Setup: Part 1 of this lab involves uploading a file to a website and observing the TCP traffic captured during the process, demonstrating various TCP functionalities.

a. To begin, I downloaded the copy of Alice in Wonderland, navigated to the appropriate website, started my packet sniffer, and uploaded the file. Then I stopped the packet capture once I received the confirmation message.

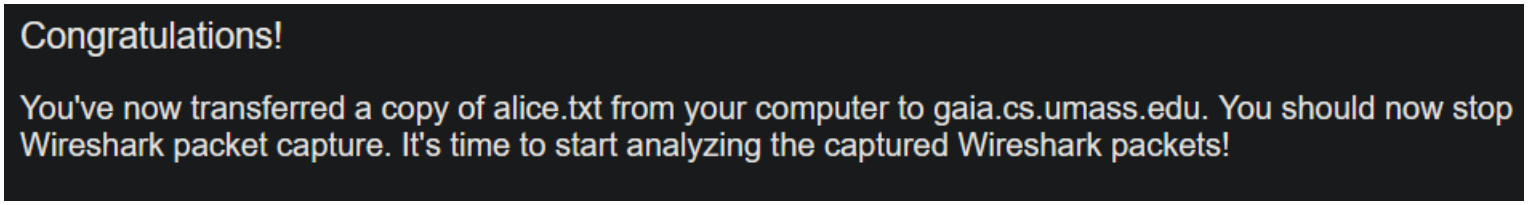
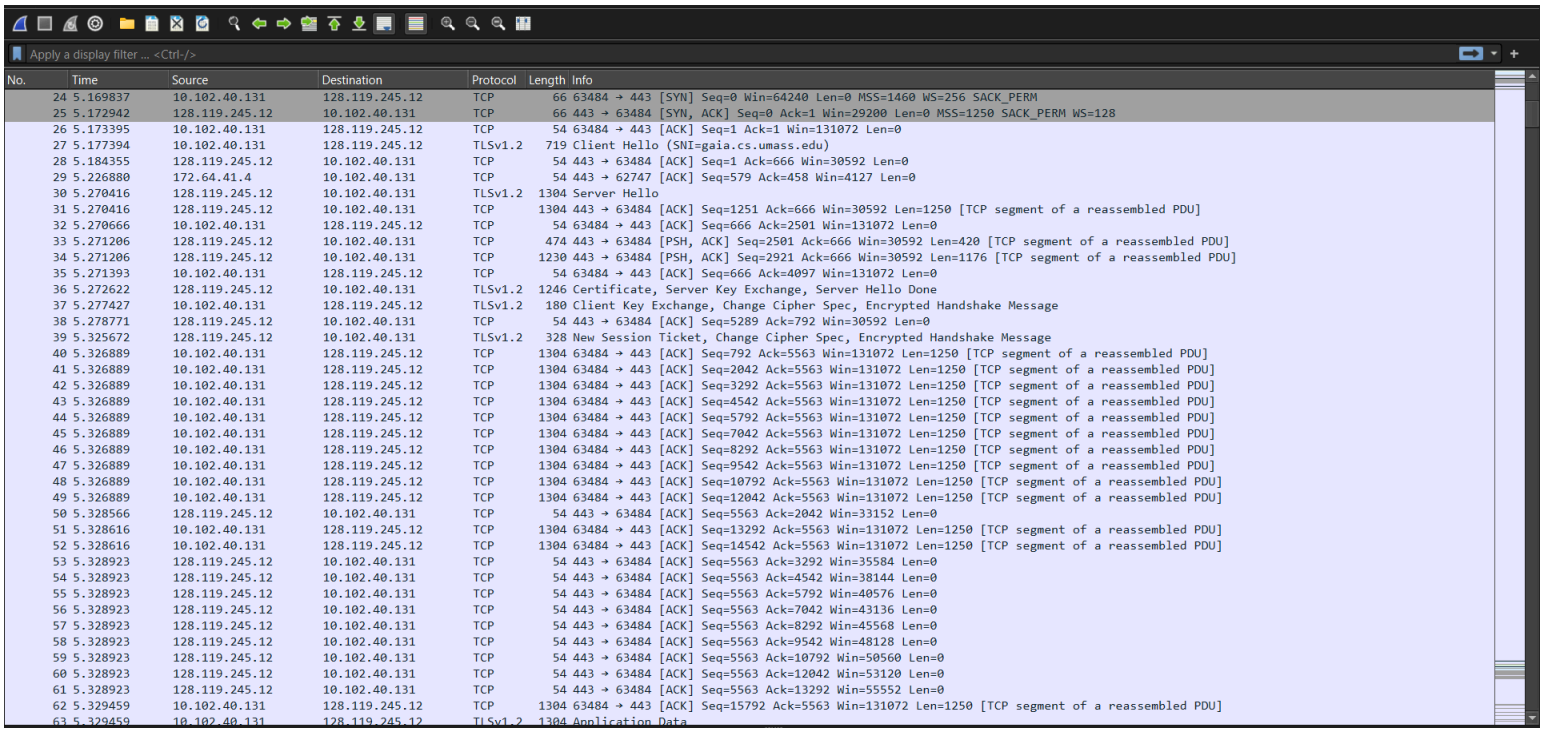
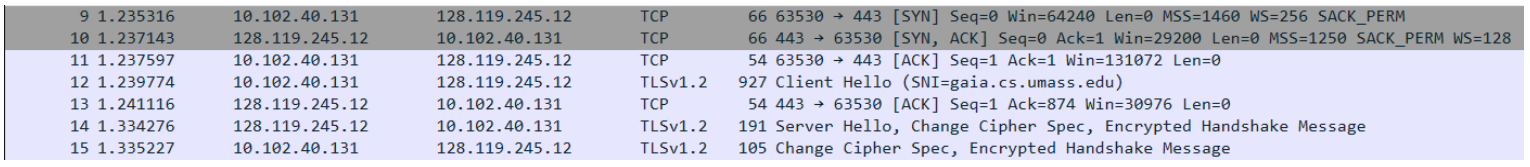


Figure 1: Congratulations message from website


Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
24	5.169837	10.102.40.131	128.119.245.12	TCP	66	63484 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
25	5.172942	128.119.245.12	10.102.40.131	TCP	66	443 → 63484 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1250 SACK_PERM WS=128
26	5.173395	10.102.40.131	128.119.245.12	TCP	54	63484 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
27	5.177394	10.102.40.131	128.119.245.12	TLSv1.2	719	Client Hello (SNI=gaia.cs.umass.edu)
28	5.184355	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=1 Ack=666 Win=30592 Len=0
29	5.226880	172.64.41.4	10.102.40.131	TCP	54	443 → 62747 [ACK] Seq=579 Ack=458 Win=4127 Len=0
30	5.270416	128.119.245.12	10.102.40.131	TLSv1.2	1304	Server Hello
31	5.270416	128.119.245.12	10.102.40.131	TCP	1304	443 → 63484 [ACK] Seq=1251 Ack=666 Win=30592 Len=1250 [TCP segment of a reassembled PDU]
32	5.270666	10.102.40.131	128.119.245.12	TCP	54	63484 → 443 [ACK] Seq=666 Ack=2501 Win=131072 Len=0
33	5.271206	128.119.245.12	10.102.40.131	TCP	474	443 → 63484 [PSH, ACK] Seq=2501 Ack=666 Win=30592 Len=420 [TCP segment of a reassembled PDU]
34	5.271206	128.119.245.12	10.102.40.131	TCP	1230	443 → 63484 [PSH, ACK] Seq=2921 Ack=666 Win=30592 Len=1176 [TCP segment of a reassembled PDU]
35	5.271393	10.102.40.131	128.119.245.12	TCP	54	63484 → 443 [ACK] Seq=666 Ack=4097 Win=131072 Len=0
36	5.272622	128.119.245.12	10.102.40.131	TLSv1.2	1246	Certificate, Server Key Exchange, Server Hello Done
37	5.277427	10.102.40.131	128.119.245.12	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
38	5.278771	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5289 Ack=792 Win=30592 Len=0
39	5.325672	128.119.245.12	10.102.40.131	TLSv1.2	328	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
40	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=792 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
41	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=2042 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
42	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=3292 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
43	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=4542 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
44	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=5792 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
45	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=7042 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
46	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=8292 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
47	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=9542 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
48	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=10792 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
49	5.326889	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=12042 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
50	5.328566	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=2042 Win=33152 Len=0
51	5.328616	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=13292 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
52	5.328616	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=14542 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
53	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=3292 Win=35584 Len=0
54	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=4542 Win=38144 Len=0
55	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=5792 Win=40576 Len=0
56	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=7042 Win=43136 Len=0
57	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=8292 Win=45568 Len=0
58	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=9542 Win=48128 Len=0
59	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=10792 Win=50560 Len=0
60	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=12042 Win=53120 Len=0
61	5.328923	128.119.245.12	10.102.40.131	TCP	54	443 → 63484 [ACK] Seq=5563 Ack=13292 Win=55552 Len=0
62	5.329459	10.102.40.131	128.119.245.12	TCP	1304	63484 → 443 [ACK] Seq=15792 Ack=5563 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
63	5.329459	10.102.40.131	128.119.245.12	TLSv1.2	1304	Application Data

Figure 2: Sample of packet data collected with Wireshark during file upload



9	1.235316	10.102.40.131	128.119.245.12	TCP	66	63530 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
10	1.237143	128.119.245.12	10.102.40.131	TCP	66	443 → 63530 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1250 SACK_PERM WS=128
11	1.237597	10.102.40.131	128.119.245.12	TCP	54	63530 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
12	1.239774	10.102.40.131	128.119.245.12	TLSv1.2	927	Client Hello (SNI=gaia.cs.umass.edu)
13	1.241116	128.119.245.12	10.102.40.131	TCP	54	443 → 63530 [ACK] Seq=1 Ack=874 Win=30976 Len=0
14	1.334276	128.119.245.12	10.102.40.131	TLSv1.2	191	Server Hello, Change Cipher Spec, Encrypted Handshake Message
15	1.335227	10.102.40.131	128.119.245.12	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message

Figure 3: The TCP 3-way handshake exchanged between client and server at gaia.cs.mass.edu

1, 2, 3: What is the IP address and TCP port number used by the client computer and the server gaia.cs.mass.edu?

a. Now, using the provided packet file from the lab, I locate the packet which has the first portion of the file being sent to the server so that it can be further analyzed. I have discovered that the source IP address and port are 192.168.1.102 and 1161, respectively. Similarly, the destination IP address and port are 128.119.245.12 and 80, respectively.

