THE UNIVERSITY OF WINNIPEG

APPLIED COMPUTER SCIENCE

ACS-3911-050 Computer Networks
Winter 2020

Assignment 3/Lab TCP and Lab UDP
Due Date: 23rd March, 2020

The assignment is in two parts, Part A - Questions and Answer, and Part B - Wireshark Lab TCP and Lap UDP. Part A is 40 marks, and Part B is 40 marks, a total of 80 marks for this assignment. Please complete by due date. No assignment will be accepted after the due date and the date/time stamp on the email will be used to determine if the assignment was on time or late.

Email your assignment in PDF format to 3911-050@acs.uwinnipeg.ca by 11:59pm Mar 23rd, 2020. The name of your file should be named - <First name Last Name> - <Student ID> – Assignment <number>, e.g. John Doe – 12345 – Assignment 3. The subject line of your email must be: <your name><your student #>< course # with section #>Assignment 3.

Part A – Questions and Answers

Review Questions

Section 3.1 – 3.3

R3. Consider a TCP connection between Host A and Host B. Suppose that the TCP segments traveling from Host A to Host B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from Host B to Host A? (Mark 1)

R7. Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts? (Mark 1)

R8. 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. (Mark 5)

Section 3.4

R11. Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol rdt 3.0, assuming that packets can be lost? Explain. (Mark 1)

Section 3.5

R14. True or false? (Mark 7)

a) Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send Host A. Host B will not send acknowledgments to Host A because Host B cannot piggyback the acknowledgments on data.
b) The size of the TCP rwnd never changes throughout the duration of the connection.
c) Suppose Host A is sending Host B a large file over a TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receive buffer.
d) Suppose Host A is sending a large file to Host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m + 1.
e) The TCP segment has a field in its header for rwnd.
f) Suppose that the last SampleRTT in a TCP connection is equal to 1 sec. The current value of TimeoutInterval for the connection will necessarily be ≥ 1 sec.
g) Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment the acknowledgment number is necessarily 42.

Section 3.7

R18. True or false? Consider congestion control in TCP. When the timer expires at the sender, the value of ssthresh is set to one half of its previous value. (Mark 1)

Problems

P3. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error? (Marks 5)

P24. Answer true or false to the following questions and briefly justify your answer: (Marks 8)

a) With the SR protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.
b) With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.
c) The alternating-bit protocol is the same as the SR protocol with a sender and receiver window size of 1.
d) The alternating-bit protocol is the same as the GBN protocol with a sender and receiver window size of 1.

P40. Consider Figure 3.58. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer. (Marks 11)

a) Identify the intervals of time when TCP slow start is operating.
b) Identify the intervals of time when TCP congestion avoidance is operating.
c) After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
d) After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
e) What is the initial value of ssthresh at the first transmission round?
f) What is the value of ssthresh at the 18th transmission round?
g) What is the value of ssthresh at the 24th transmission round?
h) During what transmission round is the 70th segment sent?
i) Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of ssthresh?

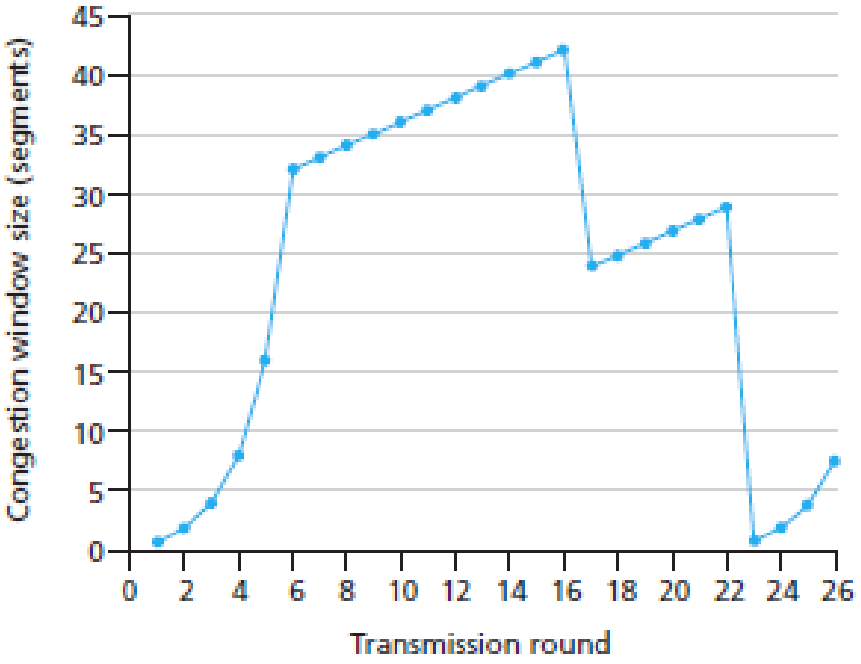


Figure 3.58 – TCP window size as a function of time

j) Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the ssthresh and the congestion window size at the 19th round?

