

Student name: Gökay Gülsoy
Student number: 270201072

-CENG312-ASSIGNMENT 2 TCP and UDP-

Q1-) Sender has sent packet with SYN to server and server has returned SYNACK to sender, but when we look at the rest of this traffic we can see that third packet of the handshake never arrived. Each duplicate ACK is querying for ACK 1 with sequence number 1 in the handshake, but as the client has two different packets with sequence number 1, this issue never solved.

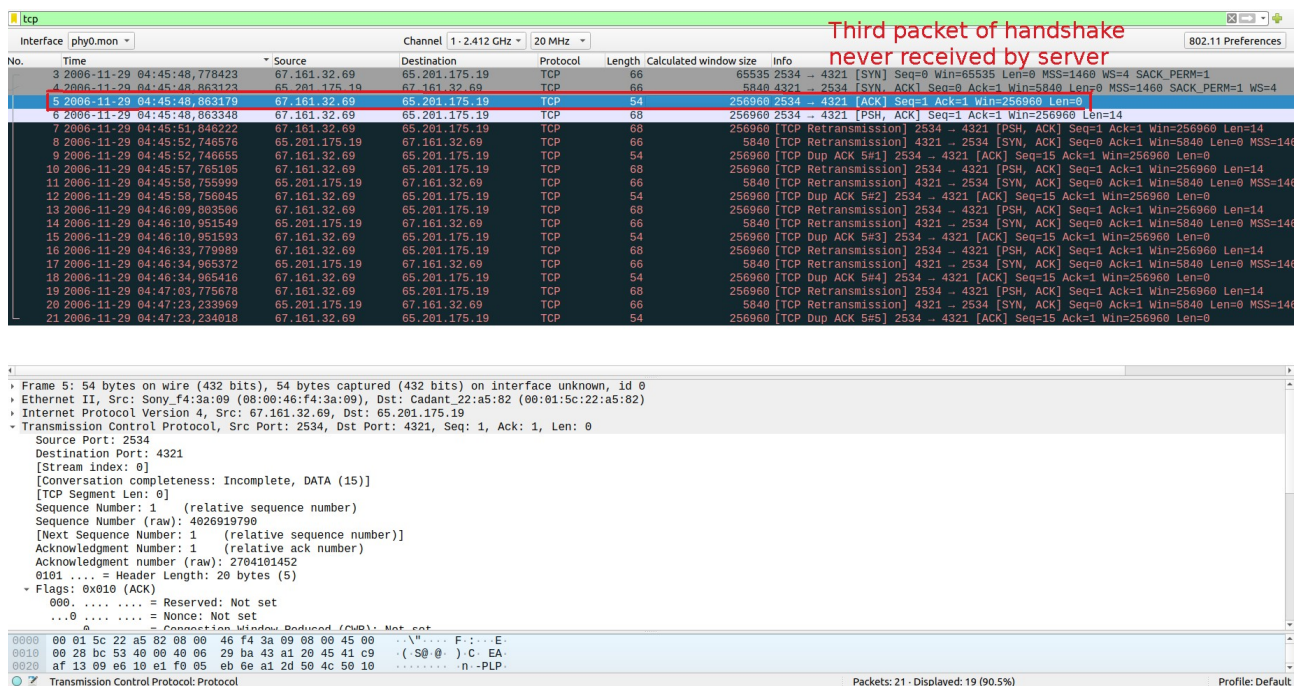


Figure 1: assignment2-tcp-1 third packet in handshake never arrived

Q2-) Firstly server sends an ACK 1 for a packet with sequence number 1, then client with IP address 192.168.0.104 sends a packet with sequence number 6349, but then server has sent three duplicate ACKs for a packet with sequence number 1. Then client used fast-retransmission to send the packet with sequence number 1, this indicates that it was a lost packet with high probability. Then client retransmits packets with sequence numbers 1461, 2921, 4381, 1 respectively. Server ACKed the packet with sequence number 1, and client retransmitted packets with sequence numbers 4381 and 5841. Server has returned cumulative ACK 4381, then client has sent packet with sequence numbers 7301 and 8761. Then server returned cumulative ACK 6861 and client has sent packets with sequence numbers 10221, 11681, and 12621. Server

Server then sent three duplicate ACK for cumulative ACK 6861 and as no ACK has returned from client, client fast-retransmitted packet with sequence number 6861 in order to avoid congestion which may be caused by a lot of retransmissions. Finally server retransmitted cumulative ACK 6861. In summary we can see that there were a lot of packet loss and in order to avoid congestion, fast-retransmissions used so that no more than three duplicate ACK's will accumulate and this avoids congestion.

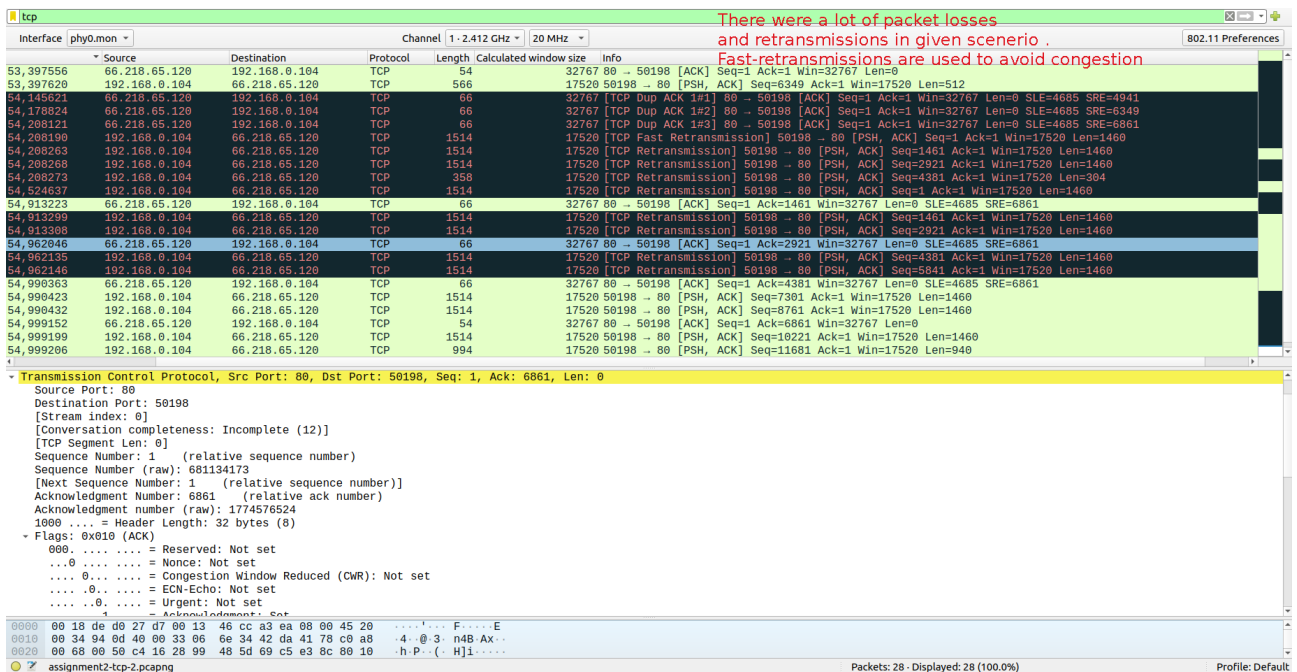


Figure 2: assignment2-tcp-2 a lot of packet loss and retransmission scenerio

Q3-) When we monitor the traffic we can see 41 instances of “TCP ZeroWindow” segment This means that due to lack of available buffer space, server is not able to receive any more data. We can see that server sends a TCP segment with “TCP Window Full” 9 times. During that time interval to keep TCP connection open even if no packet can be received, server also sends TCP segment with “TCP-keep-alive” info.