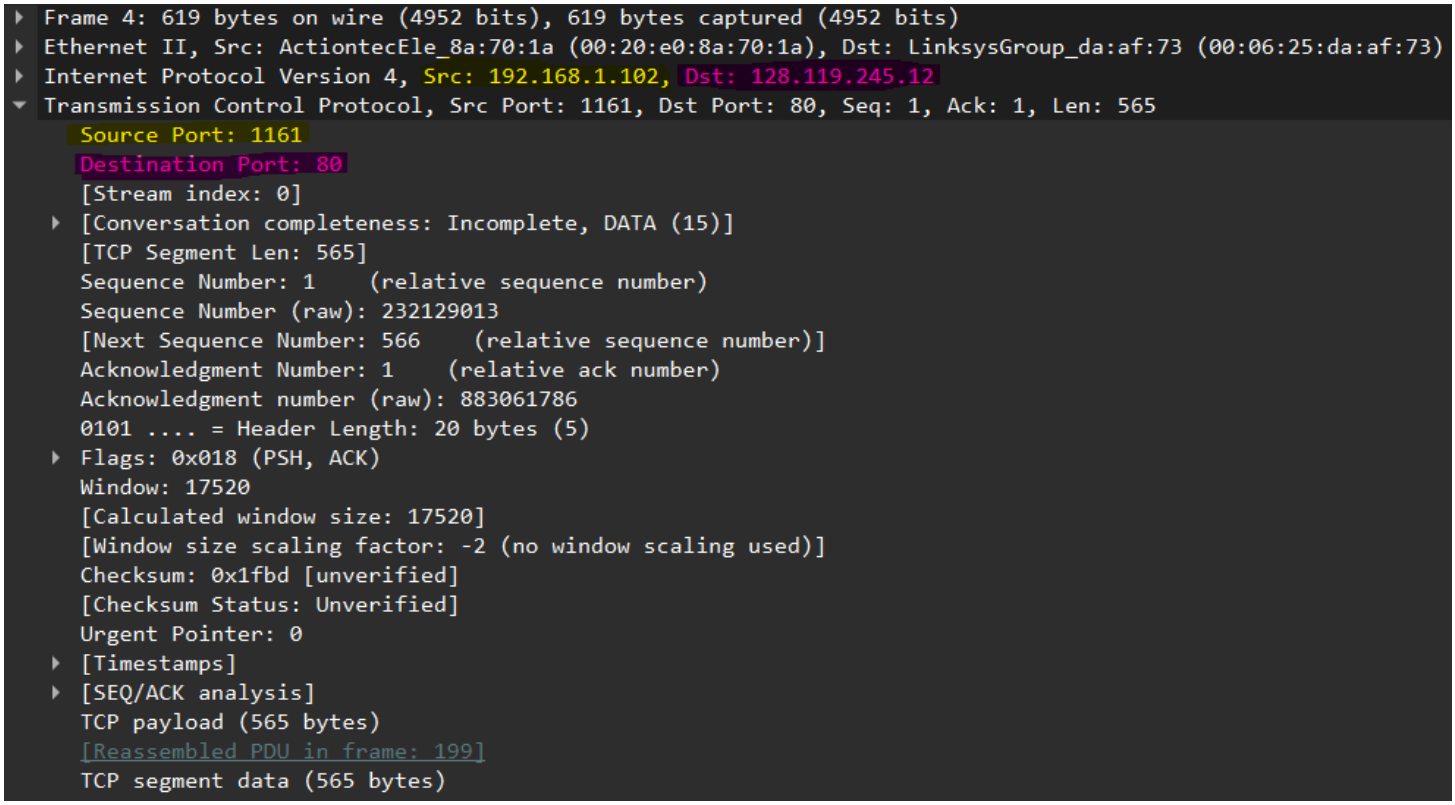


Figure 3: Packet being analyzed from the lab provided packet capture file

a. I have also captured my own packets for this process, and using these I locate the packet which has the first portion of the file being sent to the server so that it can be further analyzed. I have discovered that the source IP address and port of my client are 10.102.40.131 and 63530, respectively. Similarly, the destination IP address and port are 128.119.245.12 and 443, respectively. I do not know why the destination port is not 80, the TCP port.

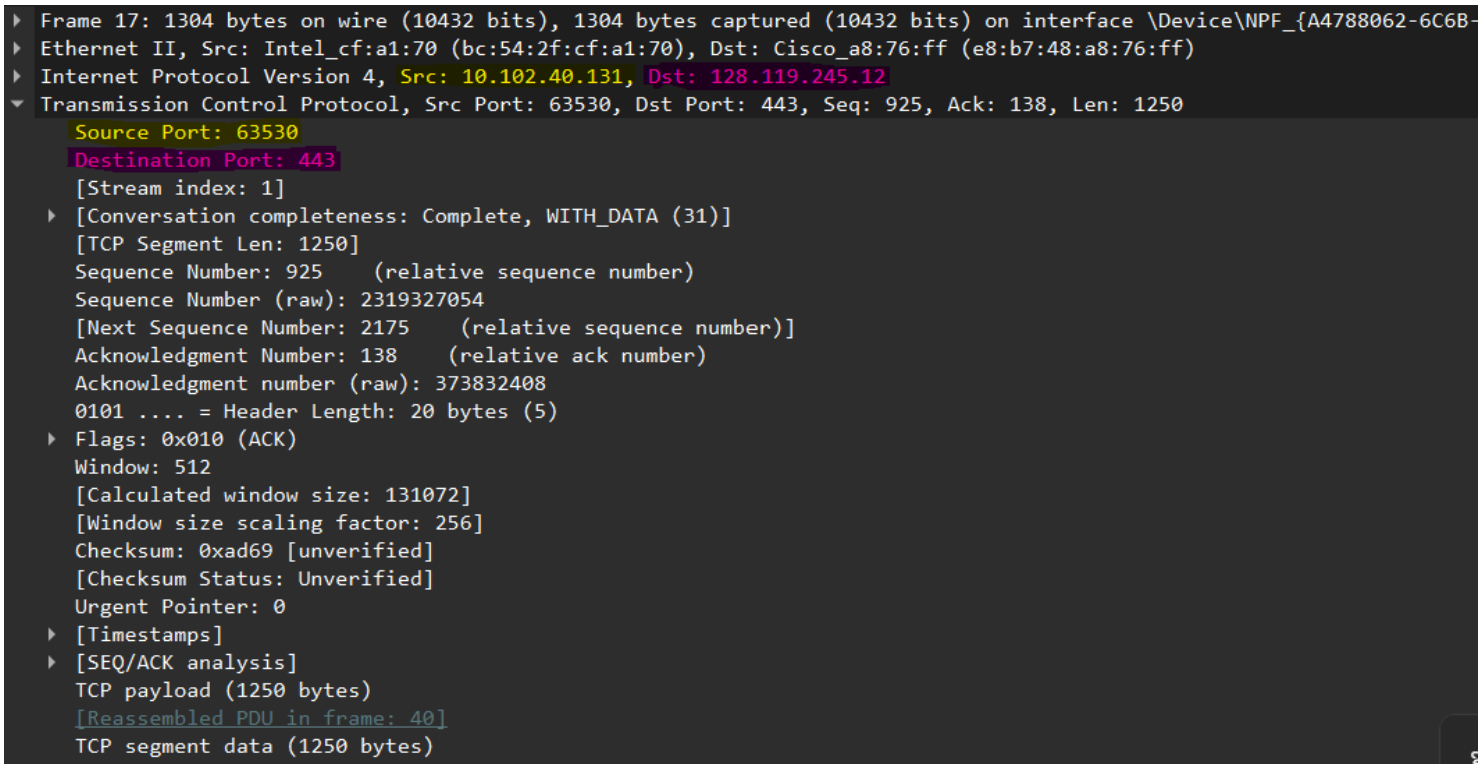


Figure 3: Packet being analyzed from my own packet capture

Setup: Navigate to Analyze -> Enabled Protocols, and uncheck HTTP to view only information about the TCP segments.

