**Deadline: Monday, October 18, 2018, 11:55 PM**

**Fall 2018**

*Total Points: 100*

**Section I (55 points)**

1. Suppose a process in Host C has a UDP socket with port number 6789. Sup- pose 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? **(5 points)**

Yes, both segments A and B will be directed to the same socket at Host C. Host C can differentiate who sent what by their IP addresses.

1. 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. **(5 points)**

The web server will create a different connection for each persistent connection. Using message info like IP address to differentiate the connections. However, they will both use the same socket number 80.

1. Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TCP receive buffer at a maximum rate of 50 Mbps. Describe the effect of TCP flow control. **(10 points)**

Host A will transmit data at 100Mbps as that is the max that the connection can handle. As the messages get sent to Host B, the receive buffer window will fill up. As Host B processes message, host B will send acknowledgement message. Then Host A sends next few messages and cycle repeats

1. Suppose that the five measured SampleRTT values (see Section 3.5.3 of textbook) are 106 ms, 120 ms, 140 ms, 90 ms, and 115 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value of α = 0.125 and assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained. Compute also the DevRTT after each sample is obtained, assuming a value of β = 0.25 and assuming the value of DevRTT was 5 ms just before the first of these five samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained. **(10 points)**

(First SAMPLE)

EstimatedRTT = 0.125 \* 106 + (1-0.125) \* 100 = 100.75ms

DevRTT = .25 \* | 106 - 100.75 | + (1-0.25) \* 5 = 5.0625ms

TimeoutInterval = 100.75 + 4 \* 5.0625 = 121ms

(SECOND SAMPLE)

EstimatedRTT = 0.125 \* 120 + (1-0.125) \* 100.75 = 103.156ms

DevRTT = 0.25 \* |120-103.15625| + (1-0.25) \* 5.0625 = 8ms

TimeoutInterval = 103.156 + 4 \* 8 = 135ms

(THIRD SAMPLE)

EstimatedRTT = 0.125 \* 140 + (1-0.125) \* 103.15 = 107.75ms

DevRTT = 0.25 \* |140-107.75| + (1-0.25) \* 8 = 14.06ms

TimeoutInterval = 107.75 + 4 \* 14.06 = 164ms

(FOURTH SAMPLE)

EstimatedRTT = 0.125 \* 90 + (1-0.125) \* 107.75 = 105.53ms

DevRTT = 0.25 \* |90-105.53| + (1-0.25) \* 14.06 = 14.42ms

TimeoutInterval = 105.53 + 4 \* 14.42 = 163.21ms

(FIFTH SAMPLE)

EstimatedRTT = 0.125 \* 115 + (1-0.125) \* 105.53 = 106.715ms

DevRTT = 0.25 \* |115-106.715| + (1-0.25) \* 14.06 = 12.885ms

TimeoutInterval = 106.715 + 4 \* 12.885 = 158.255ms

1. Visit the Go-Back-N Java applet uploaded on Moodle page of CSI 2470. a. Have the source send five packets, and then pause the animation before any of the five packets reach the destination. Then kill the first packet and resume the animation. Describe what happens. b. Repeat the experiment, but now let the first packet reach the destination and kill the first acknowledgment. Describe again what happens. c. Finally, try sending six packets. What happens?. B) Now visit Selective Repeat applet. Repeat above three steps (step a, b and c) , but now with the Selective Repeat Java applet. How are Selective Repeat and Go-Back-N different? **(5+5 points)**

(Go back N)

1. All packets are not confirmed and packets are re-sent.
2. All packets are confirmed and window shifts
3. The applet won’t let you, the window only allows 5 packets to be sent

(Selective Repeat)

1. The last 4 packets are acknowledged and the first packet is resent
2. The last 4 packets are acknowledged and the first packet is resent
3. The applet won’t let you, the window only allows 5 packets to be sent
4. Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.
5. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?
   1. The sequence number is 207, source port is 302 and destination port is 80
6. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?
   1. The sequence number is 207, source port is 302 and destination port is 80
7. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?
   1. The sequence number is 128, and will wait for 1st segment to arrive

**( 15 points)**

**Section II TCP Wireshark Lab (Q4=10 points, remaining 7Q\*5points =45 points)**

Analyze 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. You are supposed to study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer; 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.

**Part I: Capturing a bulk TCP transfer from your computer to a remote server**

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 le 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 (recall this from the last lab). Do the following:

1. Start up your web browser. Go <http://gaia.cs.umass.edu/wiresharklabs/alice.txt> and retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.
2. Next go to http:// <http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>.
3. Your web browser should show a web page which permits you to upload a file. 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.
4. Now start up Wireshark and begin packet capture (Capture-->Options). Select the correct interface (usually it's the network interface card of your computer) and then press START on the Wireshark Packet Capture Options screen. Before starting, it's best to close the webpage tabs other than the upload page.
5. 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.
6. Stop Wireshark packet capture. In order to analyze the trace, you can filter the packets displayed in the Wireshark window by entering “tcp" (lowercase, no quotes, and press return after entering) into the display filter specification window towards the top of the Wireshark window. Then you should be able to see both TCP and HTTP packets. Export and save the trace file for further analysis.