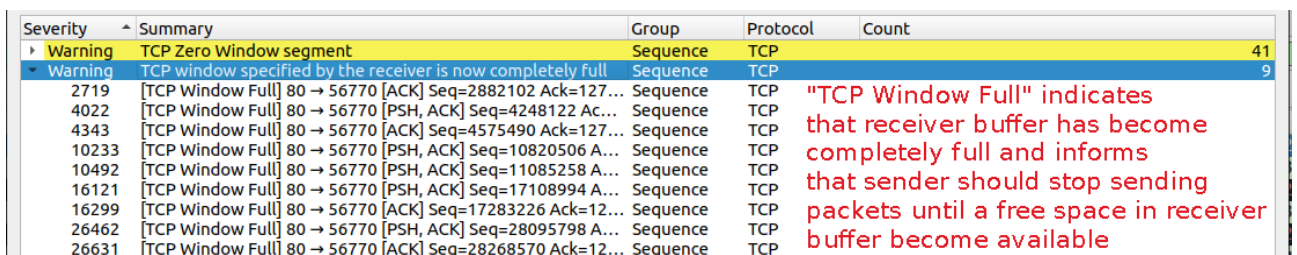


Figure 3: assignment2-tcp-3 “TCP Window Full”

Wireshark · Expert Information · assignment2-tcp-3.pcapng					
Packet	Summary	Group	Protocol	Count	
Warning	TCP Zero Window segment	Sequence	TCP	41	
2720	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28835...	Sequence	TCP		
2722	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28835...	Sequence	TCP		
2724	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28835...	Sequence	TCP		
4023	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=42484...	Sequence	TCP		
4025	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=42484...	Sequence	TCP		
4027	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=42484...	Sequence	TCP		
4029	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=42484...	Sequence	TCP		
4031	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=42484...	Sequence	TCP		
4033	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=42484...	Sequence	TCP		
4035	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=42484...	Sequence	TCP		
4344	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=45769...	Sequence	TCP		
4346	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=45769...	Sequence	TCP		
4348	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=45769...	Sequence	TCP		
10234	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=10820...	Sequence	TCP		
10236	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=10820...	Sequence	TCP		
10238	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=10820...	Sequence	TCP		
10240	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=10820...	Sequence	TCP		
10242	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=10820...	Sequence	TCP		
10244	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=10820...	Sequence	TCP		
10493	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=11085...	Sequence	TCP		
10495	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=11085...	Sequence	TCP		
16122	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17109...	Sequence	TCP		
16124	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17109...	Sequence	TCP		
16126	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17109...	Sequence	TCP		
16128	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17109...	Sequence	TCP		
16130	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17109...	Sequence	TCP		
16132	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17109...	Sequence	TCP		
16134	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17109...	Sequence	TCP		
16300	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17284...	Sequence	TCP		
16302	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17284...	Sequence	TCP		
16304	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=17284...	Sequence	TCP		
26463	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28096...	Sequence	TCP		
26465	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28096...	Sequence	TCP		
26467	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28096...	Sequence	TCP		
26469	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28096...	Sequence	TCP		
26471	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28096...	Sequence	TCP		
26473	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28096...	Sequence	TCP		
26475	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28096...	Sequence	TCP		
26632	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28270...	Sequence	TCP		
26634	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28270...	Sequence	TCP		
26636	[TCP ZeroWindow] 56770 → 80 [ACK] Seq=1270 Ack=28270...	Sequence	TCP		

"TCP Zero Window" indicates that receiver can not receive any more segment as the window has become filled and sender stops sending until sufficient space become available in receiver buffer

Figure 4: assignment2-tcp-3 "TCP Zero Window segment"

Severity	Summary	Group	Protocol	Count	
Warning	TCP Zero Window segment	Sequence	TCP	41	
Warning	TCP window specified by the receiver is now completely full	Sequence	TCP	9	
Note	This frame undergoes the connection closing	Sequence	TCP	1	
Note	This frame initiates the connection closing	Sequence	TCP	1	
Note	TCP keep-alive segment	Sequence	TCP	32	
2721	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=2883561 Ack=1270...	Sequence	TCP		
2723	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=2883561 Ack=1270...	Sequence	TCP		
4024	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4248449 Ack=1270...	Sequence	TCP		
4026	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4248449 Ack=1270...	Sequence	TCP		
4028	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4248449 Ack=1270...	Sequence	TCP		
4030	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4248449 Ack=1270...	Sequence	TCP		
4032	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4248449 Ack=1270...	Sequence	TCP		
4034	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4248449 Ack=1270...	Sequence	TCP		
4345	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4576949 Ack=1270...	Sequence	TCP		
4347	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=4576949 Ack=1270...	Sequence	TCP		
10235	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=10820997 Ack=127...	Sequence	TCP		
10237	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=10820997 Ack=127...	Sequence	TCP		
10239	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=10820997 Ack=127...	Sequence	TCP		
10241	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=10820997 Ack=127...	Sequence	TCP		
10243	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=10820997 Ack=127...	Sequence	TCP		
10494	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=11085809 Ack=127...	Sequence	TCP		
16123	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17109485 Ack=127...	Sequence	TCP		
16125	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17109485 Ack=127...	Sequence	TCP		
16127	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17109485 Ack=127...	Sequence	TCP		
16129	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17109485 Ack=127...	Sequence	TCP		
16131	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17109485 Ack=127...	Sequence	TCP		
16133	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17109485 Ack=127...	Sequence	TCP		
16301	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17284685 Ack=127...	Sequence	TCP		
16303	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=17284685 Ack=127...	Sequence	TCP		
26464	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28096289 Ack=127...	Sequence	TCP		
26466	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28096289 Ack=127...	Sequence	TCP		
26468	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28096289 Ack=127...	Sequence	TCP		
26470	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28096289 Ack=127...	Sequence	TCP		
26472	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28096289 Ack=127...	Sequence	TCP		
26474	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28096289 Ack=127...	Sequence	TCP		
26633	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28270029 Ack=127...	Sequence	TCP		
26635	[TCP Keep-Alive] 80 → 56770 [ACK] Seq=28270029 Ack=127...	Sequence	TCP		

"TCP Keep-Alive" indicates that even if no packets can be received by receiver keep the TCP connection between sender and server open. This also indicates that no space is available in receiver buffer

Figure 5: assignment2-tcp-3 "TCP Keep Alive"

Q4-a-1) We can see from Time Sequence Stevens Graph that sequence numbers of packets sent start with low sending rate, but after that it has increased exponentially until approximately 4.127 second. After 4.193 second we can see that there is a congestion up until to 4.418 second, then again sequence number of packets sent started to increase at low rate, after that it continued with exponential increase until approximately 5.983 second after which increase switched to linear.

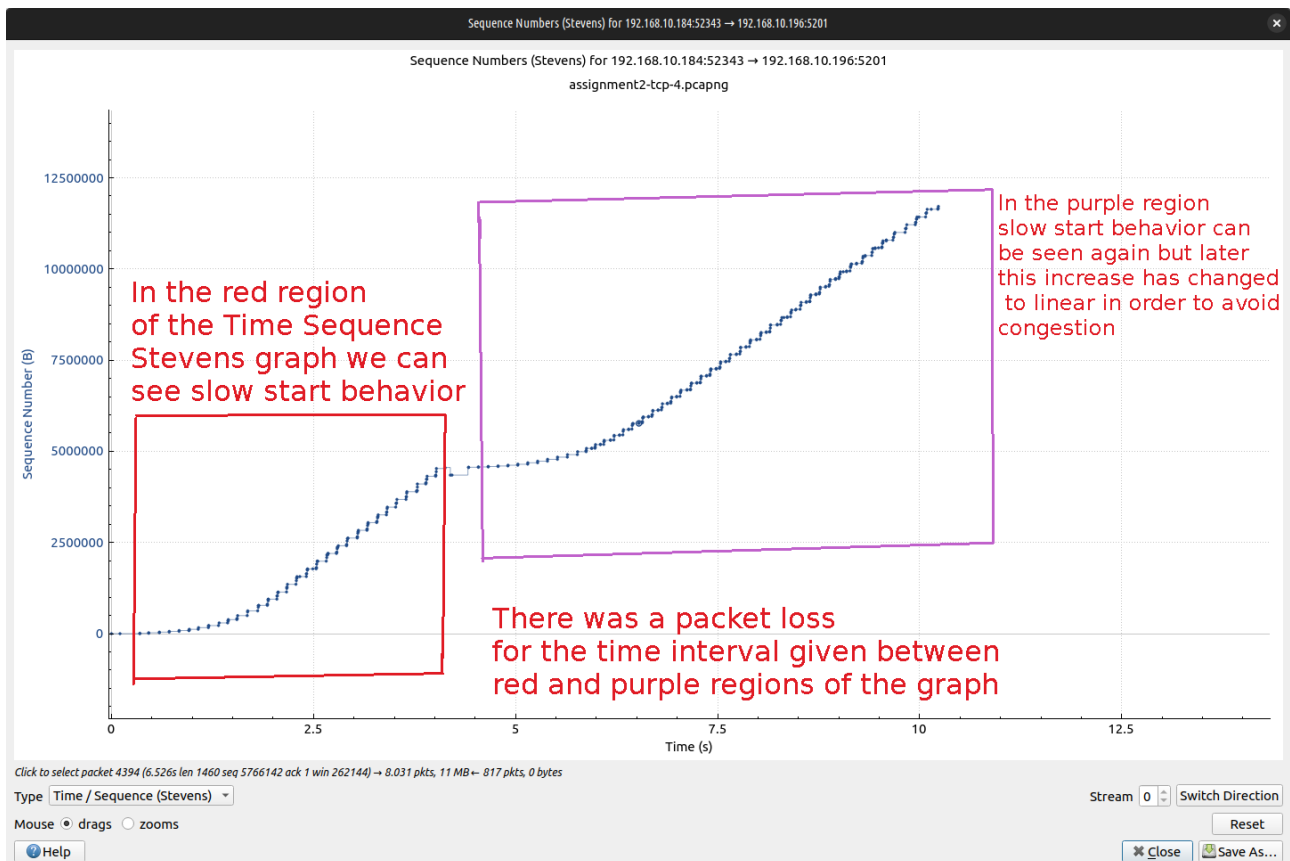


Figure 6: assignmet2-tcp-4-a-1 Time Sequence Stevens Graph

Q4-a-2) In the Window Scaling Graph we can see that receiving window is a flat green line and it denotes the window's size at a given time, so relation between receiving window and bytes out (which denotes bytes in flight) is that number of bytes out can never exceed current receiving window size. If receiving window scales, then maximum number of bytes out can scale accordingly.