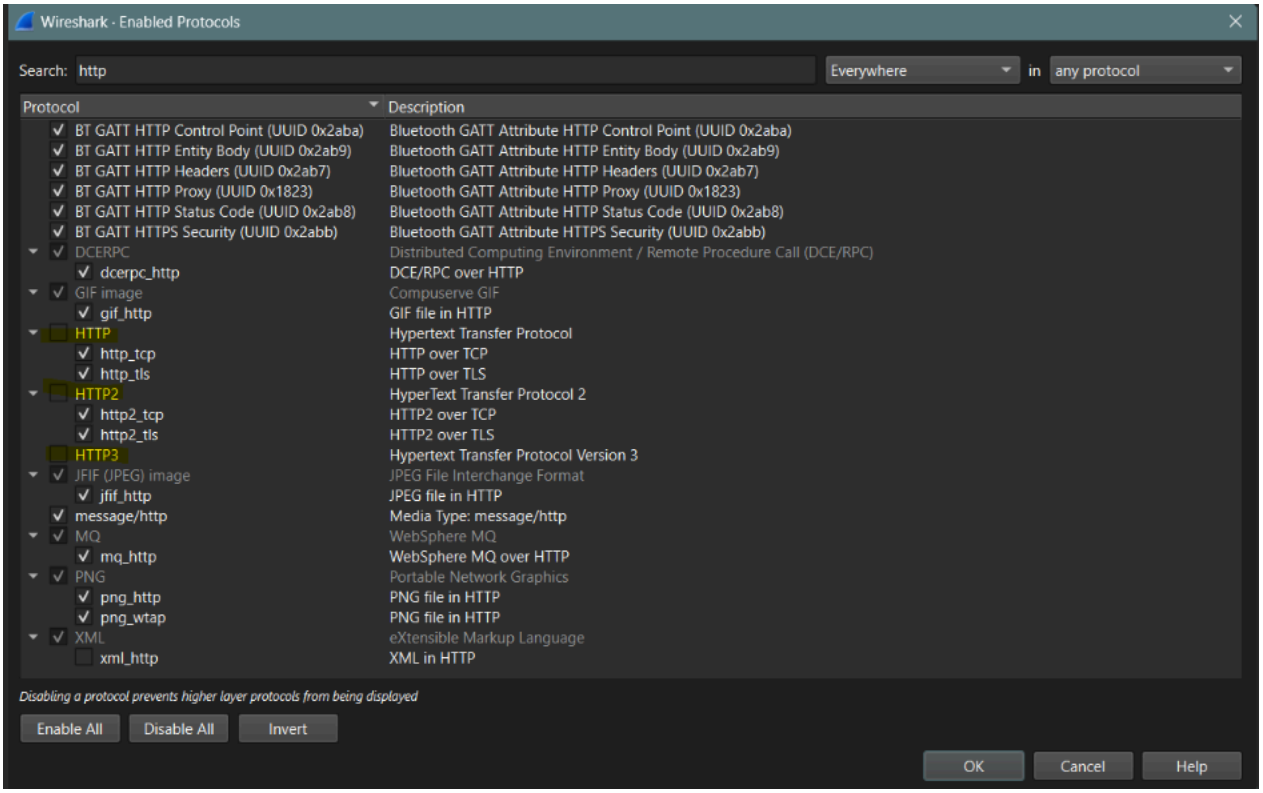


Figure 4: Unchecking HTTP protocols in the Analyze window

4: What is the sequence number of the TCP SYN segment used to initiate the TCP connection being client and server? What part of the segment identifies it as a SYN segment?

a. The sequence number of the TCP SYN segment used to initiate the connection from client to server is 0, and the Syn flag being set is what identifies it as a SYN segment, as highlighted below.

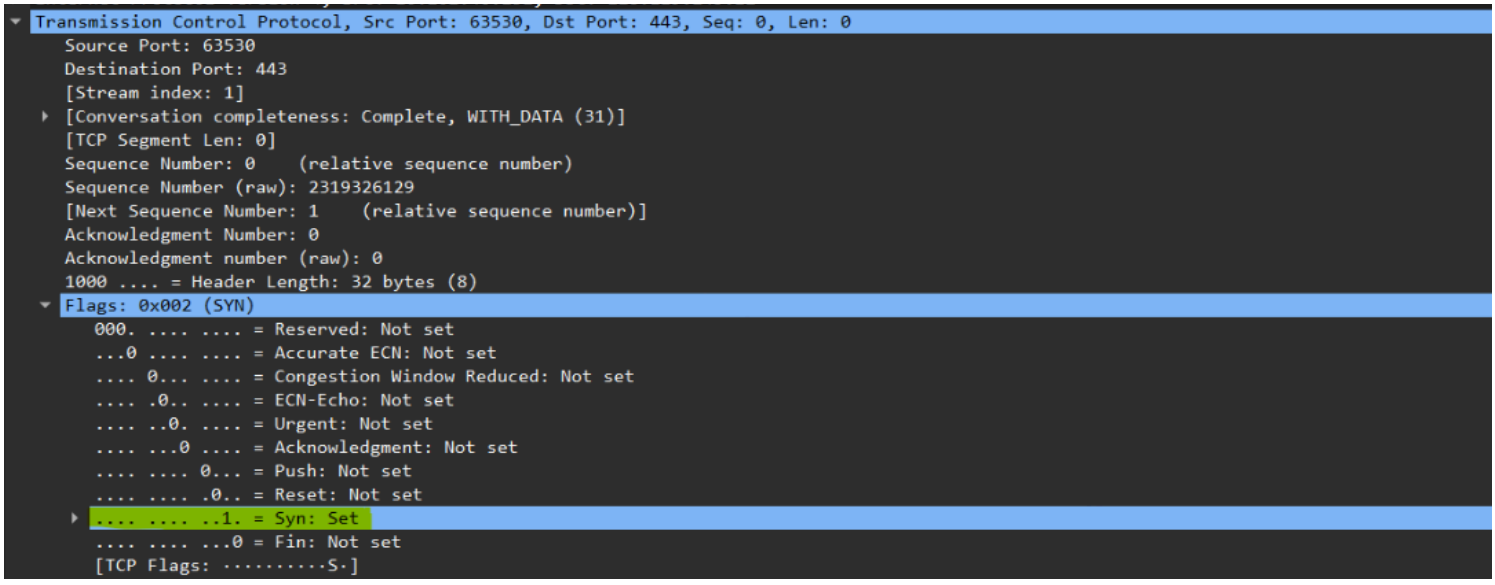


Figure 5: TCP handshake initiation segment from client to server with SYN flag set

5. What is the sequence number of the SYNACK segment returned from the server in reply to the above SYN segment? What is the value of the Acknowledgement field in this segment? How was the value determined? What part of the segment identifies it as a SYNACK segment?

a. The sequence number is 0. The value of the acknowledgement field is 1, and this is because the last segment received by the server was segment 0, so it is ready for segment 1. The segment is identified as a SYNACK segment because the Acknowledgement and Syn flags are both set, as highlighted below.

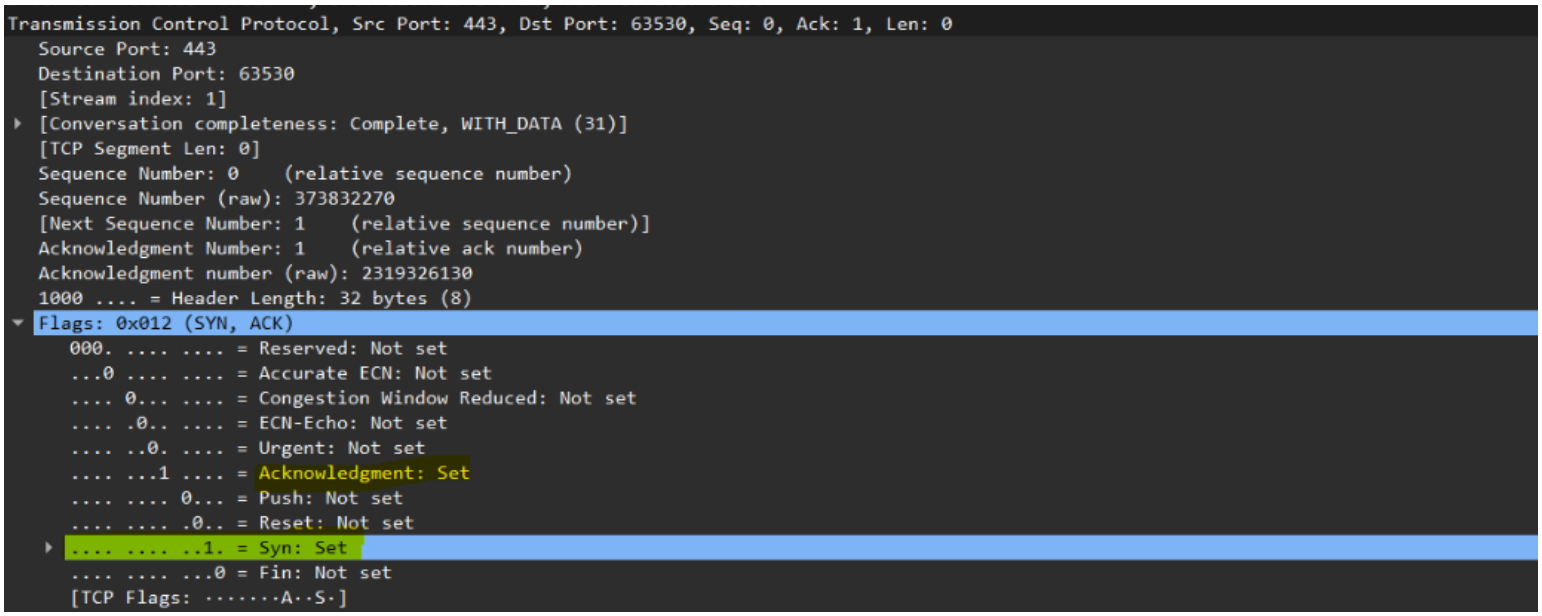


Figure 5: TCP handshake SYNACK segment returned from server with SYN and Acknowledgement flag set

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

a. The sequence number is 164041, as highlighted below.

