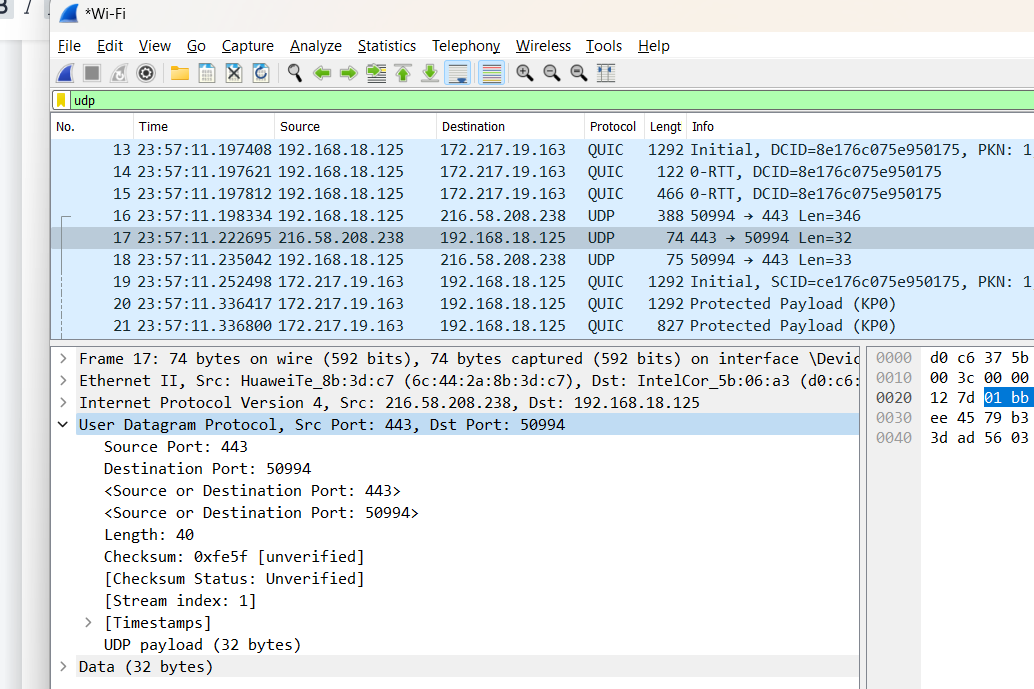
**Assignment # 2**

**K20-1052 BSE-6B**

**HASSAN ALI**

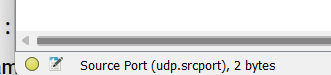
**UDP**

1-



UDP contains 4 header fields (Source Port, Destination Port, Length and Checksum)

2-

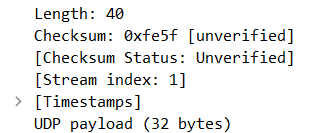


Each of 2 bytes hence total is 4 \* 2 = 8 bytes

3-

The length field indicates how many bytes are in the UDP segment that is the sum of header and data. Since the size of the data field can vary from one UDP segment to the next, an explicit length number is required.

40 - 32 = 8 bytes



4-

Here, The largest possiable source port number is (2^16 - 1) = 65535.

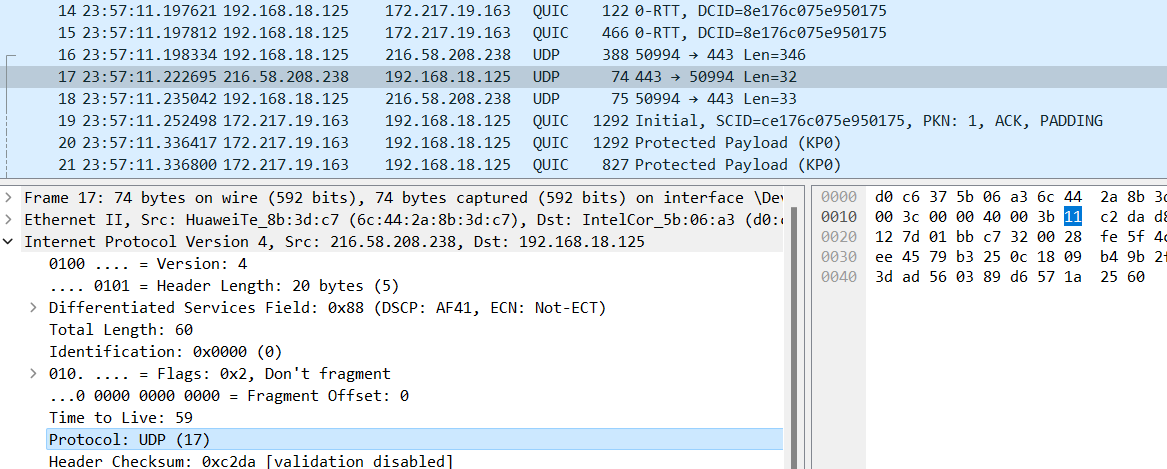
And the header bytes is 8 bytes. So the maximum number of bytes that can be included in a UDP

payload is = 65535 - 8 = 65527 bytes.

