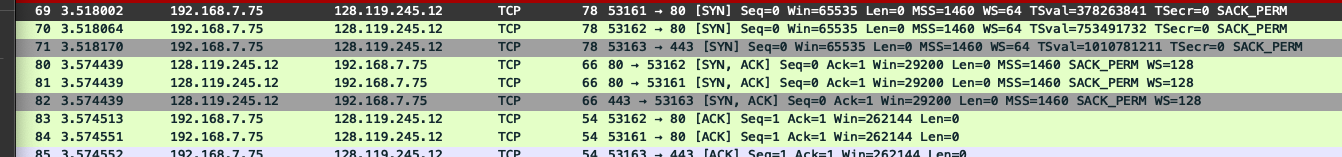
Marcos Ondruska

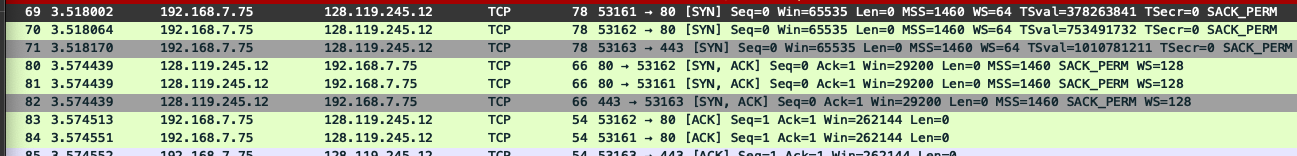
Student id: 2685885

Project 2

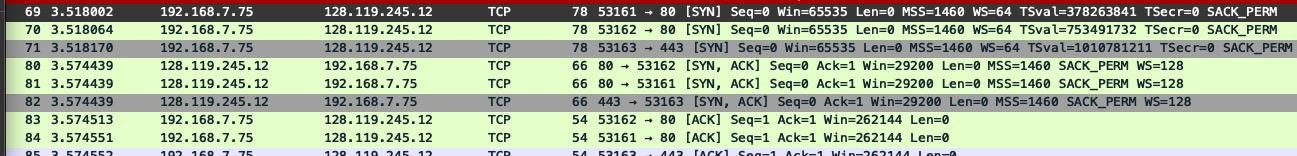
2) The IP address of gaia.cs.umass.edu is 128.119.245.12, and the port numbers are Port 80 for HTTP and Port 443 for HTTPS.



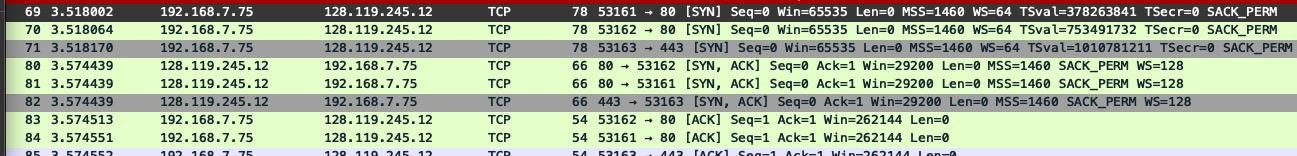
3) The IP address of my client computer is 192.168.7.75, and the port numbers used are Port 80 for HTTP and Port 443 for HTTPS.



4) The sequence number of the TCP SYN that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu is “0”. The [SYN} in the info section identifies the SYN segment.



5) The sequence number of the SYNACK segment sent by gaia.cs.umass.edu is “0”. The value of the acknowledgement field in the SYNACK segment is “1”. Gaia.cs.umass.edu determined the acknowledgement value from the client sequence number (mine) and then adds 1 (0 + 1 = 1). The item that identifies the SYNACK is the [SYN, ACK] in the info section



6) The sequence number of the TCP segment containing the HTTP POST is “1”.

A screenshot of a computer

Description automatically generated

7) The sequence numbers of the first six segments starting with the HTTP POST are (1, 566, 2026, 3486, 4946, 6406). Each of the segments was sent at the following times (0.026477, 0.041737, 0.054026, 0.054690, 0.077405, 0.078157). Each ACK was received as follows respectively (0.053937, 0.077294, 0.124085, 0.124185, 0.169118, 0.217299). RTT for each of the 6 segments is as follows (0.053937-0.026477=0.02746, 0.077294-0.041737 = 0.035557, 0.124085-0.054026=0.070059, 0.124185-0.054690=0.069495, 0.169118-0.077405=0.091713, 0.217299-0.078157=0.139142).

The equation on Page 242 of the textbook for Estimated Round Trip Time is:

EstimatedRTT = (1 – alpha) \* EstimatedRTT + (alpha \* sample RTT)

The sample value for alpha in the book is 0.125, therefore I will use the same value as per instructions. Estimated RTT for the first segment will equal its RTT, all others are EstimatedRTT.

Therefore Estimated RTT for the segments are (0.027460,

(1-0.125) \* 0.027460 + 0.125\*0.035557 = 0.028498,

(1-0.125) \* 0.028498 + 0.125\*0.070059 = 0.0.033607,

(1-0.125) \* 0.033607 + 0.125\*0.069495 = 0.037684,

(1-0.125) \* 0.037684 + 0.125\*0.0.091713 = 0.044731,

(1-0.125) \* 0.0.044731 + 0.125\*0.0.139142 = 0.0.056836)

8) A screenshot of a computer

Description automatically generated

The lengths of the first six segments are:  
(565, 1460, 1460, 1460, 1460, 1460)

9) A screenshot of a computer

Description automatically generated

The initial window size is initially advertised as 5840 bytes in packet no 2. This 5840 looks to be the minimum amount of available buffer space advertised.

It does not seem that there is throttling due to a lack of available buffer space. The Advertised window space starts at 5840 bytes and steadily increases to a maximum of 62780 over time. There don’t seem to be any instances where the window size drops to a very low value or zero, and the client doesn’t seem to be affected as they continue to send full sized segments (1460 bytes) all the way through the trace. There also don’t seem to be any long pauses in the data transmission.

10) There seem to be no duplicate sequence numbers in any of the transmissions, there also seem to be no duplicate ACKs that might seem like packet loss. I have also checked Wireshark for retransmissions at Analyze-> Expert information, and there seem to be no retransmissions there. So in short, I checked for Duplicate Sequence numbers, Duplicate ACK’s, and Wireshark’s Expert Analysis for any flags.

11) The receiver typically acknowledges 1460 bytes in an ACK, which represents the MSS (maximum segment size) of the sender’s packets. I don’t see any cases where the receiver is ACKing every other received segment.

12) Packet 202 shows that the seq ends with a total of 164091 bytes of data sent. The total time of the transfer would be calculated by taking the time at packet 202 of 5.45583 and subtracting the start time in packet 4 of 0.026477. So total time would be 5.45583-0.026477 =5.429353 seconds. To calculate throughput you would use the formula:

Throughput = total data transferred / total time

= 164091 / 5.429353

= approx 30223 bytes/second

13) A graph showing a number of numbers

Description automatically generated

A graph with a line

Description automatically generated with medium confidence

Given the graph and the zoom in of the graph, the slow start phase starts at 0 seconds and seems to end just before 0.1 seconds. This is where the sequence numbers increase in smaller sizes initially and then end up increasing quickly. The congestion window increases exponentially during this phase.

When slow start ends (just before 0.1 seconds) congestion avoidance takes over.

The biggest differences from the idealized behavior of TCP that is in the material is that all examples seemed to have a situation where there was packet loss and then the congestion window was reset. There doesn’t seem to be a reset at all and potentially another slow start. The graph itself also seem very linear, which probably shows that the MSS of the client doesn’t overwhelm the server at all.