k) Again suppose TCP Tahoe is used, and there is a timeout event at 22nd round. How many packets have been sent out from 17th round till 22nd round, inclusive?

Part B – Wireshark Lab

Supplement to Computer Networking: A Top-Down Approach, 6th ed., J.F. Kurose and K.W. Ross

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb

Wireshark Lab 1: TCP

In this lab, we'll investigate the behavior of the celebrated 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; 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].

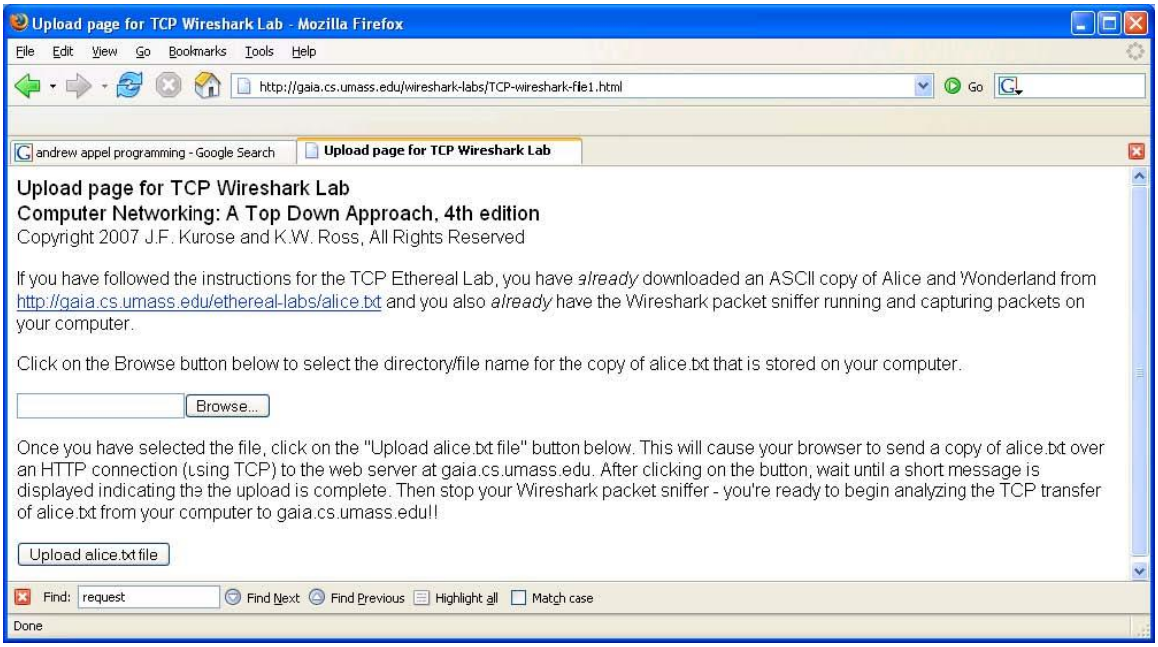
1. Capturing a bulk TCP transfer from your computer to a remote server

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.