5-

The largest possiable source port number is (2^16 - 1) = 65535.





The protocol number for UDP is 17 in decimal notation which in hexadecimal notation is 0x11.

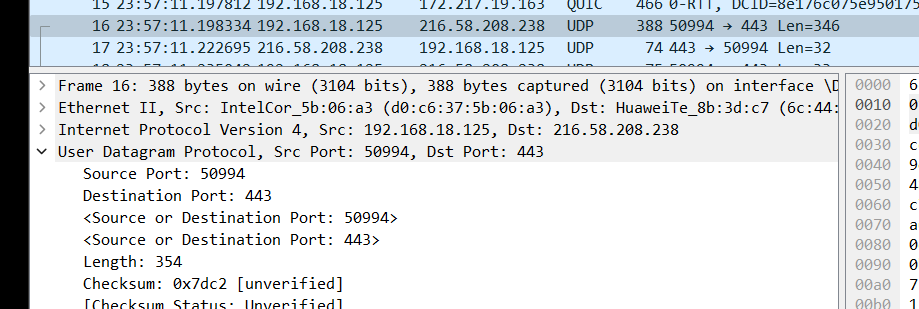
7-

In the process of calculating the 16-bit one's complement of the one's complement sum of the UDP checksum, the IP header, the UDP header, and the data are combined to create a pseudo-header. This pseudo-header is used to generate the checksum, and if the resulting checksum value is not a multiple of two bytes, zero bytes are added to the end of the pseudo-header to make it a multiple of two bytes.

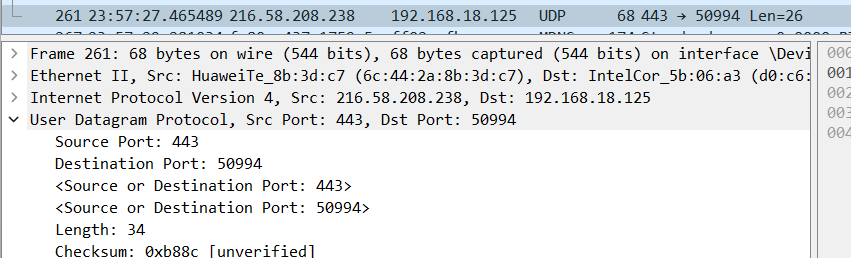
In some cases, the resulting checksum value may be calculated to be 0. In such a case, it is not appropriate to set the checksum value to 0xFFFF. Instead, the checksum value should be left as 0, as changing it to 0xFFFF would alter the integrity of the data and result in an incorrect checksum value.

