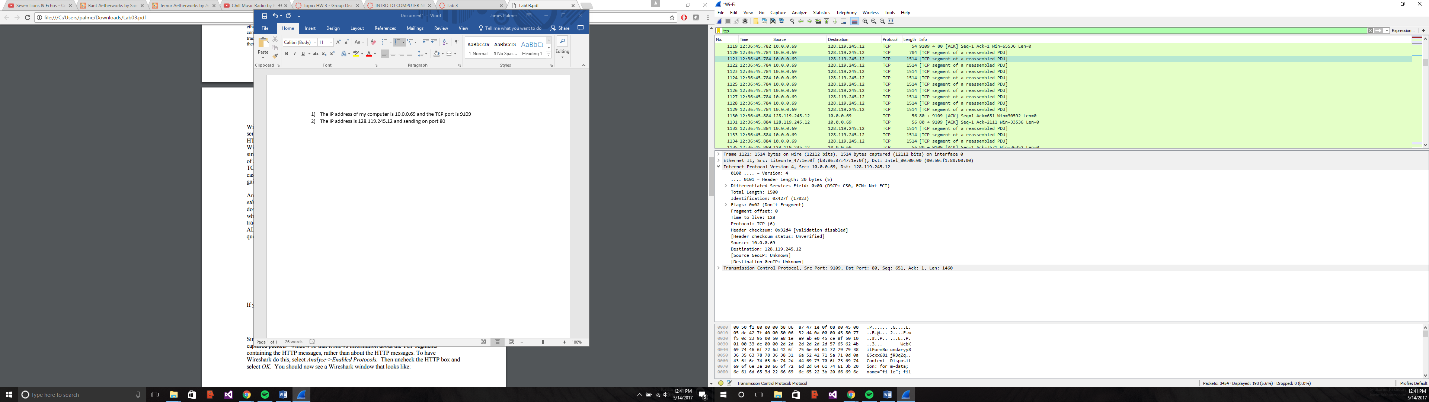
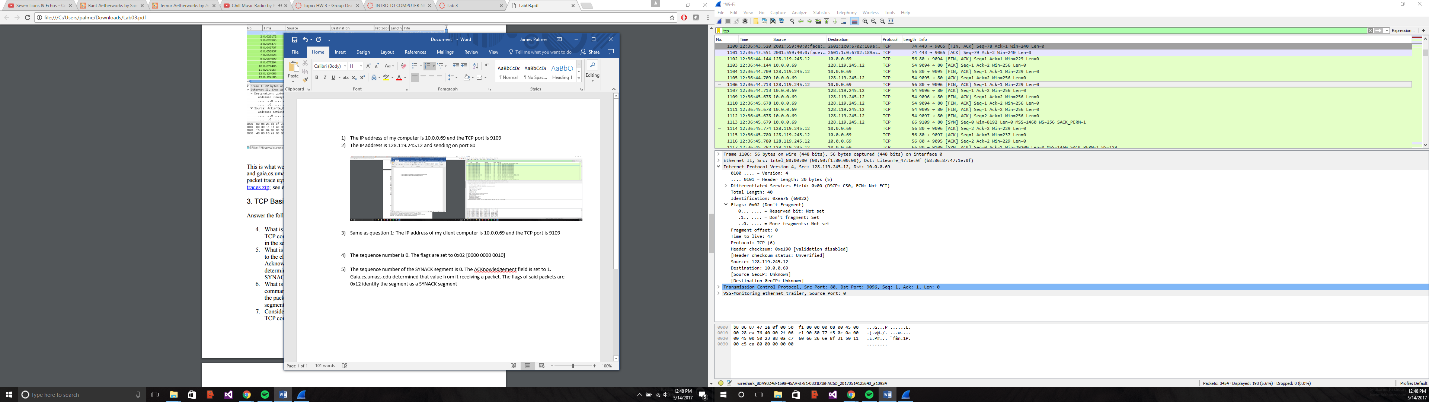
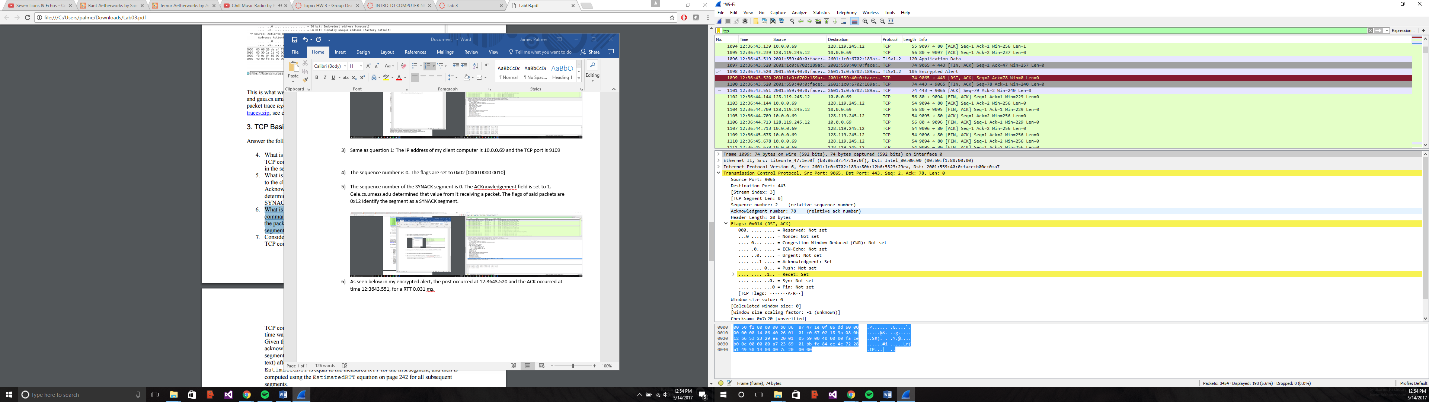
1. The IP address of my computer is 10.0.0.69 and the TCP port is 9109
2. The IP address is 128.119.245.12 and sending on port 80