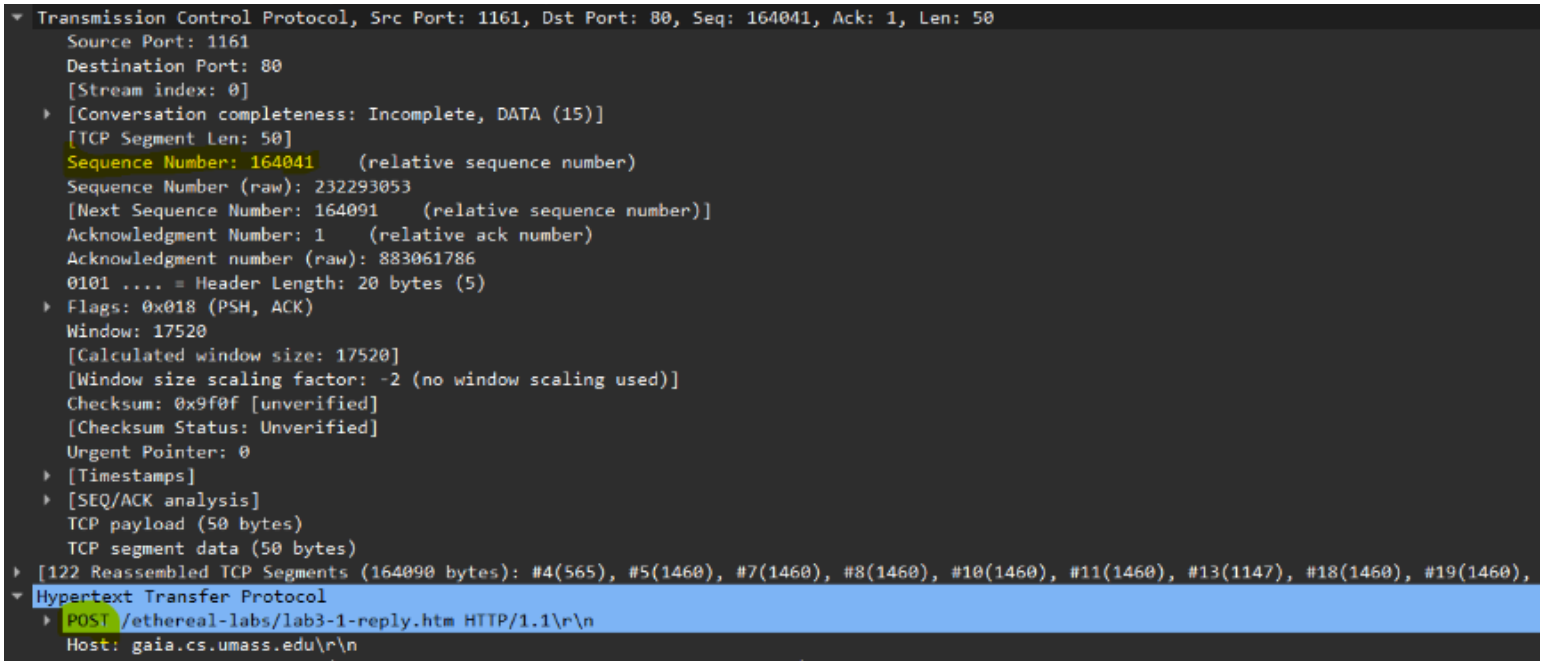


Figure 6: Datagram with HTTP POST command, reference point for Question 7

7. What are the sequence numbers of the first 6 segments (considering POST segment as first segment)? What time was each received, and what time was the ACK for each received? What is the RTT for each of the segments? What is the EstimatedRTT value after the receipt of each ACK? (Assume EstimatedRTT is equal to

measured RTT for first segment, then calculate rest using equation on p. 242). For the purposes of these questions, each segment will have its own answer section (a - f).

a. packet no. = 199, seqno = 164041, time received = 5.297341, RTT = .087369, EstimatedRTT =
b. packet no. = 200, seqno = 1, time received = 5.389471, RTT = .058416, EstimatedRTT =
c. packet no. = 201, seqno = 1, time received = 5.447887, RTT = .058416, EstimatedRTT =
d. packet no. = 202, seqno = 1, time received = 5.455830, RTT = .035345, EstimatedRTT =
e. packet no. = 203, seqno = 1, time received = 5.461175, RTT = .189966, EstimatedRTT =
f. packet no. = 206, seqno = 164091, time received = 5.651141, RTT = N/A, EstimatedRTT =

packet no	Sent time	ACK received time	RTT (seconds)
199	5.297341	5.389471	.087369
200	5.389471	5.447887	.058416
201	5.447887	5.455830	.007943
202	5.455830	5.491175	.035345
203	5.461175	5.651141	.189966
206	5.651141	N/A, last packet in sequence	N/A, last packet in sequence

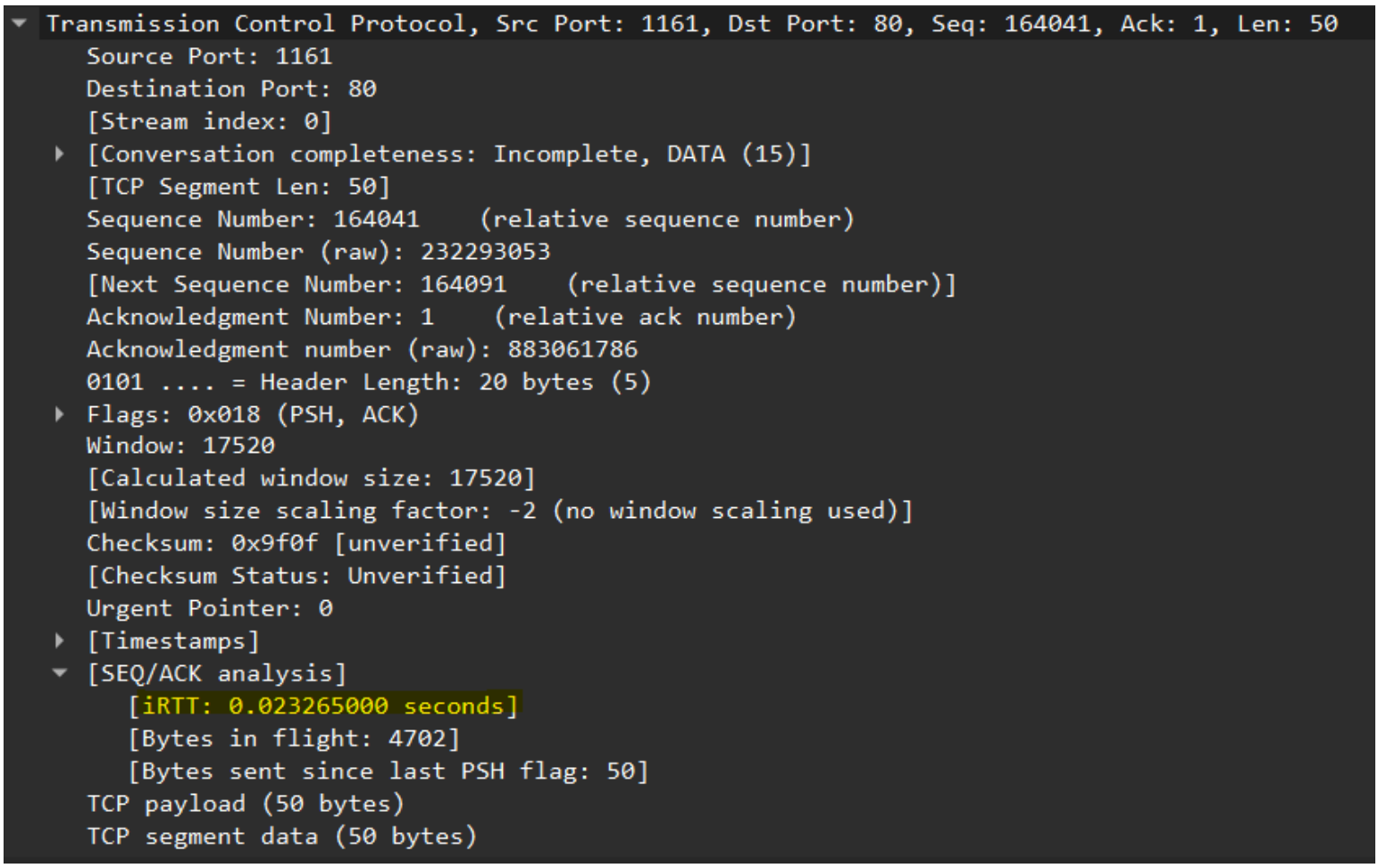
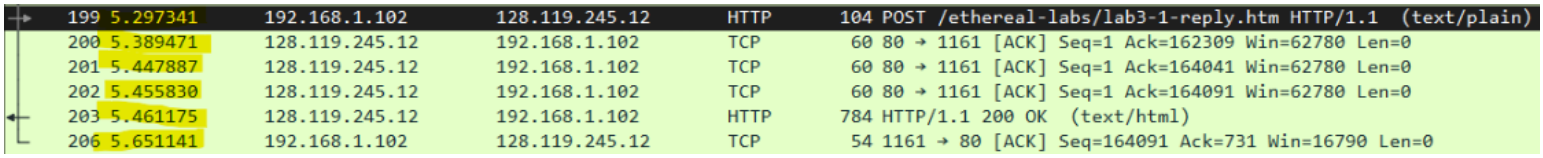


Figure 7: First packet in sequence for Question 7, with iRTT .023265 seconds

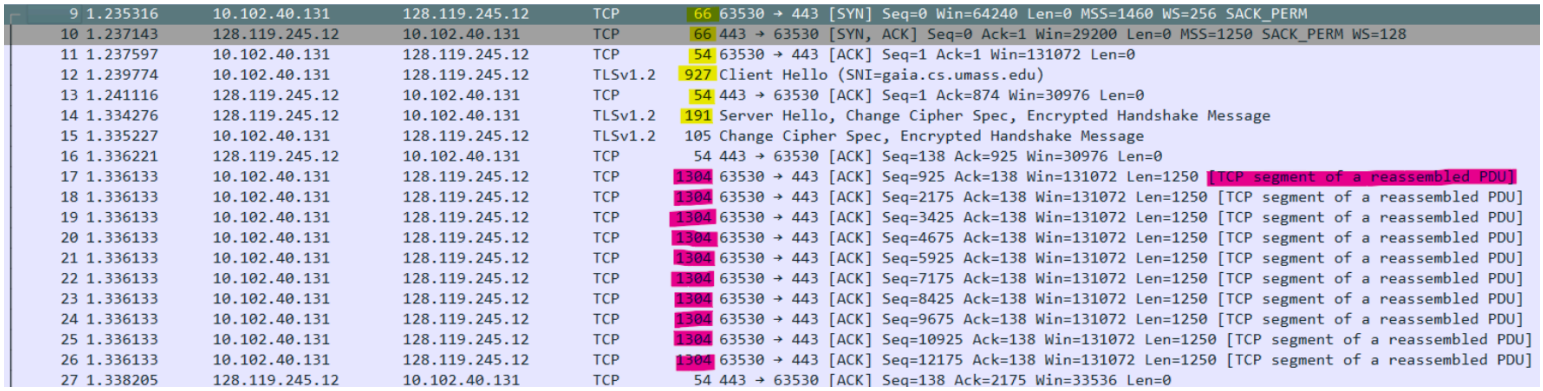


199	5.297341	192.168.1.102	128.119.245.12	HTTP	104 POST /ethereal-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
200	5.389471	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=162309 Win=62780 Len=0
201	5.447887	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=164041 Win=62780 Len=0
202	5.455830	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=164091 Win=62780 Len=0
203	5.461175	128.119.245.12	192.168.1.102	HTTP	784 HTTP/1.1 200 OK (text/html)
206	5.651141	192.168.1.102	128.119.245.12	TCP	54 1161 → 80 [ACK] Seq=164091 Ack=731 Win=16790 Len=0

Figure 8: Full sequence of segments for Question 7, with time highlighted

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

a. Including the handshake, 66, 66, 54, 927, 54, 191. However for the TCP segments which are part of the fragmented payload they are all of length 1304.