8-

Send Packet

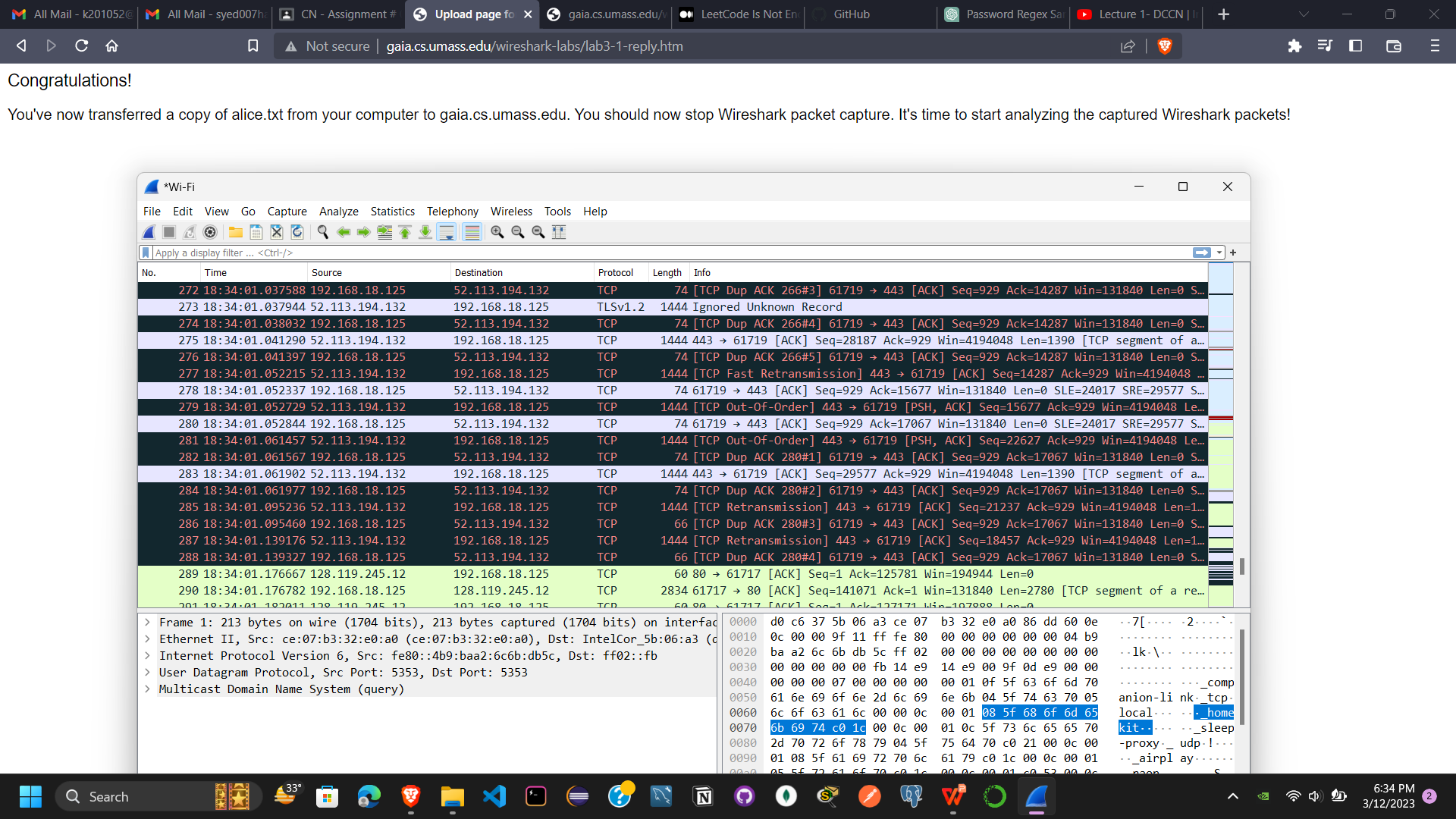


Receive Packet

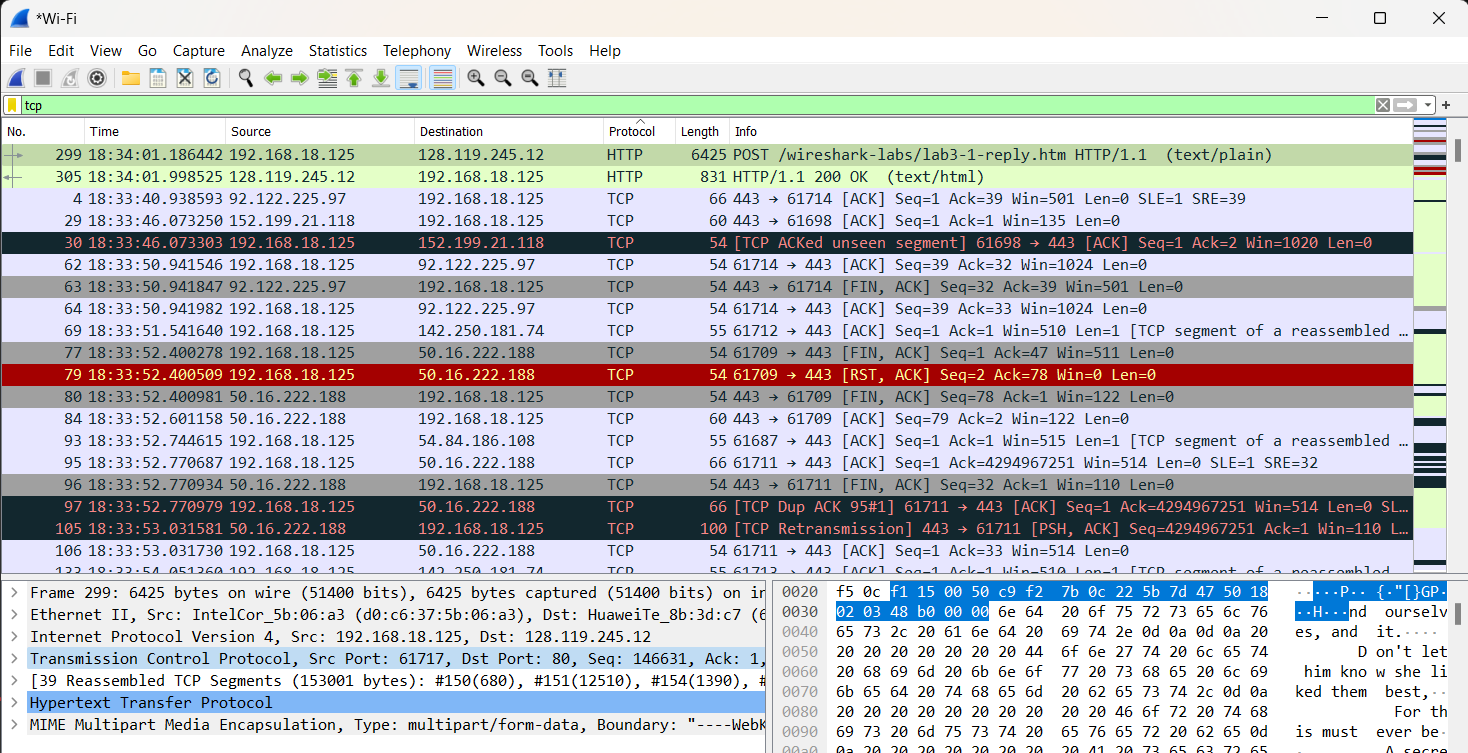


Reply packet destination port = send packet source port

