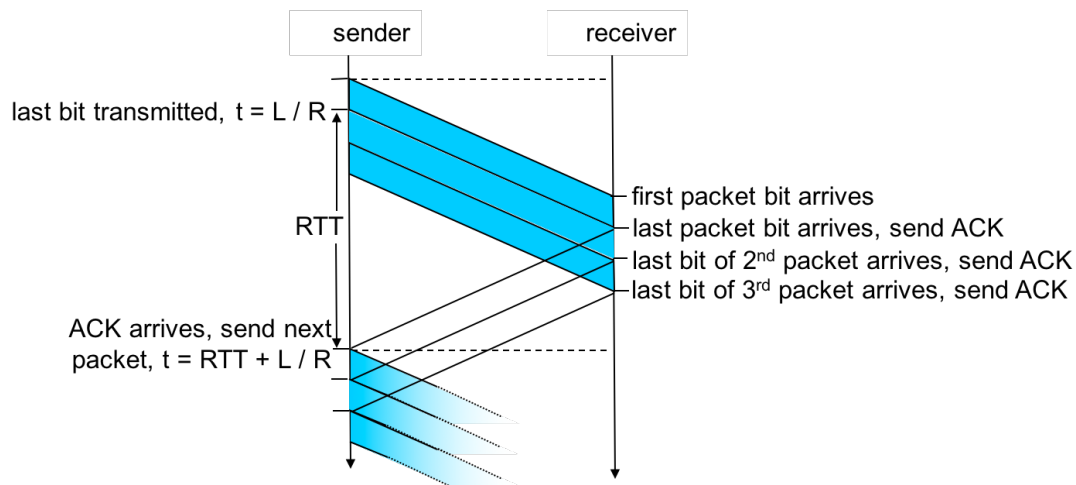

Figure1. Illustration of the pipelined protocol from theoretical problem 3.

5. [1 point] What are the similarities and differences between congestion control and flow control?

6. [1 point] We developed our realistic reliable data transfer (rdt3.0) protocol by assuming increasingly complicating circumstances (e.g. channel with error and loss) that required additional parameters to handle these circumstances. Why did we need to introduce sequence numbers? How about timers?

7. [2 points] How does congestion manifest itself in computer communication networks? Draw the delay vs. offered load (λ_{in}) curve for an idealized scenario with three senders and three receivers, one router in between, output link capacity of R, infinite buffer queues and no retransmissions.

8. [5 points] In this question we will apply our knowledge of the operations of TCP Reno in order to study its behavior through a real-world measurement. Figure 2 shows the congestion window of the protocol over multiple transmission rounds. Study the graph and answer the following questions:
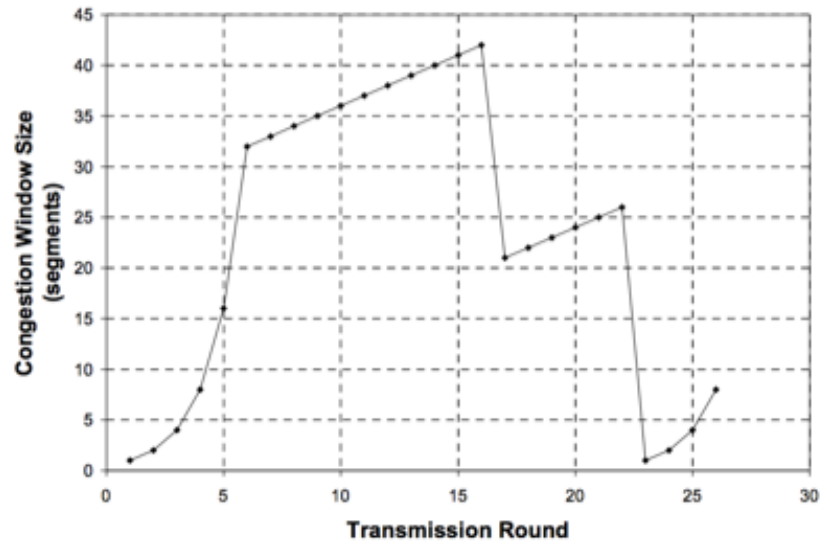

Figure 2. Graph of TCP Reno's congestion window over consecutive transmission rounds.

a. In which round intervals is *slow start* operating?

b. Which *congestion avoidance* mechanism can be observed here?

c. In which intervals is *congestion avoidance* operating?

d. What caused the cwnd decrease in round 16: a triple duplicate ACK or a timeout?

e. What is the value of ssthresh in round 1?

f. What is the value of ssthresh in round 18?

g. What is the value of ssthresh in round 24?

h. In which transmission round is the 70th segment sent?

i. Imagine that the sender received a triple duplicate ACK after the 26th round. What will be the values of cwnd and ssthresh following this event?

