Computer Networking and Network Programming (ECE 5650)

Lab 4

Instructions:

- **Original Work**: this assignment must represent the original work of the team members.
    - You must not look at other solutions or show your solutions to anyone else.
    - You must not collaborate or discuss the assignment with other groups or individuals.
    - You must not get any portion of the code from Internet sources, other current or prior students, or other groups or individuals.
    - You must not ask or pay others to help you with the solution.

  Policy on Cheating, Fabrication, and Plagiarism: Cheating, fabrication, plagiarism, and helping others to commit these acts are all forms of academic dishonesty, and they are wrong. Therefore, avoid all appearance of improper behavior! Students who witness cheating should report the incident to the instructor as soon as possible. Academic misconduct will result in at least failing the course.

- **Late Submissions**: no late submissions will be accepted.

- **Teamwork**: You must work with **one other student** and **must submit only one copy with both your names.**

- **Have questions?** Please contact the GTA/Grader. You can ask him by e-mail or send an e-mail to schedule a meeting on Microsoft Teams. No help will be given on the day the lab is due.

Objectives:

In this lab, we'll investigate the behavior of the celebrated TCP protocol in detail.

Description:

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 also see TCP's congestion control algorithm – slow start and congestion avoidance – in action. We'll also look at TCP's receiver-advertised flow control mechanism. We'll also briefly consider TCP connection setup and investigate the performance (throughput and round-trip time) of the TCP connection between your computer and the server.

Prerequisites:

- Review the class notes on TCP.

Submission Requirements:

- Submit a professional report in **pdf format** including your name(s) and your answers to all questions in the lab. **Only one file must be submitted with all screenshots included within.**

- The answers must be numbered in a consistent way to the questions in this assignment.