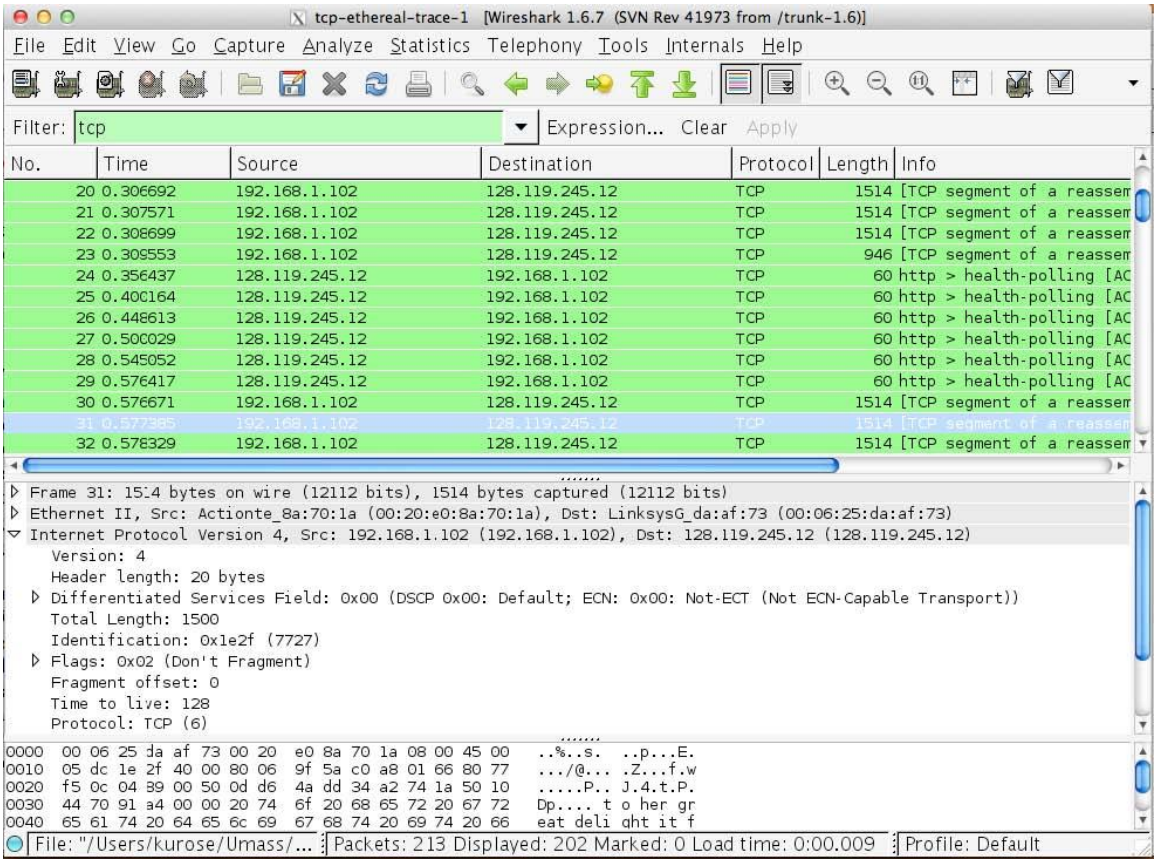
Do the following:

- Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark¬labs/alice.txt and retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.
- Next go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.
- You should see a screen that looks like:

[1] References to figures and sections are for the 6th edition of our text, Computer Networks, A Top-down Approach, 6th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2012.



- 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 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.



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 on one of the author's computers[2]. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions 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" (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.

[2] Download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file tcp¬ethereal-trace-1. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark 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.

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 gaia.cs.umass.edu. Recall from our discussion in the earlier HTTP Wireshark lab, that 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. 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 gaia.cs.umass.edu to your computer.

Answer the following questions, by opening the Wireshark captured packet file tcp¬ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip (that is download the trace and open that trace in Wireshark; see footnote 2). 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[3] 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.

