Computer Network 1

LAB 3a

Wireshark lab TCP

Student Name: Alexandre Rousseau

Student ID: 1952001

I. 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" (refer to Figure 2 in the "Getting Started with Wireshark" Lab if you're uncertain about the Wireshark windows.

The client IP address is 172.19.217.58, TCP port number is 50403.

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?

Gaia.cs.umass.edu's IP address is 128.119.245.12, port number is 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?

IP address: 192.168.111.1 Port: 5000

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?

Sequence number of the TCP SYN segment is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu. The value is 0 in this trace

The Syn flag is set to 1 and it indicates that this segment is 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?

Sequence number of the SYNACK segment from gaia.cs.umass.edu to the client computer in reply to the SYN has the value of 0 in this trace.

The value of the Acknowledgement field in the SYNACK segment is determined by gaia.cs.umass.edu by adding 1 to initial sequence number of SYN segment from the client computer is 0.

The SYN flag and Acknowledgement flag in the segment are set to 1 and they indicate that this segment is a SYNACK segment.

6) What is the sequence number of the TCP segment containing the HTTP POST command? Note that 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.

No. 4 segment is the TCP segment containing the HTTP POST command. The sequence number of this segment has the value of 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 242 in text) after the receipt of each ACK?

The HTTP POST segment is considered as the first segment. Segments 1 – 6 are No. 4,5,7,8,10, and 11 in this trace respectively. The ACKs of segments 1 – 6 are No. 6,9,12,14,15, and 16 in this trace.

Segment 1 sequence number: 1

Segment 2 sequence number: 566

Segment 3 sequence number: 2026

Segment 4 sequence number: 3486

Segment 5 sequence number: 4946

Segment 6 sequence number: 6406

The sending time and the receiving time of ACKs are tabulated in the following table.

	Sent time	ACK receiving time	RTT (seconds)
Segment 1	0.026477	0.054937	0.02746
Segment 2	0.041737	0.077294	0.035557
Segment 3	0.054026	0.124085	0.070059
Segment 4	0.054690	0.169118	0.11443
Segment 5	0.077405	0.217299	0.13989
Segment 6	0.078157	0.267802	0.18964

EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT

EstimatedRTT after the receipt of the ACK of segment 1:

EstimatedRTT = RTT for Segment 1 = 0.02746 second

EstimatedRTT after the receipt of the ACK of segment 2:

EstimatedRTT = 0.875 * 0.02746 + 0.125 * 0.035557 = 0.0285 second

EstimatedRTT after the receipt of the ACK of segment 3:

EstimatedRTT = 0.875 * 0.0285 + 0.125 * 0.070059 = 0.0337 second

EstimatedRTT after the receipt of the ACK of segment 4:

EstimatedRTT = 0.875 * 0.0337 + 0.125 * 0.11443 = 0.0438 second

EstimatedRTT after the receipt of the ACK of segment 5:

EstimatedRTT = 0.875 * 0.0438 + 0.125 * 0.13989 = 0.0558 second

EstimatedRTT after the receipt of the ACK of segment 6:

EstimatedRTT = 0.875 * 0.0558 + 0.125 * 0.18964 = 0.0725 second

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

Length of the first TCP segment (containing the HTTP POST): 565 bytes.

Length of each of the other five TCP segments: 1460 bytes (MSS).

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?

The minimum amount of buffer space (receiver window) advertised at gaia.cs.umass.edu for the entire trace is 5840 bytes, which shows in the first acknowledgement from the server. This receiver window grows steadily until a maximum receiver buffer size of 62780 bytes. The sender is never throttled due to lacking receiver buffer space by inspecting this trace.

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

There are no retransmitted segments in the trace file. We can verify this by checking the sequence numbers of the TCP segments in the trace file. In the Time-Sequence-Graph pf this trace, all sequence numbers from the source to the destination are increasing monotonically with respect to time. If there is a retransmitted segment, the sequence number of this retransmitted segment should be smaller than those of its neighboring segments.

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)?

The acknowledged sequence numbers of the ECKs are listed as follows.

	Acknowledged sequence number	Acknowledged data
ACK 1	566	566
ACK 2	2026	1460
ACK 3	3486	1460
ACK 4	4946	1460
ACK 5	6406	1460
ACK 6	7866	1460
ACK 7	9013	1147
ACK 8	10473	1460
ACK 9	11933	1460
ACK 10	13393	1460
ACK 11	14853	1460
ACK 12	16313	1460

The difference between the acknowledged sequence numbers of two consecutive ACKs indicates the data received by the server between these two ACKs. By inspecting the amount of acknowledged data by each ACK, there are cases where the receiver is ACKing every other segment.

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

The computation of TCP throughput largely depends on the selection of averaging time period. As a common throughput computation, in this question, we select the average time period as the whole connection time. Then, the average throughput for this TCP connection is computed as the ratio between the total amount data and the total transmission time. The total amount data transmitted can be computed by the difference between the sequence number of the first TCP segment and the acknowledged sequence number of the last ACK. Therefore, the total data are 164091 – 1 = 164090 bytes. The whole transmission time is the difference of the time instant of the first TCP segment and the time instant of the last ACK. Therefore, the total transmission time is 5.455830 – 0.026477 = 5.4294 seconds. Hence, the throughput for the TCP connection is computed as 164090/5.4294 = 30.222 Kbytes/sec.

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 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.

By observing the plot, we can see that the slow-start phase only lasts for first 1-1.5 second. Afterwards, it seems that the TCP session is always in congestion avoidance state. In this case, we do not observe the expected linear increase behaviors, i.e., the TCP transmit window does not grow linearly during this phase. In fact, it appears that the sender transmits packets in batches of 6. This does not seem to be caused by flow control since the receive advertised window is significantly larger than 5 packets. The reason for this behavior might be due to the fact that the HTTP server has enforced a rate-limit of some sort.