Bryan Arnold

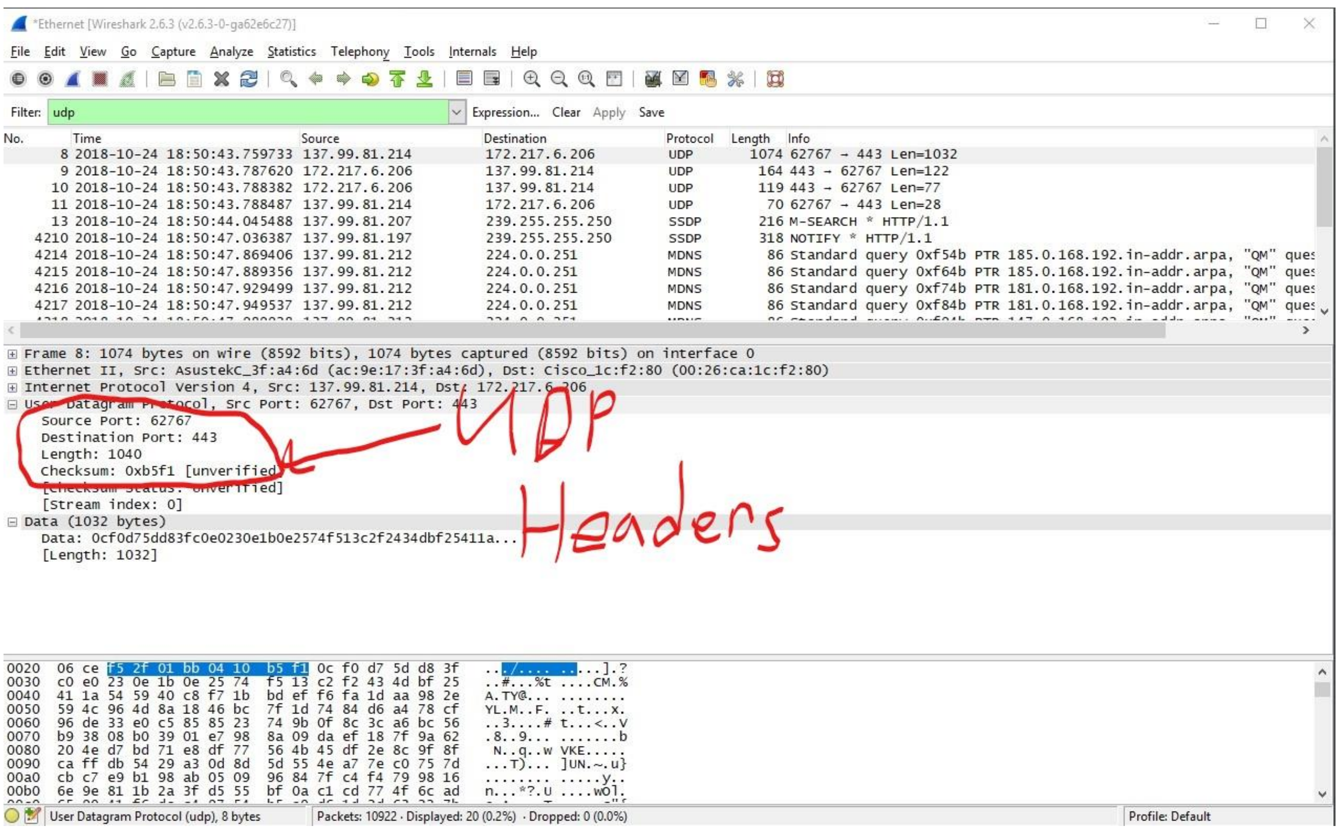
CSE 3300

10/24/18

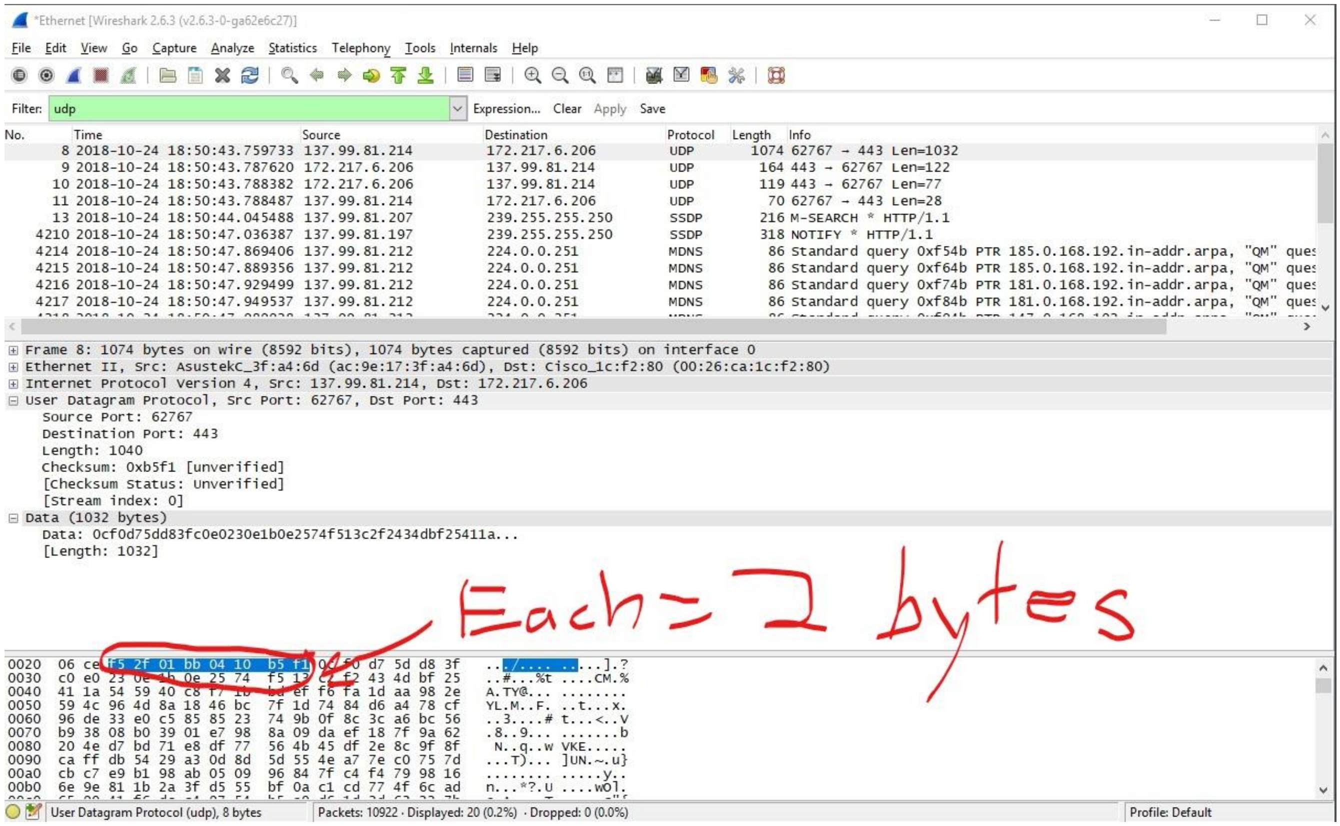
Programming Assignment 2

Part 1: UDP

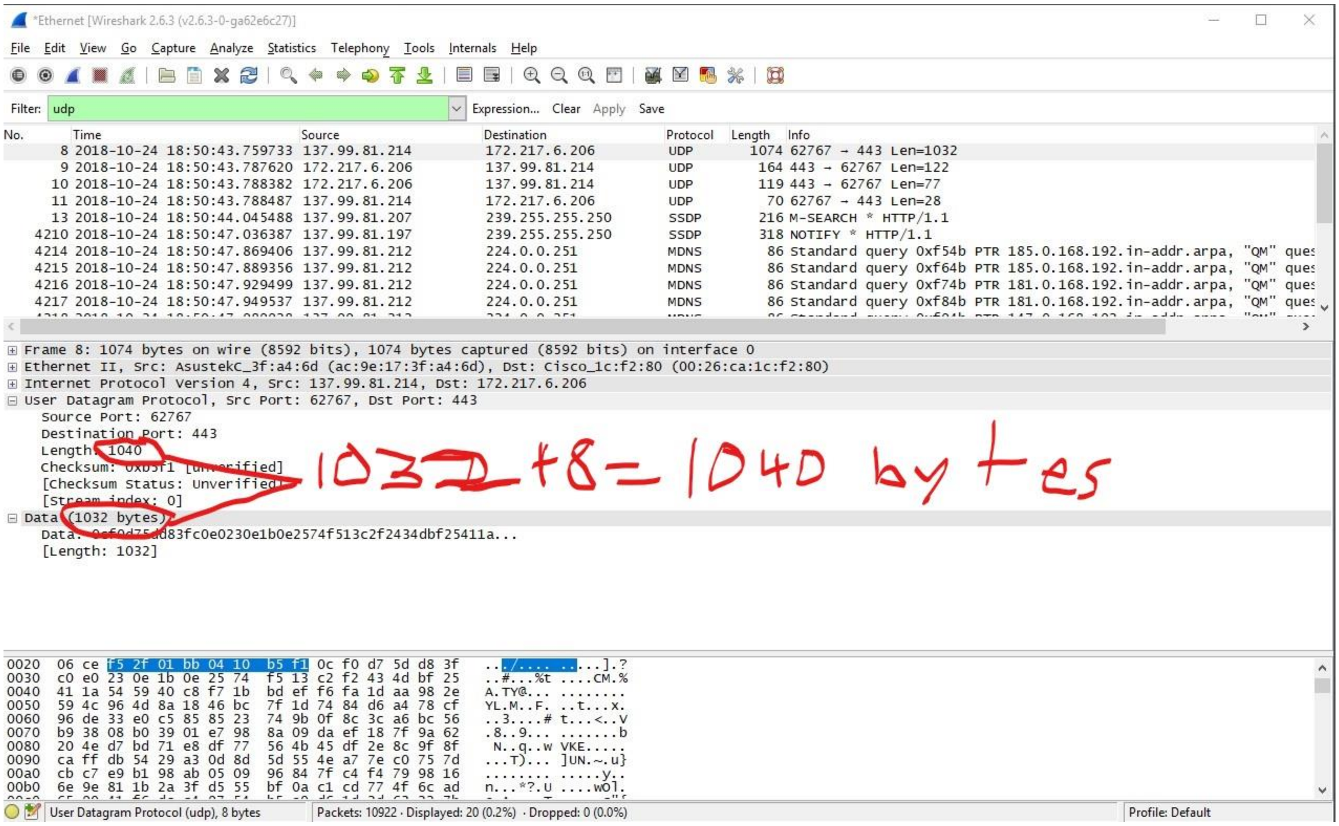
1) There is a total of 4 headers in this UDP packet: Source Port, Destination Port, Length, and Checksum:



2) Each UDP header field is 2 bytes long:



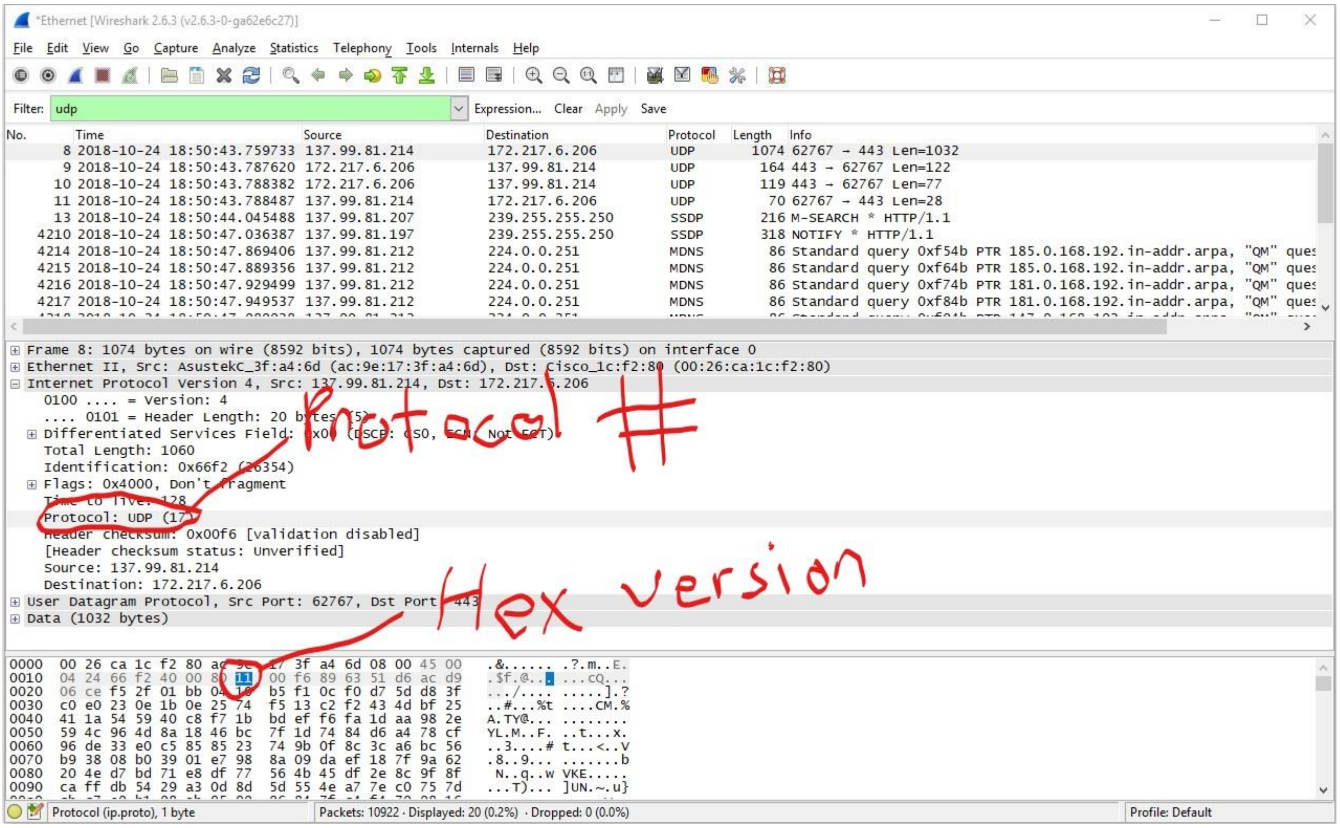
3) The Length header value, 1040 bytes, is the sum of two things. First, it is the sum of the length of each header in the UDP packet, 2 * 4 = 8 bytes. The rest of the bytes are the data included in the packet in the Data section of the packet (1032 + 8 = 1040):



4) The maximum number of bytes that can be sent in a UDP packet is 2 ^ 16 bytes. But, since there needs to be 8 reserve bytes for the headers, there is a maximum payload size of 2^16 – 8 = 65527 bytes.

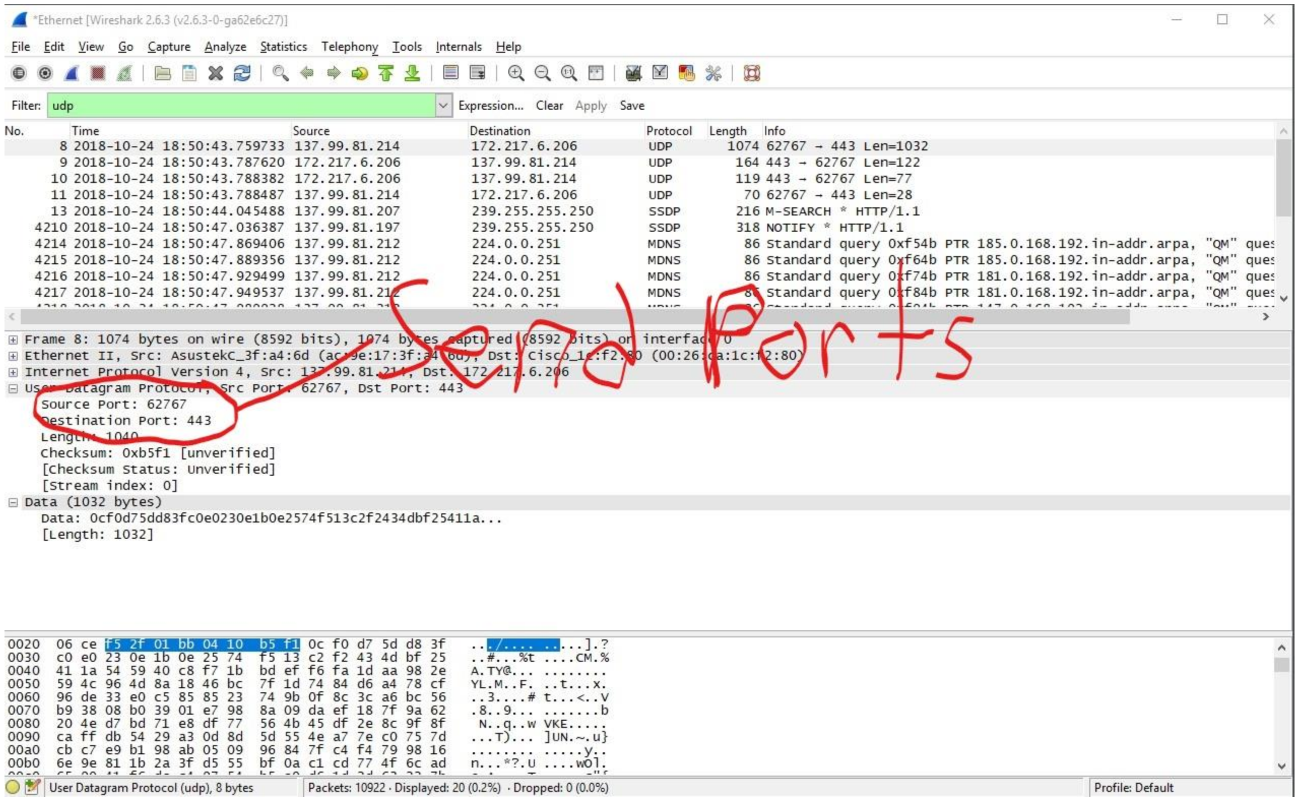
5) The largest port number corresponds to the maximum payload size not including the header bytes, so the largest port number is 65535.

6) The protocol for UDP is indicated by 17 and the hexadecimal notation of this value can be found below as 0x11:

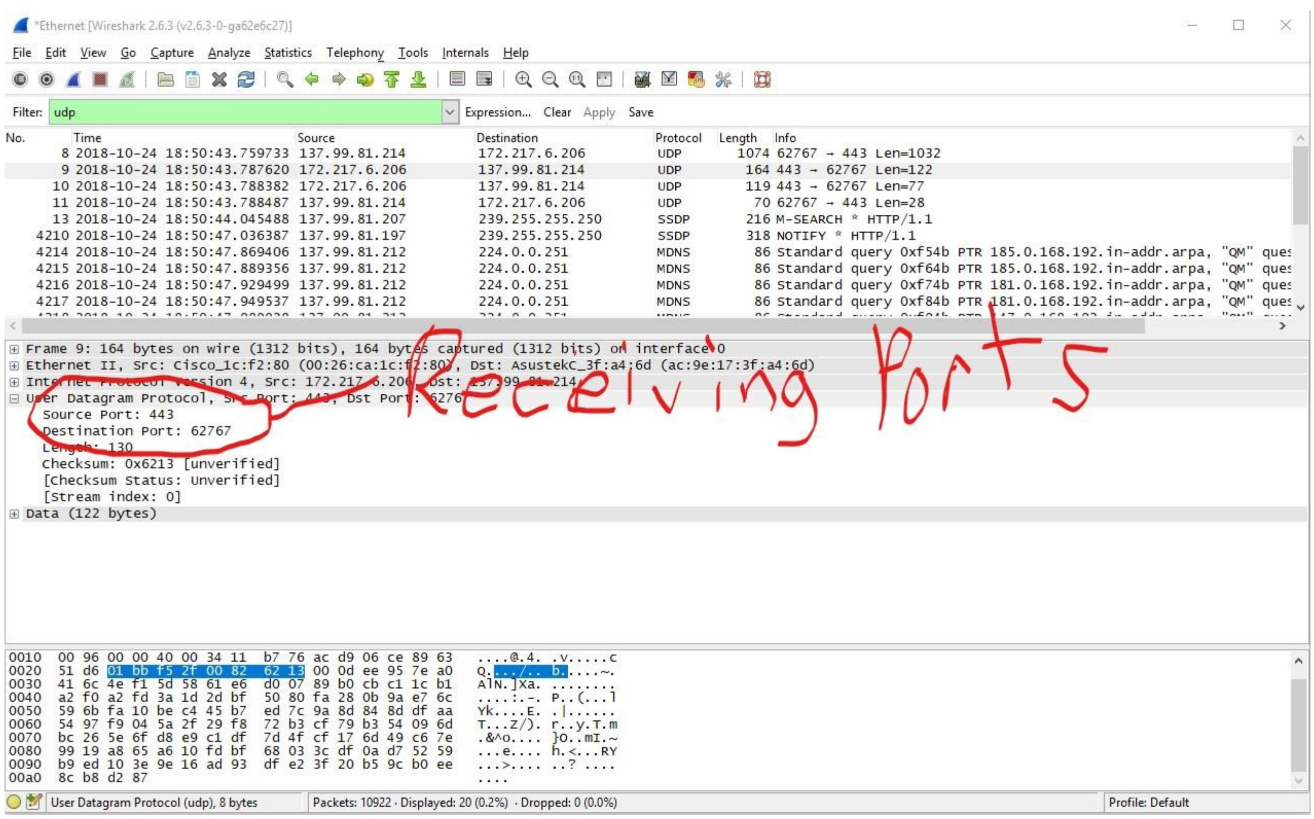


7) The relationship between these two UDP packets is the port numbers. The Source Port of the UDP packet that sent the packet is the Destination Port of the receiving message UDP packet. Hence, the Destination Port for the packet that sent the message is the Source Port of the receiving message packet:

Sending Message Packet:

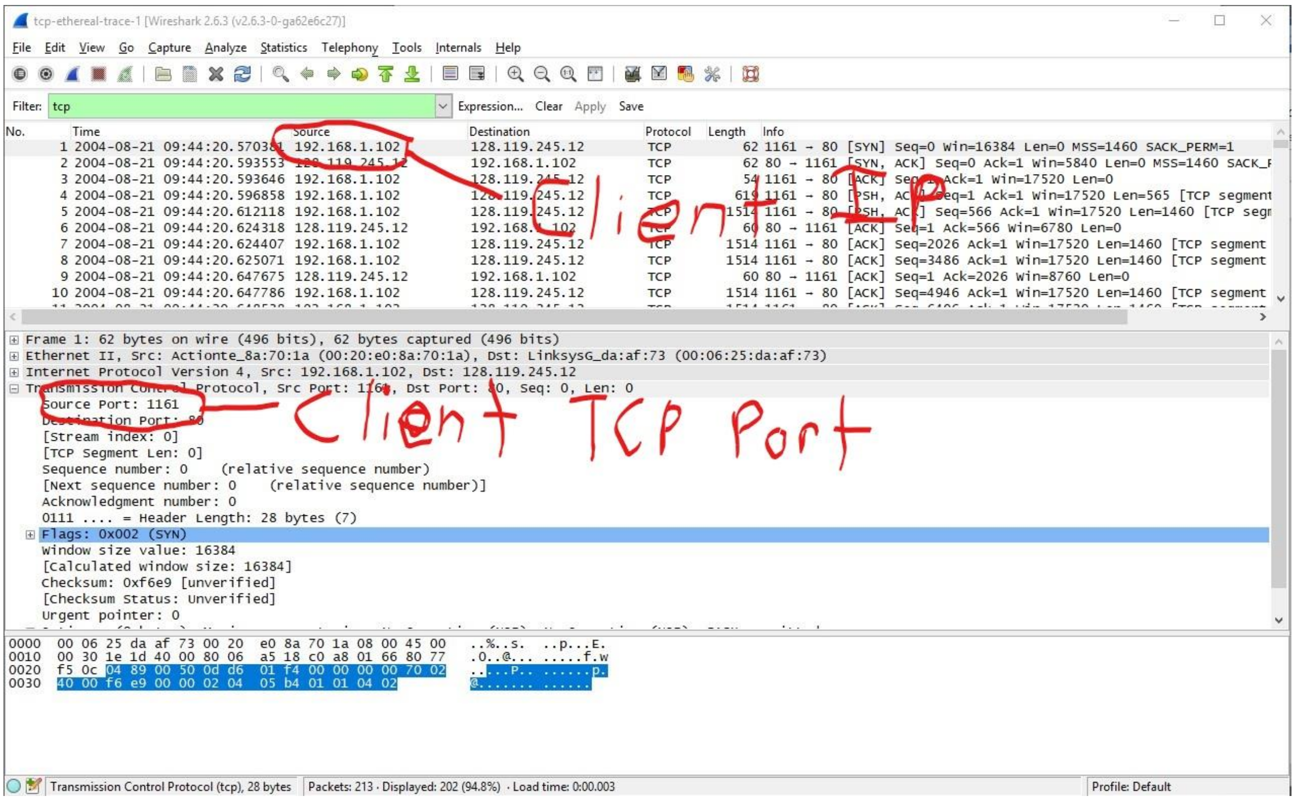


Receiving Ports:

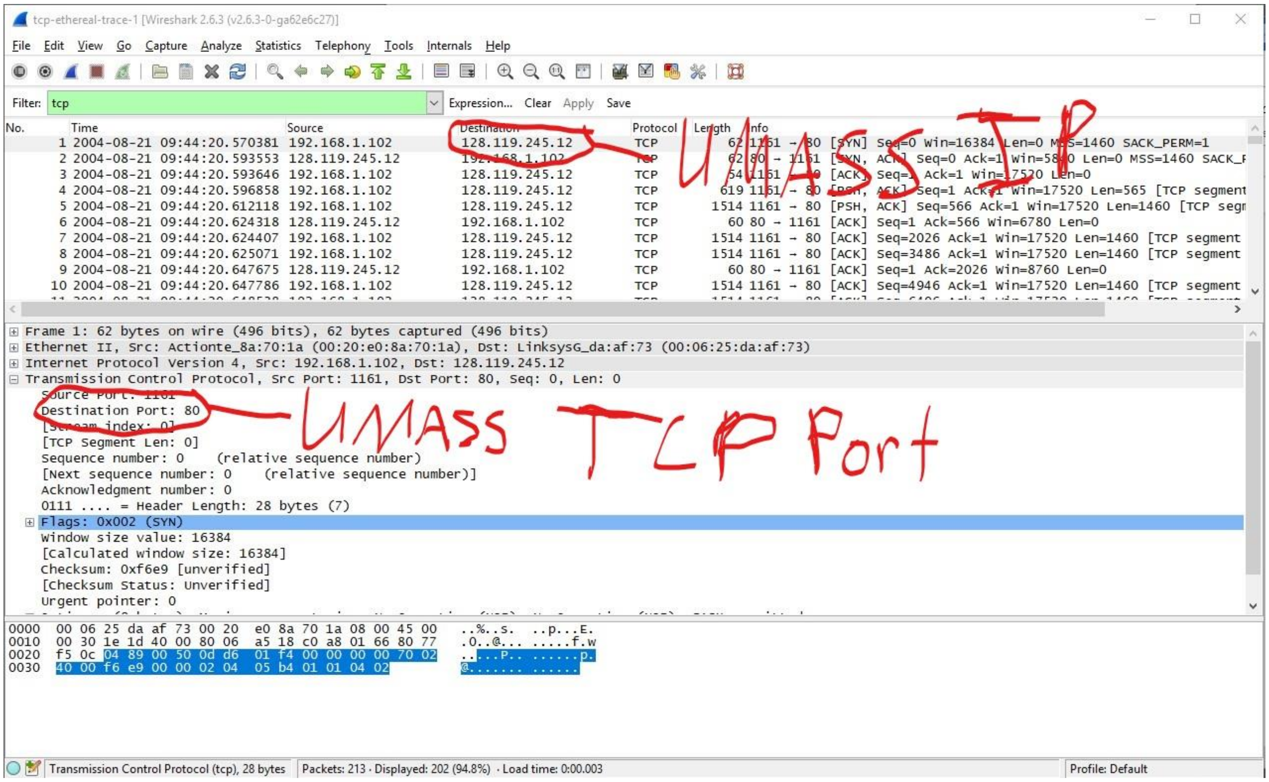


Part 2: TCP Bulk Transfer

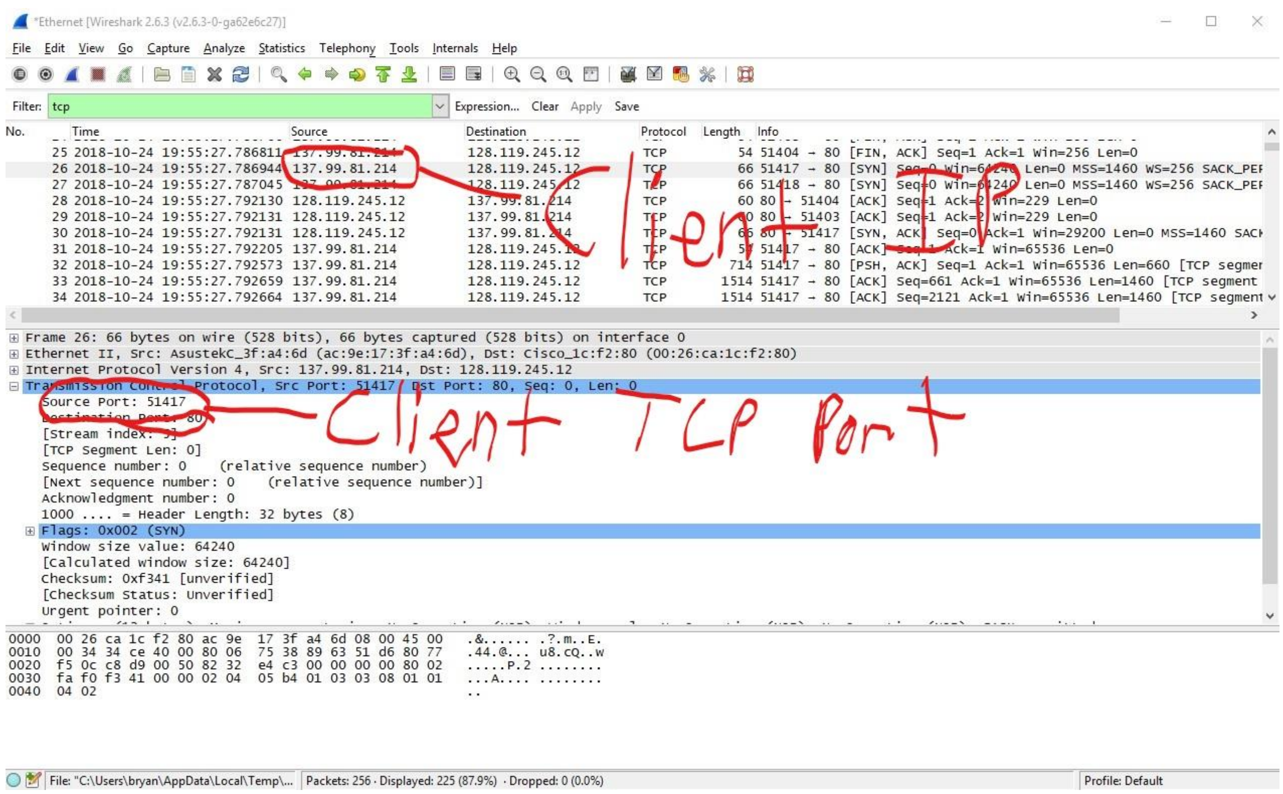
1) The IP address of the client computer is 192.168.1.102 and the TCP port number is 1161:



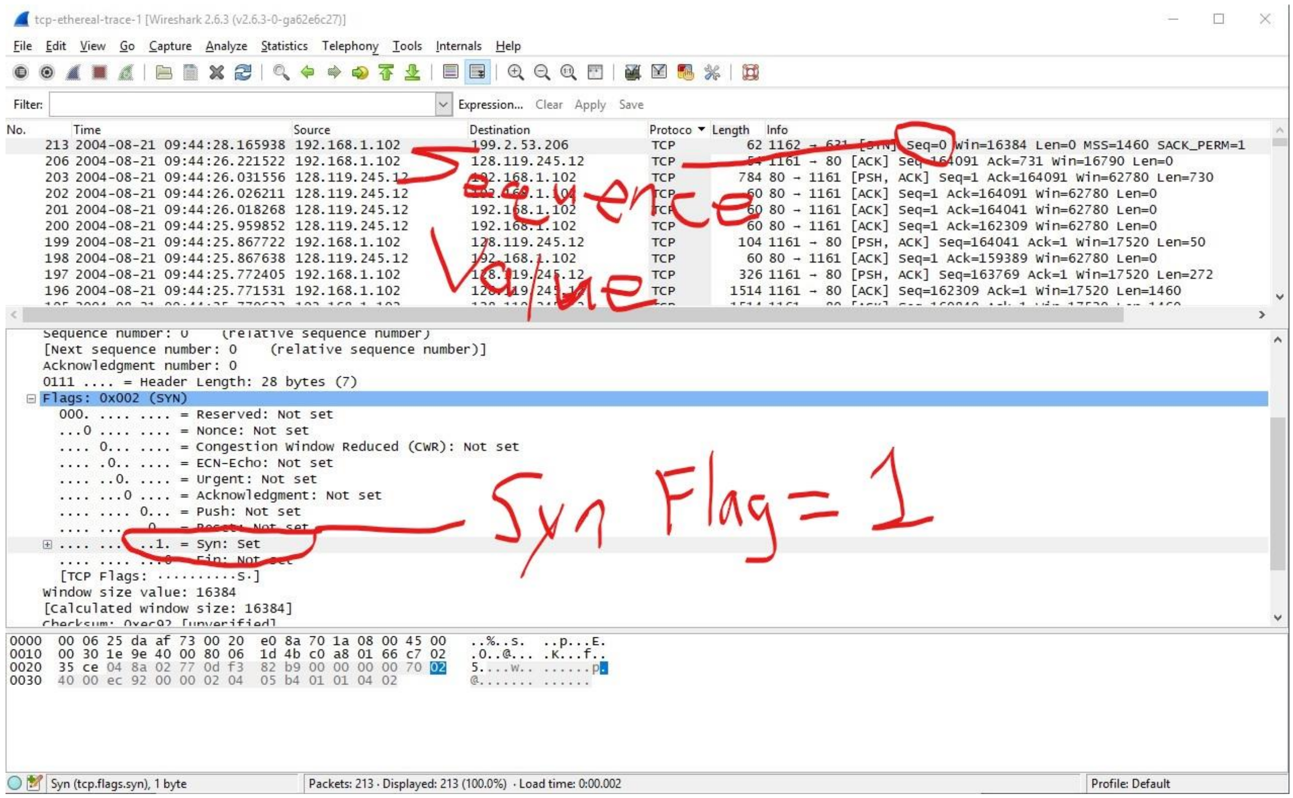
2) The IP address of gaia.cs.umass.edu is 128.119.245.12 and the TCP port number is 80:



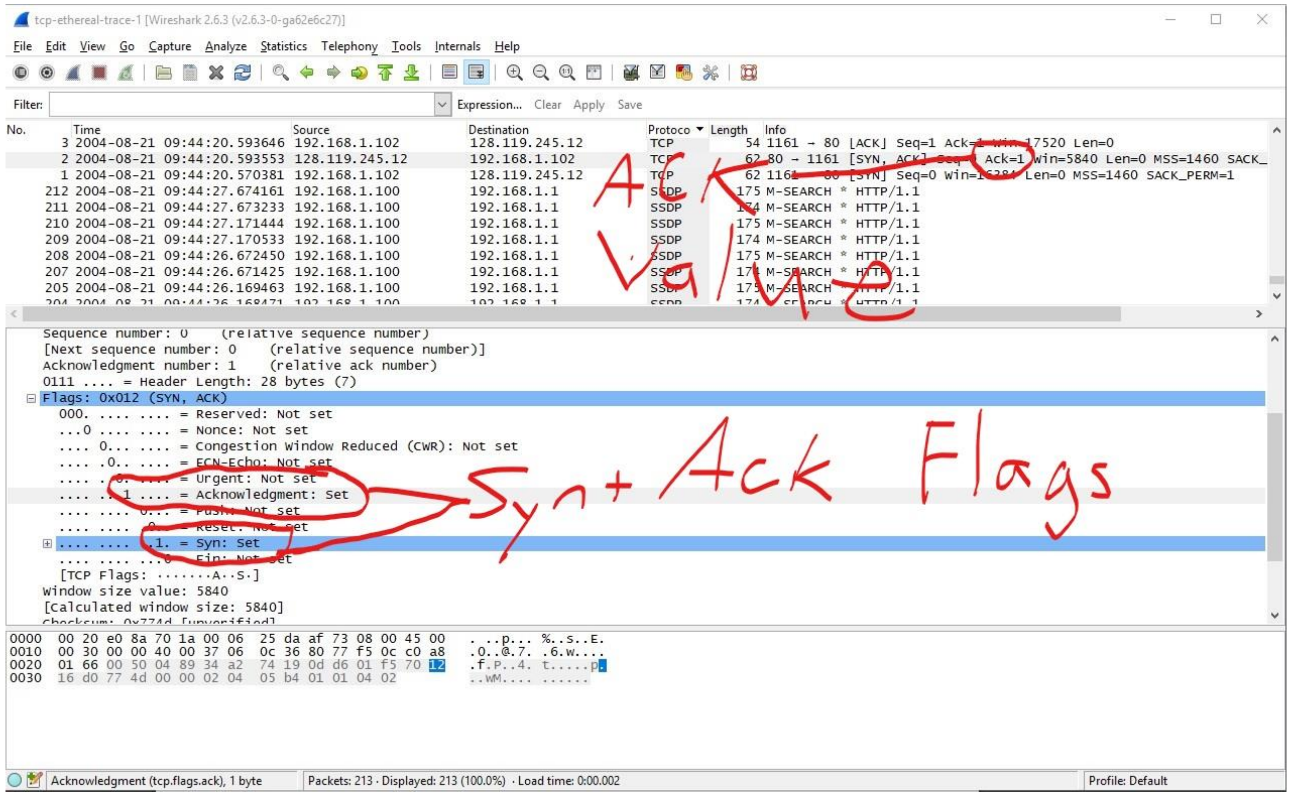
3) The IP address of my client computer is 137.99.81.214 and the TCP port number is 51417:



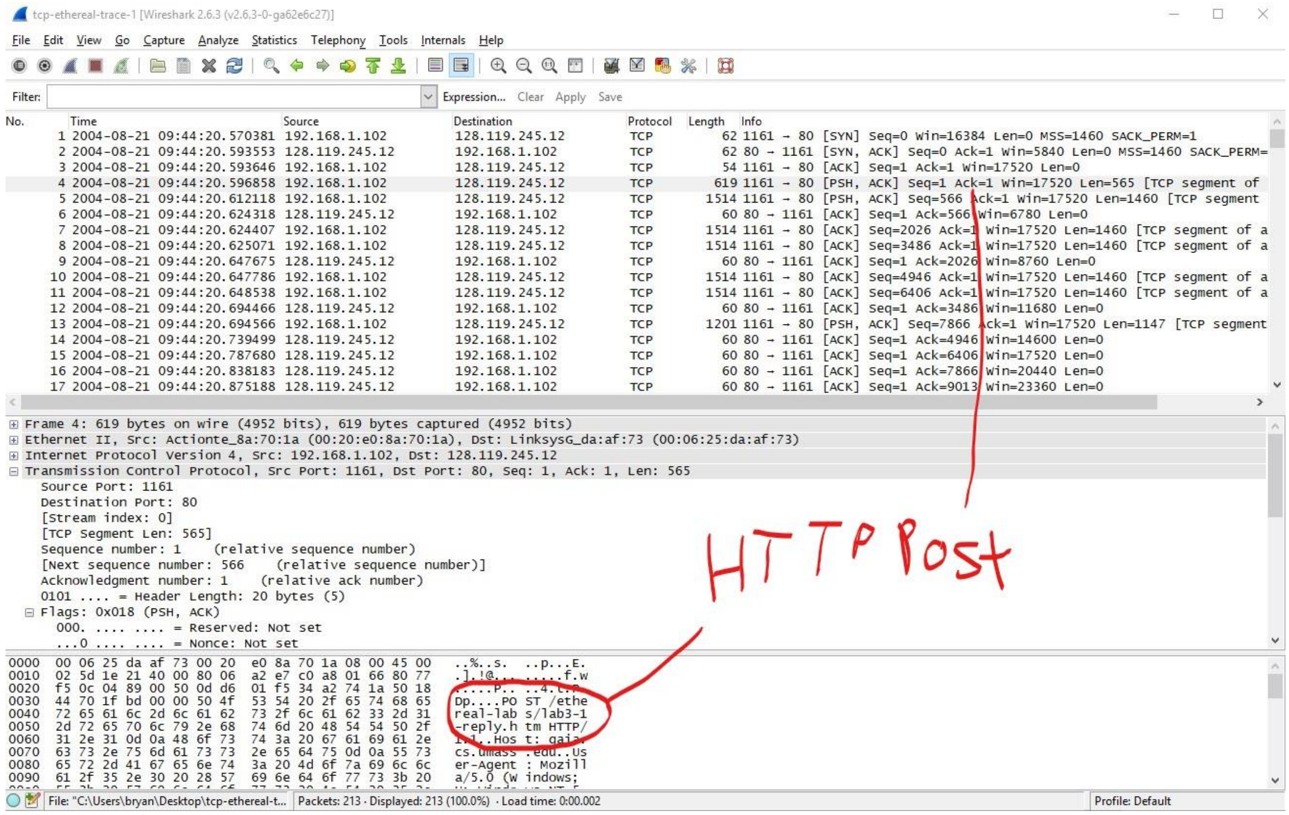
4) The sequence number of the top SYN packet is for the initiation of a connection to the destination computer. In this specific trace, the sequence number value is 0. The segment that indicates that this packet is a SYN one, is in the Flags section of the protocol information. It is 1 in this case, to indicate the packet is a SYN:



5) The value of Acknowledgement field in the SYNACK segment is 1. The destination server determines this value by adding on 1 to the initial sequence value of the SYN segment sent earlier. The segment that determines that the segment is a SYNACK one is in the flags section in the protocol information. Both the Acknowledgement and Syn flags are equal 1, which means the packet is flagged to be a SYNACK:

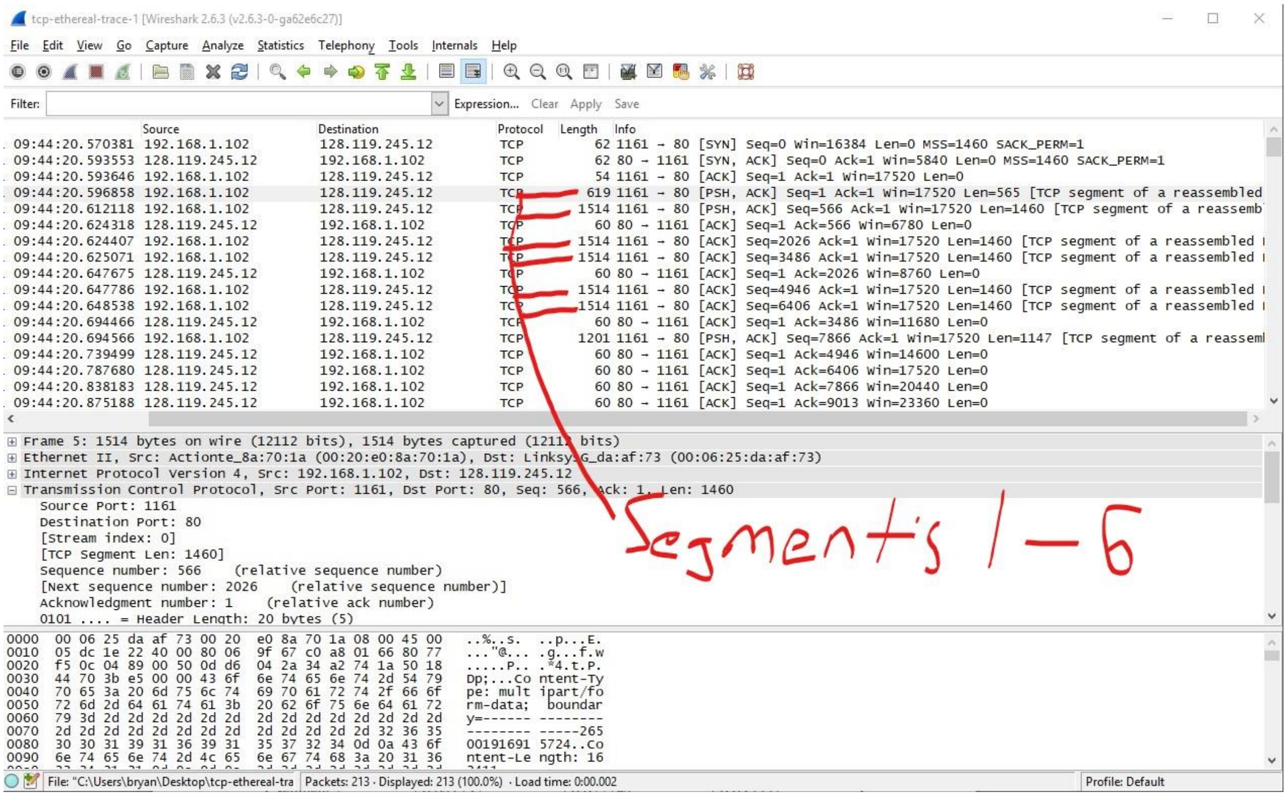


6) The 4th packet in the trace is the packet that contains the HTTP POST. The sequence value for this segment is 1:

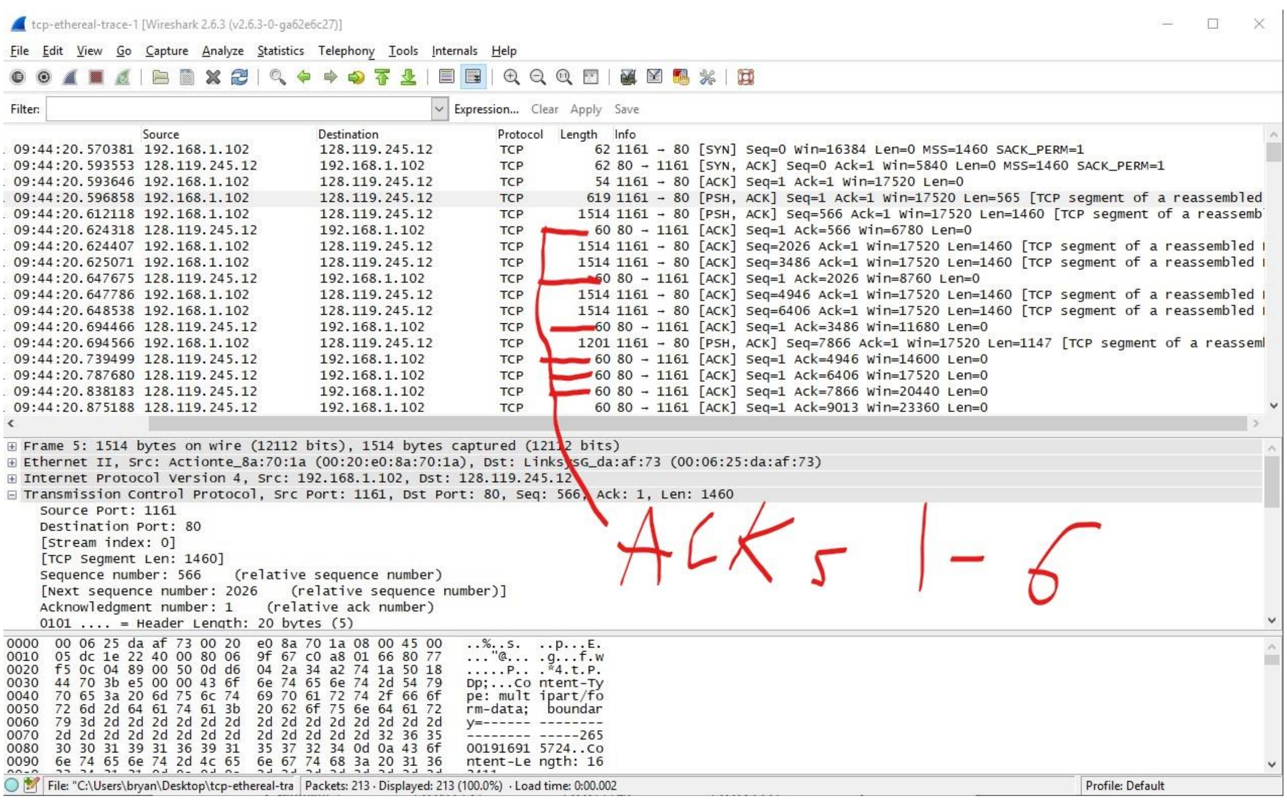


7) The first six segments in the HTTP POST are packet numbers 4, 5, 7, 8, 10, and 11. The corresponding ACK segments of these segments are packets numbers 6, 9, 12, 14, 15, and 16 respectively.

(Segments 1-6):



(ACK segments of segments 1-6):



Segment 1 Seq = 1, Segment 2 Seq = 566, Segment 3 Seq = 2026, Segment 4 Seq = 3486, Segment 5 Seq = 4946, Segment Seq 6 = 6406.

Here are the times of when the segments were sent, the ACK was received back, and the RTT time:

Segment 1: Sent = 0.026477s, ACK received = 0.053937s, RTT = 0.02746s
Segment 2: Sent = 0.041737s, ACK received = 0.077294s, RTT = 0.035557s
Segment 3: Sent = 0.054026s, ACK received = 0.124085s, RTT = 0.070059s
Segment 4: Sent = 0.054690s, ACK received = 0.169118s, RTT = 0.11443s
Segment 5: Sent = 0.077405s, ACK received = 0.217299s, RTT = 0.13989s
Segment 6: Sent = 0.078157s, ACK received = 0.267802s, RTT = 0.18964s

Now, here are the calculations for each EstimatedRTT:

Segment 1 EstimatedRTT = RTT Segment 1 = 0.02746 seconds

Segment 2 EstimatedRTT = 0.875 * Segment 1 EstimatedRTT + 0.125 * RTT Segment 2 = 0.0285 seconds

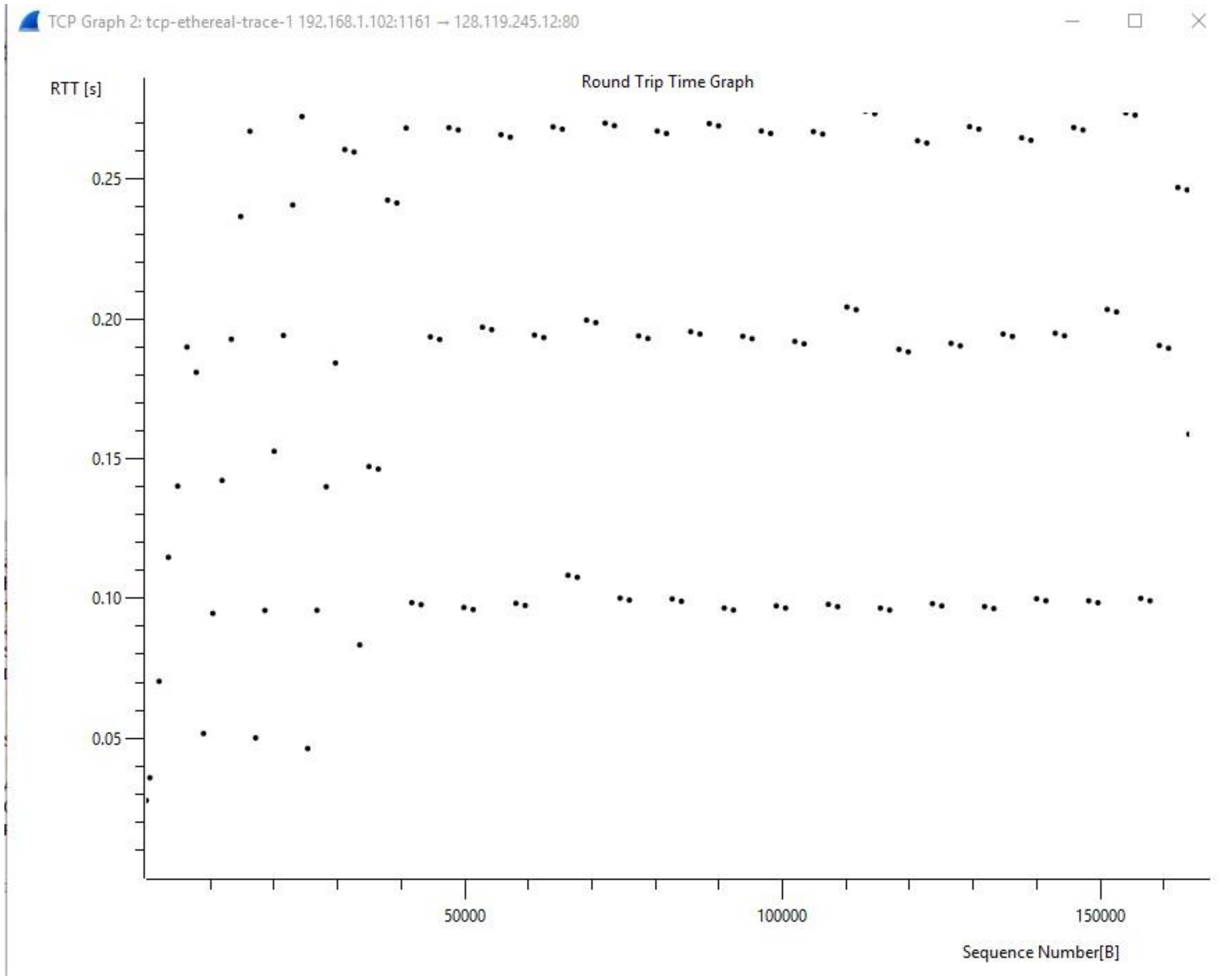
Segment 3 EstimatedRTT = 0.875 * Segment 2 EstimatedRTT + 0.125 * RTT Segment 3 = 0.0337