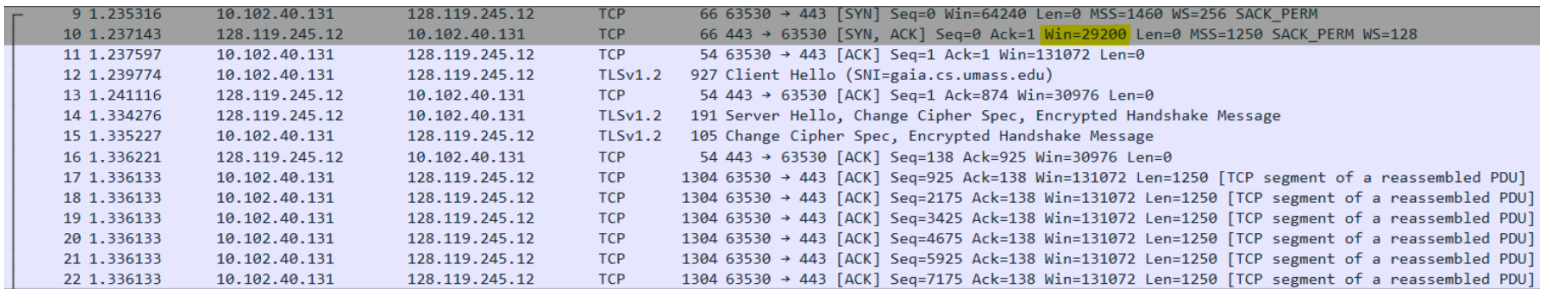


9	1.235316	10.102.40.131	128.119.245.12	TCP	66 63530 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
10	1.237143	128.119.245.12	10.102.40.131	TCP	66 443 → 63530 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1250 SACK_PERM WS=128
11	1.237597	10.102.40.131	128.119.245.12	TCP	54 63530 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
12	1.239774	10.102.40.131	128.119.245.12	TLSv1.2	927 Client Hello (SNI=gaia.cs.umass.edu)
13	1.241116	128.119.245.12	10.102.40.131	TCP	54 443 → 63530 [ACK] Seq=1 Ack=874 Win=30976 Len=0
14	1.334276	128.119.245.12	10.102.40.131	TLSv1.2	191 Server Hello, Change Cipher Spec, Encrypted Handshake Message
15	1.335227	10.102.40.131	128.119.245.12	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
16	1.336221	128.119.245.12	10.102.40.131	TCP	54 443 → 63530 [ACK] Seq=138 Ack=925 Win=30976 Len=0
17	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=925 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
18	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=2175 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
19	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=3425 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
20	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=4675 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
21	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=5925 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
22	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=7175 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
23	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=8425 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
24	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=9675 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
25	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=10925 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
26	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=12175 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
27	1.338205	128.119.245.12	10.102.40.131	TCP	54 443 → 63530 [ACK] Seq=138 Ack=2175 Win=33536 Len=0

Figure 9: Size of TCP segments highlighted

9. What is the minimum amount of buffer space advertised at the receiver for the entire trace? Does the lack of buffer space ever throttle the sender?

a. 29200, as seen in second packet of the exchange



9	1.235316	10.102.40.131	128.119.245.12	TCP	66 63530 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
10	1.237143	128.119.245.12	10.102.40.131	TCP	66 443 → 63530 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1250 SACK_PERM WS=128
11	1.237597	10.102.40.131	128.119.245.12	TCP	54 63530 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
12	1.239774	10.102.40.131	128.119.245.12	TLSv1.2	927 Client Hello (SNI=gaia.cs.umass.edu)
13	1.241116	128.119.245.12	10.102.40.131	TCP	54 443 → 63530 [ACK] Seq=1 Ack=874 Win=30976 Len=0
14	1.334276	128.119.245.12	10.102.40.131	TLSv1.2	191 Server Hello, Change Cipher Spec, Encrypted Handshake Message
15	1.335227	10.102.40.131	128.119.245.12	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
16	1.336221	128.119.245.12	10.102.40.131	TCP	54 443 → 63530 [ACK] Seq=138 Ack=925 Win=30976 Len=0
17	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=925 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
18	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=2175 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
19	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=3425 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
20	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=4675 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
21	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=5925 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
22	1.336133	10.102.40.131	128.119.245.12	TCP	1304 63530 → 443 [ACK] Seq=7175 Ack=138 Win=131072 Len=1250 [TCP segment of a reassembled PDU]

Figure 10: Smallest buffer size highlighted

10. Are there any retransmitted segments? How did you find the answer to this question?

a. No packets were dropped, this was verified by checking that sequence numbers were ordered and sequential, and via inspection of the Time-Sequence-Graph (Stevens).