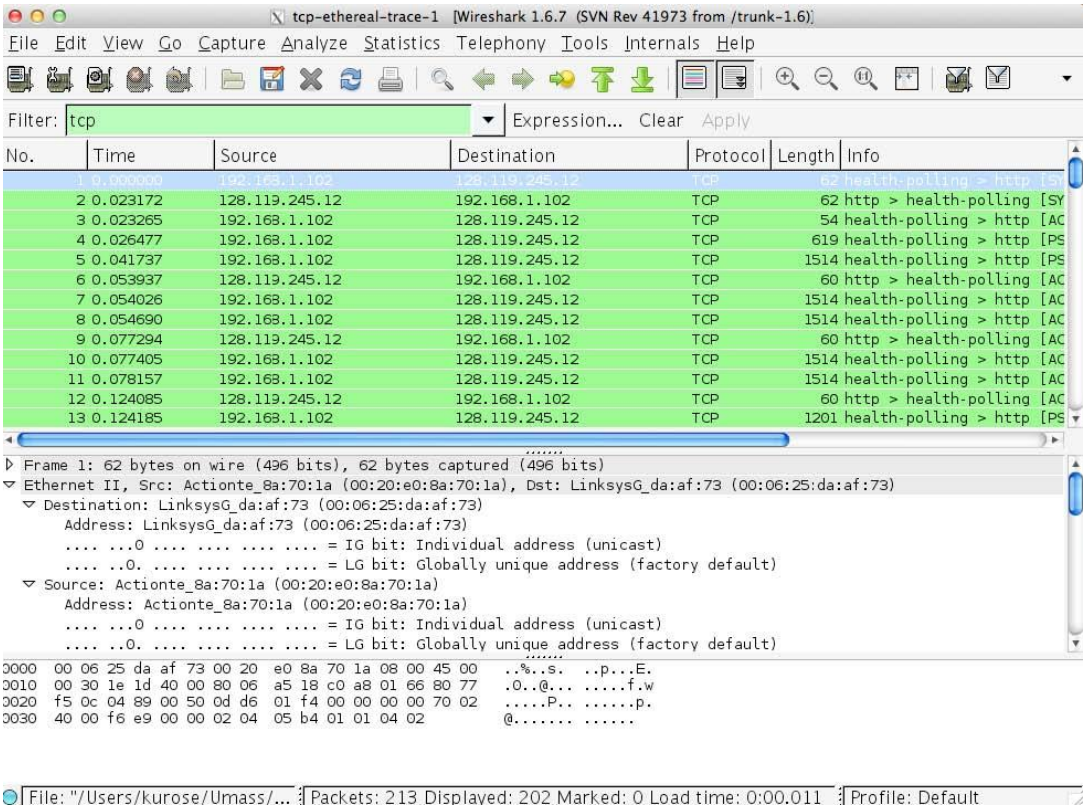
1. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details 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. (Marks 2)
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection? (Marks 2)

If you have been able to create your own trace, answer the following question:

3. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu? (Marks 2)

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:

[3] 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 gaia.cs.umass.edu. We will use the packet trace that you have captured (and/or the packet trace tcp-ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark¬traces.zip; see earlier footnote) to study TCP behavior in the rest of this lab.

3. TCP Basics

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? (Mark 1)
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? (Marks 4)
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. (Mark 1)
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 EstimatedRTT value (see Section 3.5.3, page 239 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 239 for all subsequent segments. (Marks 4)

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?[4] (Mark 1)
9. 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? (Mark 1)
10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question? (Mark 1)
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 text). (Marks 2)
12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value. (Mark 1)