Segment 4 EstimatedRTT = 0.875 * Segment 3 EstimatedRTT + 0.125 * RTT Segment 4 = 0.0438

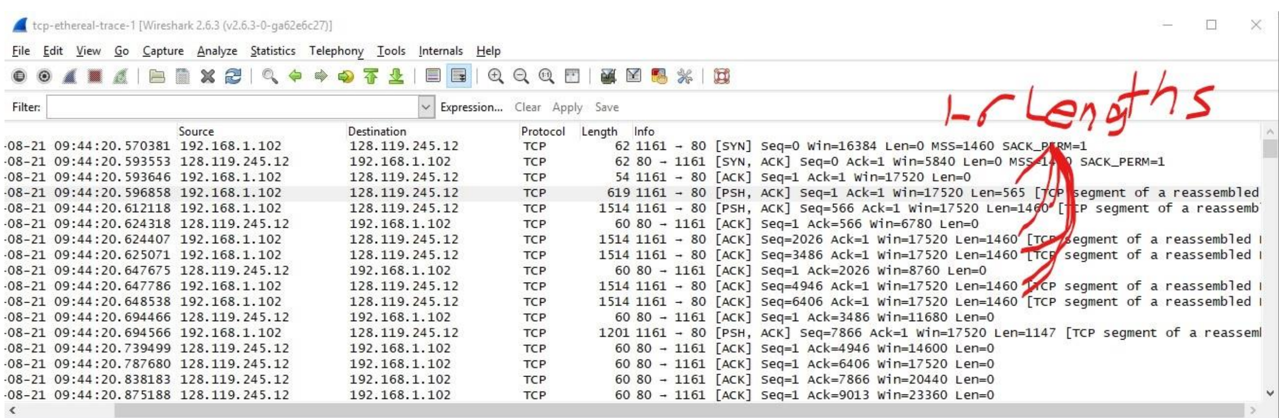
Segment 5 EstimatedRTT = 0.875 * Segment 4 EstimatedRTT + 0.125 * RTT Segment 5 = 0.0558

Segment 6 EstimatedRTT = 0.875 * Segment 5 EstimatedRTT + 0.125 * RTT Segment 6 = 0.0725

(Graph of TCP Segments and ACKS)

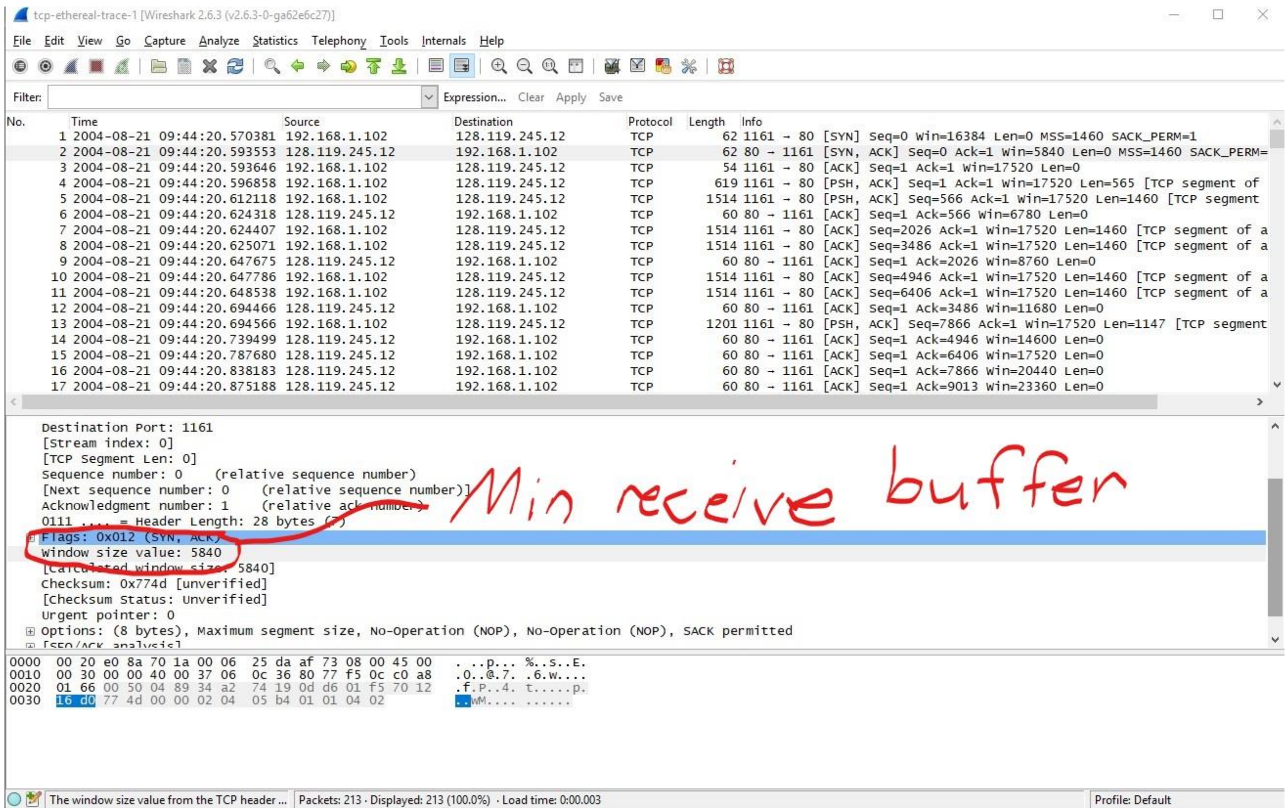


8) The Lengths of the first 6 TCP segments are 565, 1460, 1460, 1460, 1460, and 1460 respectively:

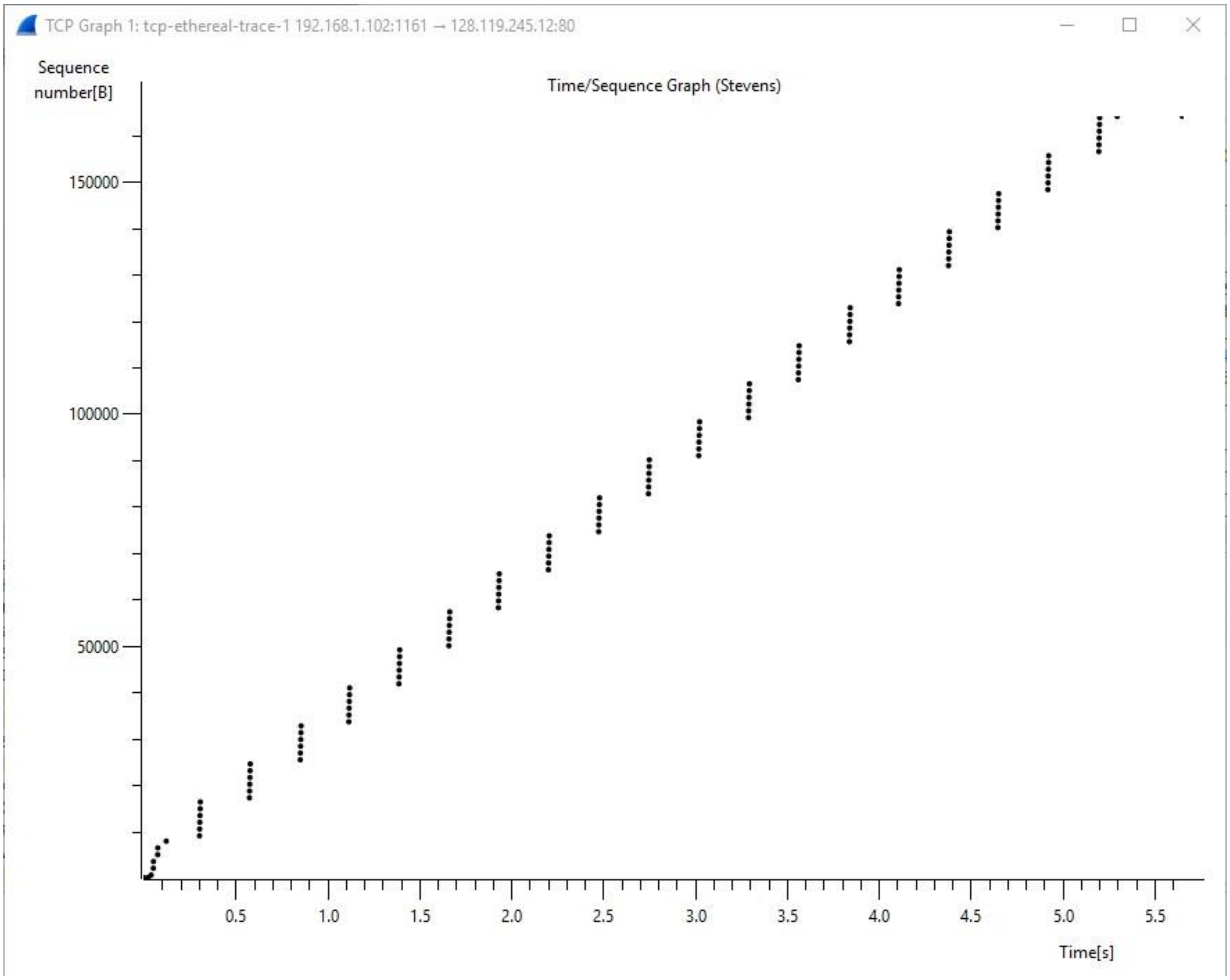


9) The minimum buffer space that the destination computer advertises usually is seen in the very first acknowledgement response from the server. This would be in the first