**NOTE: Instead of capturing your own file one can download the zip file** [**http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip**](file:///R:\CN) **and extract the file tcp­ethereal-trace-1 trace for this home assignment.**

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 and 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, 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 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.

**Note:** To answer these questions, 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"

Since this lab is about TCP rather than HTTP, 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 that shown in Fig. 1. This is what we're looking for a series of TCP segments sent between your computer and gaia.cs.umass.edu.



Fig1 series of TCP between your computer and server.

**Part II: TCP Basics**

Answer the following questions for the TCP segments:

1. 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?
   1. The sequence number is 0.
   2. The sequence number is 1.
2. 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 acknowledgement number (Ack=?) 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?
   1. The sequence number is 0.
   2. The acknowledgment number is 1. This is because gaia.cs.umass.edu added 1 to the previous sequence number.
   3. The segment that identifies SYN is segment 1, which is labelled as Syn.
3. What is the sequence number of the TCP segment containing the HTTP POST command? What is the IP address and TCP This can be disabled by going to Edit--> Preferences --> Protocols --> HTTP and unchecking the “Reassemble HTTP bodies spanning multiple TCP segments" box.

1. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. Calculate the Round Trip Time (RTT). Note that the RTT time is the time difference between the time of the POST message and the corresponding ACK.
   1. 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? Note: you may want to re-enable “Reassemble HTTP bodies spanning multiple TCP segments" if you disabled this setting previously. Also note that there may be multiple ACKs associated with each TCP segment. You should list the time of the final ACK for each segment. The Statistics --> Flow Graph --> TCP flow view can also be useful.
   2. 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?
   3. Plot the Round Trip Time Graph.

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.

For (a) and (b), fill in this table for 6 segments. For (c), hand in the graph

|  |  |  |  |
| --- | --- | --- | --- |
| Segment Seq. # | Sent time | ACK Receive Time | Actual RTT |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

Note: Generally, the TCP segments will all be less than 1460 bytes. This is because most computers have 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.

1. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question? (hint: plot the time sequence graph from the statistics menu)

Note: Select a TCP segment sent from your computer to the server 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 plot in Figure 2. 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.

1. How much data does the receiver typically acknowledge in an ACK? Show an example.
2. What is the average throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

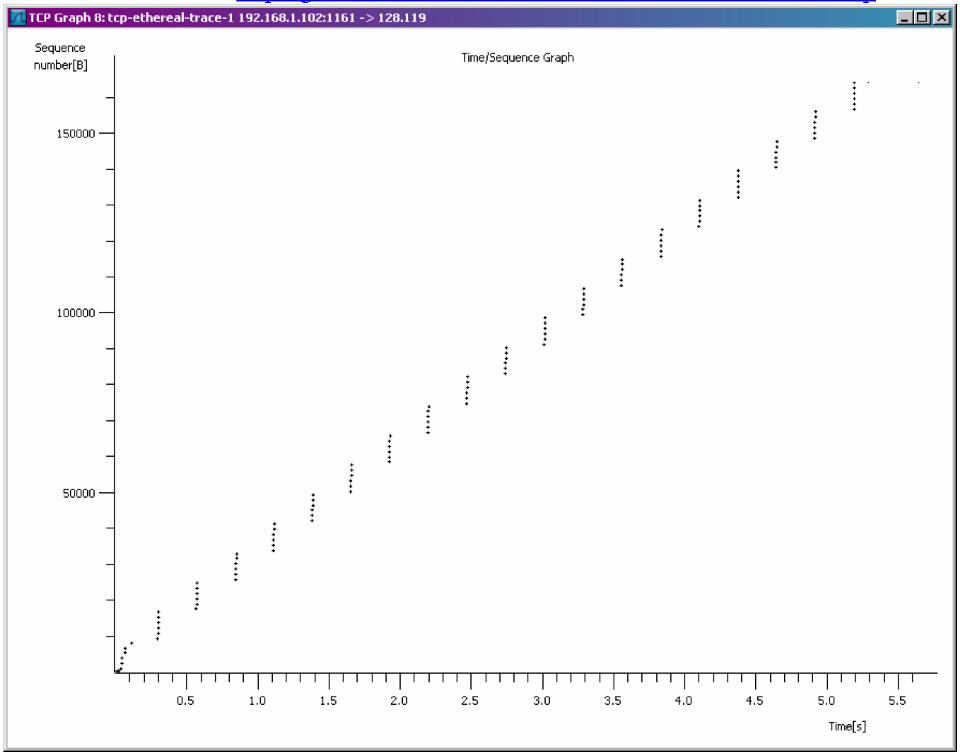
What to hand in: answers to questions 1-8. Also provide evidence for how you got the answers.

Figure 2: Example plot of TCP sequence numbers