Wireshark Lab: TCP

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 a 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 the client 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 the client computer to a remote server

To allow you to investigate TCP's behavior, Wireshark was used to obtain a packet trace of the TCP transfer of a file from a client computer to a remote server. The trace was created by accessing a Web page that will allowed the user to enter the name of a file stored on the client 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). The POST method was used rather than the GET method because we'd like to transfer a large amount of data *from* the client computer to another computer. During the transfer, Wireshark was running to obtain the trace of the TCP segments sent and received from the client computer.

The Wireshark packet trace was captured using the following steps:

- The user visited the URL http://gaia.cs.umass.edu/Wireshark-labs/TCP-Wireshark-file1.html
- The user entered the name of the file containing the text of *Alice in Wonderland*.
- Before pressing the Upload button, the packet capture with Wireshark was started.
- Then the user pressed the "*Upload alice.txt file*" button to upload the file to the gaia.cs.umass.edu server. Once the file had been uploaded, a short congratulations message was be displayed in the browser window.
- Wireshark packet capture was then stopped.

If you load the tcp_wireshark_trace_1 file into Wireshark, your Wireshark window should look like the window shown below.
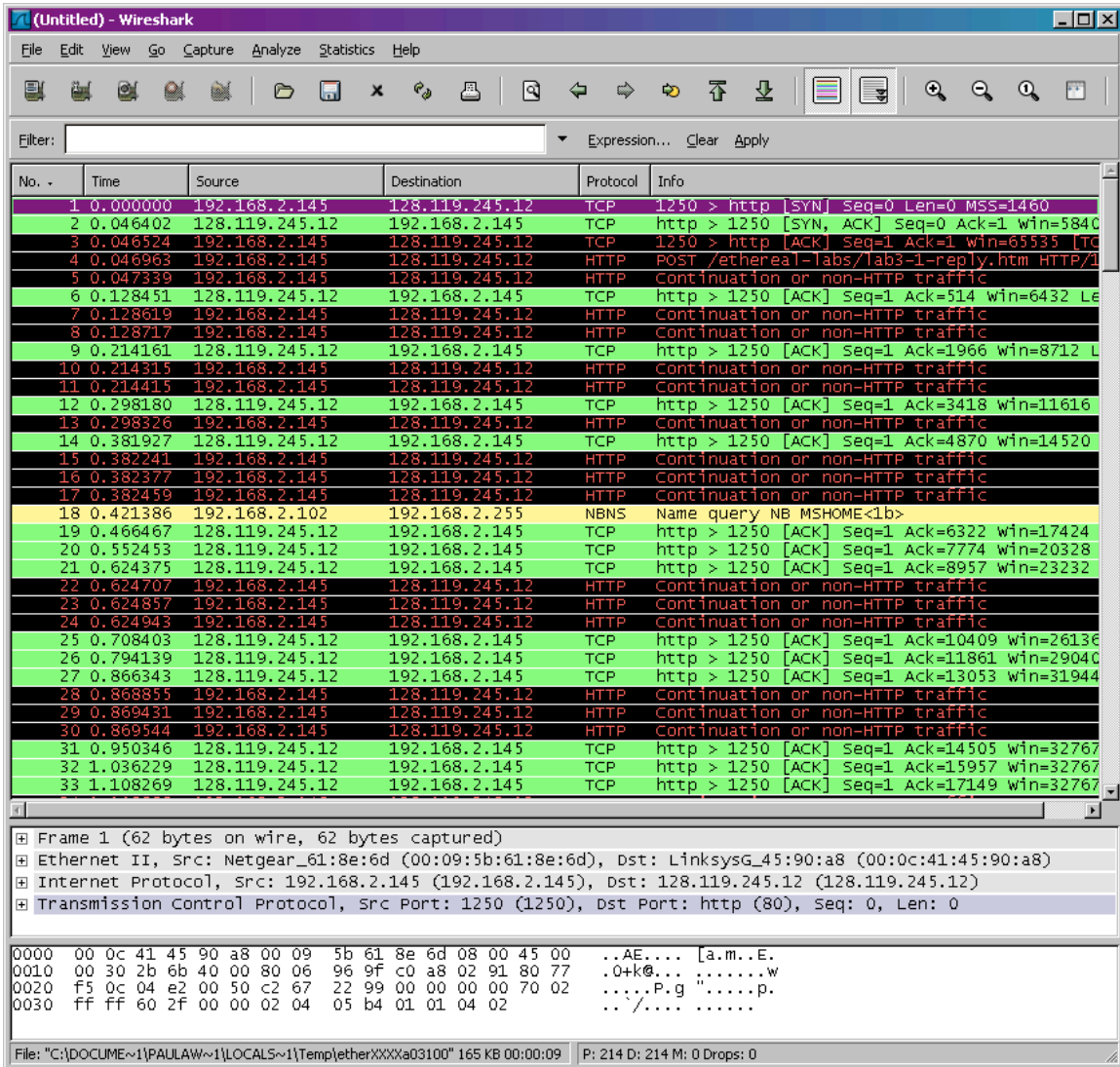