1. Same as question 1: The IP address of my client computer is 10.0.0.69 and the TCP port is 9109
2. The sequence number is 0. The flags are set to 0x02 [0000 0000 0010]
3. The sequence number of the SYNACK segment is 0. The ACKnowledgement field is set to 1. Gaia.cs.umass.edu determined that value from it receiving a packet. The flags of said packets are 0x12 identify the segment as a SYNACK segment.



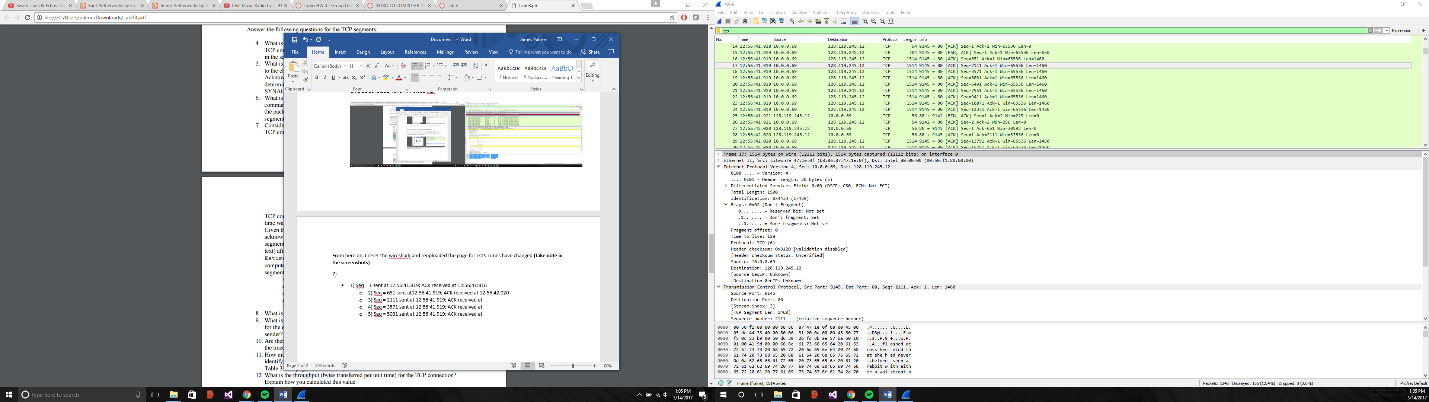
1. As seen below in my encrypted alert, the post occurred at 12:3643.520 and the ACK occurred at time 12:3643.551, for a RTT 0.031 ms.