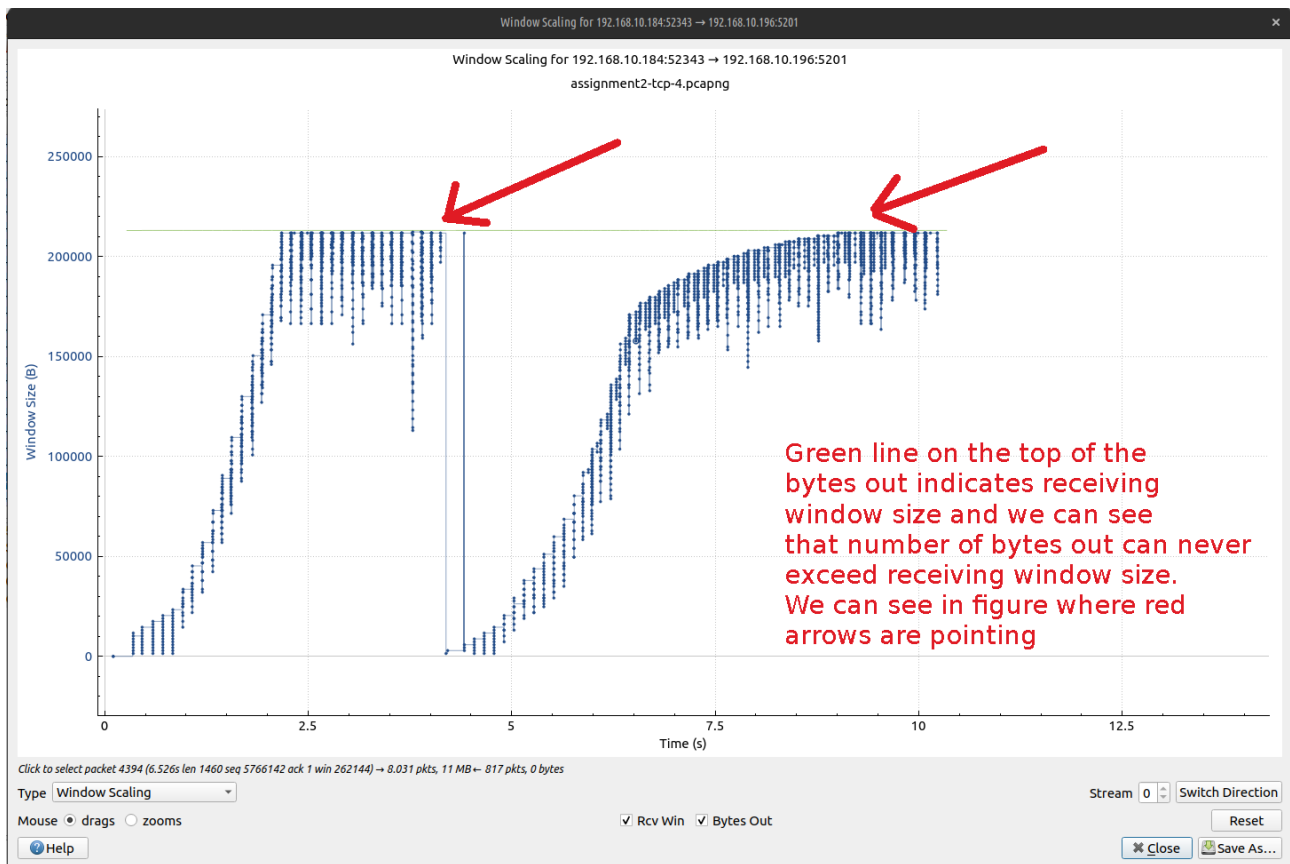


Figure 7: assignment2-tcp-4-a-2 Window Scaling Graph

Q4-b-4) We can see the slow start behavior in Time Sequence Stevens graph and Window Scaling Graph. At 0.00011 second sent packet number is 3, if we look at the 5.123 second sent packet number reaches to 926, so number of packets sent per second is approximately 176.481 in time interval 0.00011 to 5.123 second. If we look at the 67.68 second, number of last packet sent was 18805, if increase rate had remained same, then number of last packet sent would have been 11944 but actual packet number sent at 67.68 second 18805. This indicates that packet sent rate has increased over time.