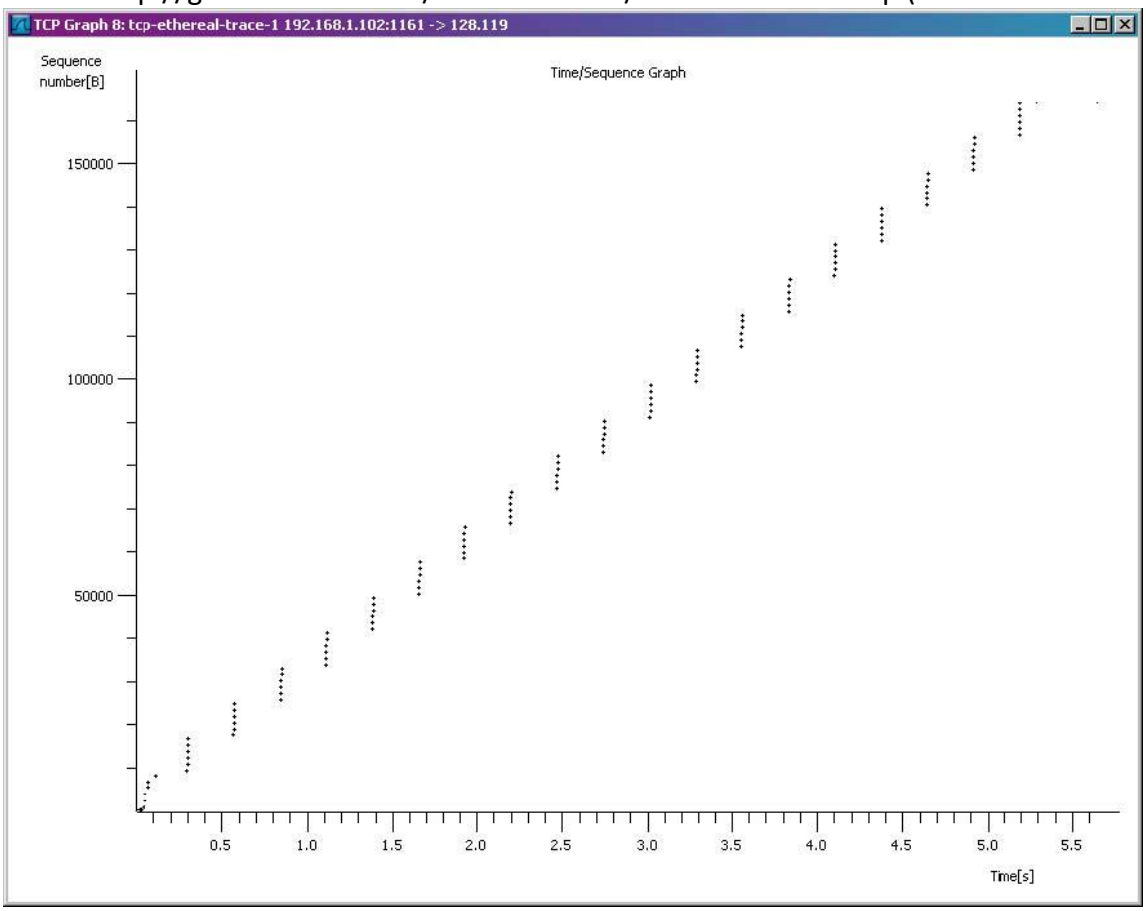
4. TCP congestion control in action

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

[4] The TCP segments in the tcp-ethereal-trace-1 trace file are all less that 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.

captured packets in the packet trace tcp-ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip (see earlier footnote):



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 questions for the TCP segments the packet trace tcp-ethereal¬trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip

13. 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 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. (Marks 2)

14. Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu (Marks 2)

Wireshark Lab 2: UDP

In this lab, we'll take a quick look at the UDP transport protocol. As we saw in Chapter 3 of the text[5], UDP is a streamlined, no-frills protocol. You may want to re-read section 3,3 in the text before doing this lab. Because UDP is simple and sweet, we'll be able to cover it pretty quickly in this lab. So if you've another appointment to run off to in 30 minutes, no need to worry, as you should be able to finish this lab with ample time to spare.
At this stage, you should be a Wireshark expert. Thus, we are not going to spell out the steps as explicitly as in earlier labs. In particular, we are not going to provide example screenshots for all the steps.

The Assignment

Start capturing packets in Wireshark and then do something that will cause your host to send and receive several UDP packets. It's also likely that just by doing nothing (except capturing packets via Wireshark) that some UDP packets sent by others will appear in your trace. In particular, the Simple Network Management Protocol (SNMP -chapter 9 in the text) sends SNMP messages inside of UDP, so it's likely that you'll find some SNMP messages (and therefore UDP packets) in your trace.

After stopping packet capture, set your packet filter so that Wireshark only displays the UDP packets sent and received at your host. Pick one of these UDP packets and expand the UDP fields in the details window. If you are unable to find UDP packets or are unable to run Wireshark on a live network connection, you can download a packet trace containing some UDP packets.[6]

Whenever possible, when answering a question below, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout3 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. Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. (You shouldn't look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields. (Marks 5)

[5] References to figures and sections are for the 6th edition of our text, Computer Networks, A Top-down Approach, 6th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2012.
[6] Download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file http-ethereal-trace-5, which contains some UDP packets carrying SNMP messages. The traces in this zip file were collected by Wireshark running on one of the author's computers. 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 http-ethereal-trace-5 trace file.

2. By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of each of the UDP header fields. (Mark 1)
3. The value in the Length field is the length of what? (You can consult the text for this answer). Verify your claim with your captured UDP packet. (Marks 2)
4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above) (Mark 1)
5. What is the largest possible source port number? (Hint: see the hint in 4.) (Mark 1)
6. What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. To answer this question, you'll need to look into the Protocol field of the IP datagram containing this UDP segment (see Figure 4.13 in the text, and the discussion of IP header fields). (Marks 2)
7. Examine a pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). Describe the relationship between the port numbers in the two packets. (Marks 2)