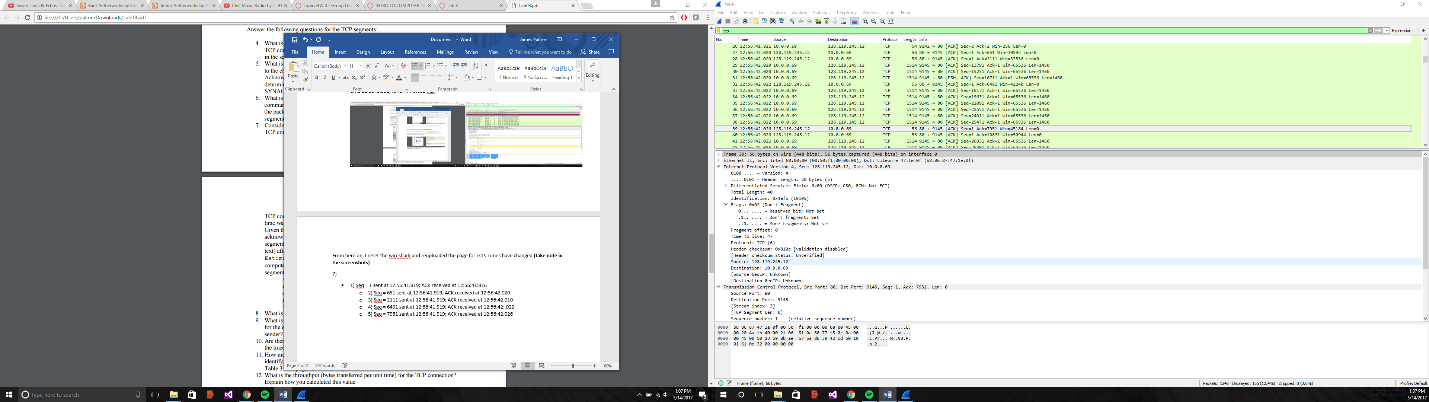


From here on, I reset the wireshark and reuploaded the page for TCP. Times have changed **(take note in the screenshots)**

7)

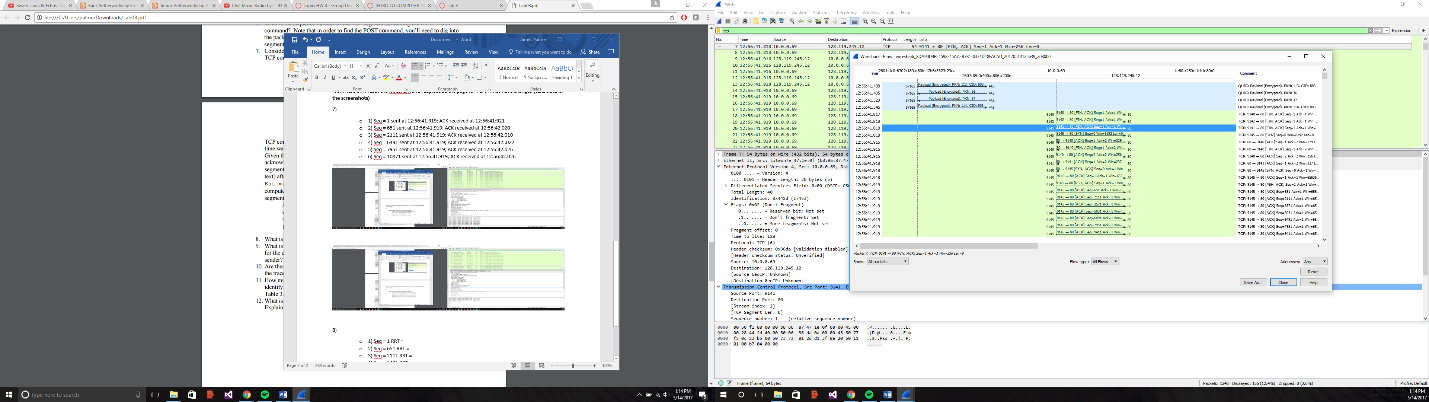
* + 1) Seq = 1 sent at 12:56:41.919; ACK received at 12:56:41:921
  + 2) Seq = 651 sent at 12:56:41.919; ACK received at 12:56:42.020
  + 3) Seq = 2111 sent at 12:56:41.919; ACK received at 12:56:42.010
  + 4) Seq = 6491 sent at 12:56:41.919; ACK received at 12:56:42.022
  + 5) Seq = 7951 sent at 12:56:41.919; ACK received at 12:56:42.026
  + 6) Seq = 10871 sent at 12:56:41.919; ACK received at 12:56:42.026