**TCP**



1-



Source IP: 198.168.18.125

TCP PORT NO: 61717

2-

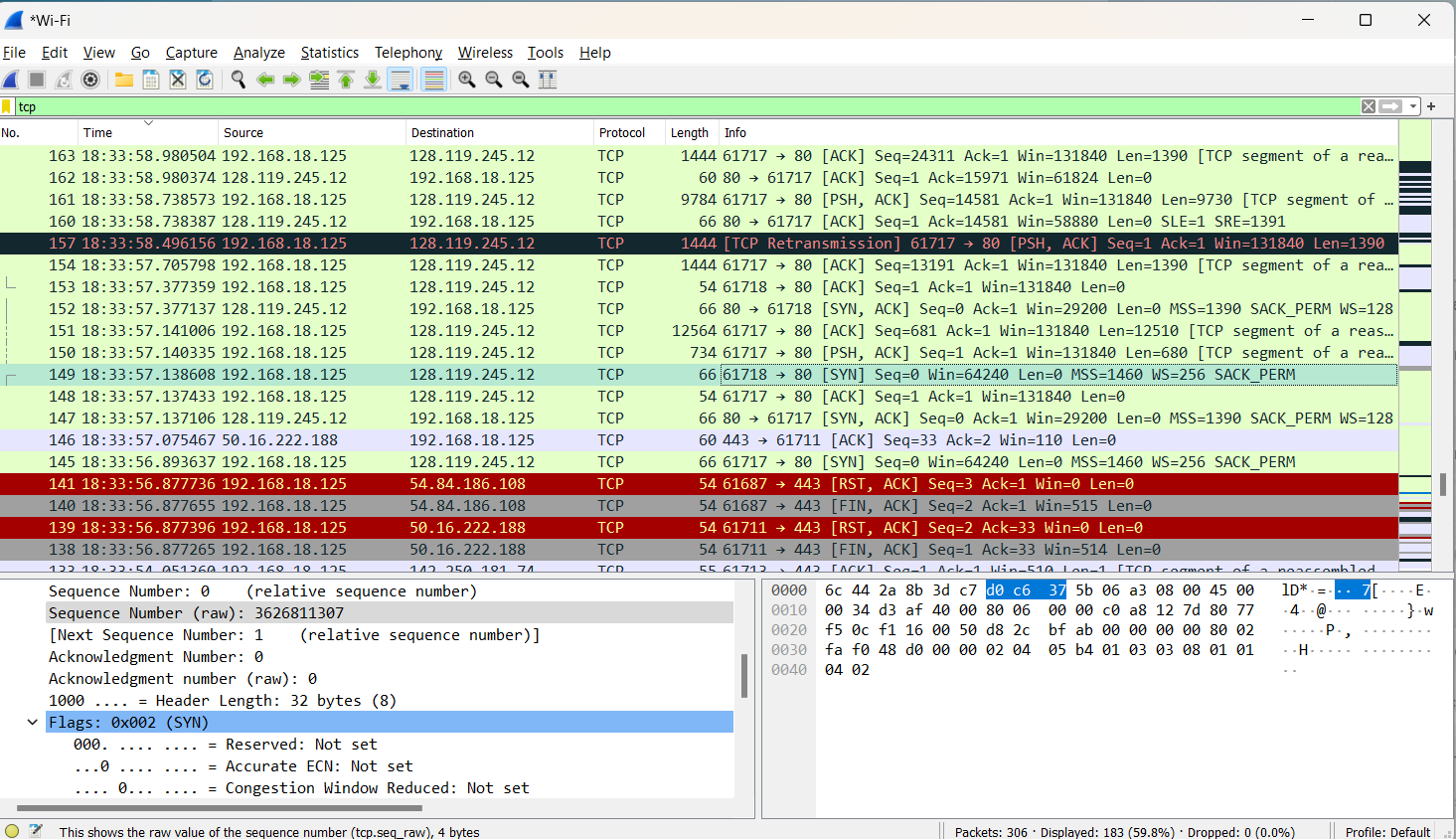
IP address of gaia.cs.umass.edu: 128.119.245.12