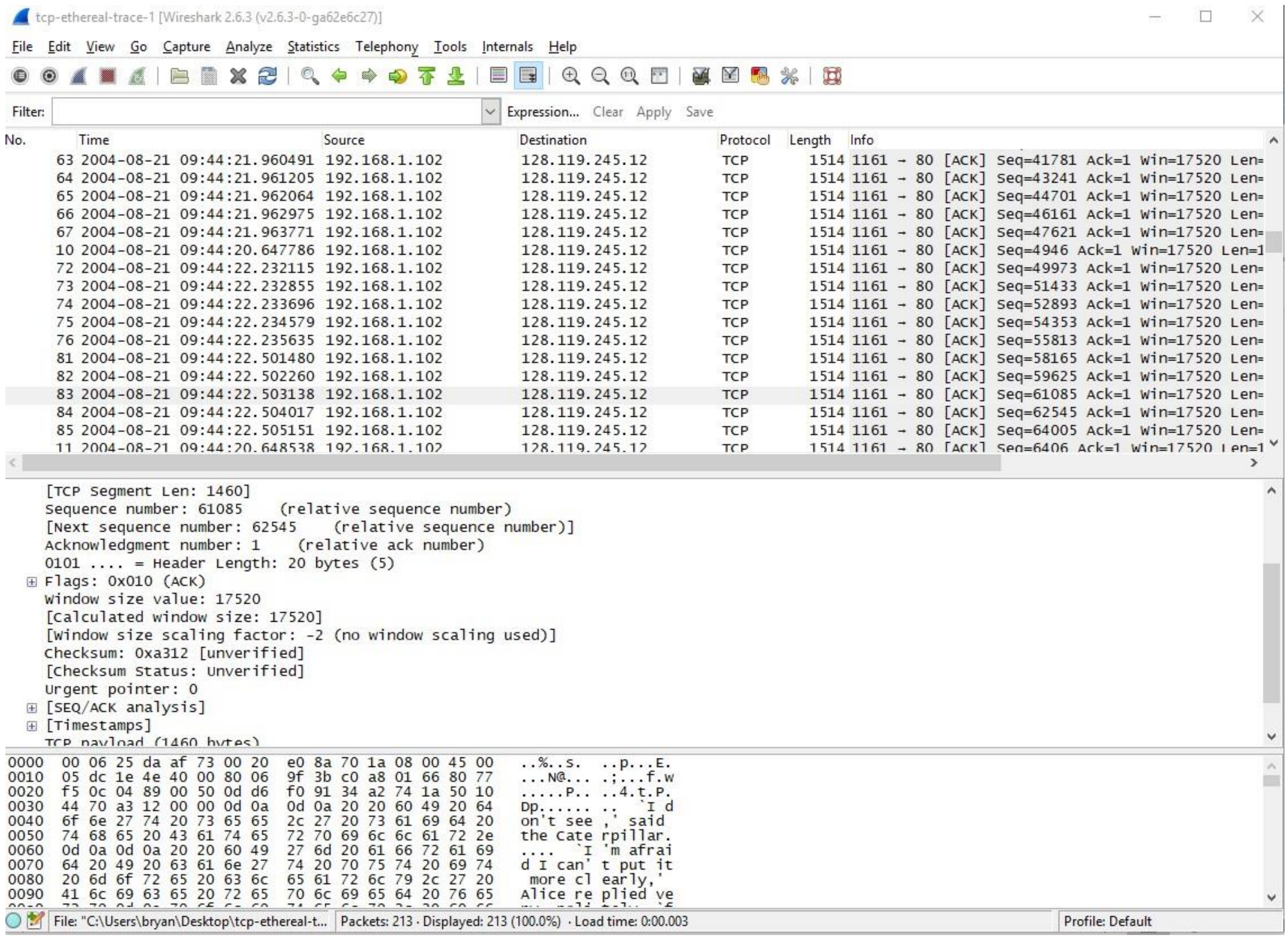
ACK segments in the trace, and the minimum buffer space is 5840 bytes. The sender is never throttled in this trace for a lack of receiving buffer:



10) None of the segments in the trace were retransmitted. By using the Time-Sequence-Graph for the entire trace, also given to us in the homework pdf, we can see the sequence numbers from the source increasing linearly with time. In order for a segment to be retransmitted, the segment must have a smaller sequence value than the segments next to it (which doesn't happen):

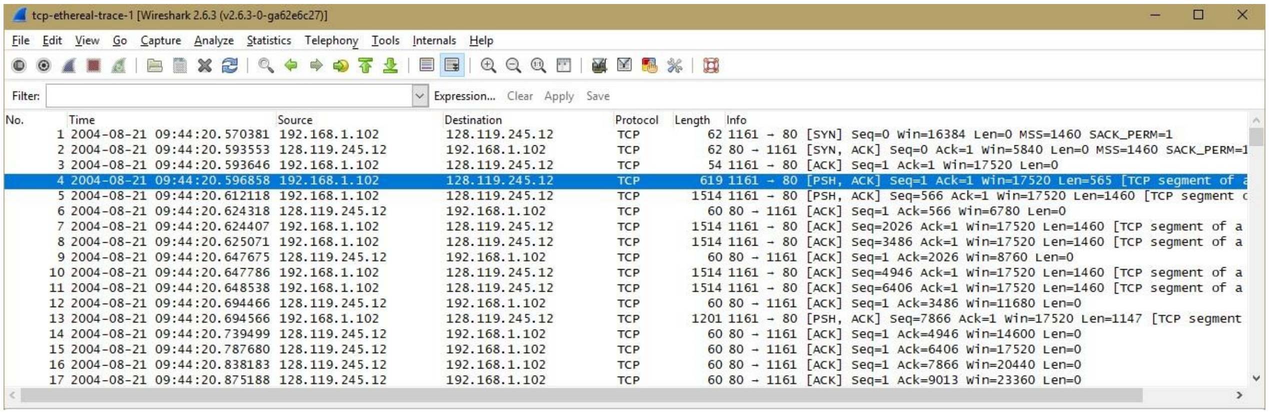


11) The amount of data that is usually received in an ACK is the difference of the sequence numbers of two ACKs. For instance, sequence number 566 for one ACK and then 2026 for the next ACK results in a total of 1460 acknowledged data. If you look closely at the data being acknowledged, sometimes the receiver is ACKing every other segment. For instance, segment 80 of the trace acknowledged 2920 bytes instead of 1460 bytes like it should have:



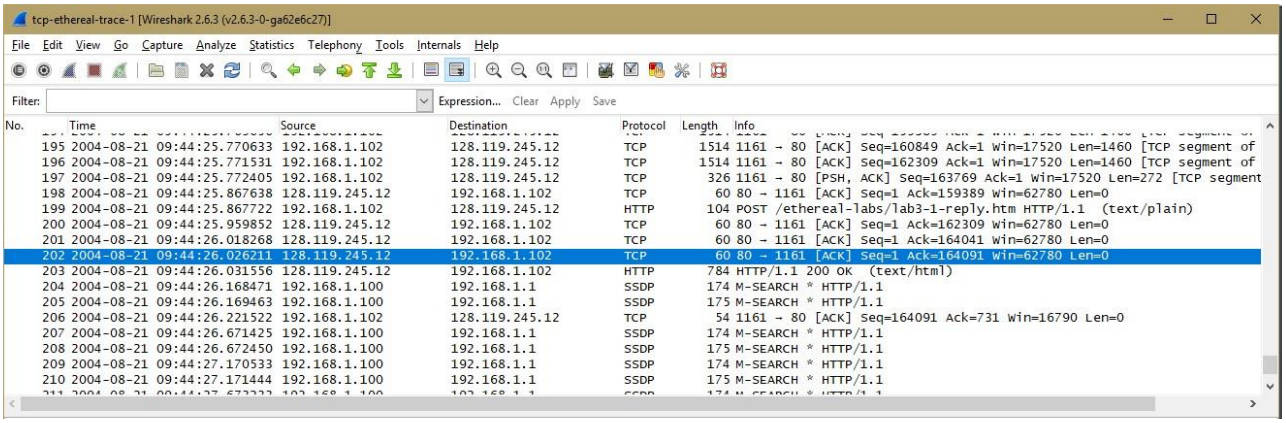
12) To find the throughput of the TCP connection, we need to know the total data sent and the time from the first ACK segment to the last ACK segment of the trace. To find the total amount of data transmitted, we need to look at sequence number of the very first ACK segment, and the sequence number of the very last ACK segment of the trace. This packet corresponds to packet 4 and 202. Sequence value for packet 4 is 1, and the sequence value for packet 202 is 164091. 164091 – 1 = 16490 gives us the total bytes transmitted. Now, we also need to look at the time the first ACK was received, and the last ACK. Looking at packets 4 and 202 again, we get 5.455830 – 0.026477 = 5.4294 seconds for the data to transmit. Finally, to get the total throughput, we do 16490 bytes / 5.4294 seconds = roughly 30.2 KBps:

Segment 1 (packet 4) sequence and time:



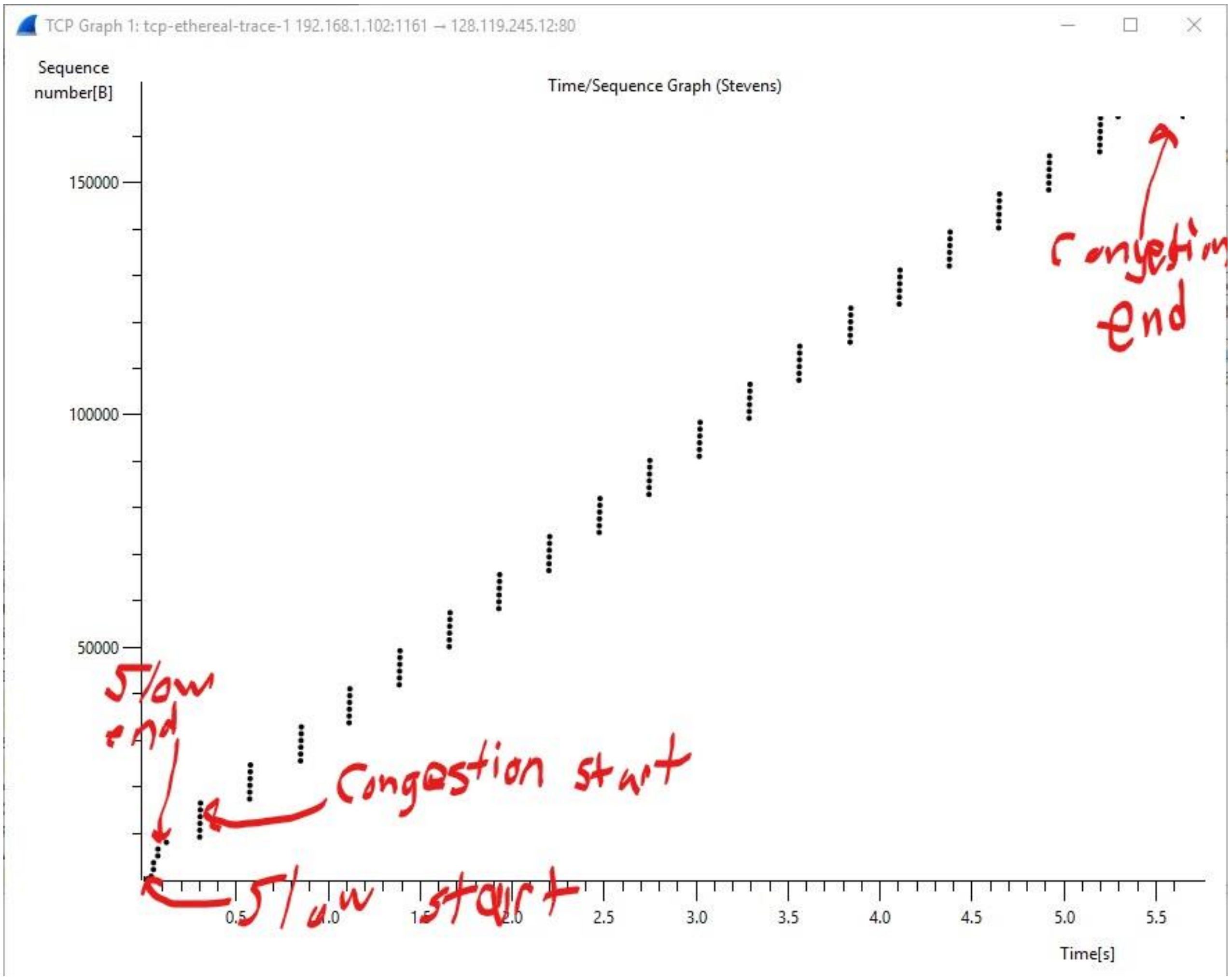
No.	Time	Source	Destination	Protocol	Length	Info
1	2004-08-21 09:44:20.570381	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1
2	2004-08-21 09:44:20.593553	128.119.245.12	192.168.1.102	TCP	62	80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
3	2004-08-21 09:44:20.593646	192.168.1.102	128.119.245.12	TCP	54	1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
4	2004-08-21 09:44:20.596858	192.168.1.102	128.119.245.12	TCP	619	1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment of a
5	2004-08-21 09:44:20.612118	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460 [TCP segment c
6	2004-08-21 09:44:20.624318	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
7	2004-08-21 09:44:20.624407	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP segment of a
8	2004-08-21 09:44:20.625071	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP segment of a
9	2004-08-21 09:44:20.647675	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
10	2004-08-21 09:44:20.647786	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP segment of a
11	2004-08-21 09:44:20.648538	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP segment of a
12	2004-08-21 09:44:20.694466	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	2004-08-21 09:44:20.694566	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 [TCP segment
14	2004-08-21 09:44:20.739499	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	2004-08-21 09:44:20.787680	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	2004-08-21 09:44:20.838183	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0
17	2004-08-21 09:44:20.875188	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=9013 Win=23360 Len=0

Last ACK segment (packet 202) size and time:

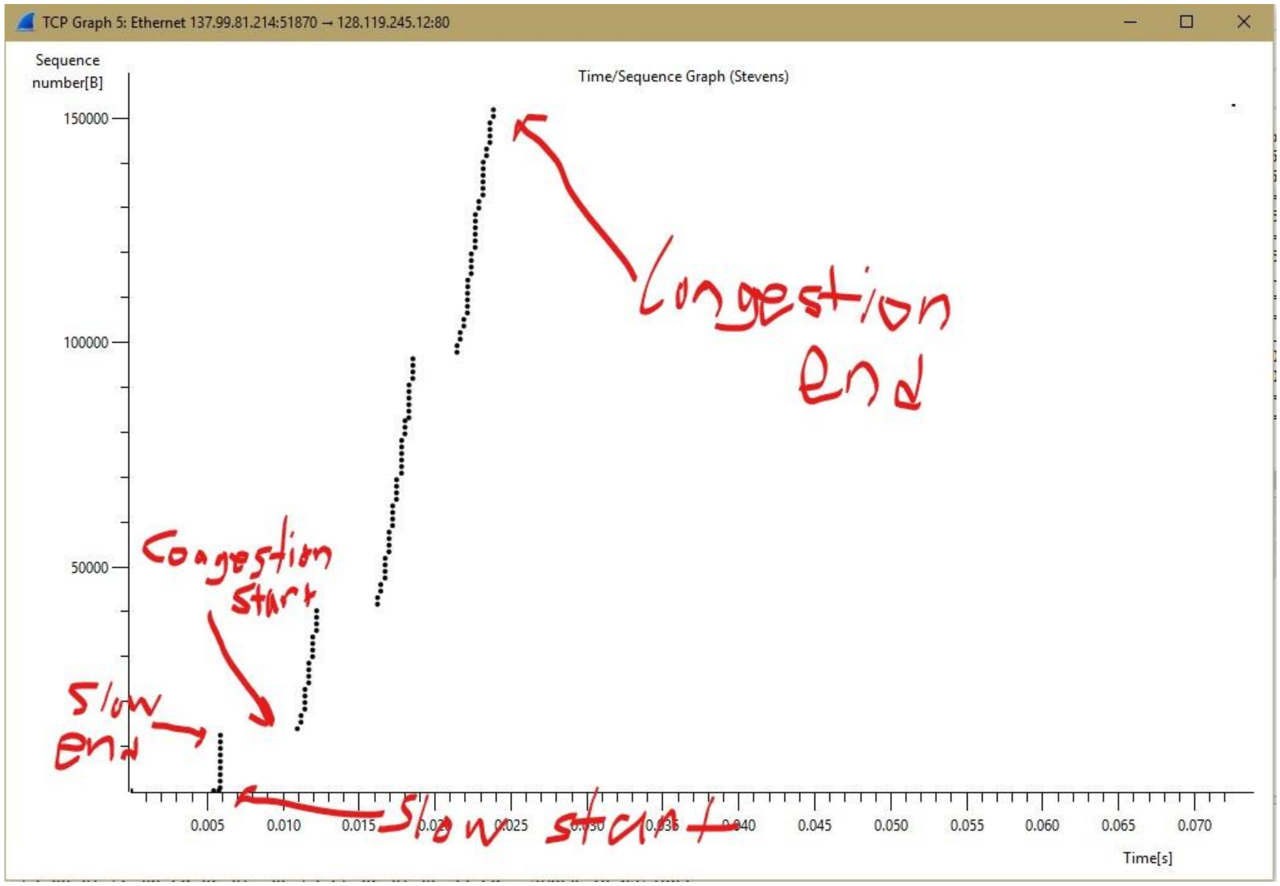


No.	Time	Source	Destination	Protocol	Length	Info
195	2004-08-21 09:44:25.770633	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=160849 Ack=1 Win=17520 Len=1460 [TCP segment of
196	2004-08-21 09:44:25.771531	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=162309 Ack=1 Win=17520 Len=1460 [TCP segment of
197	2004-08-21 09:44:25.772405	192.168.1.102	128.119.245.12	TCP	326	1161 → 80 [PSH, ACK] Seq=163769 Ack=1 Win=17520 Len=272 [TCP segment
198	2004-08-21 09:44:25.867638	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=159389 Win=62780 Len=0
199	2004-08-21 09:44:25.867722	192.168.1.102	128.119.245.12	HTTP	104	POST /ethereal-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
200	2004-08-21 09:44:25.959852	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=162309 Win=62780 Len=0
201	2004-08-21 09:44:26.018268	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=164041 Win=62780 Len=0
202	2004-08-21 09:44:26.026211	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=164091 Win=62780 Len=0
203	2004-08-21 09:44:26.031556	128.119.245.12	192.168.1.102	HTTP	784	HTTP/1.1 200 OK (text/html)
204	2004-08-21 09:44:26.168471	192.168.1.100	192.168.1.1	SSDP	174	M-SEARCH * HTTP/1.1
205	2004-08-21 09:44:26.169463	192.168.1.100	192.168.1.1	SSDP	175	M-SEARCH * HTTP/1.1
206	2004-08-21 09:44:26.221522	192.168.1.102	128.119.245.12	TCP	54	1161 → 80 [ACK] Seq=164091 Ack=731 Win=16790 Len=0
207	2004-08-21 09:44:26.671425	192.168.1.100	192.168.1.1	SSDP	174	M-SEARCH * HTTP/1.1
208	2004-08-21 09:44:26.672450	192.168.1.100	192.168.1.1	SSDP	175	M-SEARCH * HTTP/1.1
209	2004-08-21 09:44:27.170533	192.168.1.100	192.168.1.1	SSDP	174	M-SEARCH * HTTP/1.1
210	2004-08-21 09:44:27.171444	192.168.1.100	192.168.1.1	SSDP	175	M-SEARCH * HTTP/1.1

13) From the graph, we can see that the slow start phase begins when the HTTP POST command is sent, so the very first sequence number of the graph. This slow start ends when the very last sequence number in the first cluster is received an ACK. This indicates the start of the congestion avoidance phase. As seen in the graph, the clusters of sequences are spaced out from one another in their clusters, as well as other clusters, hence it is trying to avoid congestion. This phase ends when the trace stops. In terms of how the measured data compares to the aggressive nature of idealized TCP, too much traffic could interrupt the transmission of data. Idealized TCP has no packet loss, but due to the congestion in the data, it can happen, making it not match idealized TCP. Overall, the data measured here is fairly good match to idealized TCP:



14) For a file I transferred to the destination server, the same ways of telling when the slow start phase and congestion avoidance phase are the same as the previous. The measured data in this TCP trace was much quicker than the previous one. The overall transmission was a lot faster and the amount of sequences clustered together was much bigger than the previous as well. This was significantly quicker and a much more aggressive transmission of data, which aligns more with how idealized TCP should be. Hence, the file I sent is much better fit to idealized TCP.



TCP Graph 5: Ethernet 137.99.81.214:51870 → 128.119.245.12:80
Time/Sequence Graph (Stevens)
Sequence number[B]
150000
100000
50000
0.005
0.010
0.015
0.025
0.040
0.045
0.050
0.055
0.060
0.065
0.070
Time[s]
Congestion end
Congestion start
Slow end
Slow start