2. A first look at the captured trace

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" 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 the client computer and gaia.cs.umass.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 client computer to gaia.cs.umass.edu. Note that there is 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. You should also see TCP ACK segments being returned from gaia.cs.umass.edu to the client computer.

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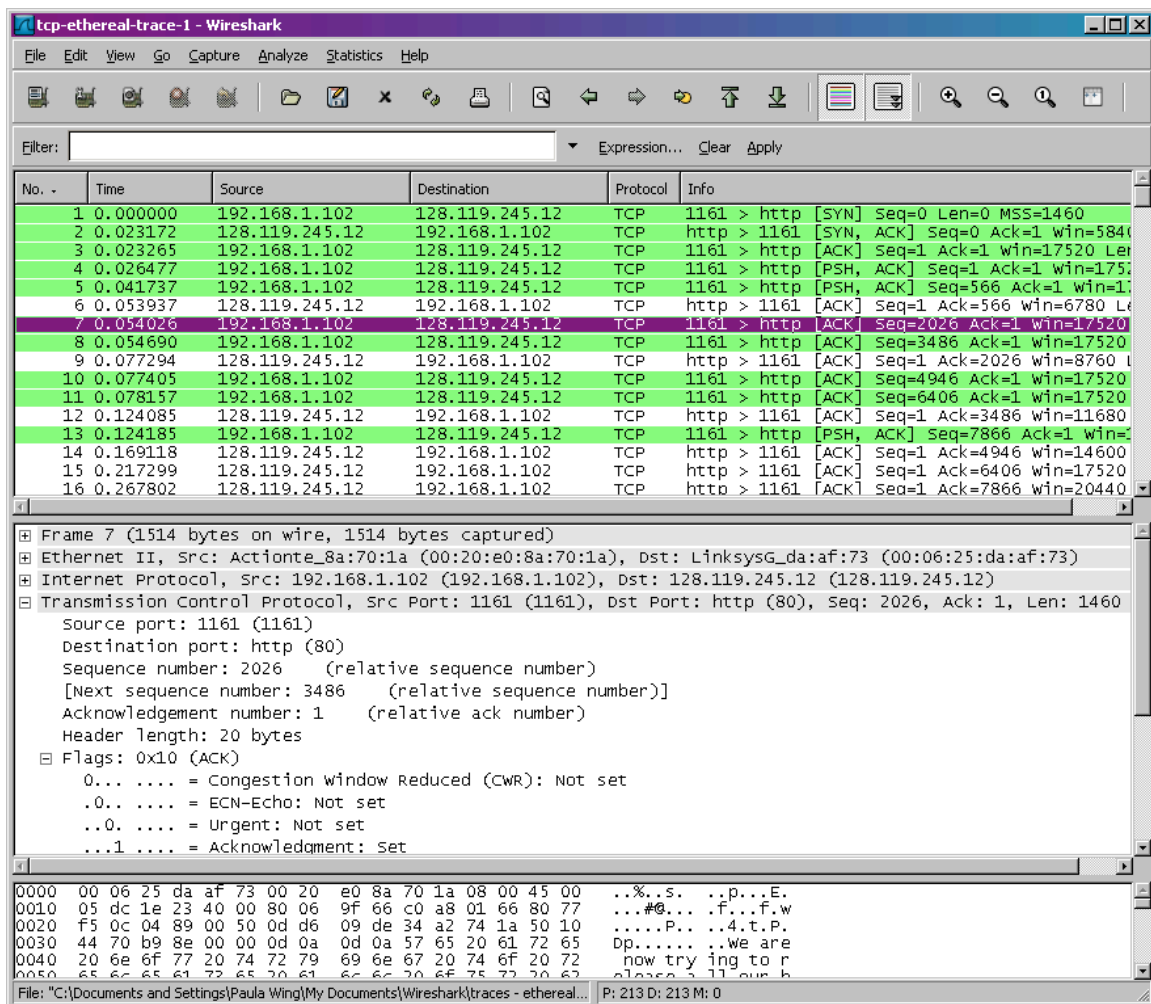
This is what we're looking for - a series of TCP segments sent between the client computer and gaia.cs.umass.edu. We will use the packet trace to study TCP behavior in the rest of this lab.