Port No: 80

Created my own trace didn’t use Zip File

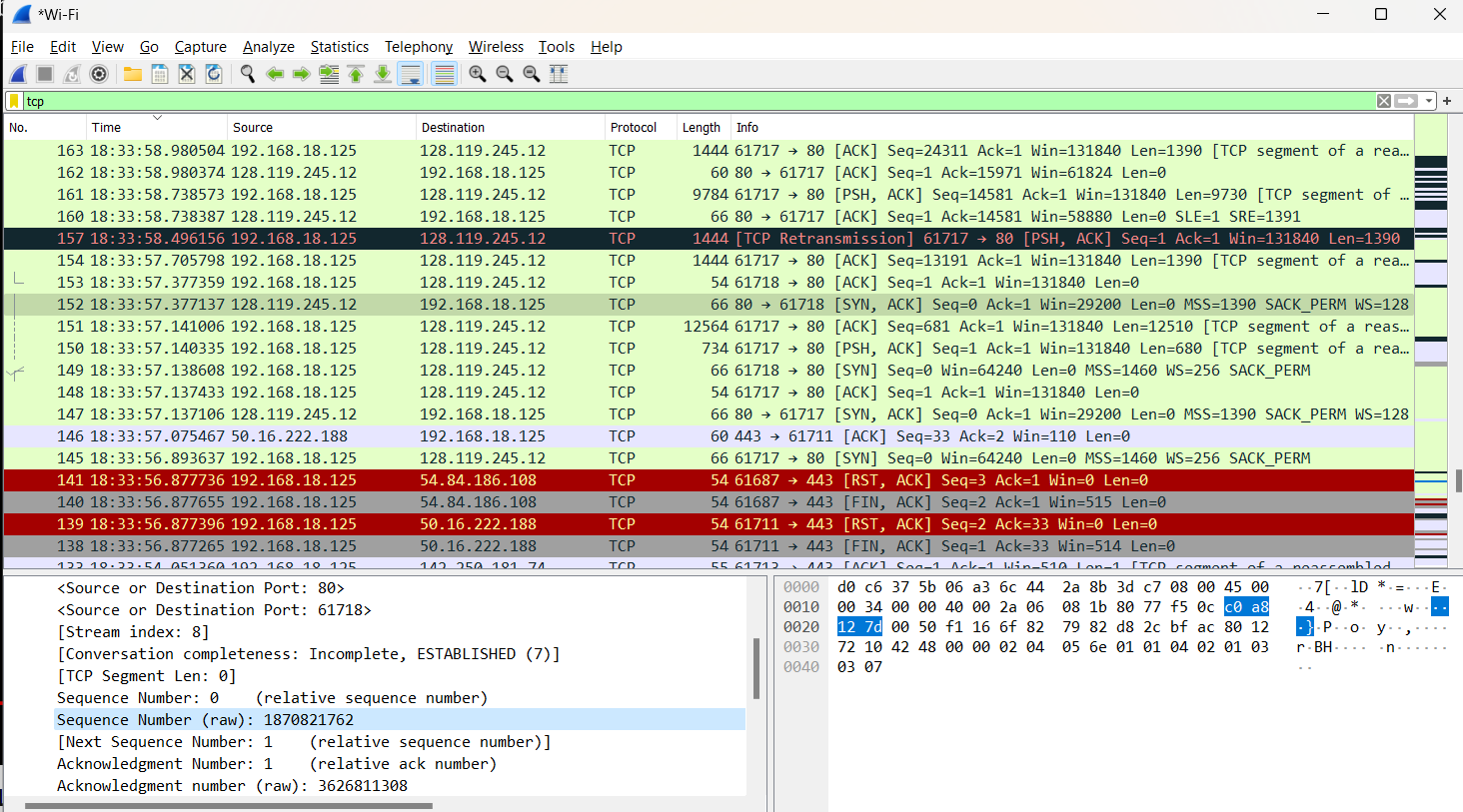
**TCP BASICS**

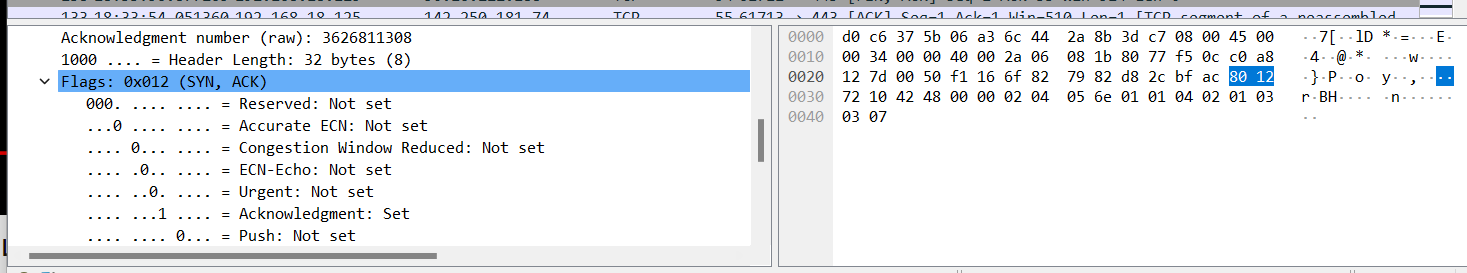
4-



Sequence No: 3626811307

5-

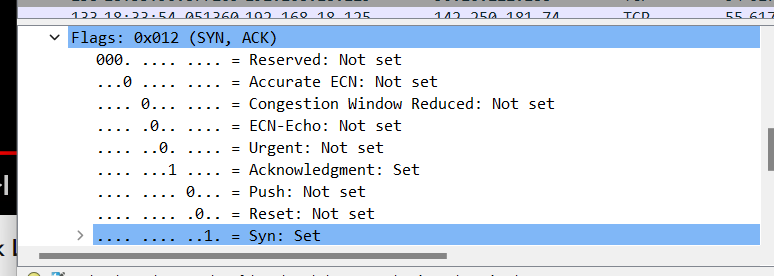




Sequence No: 1870821762

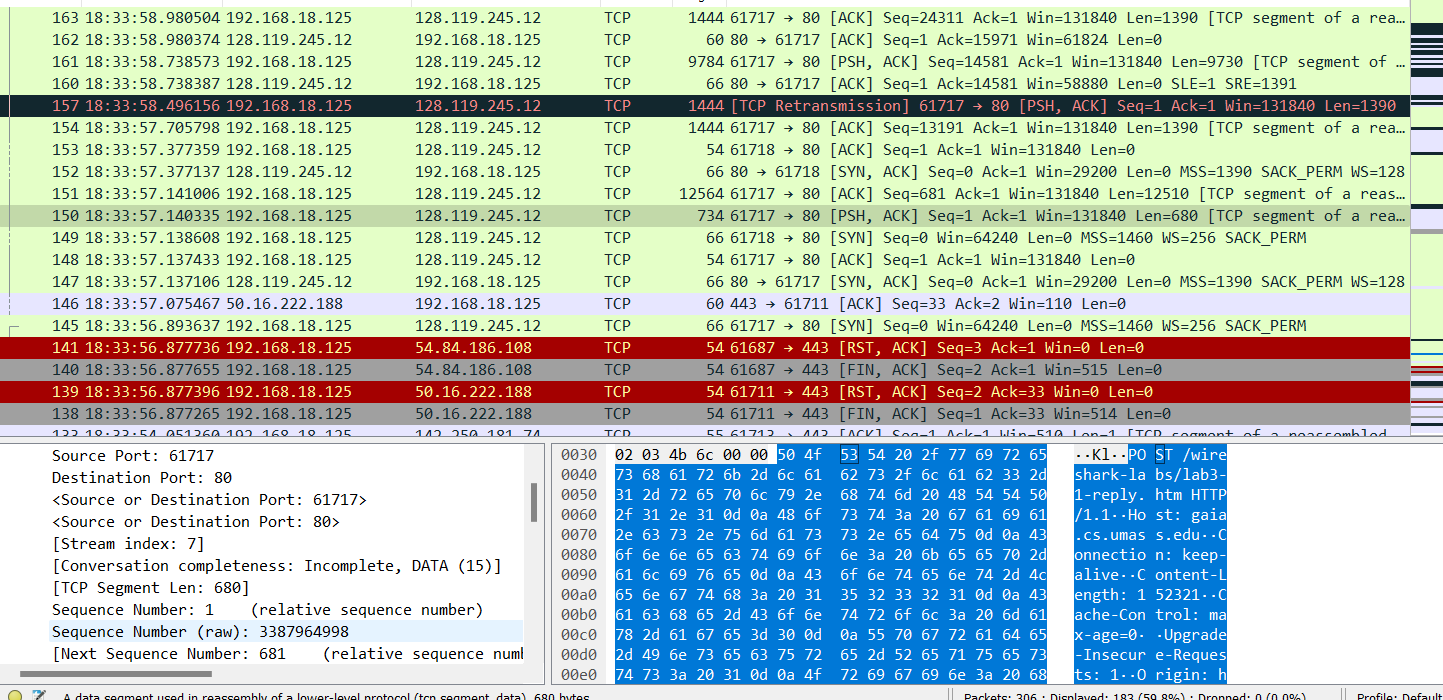
Acknowledgment number (raw): 3626811308

Ack Value for SYN and ACK is equal to sequence number of next ACK segment



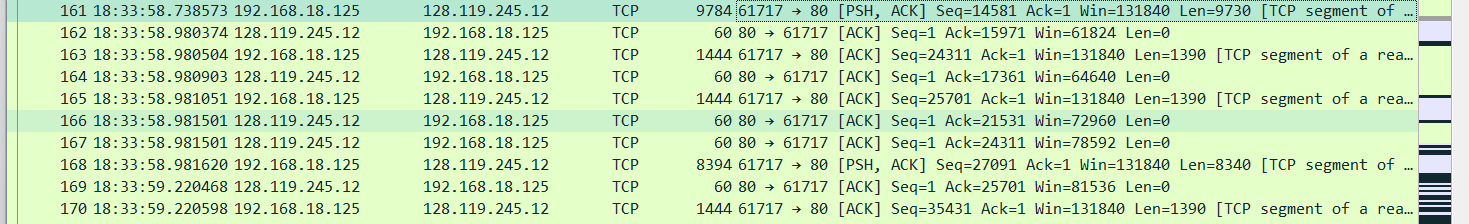
