Comp416 Report 2 - Wireshark

Batuhan Arat

68665

Part 1.1 UDP Experiment

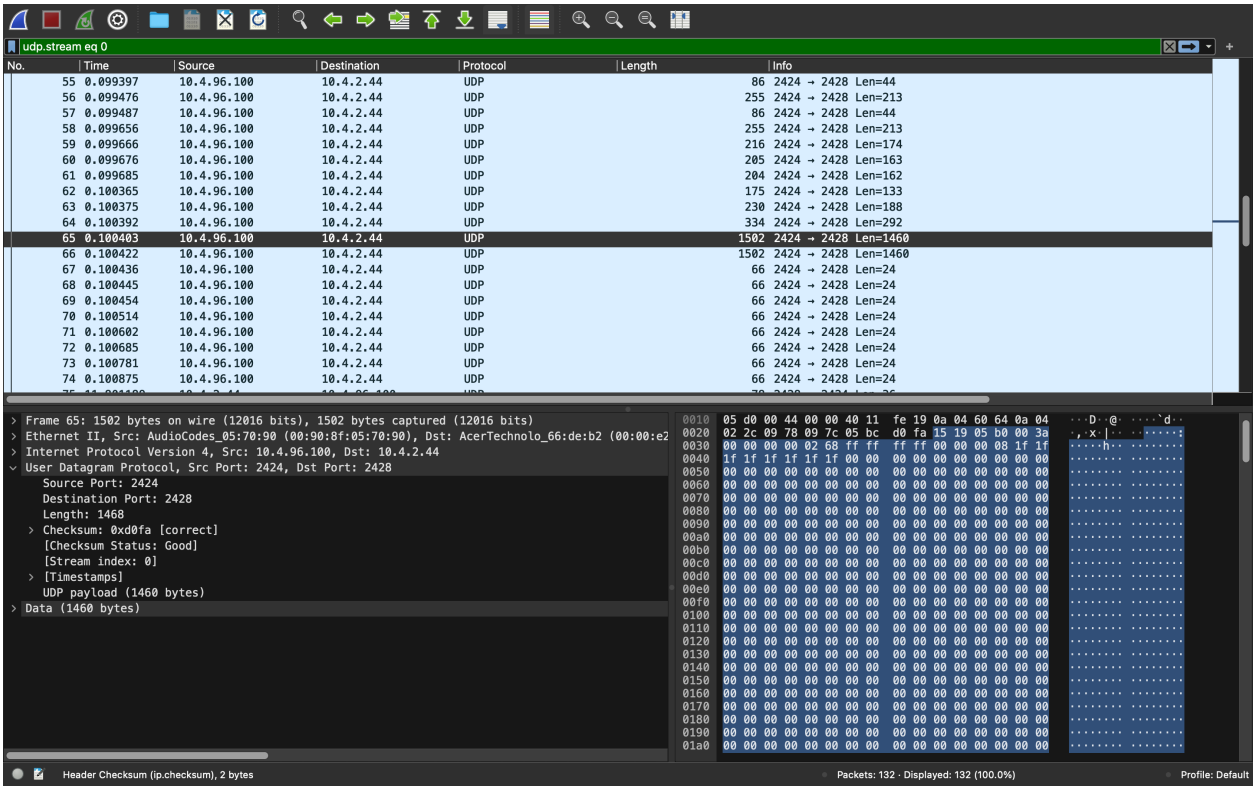


Figure 1

Here is the packet that corresponding the last two digit of my id. 68665 ‘s last two number is 65 so i check the 65.

