MPCS 54001, Winter 2021

Lab 3 - Solutions

Due: January 29 at 11:59 pm

Submit your assignment to Gradescope. If you need help, post questions to Ed Discussion. As a reminder, if you make a public post on Ed Discussion, please don't give away the answer! Please submit one pdf per group, answering the questions which appear on the last page. Gradescope will allow you to submit one pdf and add all group members to it after submitting.

Most of the grading for this assignment is checking the questions against the printout (or screen capture) that was submitted with the lab.

A First Look at the Captured Trace

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".

   Check this against the printout. Will vary from one student to the next, but should be opposite the answer for #2 in the file transfer packets.

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?

   Check this against the printout. It's likely to be 128.119.245.12, 80.

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?

   Duplicate question; disregard.

TCP Basics

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?

   Check the first question against the printout. The second should be [SYN] or that the SYN flag is set (value 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?

   Check this against the printout. The ACK field in the SYNACK segment should be the SYN sequence number + 1, which is how the server determined it. The SYN and ACK flags are set.

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.

   Check this against the printout.

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 242 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 242 for all subsequent segments.

Check against the printout that the proper ACK is paired with each sequence number. Check that the round trip time is calculated correctly (SampleRTT) and that the EstimatedRTT formula is implemented properly. You may want to build an Excel workbook to do this.

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

   Check this against the printout.

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?

   Check this against the printout. You are checking for the "window" field.

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

    Check this against the printout. You can either see that there are no repeated sequence numbers in the trace, or that the graph is monotonically increasing in the *Time-Sequence-Graph (Stevens)* of this trace.

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 250 in the text or slide 22 in Transmission Control Protocol)?

    Check this against the printout. The difference between the acknowledged sequence numbers of two consecutive ACKs indicates the data received by the server between those two ACKs.

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

    Check this against the printout. The number of bytes transferred is the difference between the first sequence number in the file-transfer trace and the last ACK in the same trace. The time is the difference in timestamps between the two. It would also be acceptable to take the file size.

TCP Congestion Control in Action

13. 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.

Check this against the printout of the graph. It should be hard to tell from the graph, but this can be solved by examining how the congestion window size reacts to the arrival of an ACK. Note that the criteria to determine the end of slow start and the beginning of the congestion avoidance is the way congestion window size reacts to the arrival of ACKs. Upon an ACK arrival, if the congestion window size increases by one MSS, TCP sender still stays in the slow start phase. In the congestion avoidance phase, the congestion window size increases at 1/(current_congestion_window_size). By inspecting the change of the congestion window upon the arrival of ACKs, we can infer the states of the TCP sender.

You can estimate the size of the congestion window by keeping track of how much unACKed data is outstanding. This is done by lining up the data packets and acks and doing arithmetic. An example is below:

Type	No.	Seq.	ACKed seq.	Outstanding data
Data	4	1		565
Data	5	566		2025
ACK	6		566	1460
Data	7	2026		2920
Data	8	3486		4380
ACK	9		2026	2920
Data	10	4946		4380
Data	11	6406		5840
ACK	12		3486	4380
Data	13	7866		5527
ACK	14		4096	4917
ACK	15		6006	3007
ACK	16		7866	1147
ACK	17		9013	0
Data	18	9013		1460
Data	19	10473		2920

It is possible that this file upload happened so quickly that all of the DATA was sent before any ACKs were received. In this case, clear from the graph, the entire transfer is in the slow start phase.

The idealized behavior of TCP in the text assumes that TCP senders are aggressive in sending data. Too much traffic may congest the network; therefore, TCP senders should follow the AIMD algorithm so that when they detect network congestion (i.e., packet loss), their sending window size should drop down. In the practice, TCP behavior also largely depends on the application. In this example, when the TCP sender can send out data, there are no data available for transmission. In the web application, some of web objects have very small sizes. Before the end of slow start phase, the transmission is over; hence, the transmission of these small web objects suffers from the unnecessary long delay because of the slow start phase of TCP.