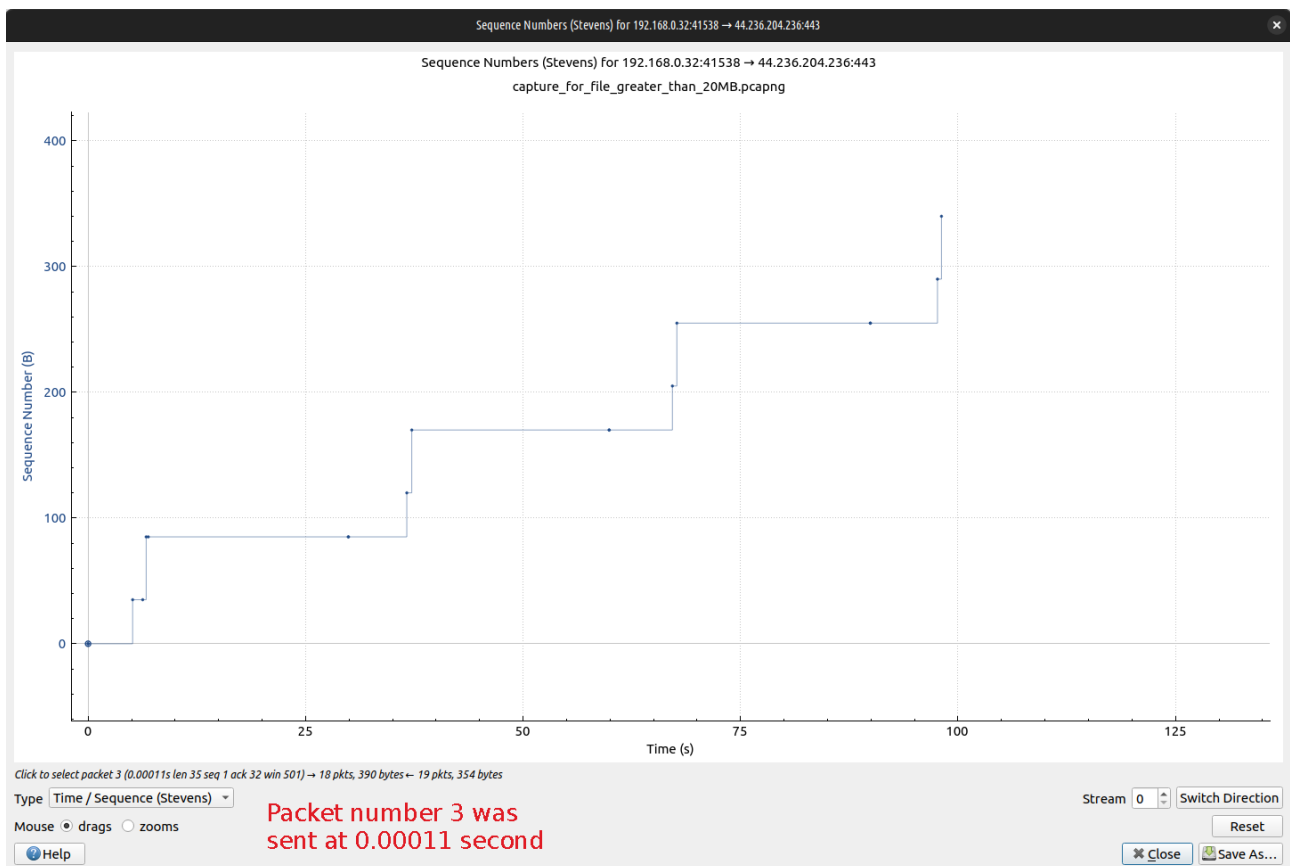


Figure 8: assignment2-tcp-4-b-4 Time Sequence Stevens Graph

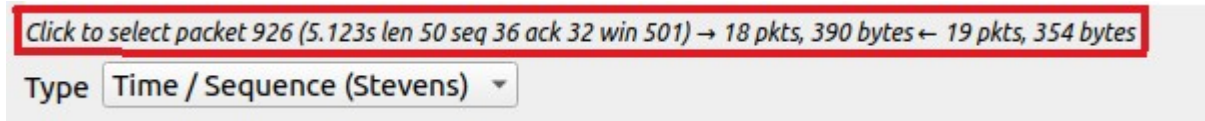


Figure 9: Packet number 926 was sent at 5.123 second

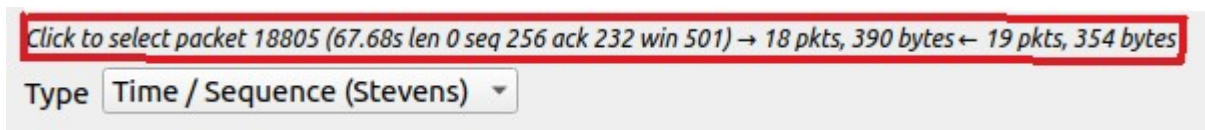


Figure 10: Packet number 18805 was sent at 67.68 second

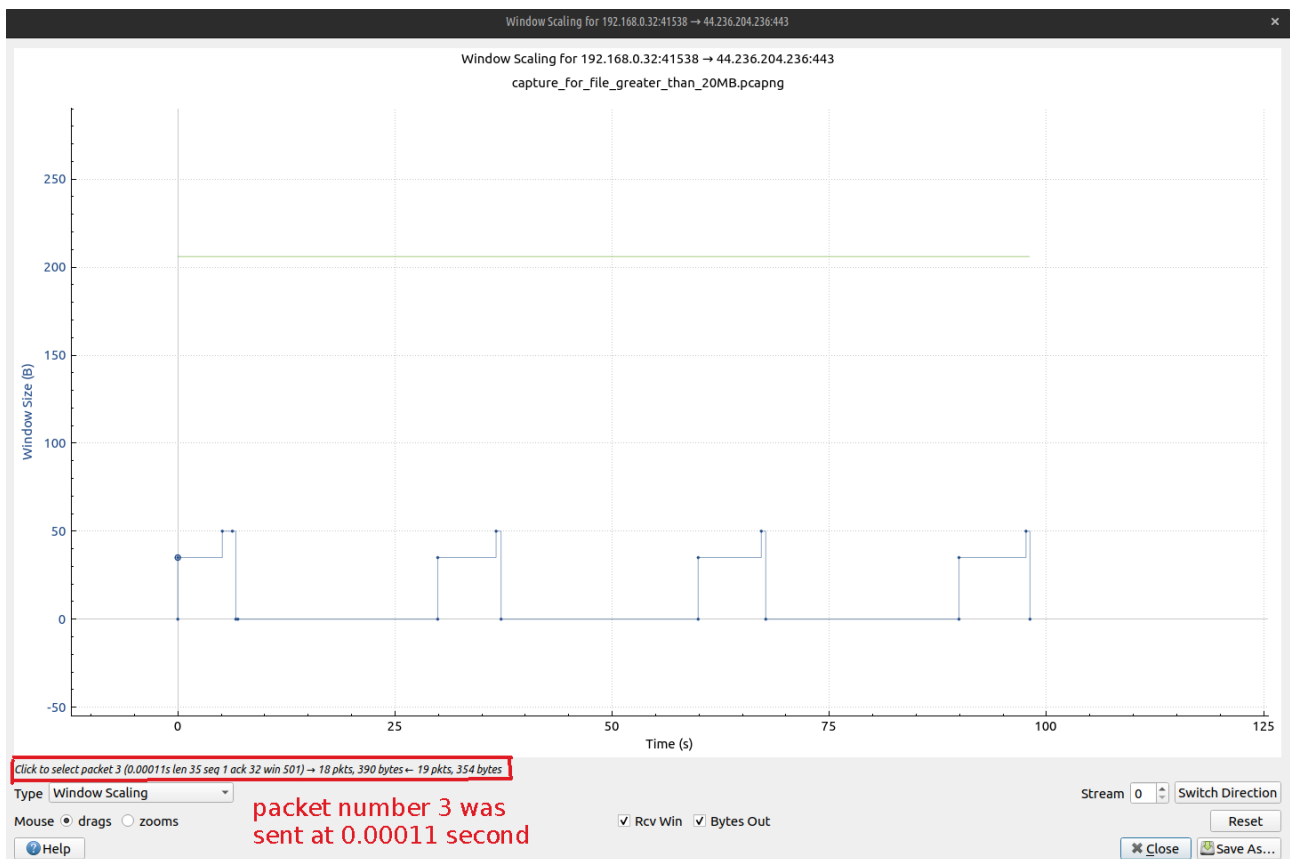


Figure 11: assignment2-tcp-4-b-4 Window Scaling Graph

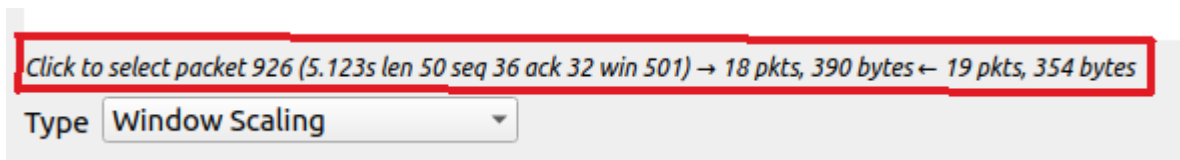


Figure 12: Packet number 926 was sent at 5.123 second

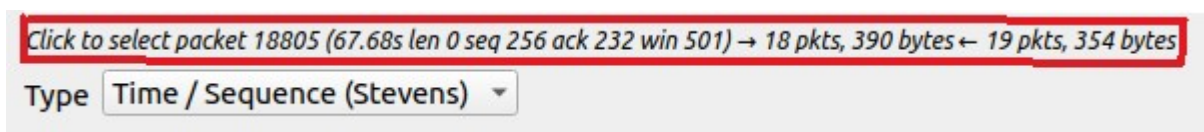
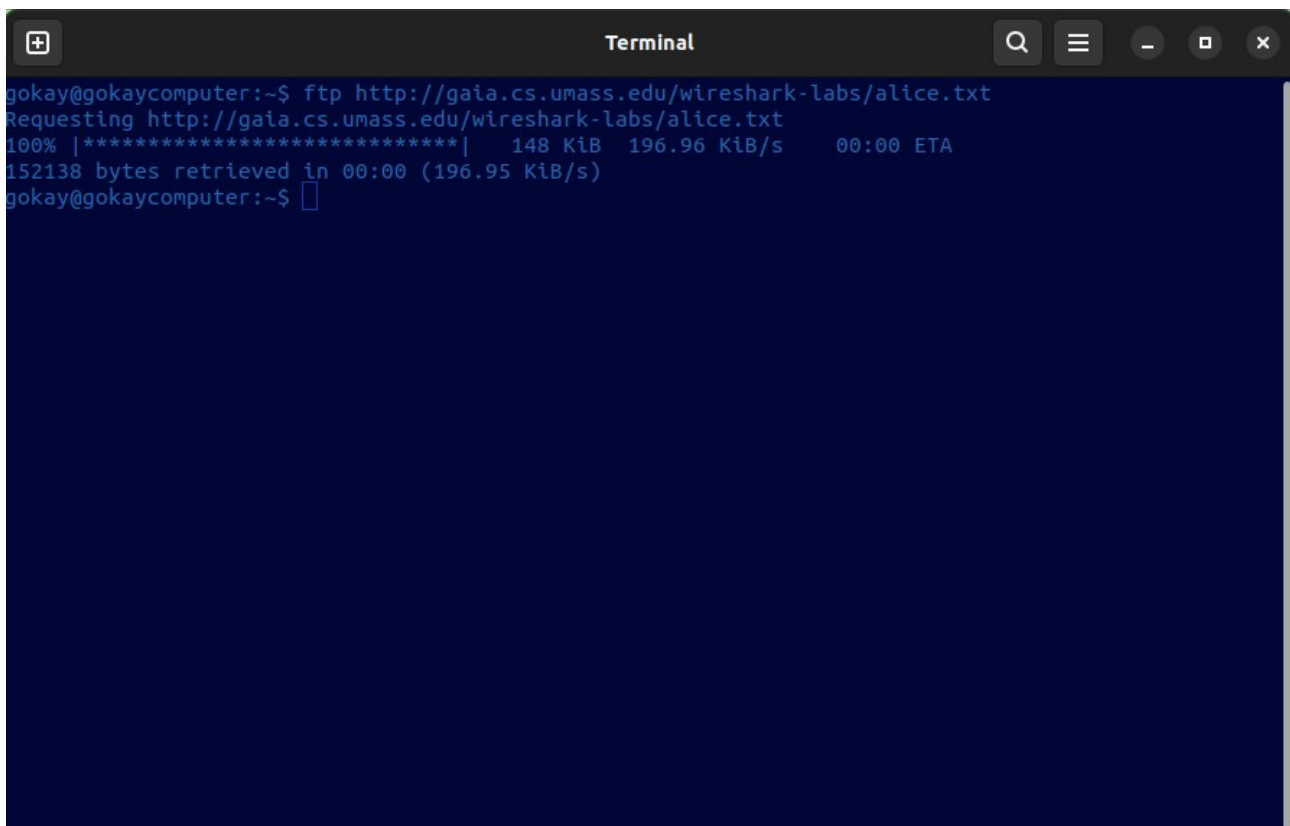