Both flag are set for SYN,ACK

6-



Sequence Number (raw): 3387964998

7-

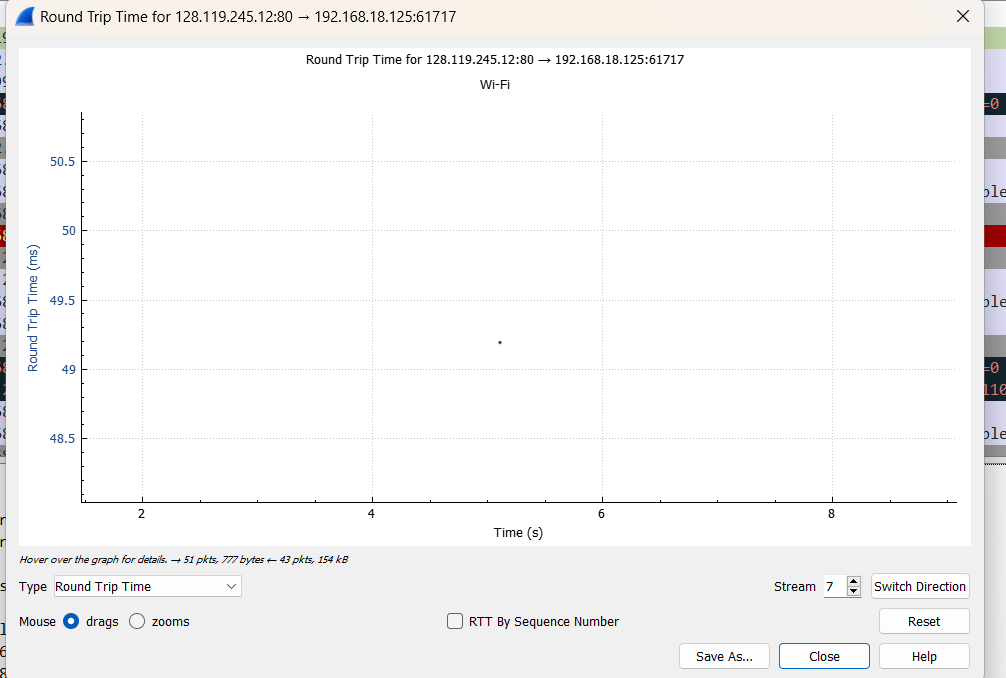


First 5 segments seq nos are: Seq No || Seq Sent time || Seq Rec Time

1. Sequence Number (raw): 3387964998 || frame.time == 18:33:57.140335 || frame.time == 18:33:57.141006
2. Sequence Number (raw): 3387992088 || frame.time == 18:33:58.738573 || frame.time == 18:33:58.980374
3. Sequence Number (raw): 3388014328 || frame.time == 18:33:58.981620 || frame.time == 18:33:59.220468