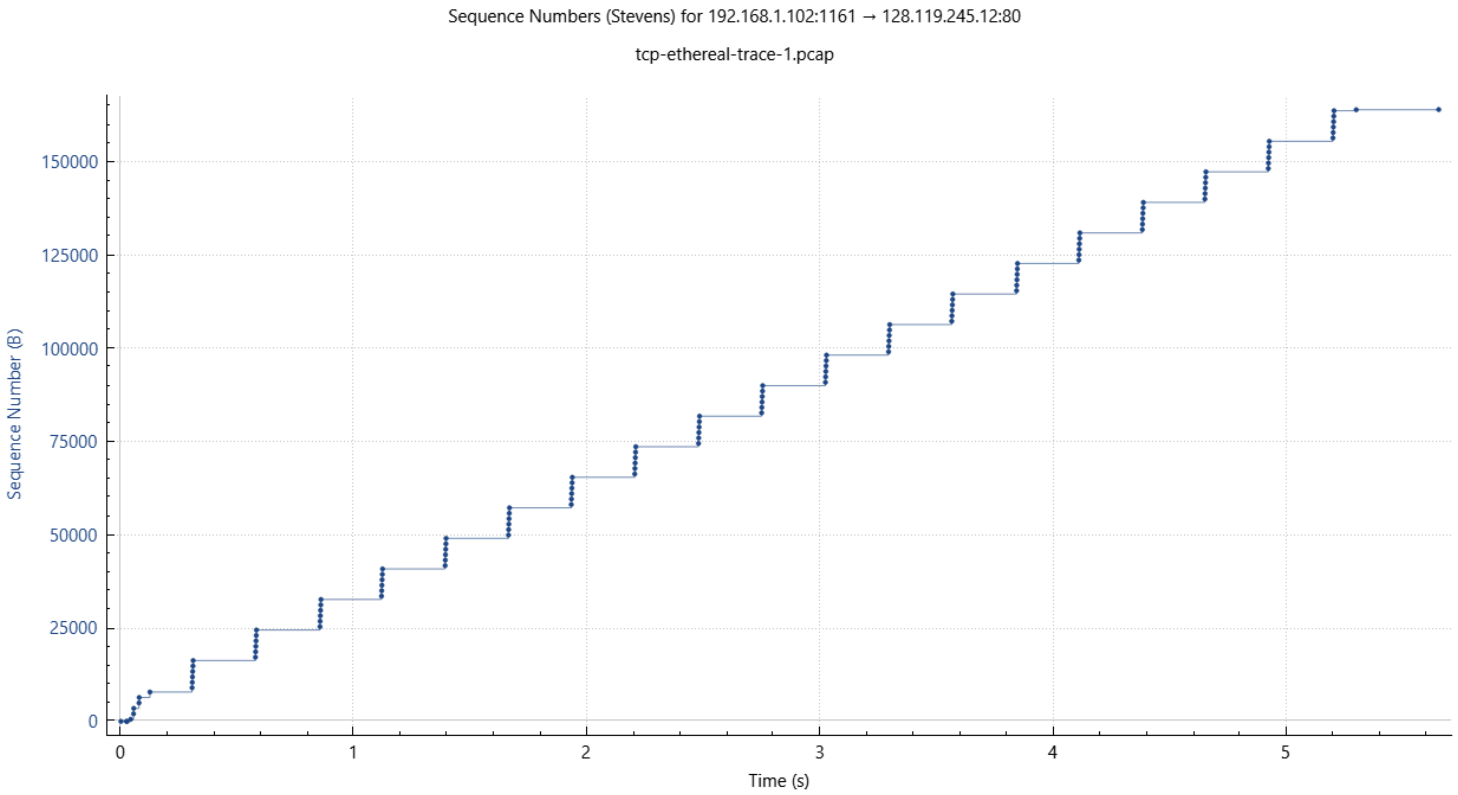


Figure 11: Time-Sequence-Graph for the book provided trace file

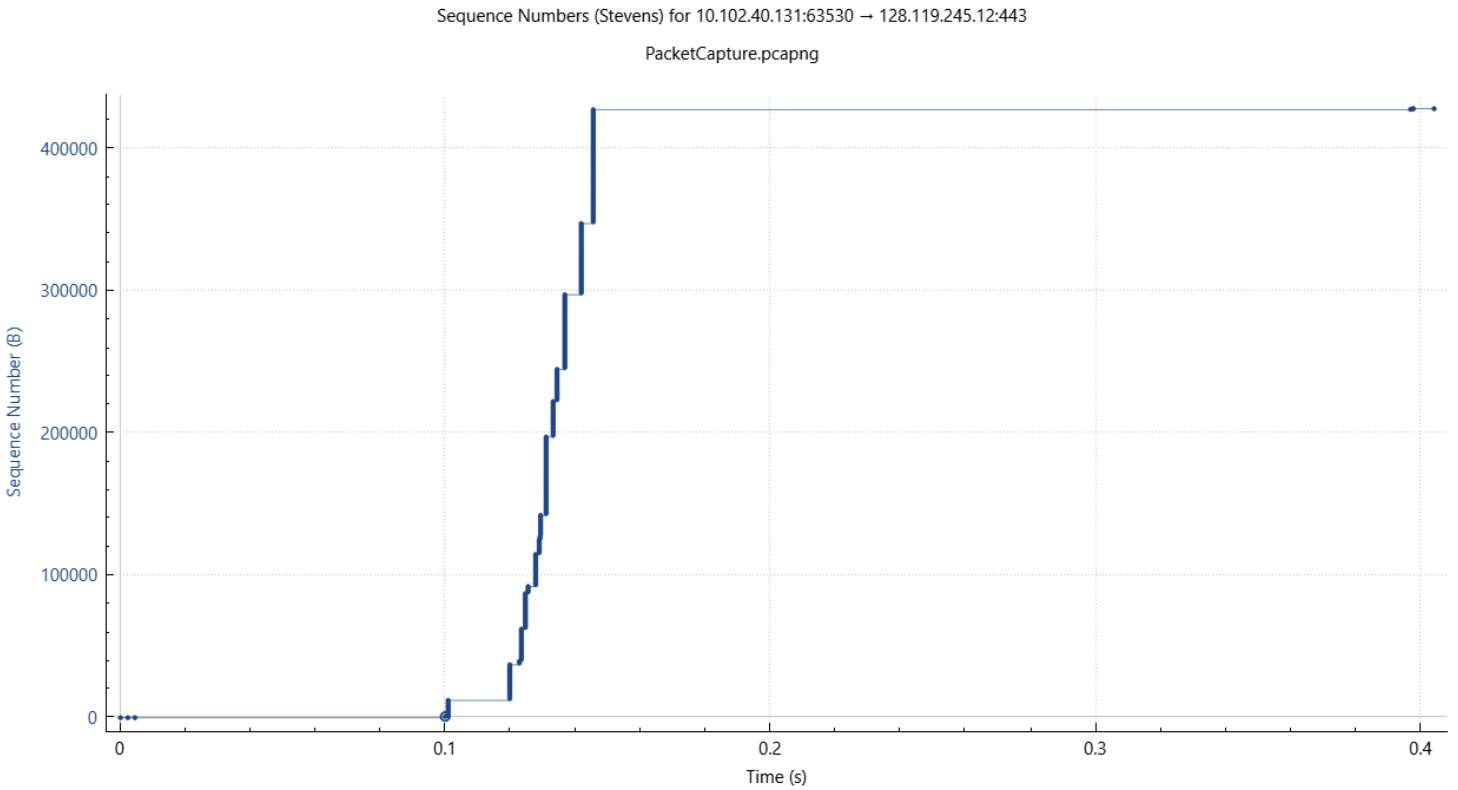


Figure 12: Time-Sequence-Graph for my packet capture file

11. How much data does the receiver typically acknowledge in an ACK? Can you identify any cases where the receiver is ACKing every other received segment?

a. It seems that the receiver is ACKing about 1250 bytes of data with each ACK, the first few ACKs are used in the handshake so they are outliers for this question. It is hard to identify where the receiver is ACKing every other segment, but I do not see evidence of that in the ACKs collected for this table.

Packet No.	ACK seqno	acknowledged data
10	0	1
13	1	874
14	138	925
16	138	2175
27	138	3425
28	138	4675
29	138	5925
30	138	7175
31	138	8425
32	138	9675
33	138	10925
34	138	12175
35	138	13425
36	138	14675
57	138	15925

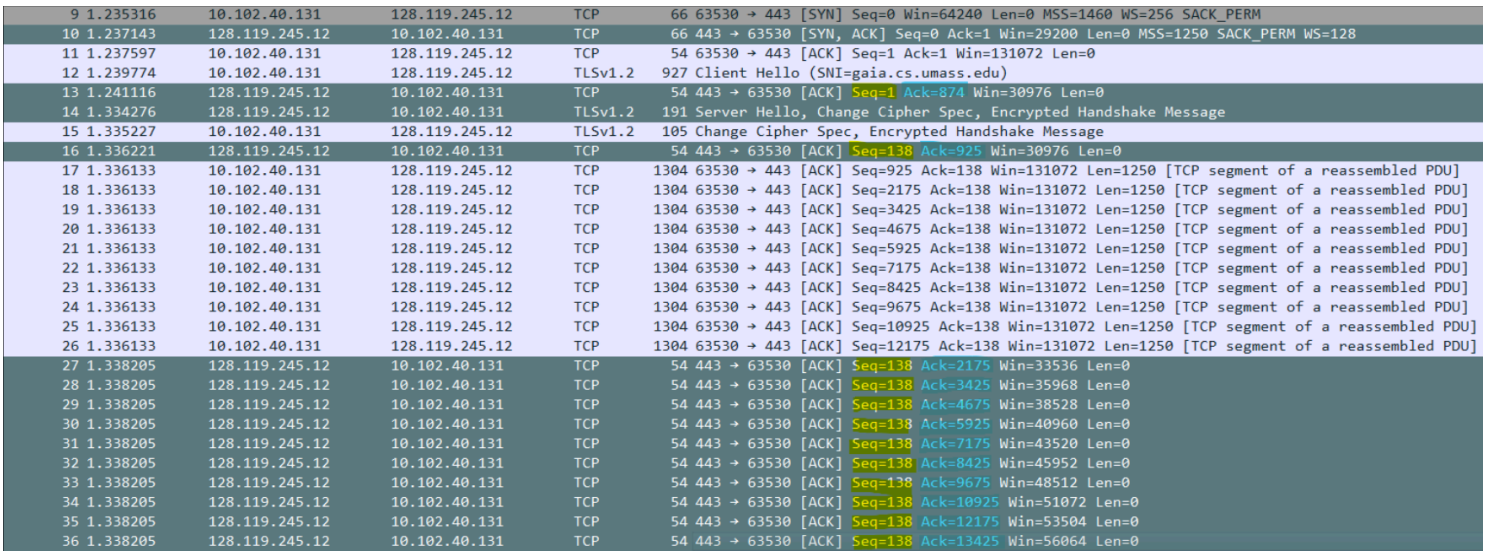


Figure 13: Some of the ACKs recorded in the above table

12. What is the throughput for the TCP connection? How was this calculated?

a. Data transmitted = last ACK amount - first ACK amount = 427889 - 1 = 127888 bytes
b. Transmission time = time elapsed (last ACK) - time elapsed (first ACK) = 1.639058 - 1.237143 = .401915 sec
c. Throughput = data transmitted / transmission time = 127888 / .401915 = 318,196 bytes/second

13. On Time-Sequence-Graph, highlight congestion control/slow start section for provided packet capture

a. See annotated photo.