Figure 13: Packet number 18805 was sent at 67.68 second

Q4-b-5) I have made an ftp request to gaia.cs.umass.edu.wireshark-labs/alice.txt to download the file `alice.txt`. According to Time Sequence Stevens Graph at 1.334 second, packet number 13 was sent and at 1.564 second packet number 25 was sent. Number of packet sent per second as a rate in time interval 1.334 to 1.564 is approximately 19. If we look, at 1.789 second sent packet number was 51 and 2.088 second sent packet

number was 75 . Number of packets sent per second as a rate in time interval 1.789 to 2.088 is approximately 67 so we can see that packet sending rate started at slower rate and it has increased gradually. This indicates slow start behaviour. Same behavior can be seen in Window Scaling Graph. Also when we look at the window size it has increased gradually in certain time intervals in response to increased number of sent packets.

A terminal window titled "Terminal" with a dark blue background. The prompt is "gokay@gokaycomputer:~\$". The command entered is "ftp http://gaia.cs.umass.edu/wireshark-labs/alice.txt". The output shows the file being requested, a progress bar at 100%, and the file size and retrieval speed. The prompt returns to "gokay@gokaycomputer:~\$".

```
gokay@gokaycomputer:~$ ftp http://gaia.cs.umass.edu/wireshark-labs/alice.txt
Requesting http://gaia.cs.umass.edu/wireshark-labs/alice.txt
100% |*****| 148 KiB 196.96 KiB/s 00:00 ETA
152138 bytes retrieved in 00:00 (196.95 KiB/s)
gokay@gokaycomputer:~$
```

Figure 14: ftp request made to gaia.cs.umass.edu to fetch alice.txt file

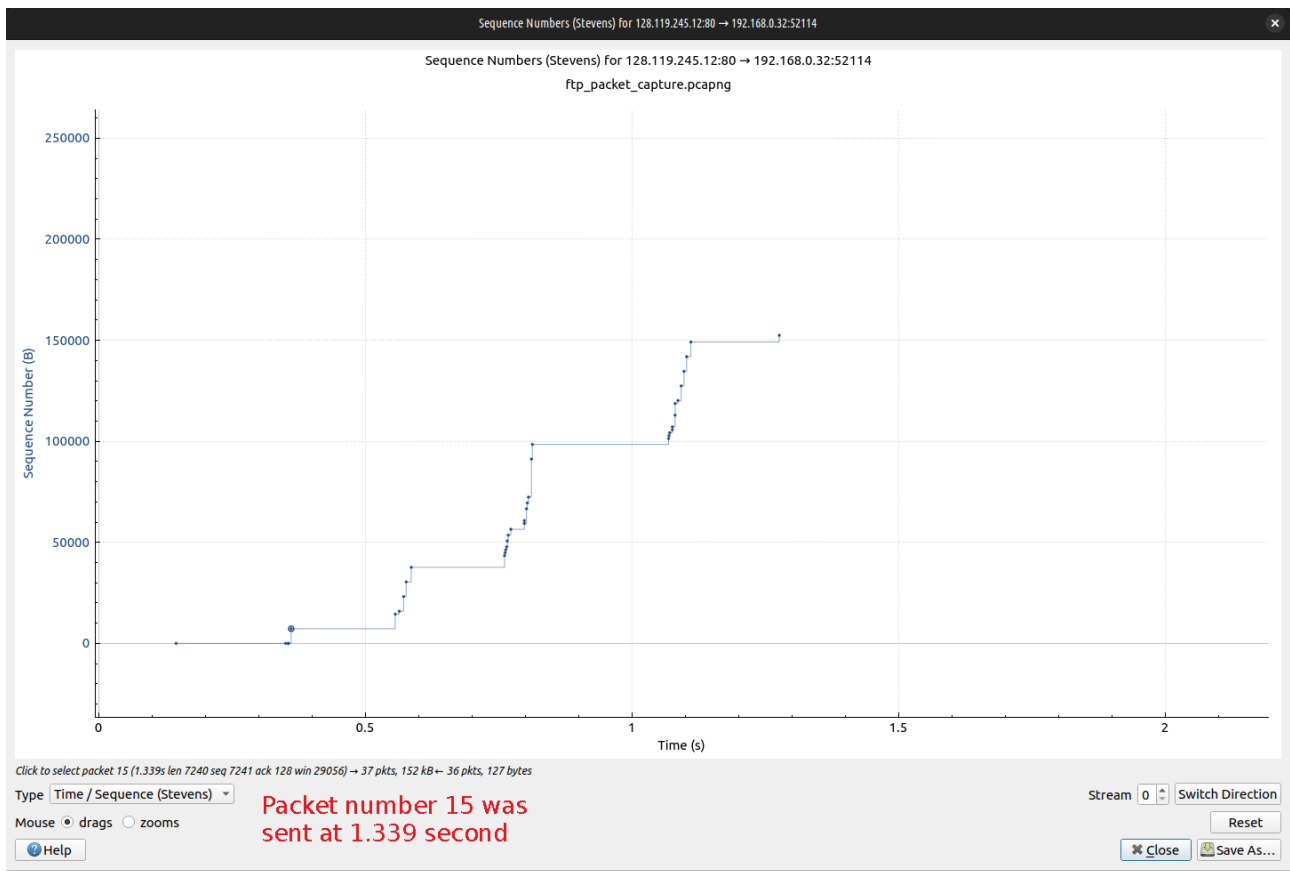


Figure 15: assignment2-tcp-4-b-5 Time Sequence Stevens Graph



Figure 16: Packet number 13 was sent at 1.334 second



Figure 17: Packet number 25 was sent at 1.564 second

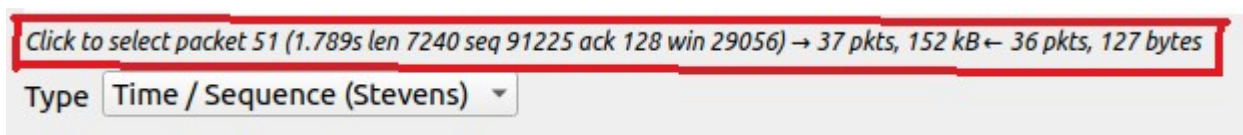


Figure 18: Packet number 51 was sent at 1.789 second

Click to select packet 75 (2.088s len 3322 seq 149145 ack 128 win 29056) → 37 pkts, 152 kB ← 36 pkts, 127 bytes