8)

* 1) Seq = 1 RRT = .002ms
  + 2) Seq = 651 RRT = .101ms
  + 3) Seq = 2111 RRT = .091ms
  + 4) Seq = 6491 RRT = .103ms
  + 5) Seq = 7951 RRT = .107ms
  + 6) Seq = 10871 RRT = .107ms



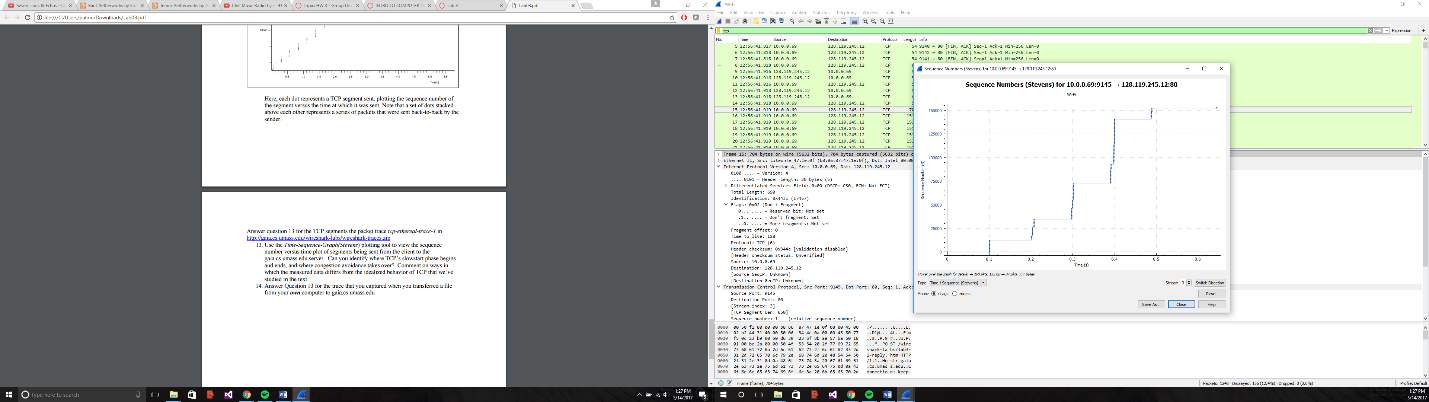
9) The minimum amount of available buffer space is advertised as 5840 bytes. The lack of receiver buffer space never throttles the sender.

10) No there aren’t any retransmitted segments.

11) 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. For example, segment of No. 80 acknowledged data with 2920 bytes = 1460\*2 bytes (times 2 bytes because they are doubled).

12) 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 (i.e. 1 byte for No. 4 segment) and the acknowledged sequence number of the last ACK (164091 bytes for No. 202 segment). 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 (i.e., 0.026477 second for No.4 segment) and the time instant of the last ACK (i.e., 5.455830 second for No. 202 segment). 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 KByte/sec.

**TCP congestion control in action**



13) The slow start phase begins around zero and ends around .15 seconds in accordance to the graph; after that congestion takes over. The measured data uses only a fraction of the window size instead of the 1/3 to a half.

14) My graph shows something a little different. The slow start phase begins around zero and ends around .10 seconds in accordance to the graph; after that there is little to no congestion. The measured data uses roughly the same window size as the graph scales accordingly. Until roughly seq 140483 around the .5 seconds’ mark. The measure data is much larger over a longer space of time before the Wireshark run sequence was stopped. This could be due to a change in song from another tab (i.e. other bandwidth needs are sent elsewhere and/or more spread out).