4. Sequence Number (raw): 3388031008 || frame.time == 18:33:59.701777 || frame.time == 18:33:59.702220
5. Sequence Number (raw): 3388079658 || frame.time == 18:33:59.942676 || frame.time == 18:33:59.943252

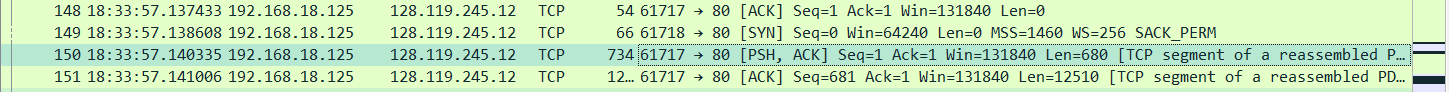
8.

RTTP = Receive Time - Sent Time

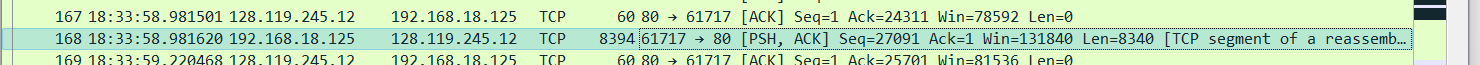


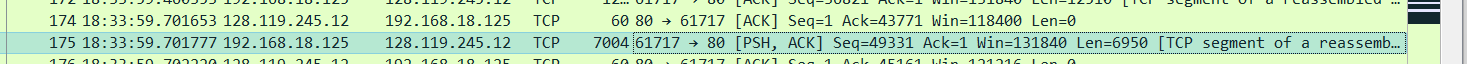
9-

Length of TCP Segments denoted by ‘Len=’