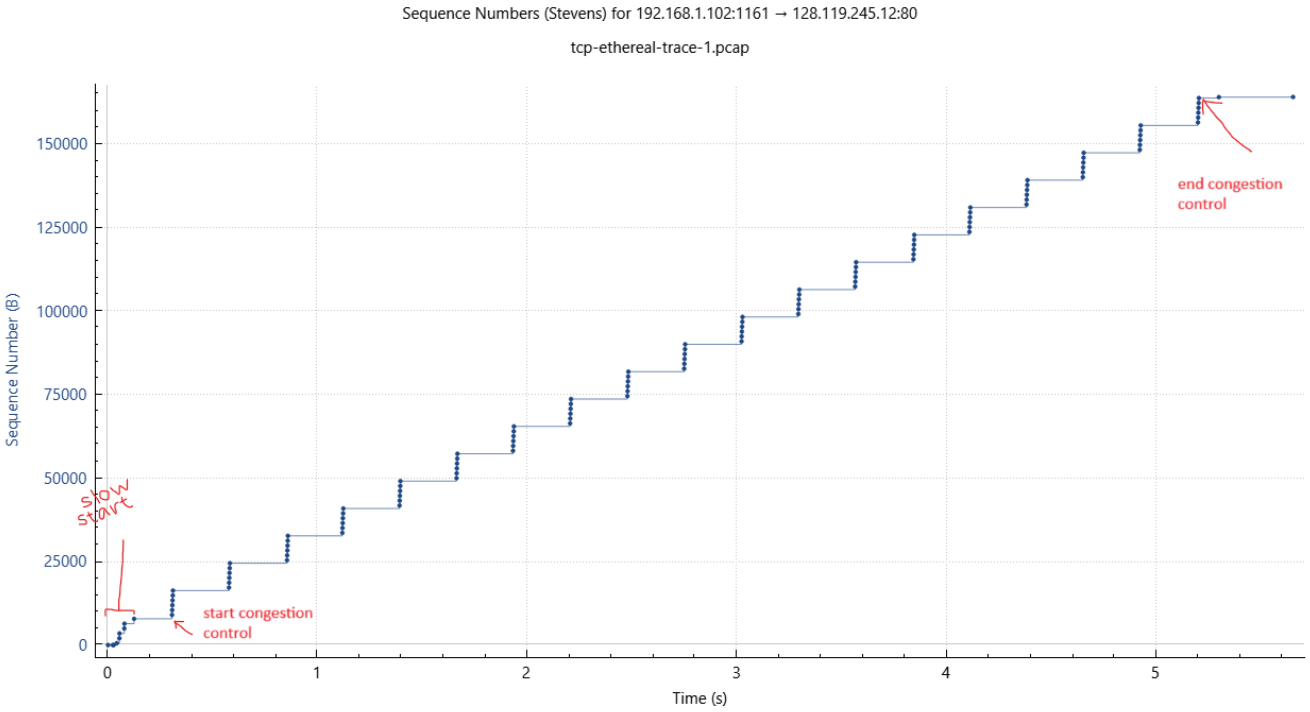


Figure 14: Annotated graph for book provided packet capture

13. On Time-Sequence-Graph, highlight congestion control and slow start section on your own capture

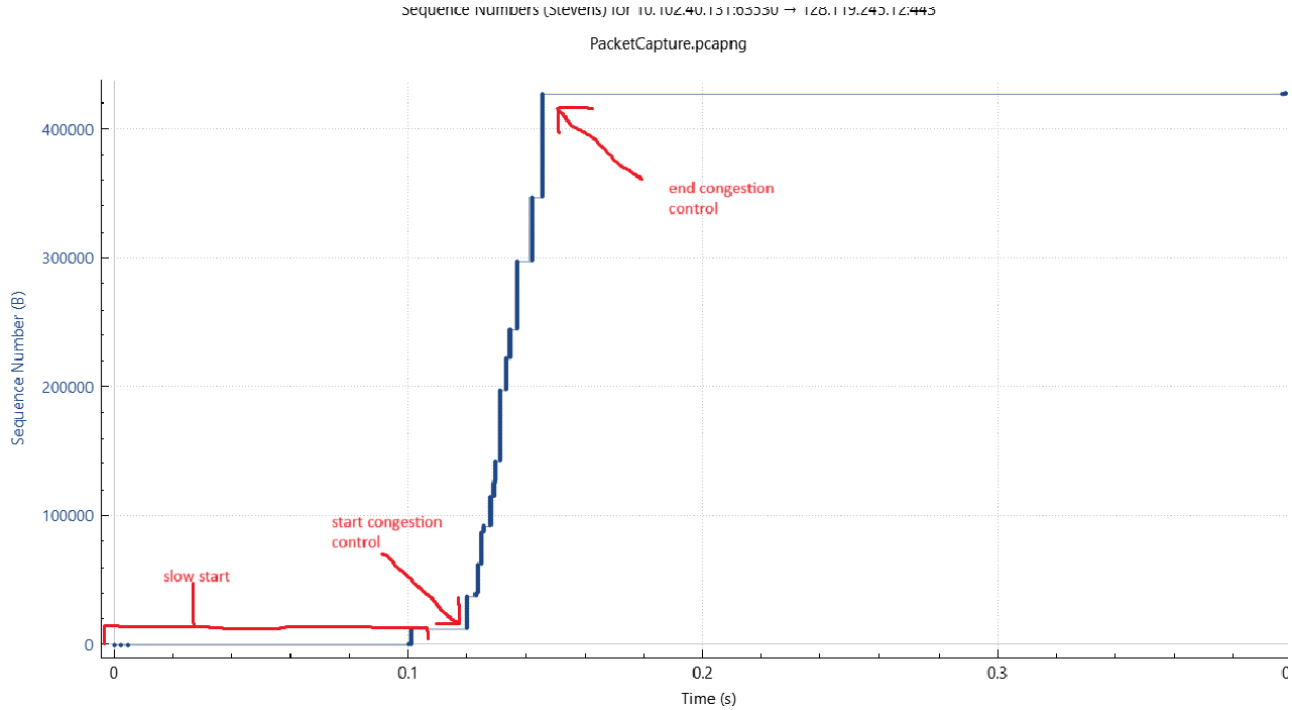


Figure 14: Annotated graph for book provided packet capture

PART II

Setup: Collect some UDP packets.

a. Since this lab requires the collection of UDP packets, I opened the UDP capture provided by the lab and will be using these.

```
  1 0.000000 192.168.1.101    68.87.71.226     DNS      85 Standard query 0x0001 PTR 226.71.87.68.in-addr.arpa
  2 0.012481 68.87.71.226     192.168.1.101    DNS     137 Standard query response 0x0001 PTR 226.71.87.68.in-addr.arpa PTR cns.chelmsfdrdc2.ma.boston.comcast.net
  3 0.014232 192.168.1.101    68.87.71.226     DNS      91 Standard query 0x0002 A www.mit.edu.hsd1.ma.comcast.net
  4 0.042641 68.87.71.226     192.168.1.101    DNS     171 Standard query response 0x0002 No such name A www.mit.edu.hsd1.ma.comcast.net SOA dns1.inflow.pa.bo.comcast.net
  5 0.044178 192.168.1.101    68.87.71.226     DNS      86 Standard query 0x0003 A www.mit.edu.ma.comcast.net
  6 0.058934 68.87.71.226     192.168.1.101    DNS      86 Standard query response 0x0003 Server failure A www.mit.edu.ma.comcast.net
  7 0.060268 192.168.1.101    68.87.71.226     DNS      71 Standard query 0x0004 A www.mit.edu
  8 0.074984 68.87.71.226     192.168.1.101    DNS      87 Standard query response 0x0004 A www.mit.edu A 18.7.22.83
 28 39.7819… 192.168.1.101    68.87.71.226     DNS      85 Standard query 0x0001 PTR 226.71.87.68.in-addr.arpa
 29 39.7948… 68.87.71.226     192.168.1.101    DNS     137 Standard query response 0x0001 PTR 226.71.87.68.in-addr.arpa PTR cns.chelmsfdrdc2.ma.boston.comcast.net
 30 39.7967… 192.168.1.101    68.87.71.226     DNS      87 Standard query 0x0002 NS mit.edu.hsd1.ma.comcast.net
 31 39.8237… 68.87.71.226     192.168.1.101    DNS     167 Standard query response 0x0002 No such name NS mit.edu.hsd1.ma.comcast.net SOA dns1.inflow.pa.bo.comcast.net
 32 39.8251… 192.168.1.101    68.87.71.226     DNS      82 Standard query 0x0003 NS mit.edu.ma.comcast.net
 33 39.8383… 68.87.71.226     192.168.1.101    DNS      82 Standard query response 0x0003 Server failure NS mit.edu.ma.comcast.net
 34 39.8397… 192.168.1.101    68.87.71.226     DNS      67 Standard query 0x0004 NS mit.edu
 35 39.8522… 68.87.71.226     192.168.1.101    DNS     176 Standard query response 0x0004 NS mit.edu NS W20NS.mit.edu NS BITSY.mit.edu NS STRAWB.mit.edu A 18.72.0.3 A 18.71.
```

Figure 1: The packets I will be focusing on for this lab

1. Select one UDP packet from the trace, determine how many fields are in the UDP header. Name the fields.

a. Source Port, Destination Port, Length, Checksum, UDP Payload, 5 fields

```
▸ Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits)
▸ Ethernet II, Src: Dell_4f:36:23 (00:08:74:4f:36:23), Dst: CiscoLinksys_f4:eb:a8 (00:16:b6:f4:eb:a8)
▸ Internet Protocol Version 4, Src: 192.168.1.101, Dst: 68.87.71.226
▾ User Datagram Protocol, Src Port: 4372, Dst Port: 53
    Source Port: 4372
    Destination Port: 53
    Length: 51
    Checksum: 0x77d4 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  ▸ [Timestamps]
    UDP payload (43 bytes)
▸ Domain Name System (query)
```

Figure 2: UDP Header fields

2. Determine length in bytes of each of the UDP header fields.

a. Source Port: 2 bytes
b. Destination Port: 2 bytes
c. Length: 2 bytes
d. Checksum: 2 bytes
e. Payload: variable (43 in this case)

```
▸ Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits)
▸ Ethernet II, Src: Dell_4f:36:23 (00:08:74:4f:36:23), Dst: CiscoLinksys_f4:eb:a8 (00:16:b6:f4:eb:a8)
▸ Internet Protocol Version 4, Src: 192.168.1.101, Dst: 68.87.71.226
▾ User Datagram Protocol, Src Port: 4372, Dst Port: 53
    Source Port: 4372
    Destination Port: 53
    Length: 51
    Checksum: 0x77d4 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  ▸ [Timestamps]
    UDP payload (43 bytes)
▸ Domain Name System (query)

0000  00 16 b6 f4 eb a8 00 08  74 4f 36 23 08 00 45 00   ........ tO6#..E.
0010  00 47 3c f9 00 00 80 11  af 66 c0 a8 01 65 44 57   .G<..... .f...eDW
0020  47 e2 11 14 00 35 00 33  77 d4 00 01 01 00 00 01   G....5.3 w.......
0030  00 00 00 00 00 00 03 32  32 36 02 37 31 02 38 37   .......2 26.71.87
0040  02 36 38 07 69 6e 2d 61  64 64 72 04 61 72 70 61   .68.in-a ddr.arpa
0050  00 00 0c 00 01                                    .....
```

Figure 3: The size of each header field highlighted

3. The “Length” field value refers to what? Verify with a captured packet.

a. Length in bytes of UDP segment including header

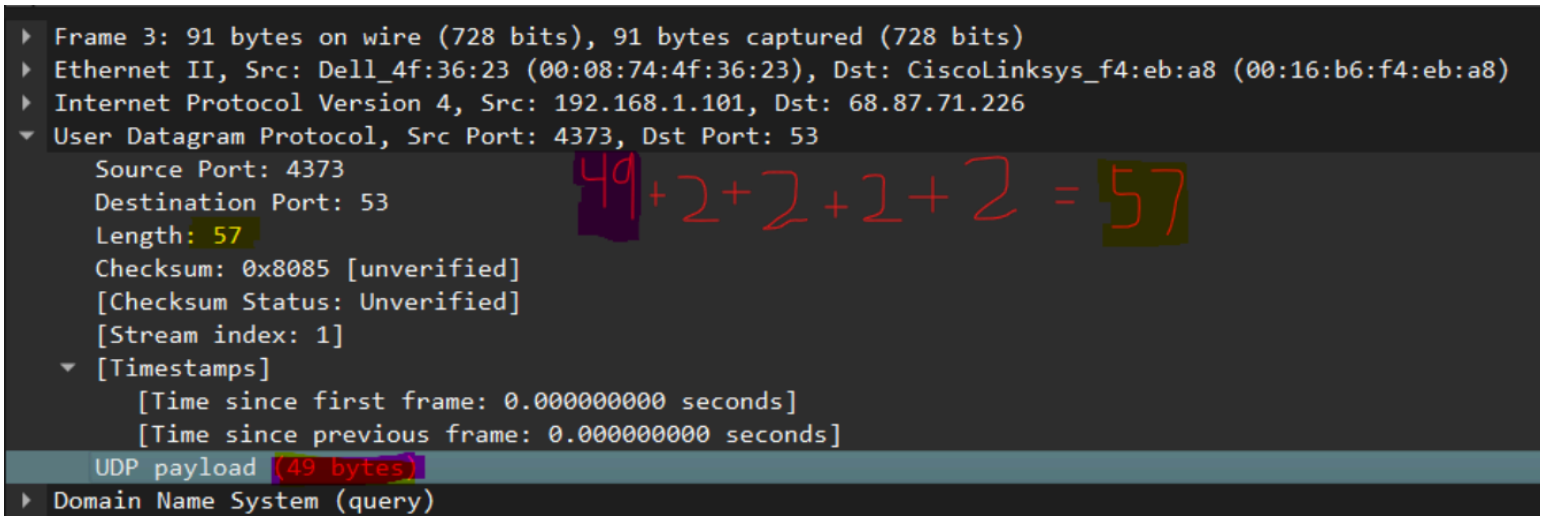


Figure 4: The Length field equals all header fields + payload

4. What is the maximum number of bytes that can be included in a UDP payload?

a. 2^16 - 1 - 8 = 65527 bytes

5. What is the largest possible source port number? (Hint: See previous hint)

a. 2^16 - 1 = 65535

6. What is the protocol number for UDP? Provide answers in decimal and hexadecimal. (Hint: Look into the protocol field of the IP datagram containing this UDP segment)

a. 17 (decimal), 11 (hex)