Type Time / Sequence (Stevens) ▾

Figure 19: Packet number 75 was sent 2.088 second

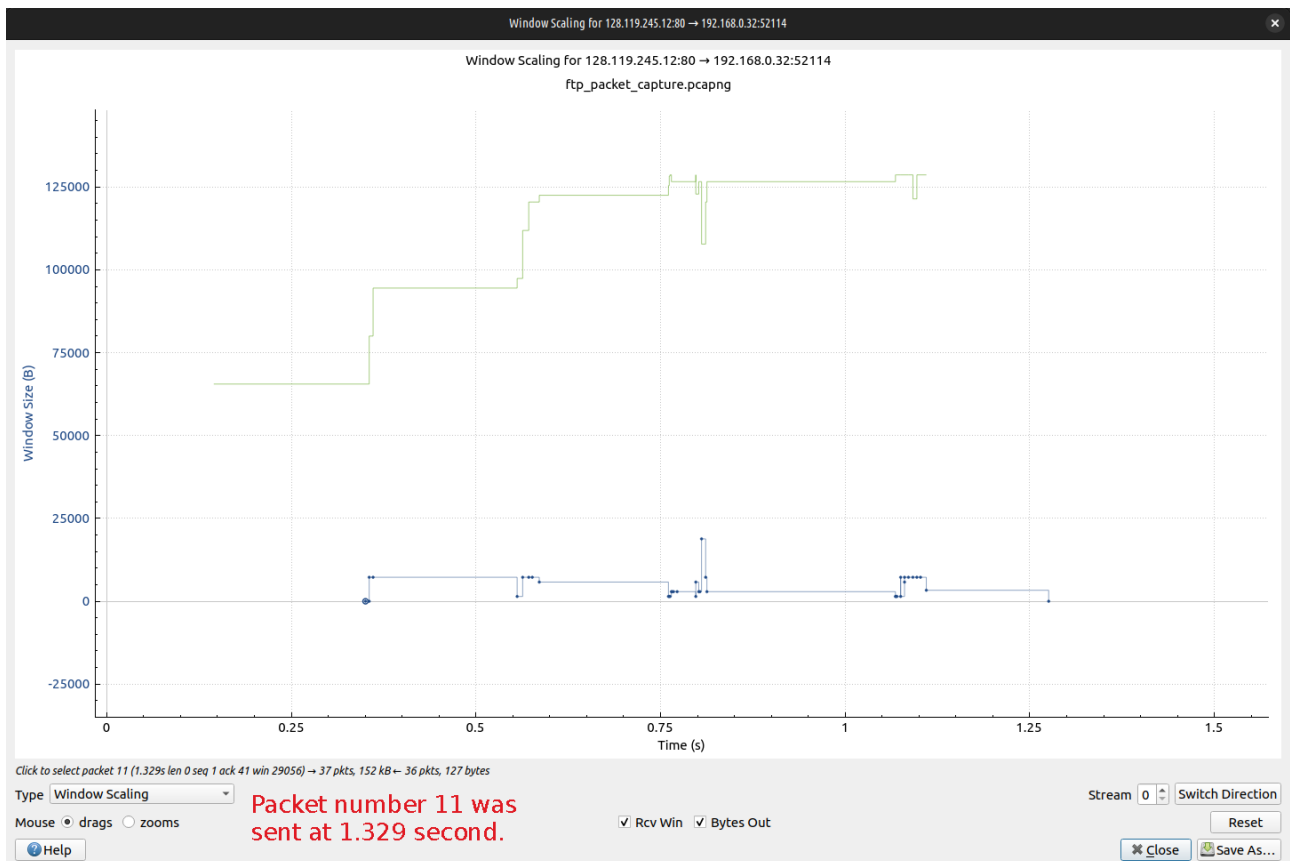


Figure 20: assignment2-tcp-4-b-5 Window Scaling Graph

Click to select packet 13 (1.334s len 7240 seq 1 ack 128 win 29056) → 37 pkts, 152 kB ← 36 pkts, 127 bytes

Type Time / Sequence (Stevens) ▾

Figure 21: Packet number 13 was sent at 1.334 second

Click to select packet 25 (1.564s len 5792 seq 37649 ack 128 win 29056) → 37 pkts, 152 kB ← 36 pkts, 127 bytes

Type Window Scaling ▾

Figure 22: Packet number 25 was sent at 1.564 second

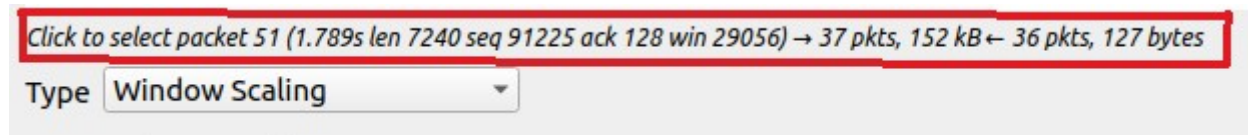


Figure 23: Packet number 51 was sent at 1.789 second

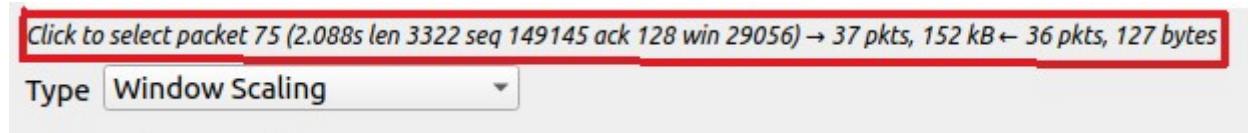


Figure 24: Packet number 75 was sent 2.088 second

-UDP SECTION-

I have created a scenerio in which I made a DNS query for iyte.edu.tr domain name with dig tool which can be used for making DNS queries from terminal in Linux. As DNS uses UDP as an underlying transport layer protocol we can analyze UDP packets send and received as well in this scenario.

Q1-) Destination IP address of the sent packet is 192.168.166.10. It is not possible to be sure that packet reaches to destination address because UDP does not use any acknowledgement mechanism. Packet may be lost and we can never know this because of unavaliabe acknowledgement mechanism.

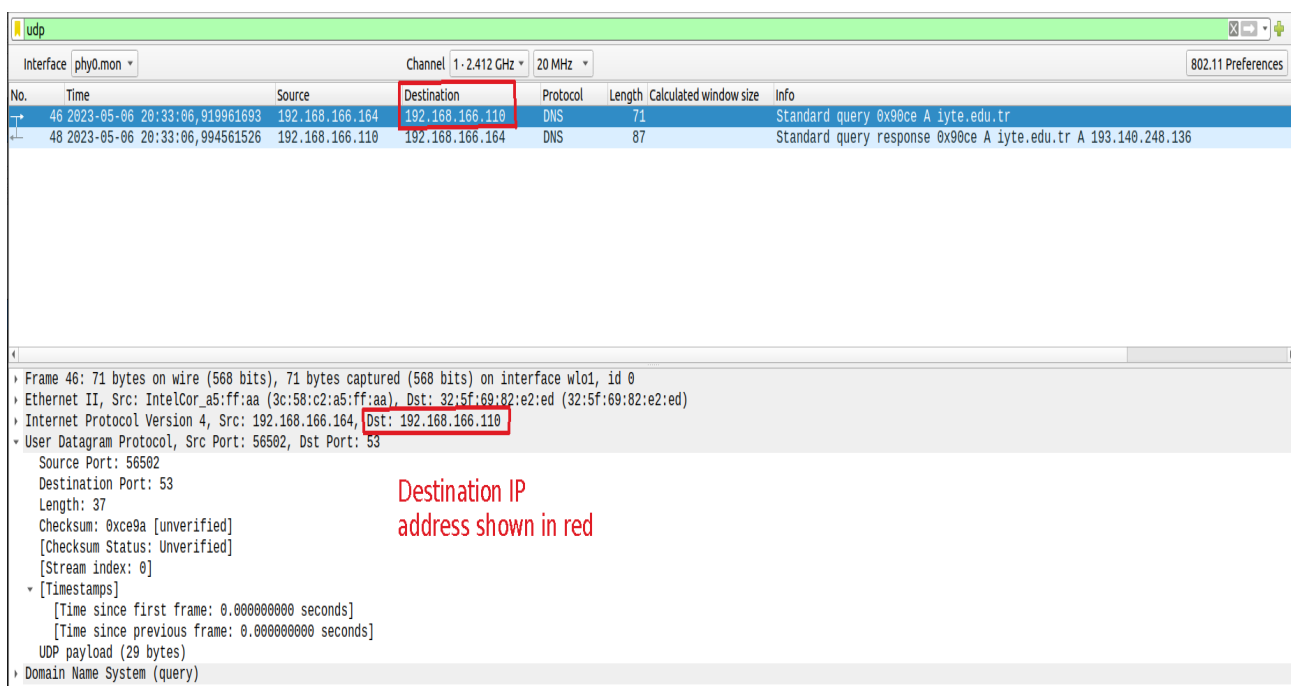


Figure 25: Destination IP address of sent UDP packet

Q2-) Source port number for sent packet from my host is 56502, and destination port number is 53. For received packet as a reply to this packet, source port number is 53, and destination port number is 56502. Source and destination port numbers are reverses of each other for the sent and received packet.