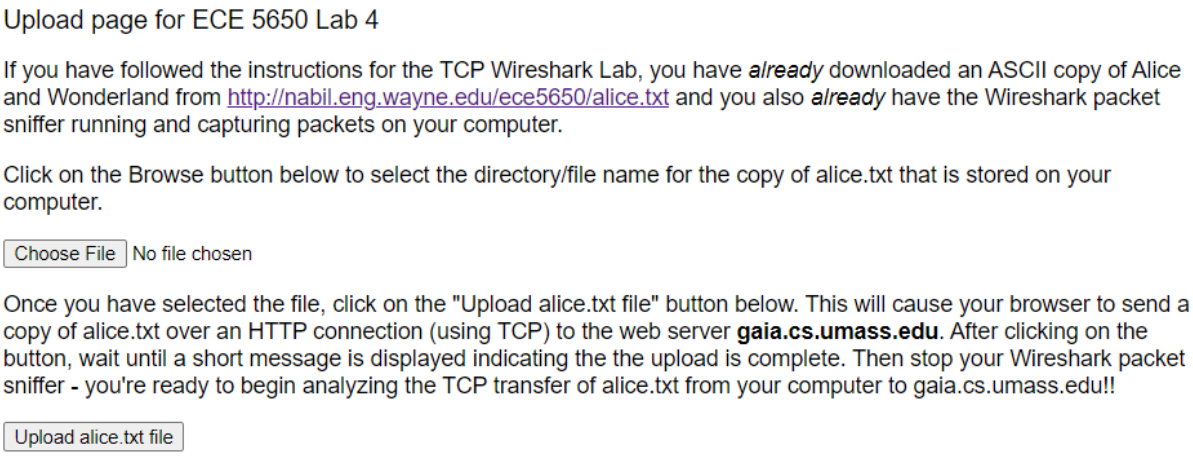
Lab Assignment

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

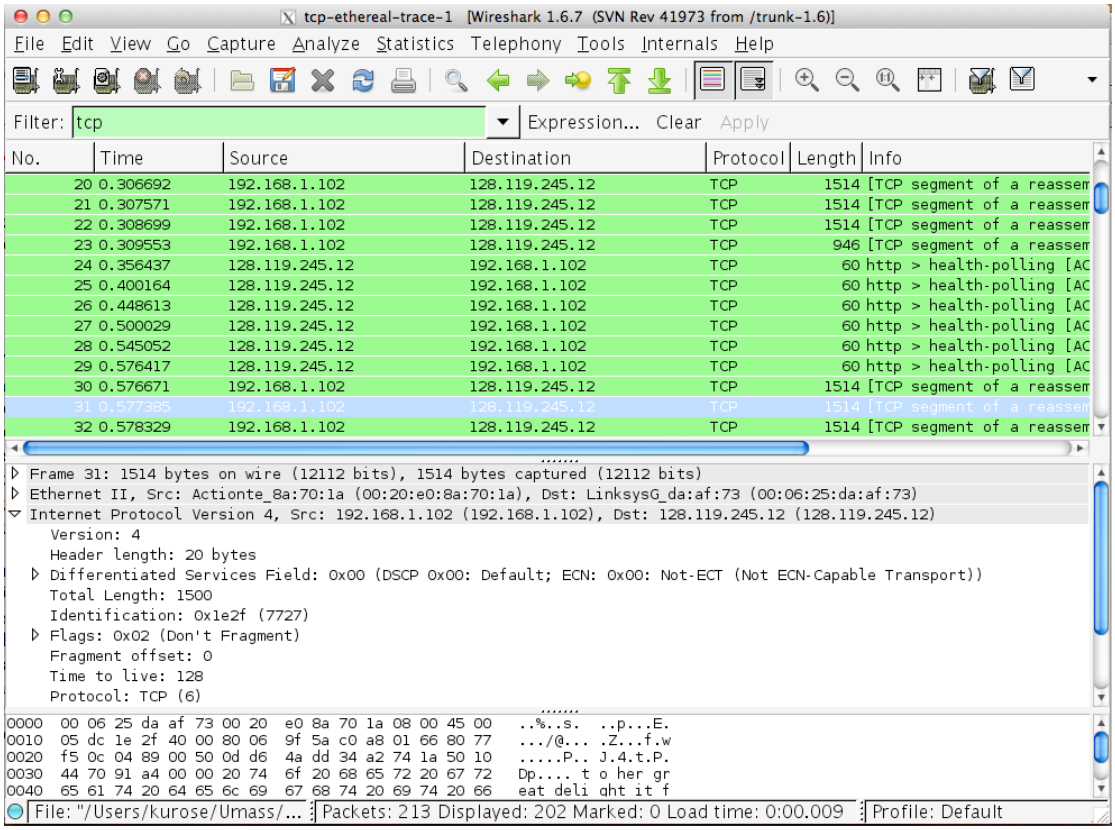
Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text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

Please preform the following tasks.

- Start up your web browser. Go to http://nabil.eng.wayne.edu/ece5650/alice.txt and retrieve an ASCII copy of *Alice in Wonderland.* Store this file somewhere on your computer.
- Next, go to http://nabil.eng.wayne.edu/ece5650/upload2.html
- You should see a screen that looks as follows.



- 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, startup 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 gaia.cs.umass.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.



In this Lab, you will use the trace that you just captured yourself as well as another trace collected on a different computer, as will be explained later.

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 gaia.cs.umass.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message.

In 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 gaia.cs.umass.edu to your computer.