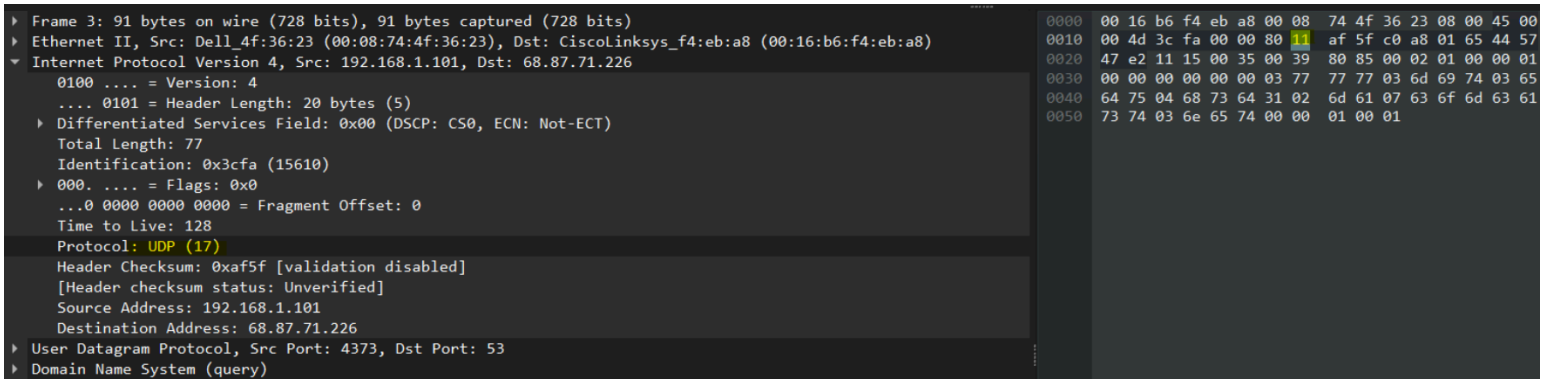
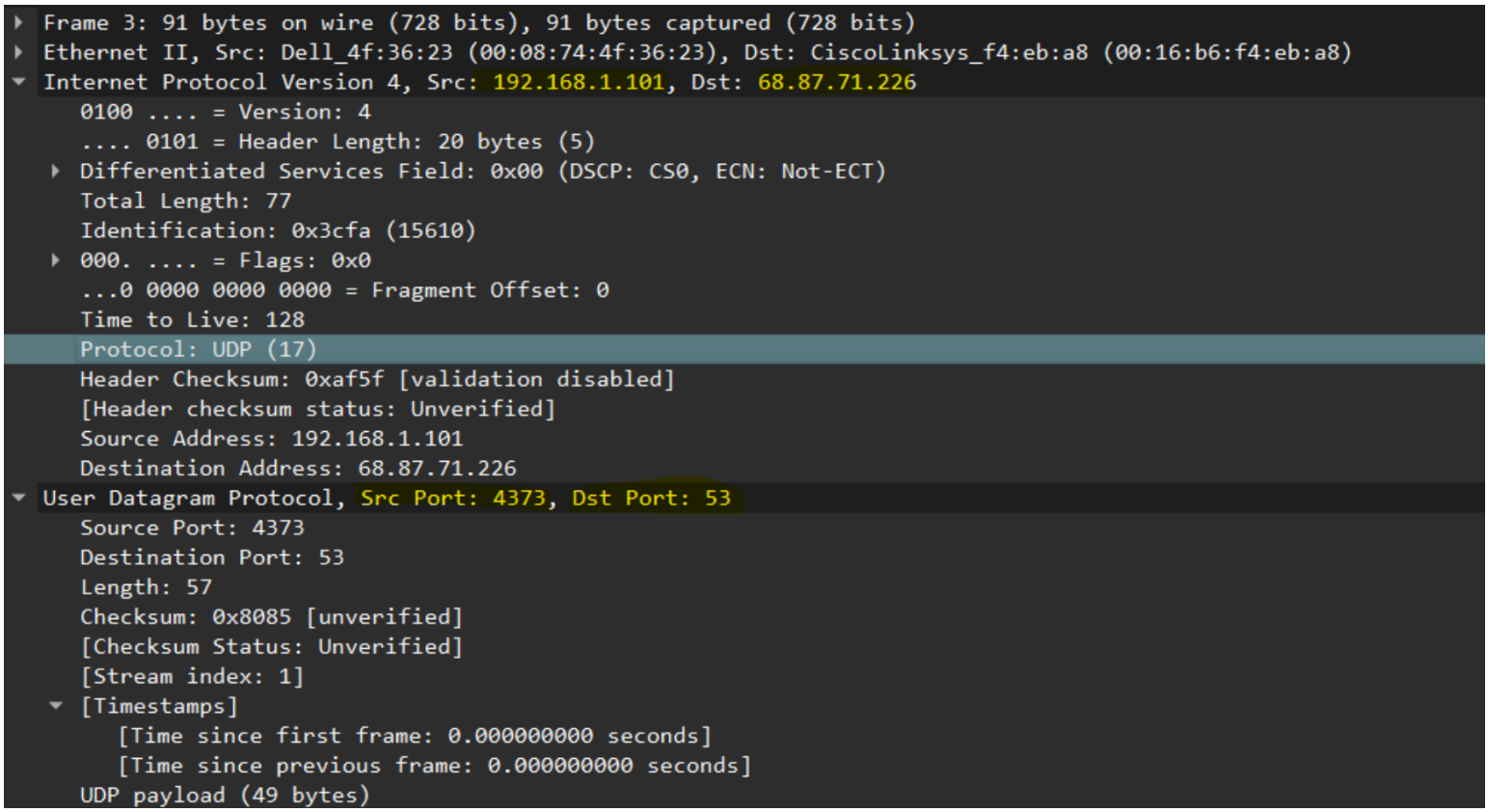
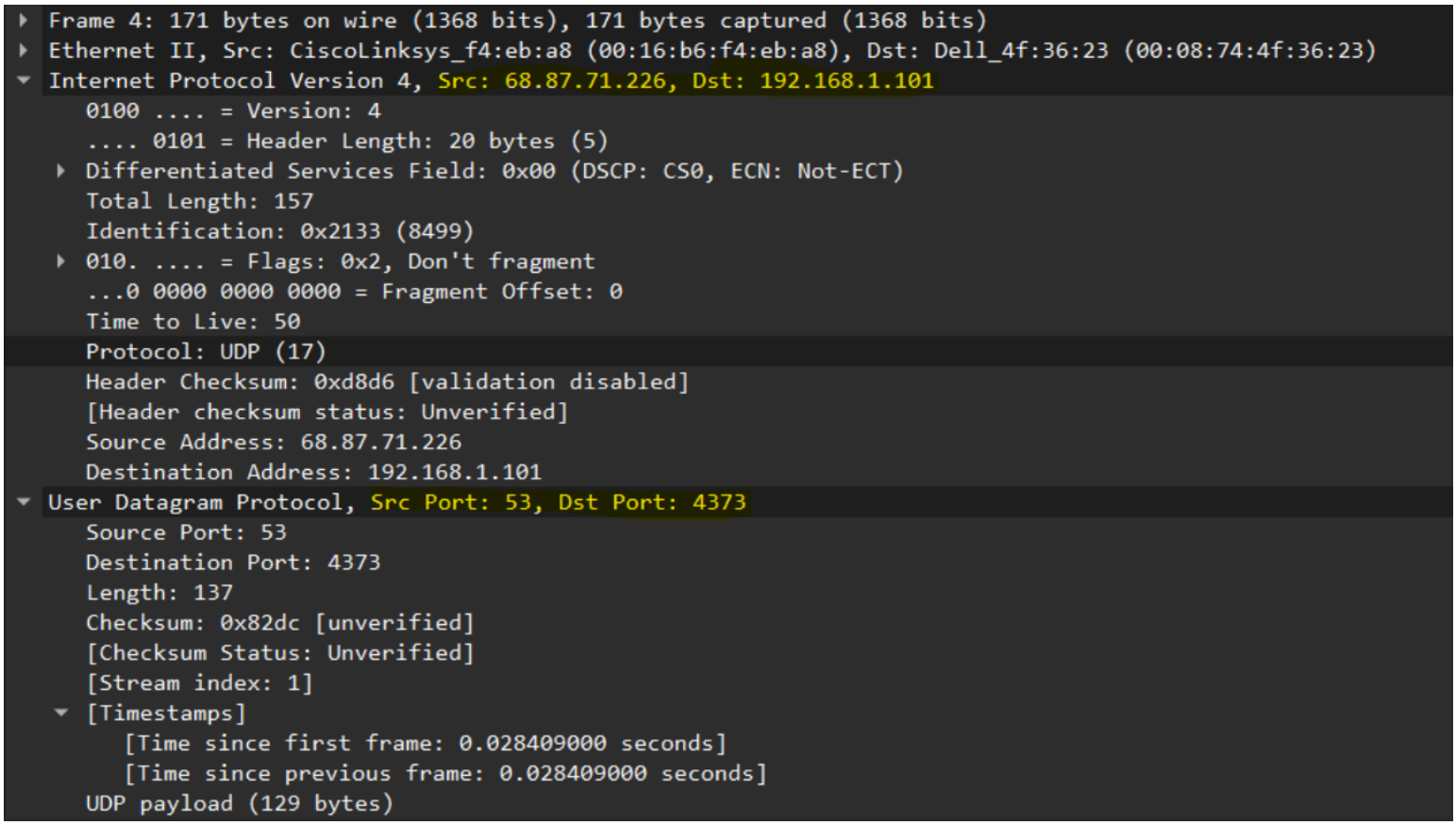


Figure 5: Protocol number identified

7. Examine a pair of UDP packets where the first is one your host sends and the second is the reply to the first. Describe the relationship between the port numbers in these two packets.

a. The relationship is that the host port is the same as the destination port on return packet, and vice versa.





Figures 5 & 6: Sent and returned segments