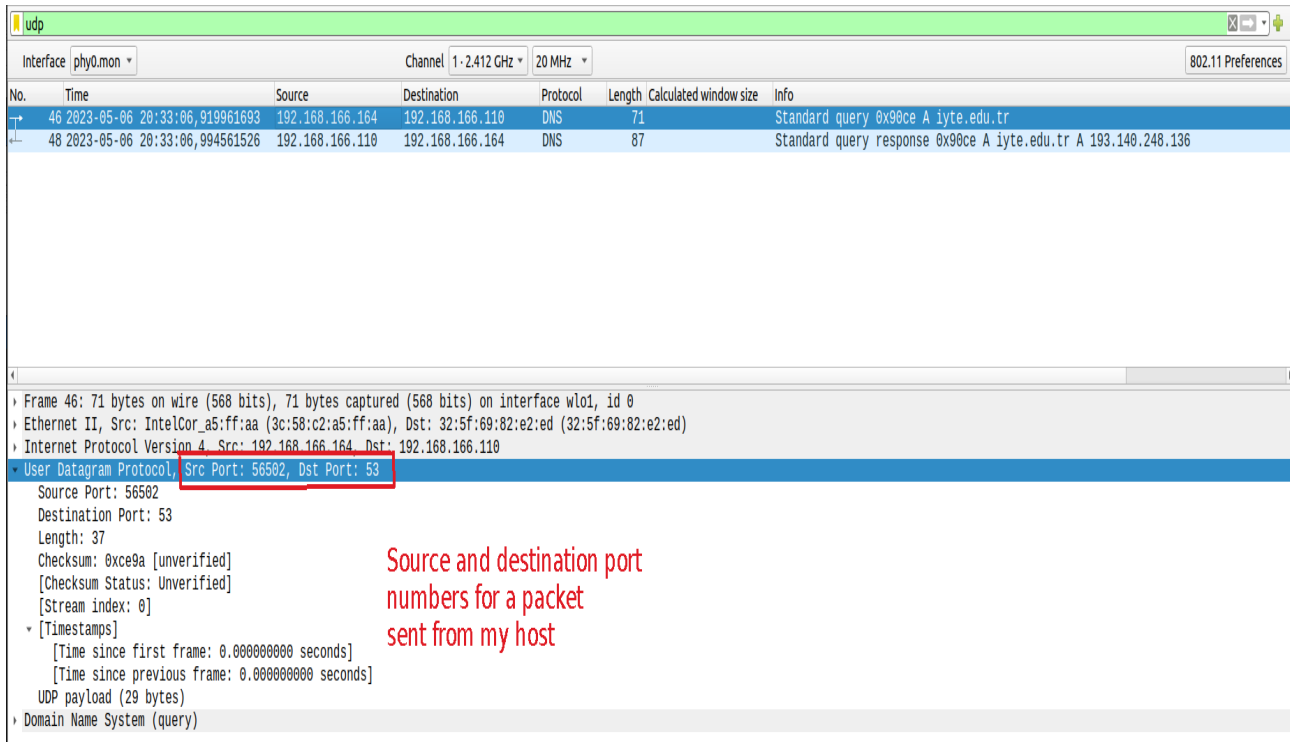


Figure 26: Source and destination port numbers for sent packet

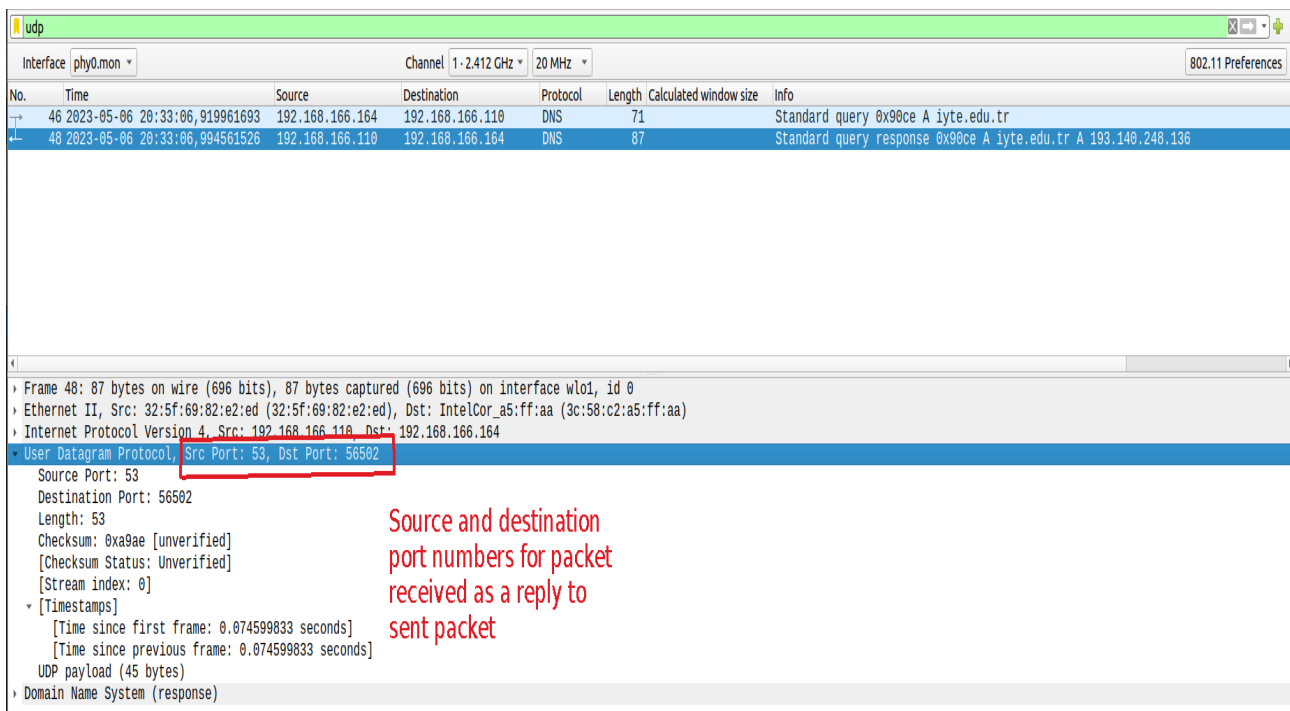


Figure 27: Source and destination port numbers for received packet as a reply

Q3-) UDP header size is 8 bytes and UDP payload size is 29 bytes for packet sent from my host. For the received packet header size is same which is 8 bytes and payload size is 45 bytes. UDP header consists of 4 fields, 16 bits for source port number, 16 bits for destination port number, 16 bits representing total length (header plus data) and 16 bits for checksum.