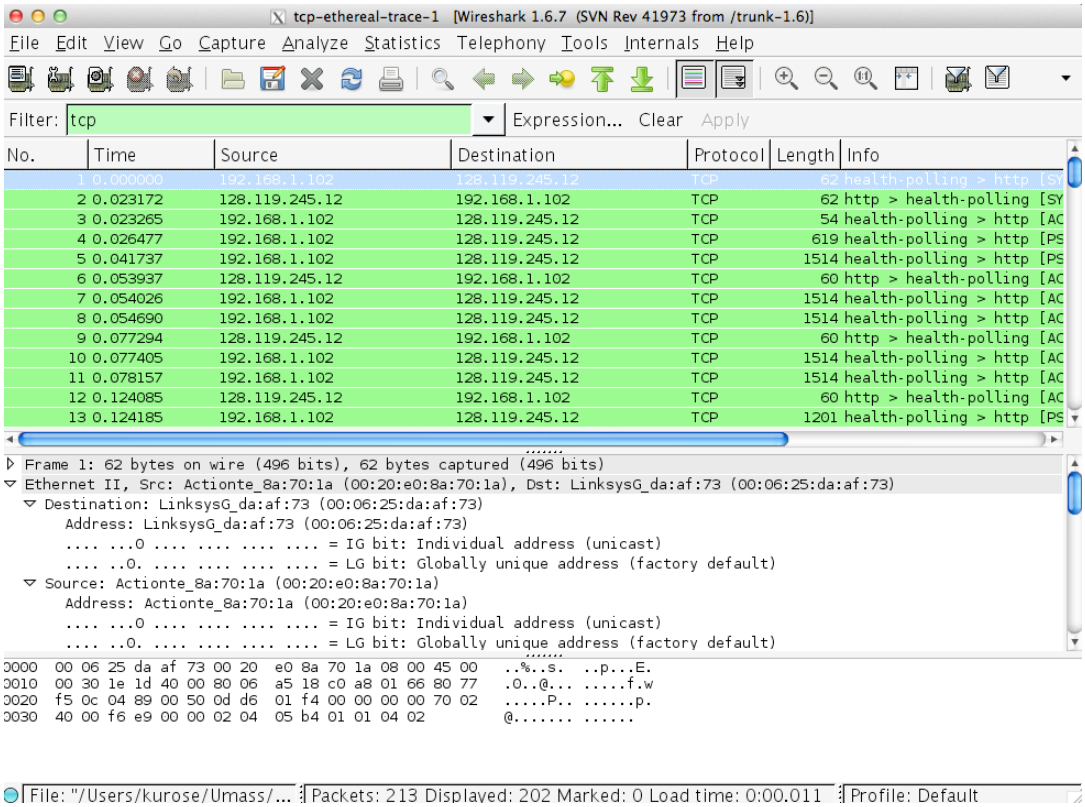
1. What is the IP address and TCP port number used by your client computer

(source) to transfer the file to gaia.cs.umass.edu?

Now, you will use a trace collected on a different computer. Download the zip file http://nabil.eng.wayne.edu/ece5650/wireshark-traces.zip and extract the file tcp-ethereal-trace-1. The traces in this zip file were collected by Wireshark running on a certain computer, while performing the steps indicated in the first part of this 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 tcp-ethereal-trace-1 trace file. Whenever possible, when answering a question, you should include a snapshot of the packet(s) within the trace that you used to answer the question asked. Preferably, annotate the printout to explain your answer by highlighting where in the printout you've found the answer and adding some text.
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.

2. What is the IP address and TCP port number used by the client computer (source) 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" (refer to Figure 2 in the "Getting Started with Wireshark" Lab if you're uncertain about the Wireshark windows.
3. What is the IP address of gaia.cs.umass.edu? On what port number is it 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 your computer and gaia.cs.umass.edu. You will use the packet trace that you have captured **and/or** the packet trace *tcp-ethereal-trace-1* in http://www.ece.eng.wayne.edu/~nabil/ece5650/wireshark-traces.zip to study TCP behavior in the rest of this lab. The questions will specify the trace that you need to use.

3. TCP Basics

For the *tcp-ethereal-trace-1* in http://nabil.eng.wayne.edu/ece5650/wireshark-traces.zip, answer the following questions for the TCP segments:

4. 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?
5. 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? What is it in the segment that identifies the segment as a SYNACK segment?
6. 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.