3. TCP Basics

Answer the following questions for the TCP segments:

1. What is the IP address and TCP port number used by the client computer (source) to transfer the file to gaia.cs.umass.edu? What is the IP address and port number used by gaia.cs.umass.edu to receive the file?
2. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?
3. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client 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?
4. 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.
5. 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 client to the gaia.cs.umass.edu server. Then select: *Statistics->TCP Stream Graph->Round Trip Time Graph.*

4. TCP congestion control in action

Let's now examine the amount of data sent per unit time from the client 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)*. To get the most interesting plot, select a packet with the source IP of 192.168.1.102, and select a packet towards the middle to end of the trace. You should see a plot that looks similar to the following plot:

TCP Graph 8: tcp-ethereal-trace-1 192.168.1.102:1161 -> 128.119

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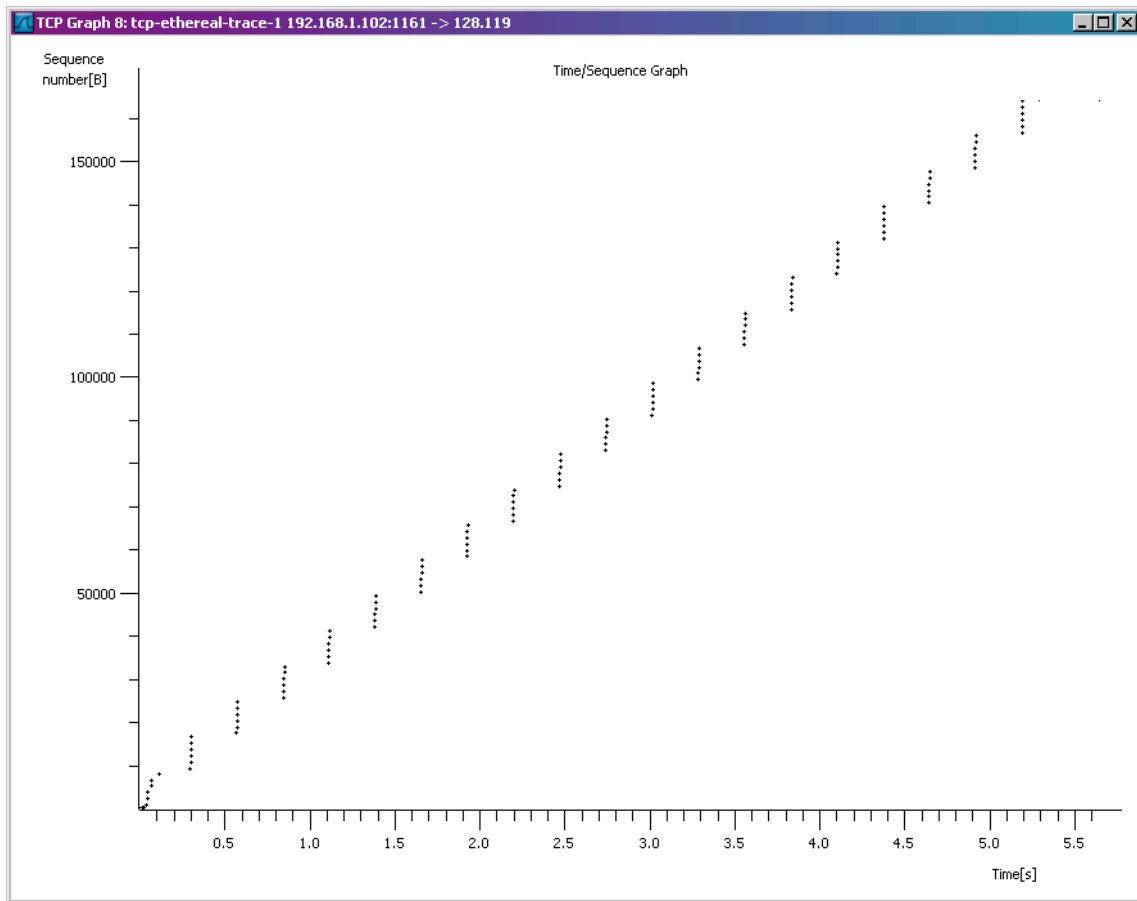
Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

Answer the following question:
6. Use the *Time-Sequence-Graph(Stevens)* plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.