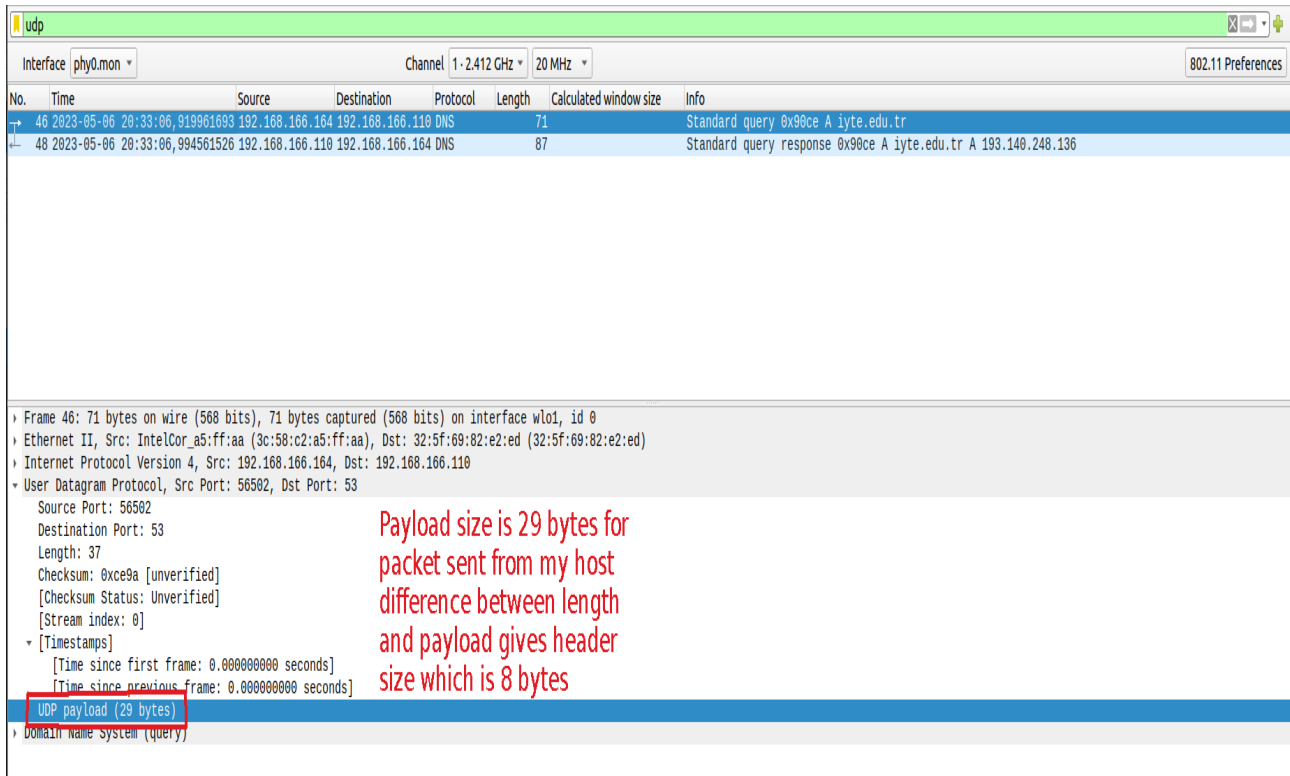


Figure 28: Header and Payload size for sent packet

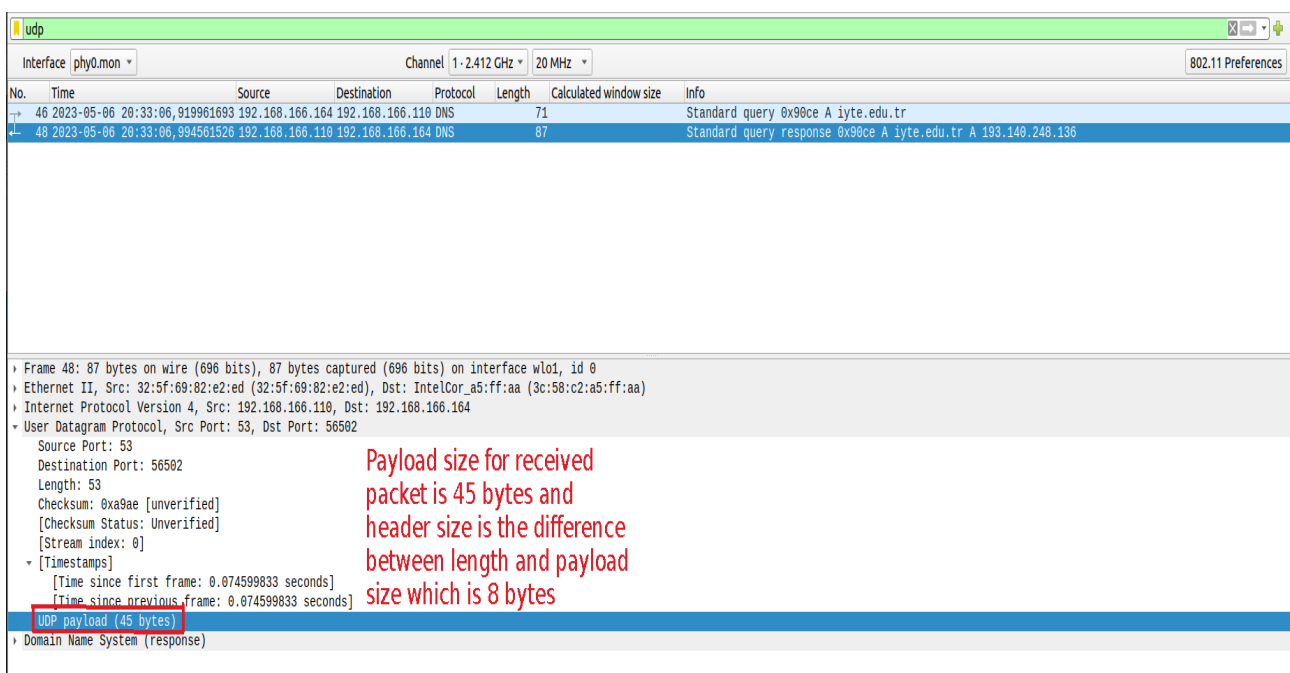
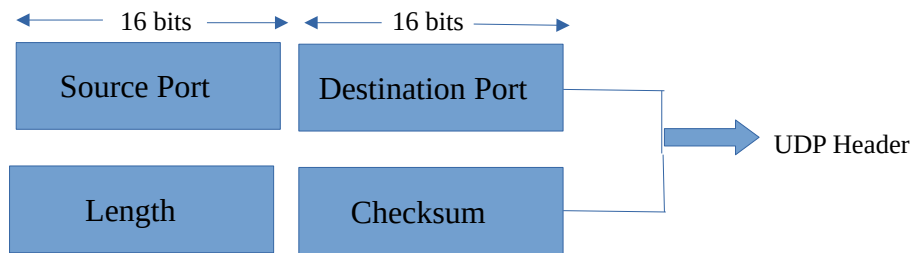


Figure 29: Header and Payload size for received packet

-UDP Header Structure-



Q4-) Checksum value of the packet sent from my host in hexadecimal notation is ce9a (1100111010011010 in binary). Checksum value of the of the packet received as a reply in hexadecimal notation is a9ae (1010100110101110 in binary). Packets sent via UDP may become corrupted while transmission. To check whether information is accurate or not checksum values are used.

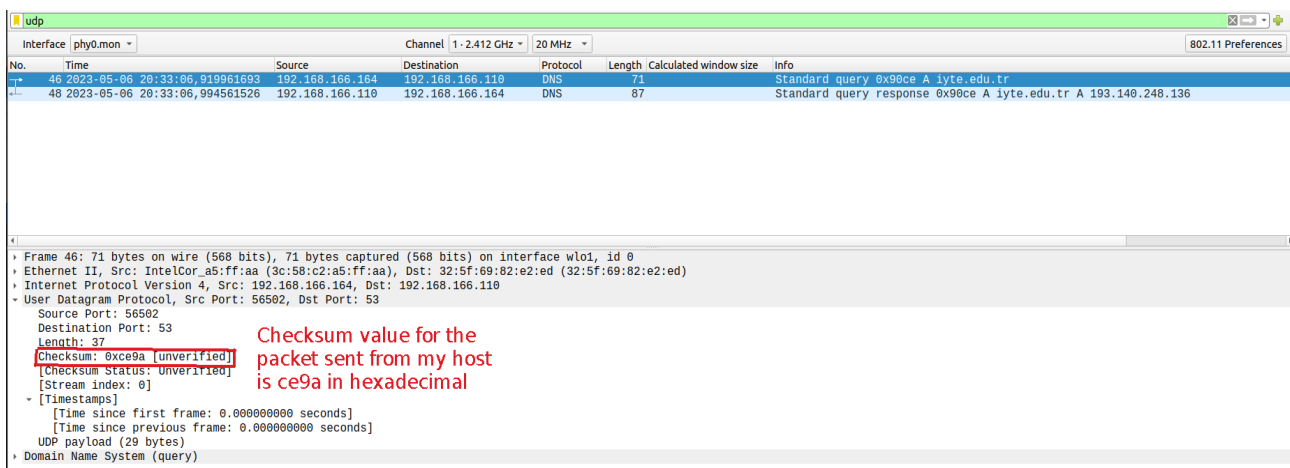


Figure 30: Checksum value for the packet sent from my host

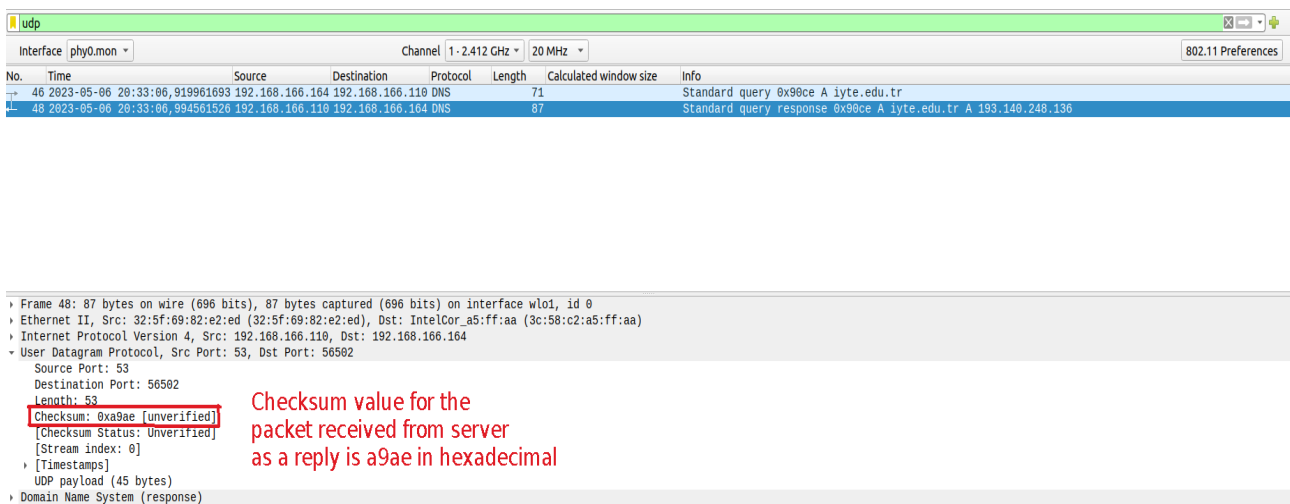


Figure 31: Checksum value for the packet received from iyte.edu.tr