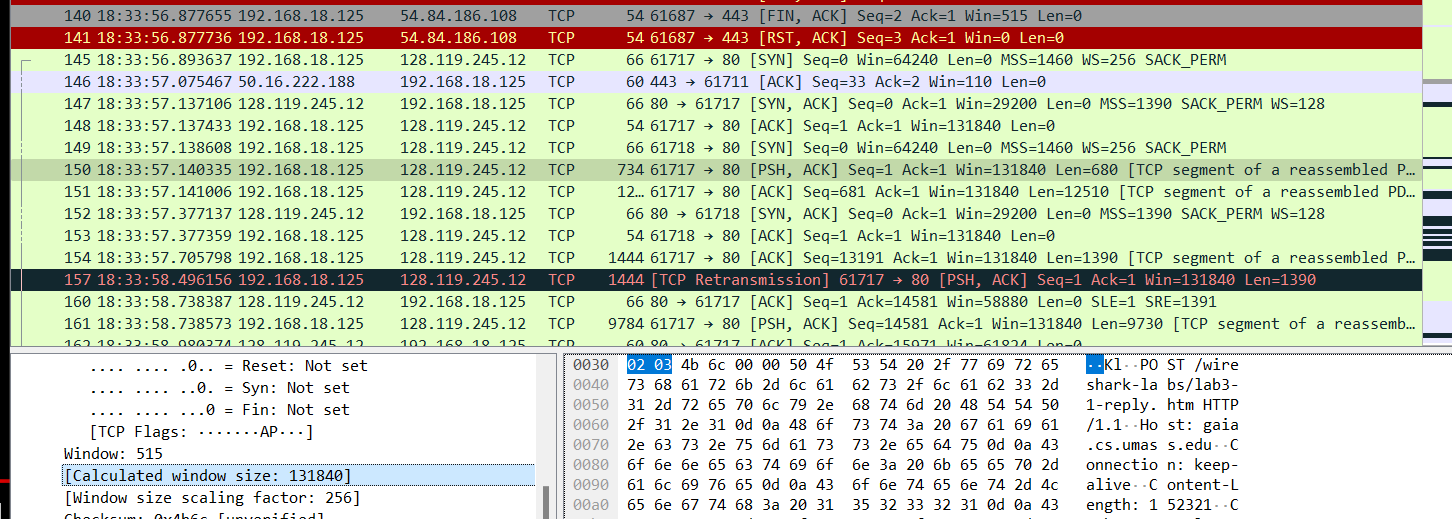




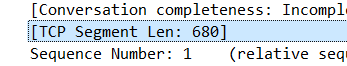




10-



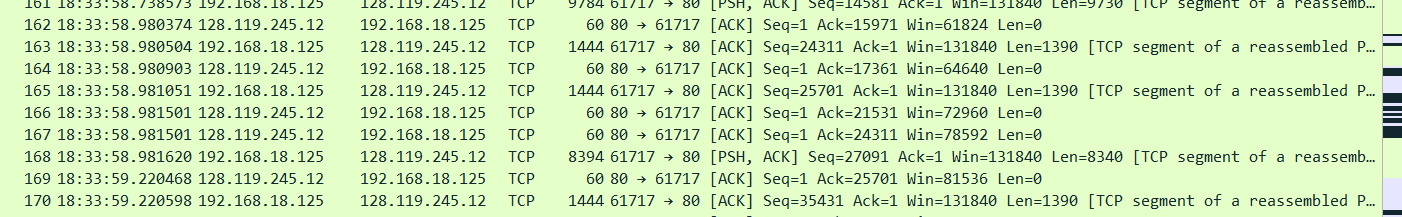
Win=131840 is the minimum amount of available buffer space advertised at the received for the entire trace



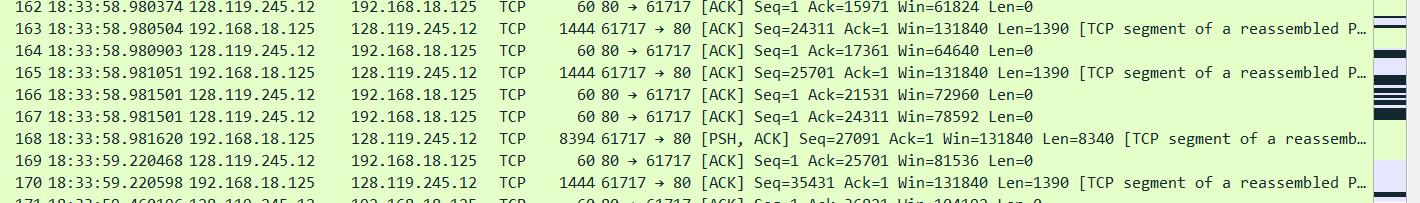
No throttle as Segment length is less than windows size

11-

No retransmitted segments in the trace file as sequence no increasing without repeating



12-



Receiver typically acknowledged Len=1390 bytes

No (Can you identify cases where the receiver is ACKing every other received segment)

13-

First TCP segment size = 1 byte

Last TCP segment size = 190274

Total data = 190274 - 1 = 190273

Transmission time for first segment = 0.014312 secs

Transmission time for last segment = 4.354567 secs

Difference = 4.340255

Throughput is = 190273 / 4.340255 = 43,839.1292677504 MB/SECS