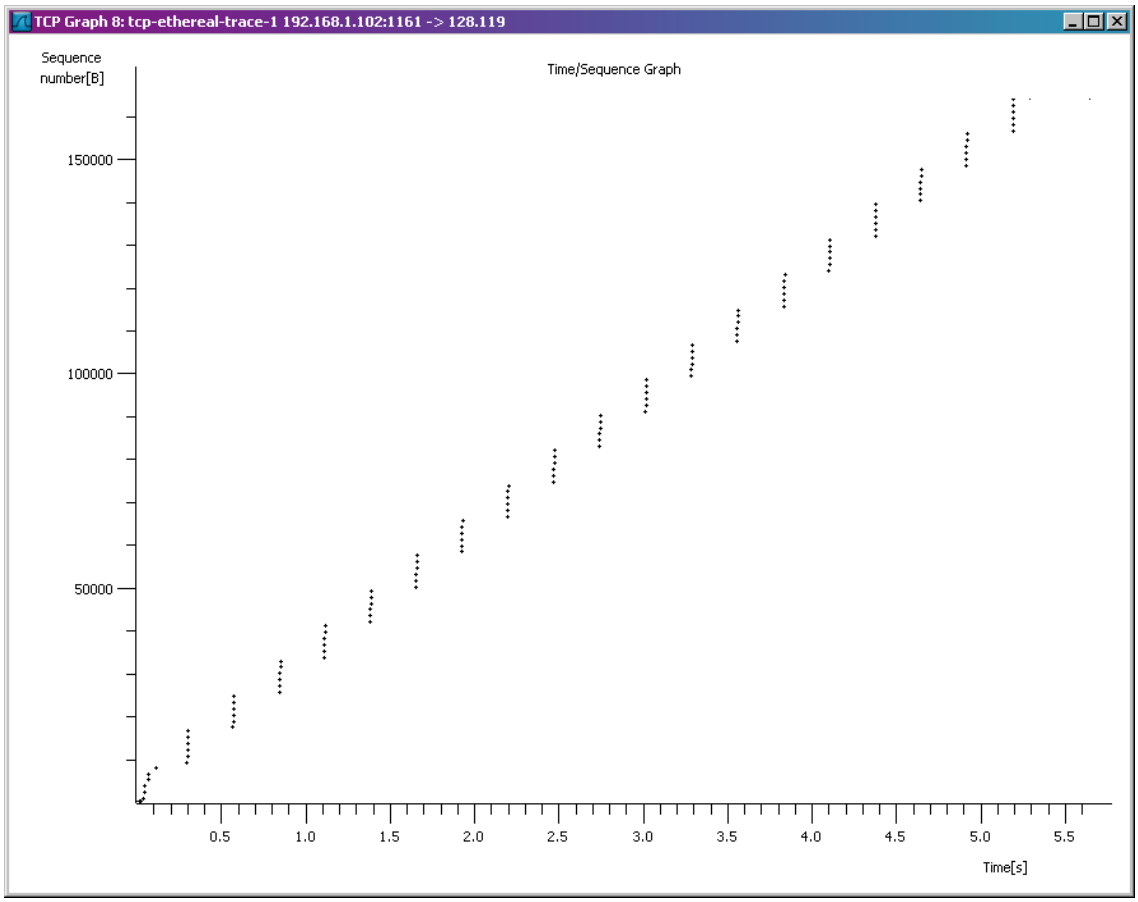
7. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? 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 EstimatedRTTvalue (see Section 3.5.3, page 239 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTTis equal to the measured RTT for the first segment, and then is computed using the EstimatedRTTequation on page 239 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.*
8. What is the length of each of the first six TCP segments?
   Note: The TCP segments in the tcp-ethereal-trace-1 trace file are all smaller than 1460 bytes. This is because the computer on which the trace was gathered has an Ethernet card that limits the length of the maximum IP packet to 1500 bytes (40 bytes of TCP/IP header data and 1460 bytes of TCP payload). This 1500-byte value is the standard maximum length allowed by Ethernet. If your trace indicates a TCP length greater than 1500 bytes, and your computer is using an Ethernet connection, then Wireshark is reporting the wrong TCP segment length; it will likely also show only one large TCP segment rather than multiple smaller segments. Your computer is indeed probably sending multiple smaller segments, as indicated by the ACKs it receives. This inconsistency in reported segment lengths is due to the interaction between the Ethernet driver and the Wireshark software. We recommend that if you have this inconsistency, that you perform this lab using the provided trace file.
9. What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
11. 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 247 in the textbook)?
12. 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 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*). You should see a plot that looks similar to the following plot, which was created from the captured packets in the packet trace *tcp-ethereal- trace-1* in http://nabil.eng.wayne.edu/ece5650/wireshark-traces.zip.



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.

13. For the TCP segments the packet trace tcp-ethereal- trace-1 in http://nabil.eng.wayne.edu/ece5650/wireshark-traces.zip, answer the following two questions for the TCP segments. 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. Can you identify where TCP's slow-start 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.
14. For the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu, answer each of the last two questions.