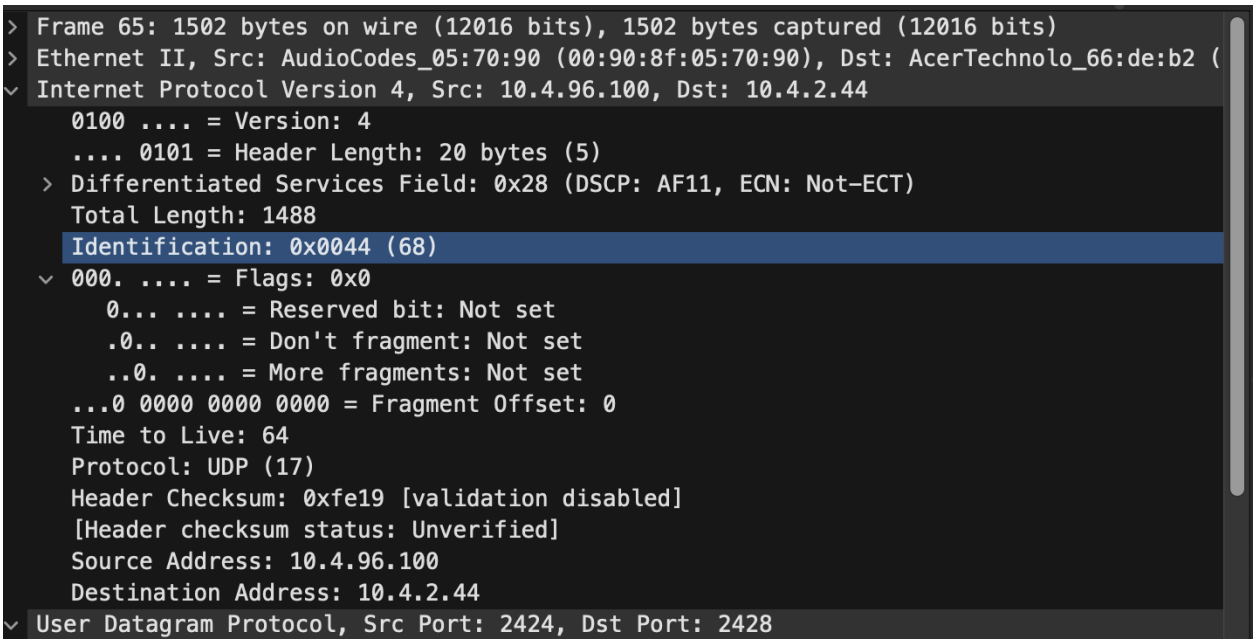


Figure 2

Part 1:

(Figure 2)

Time to live : 64

TTL is a mechanism that limits the lifespan of data in a network. If the TTL value reaches zero before the packet reaches its destination, the packet will be discarded to prevent it from circulating indefinitely.

Part 2:

(Figure 1)

Stream index = 0

The Stream Index column displays a unique number for each stream.Stream indexes are Wireshark-internal. It just uses a number to uniquely identify a TCP stream.

The stream index can help you to identify and follow a specific conversation or stream of packets between two endpoints in a network.

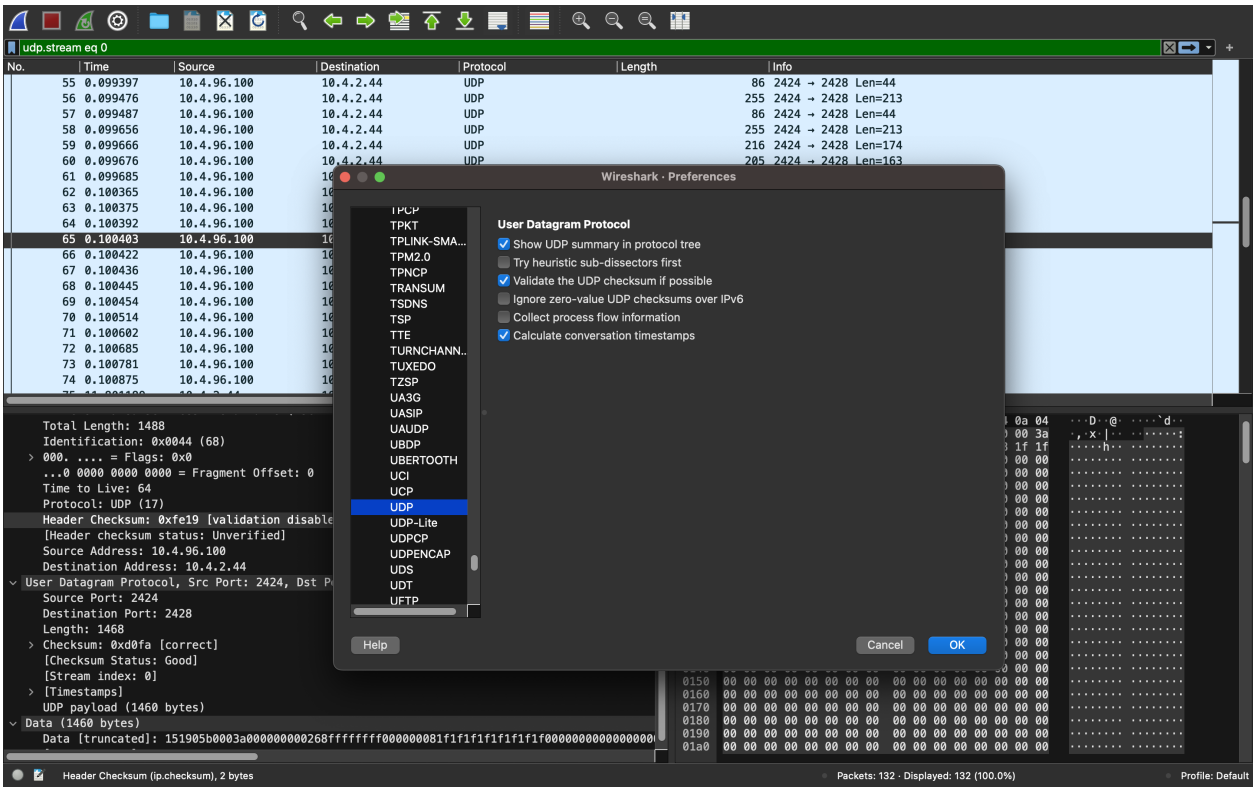
Part 3:

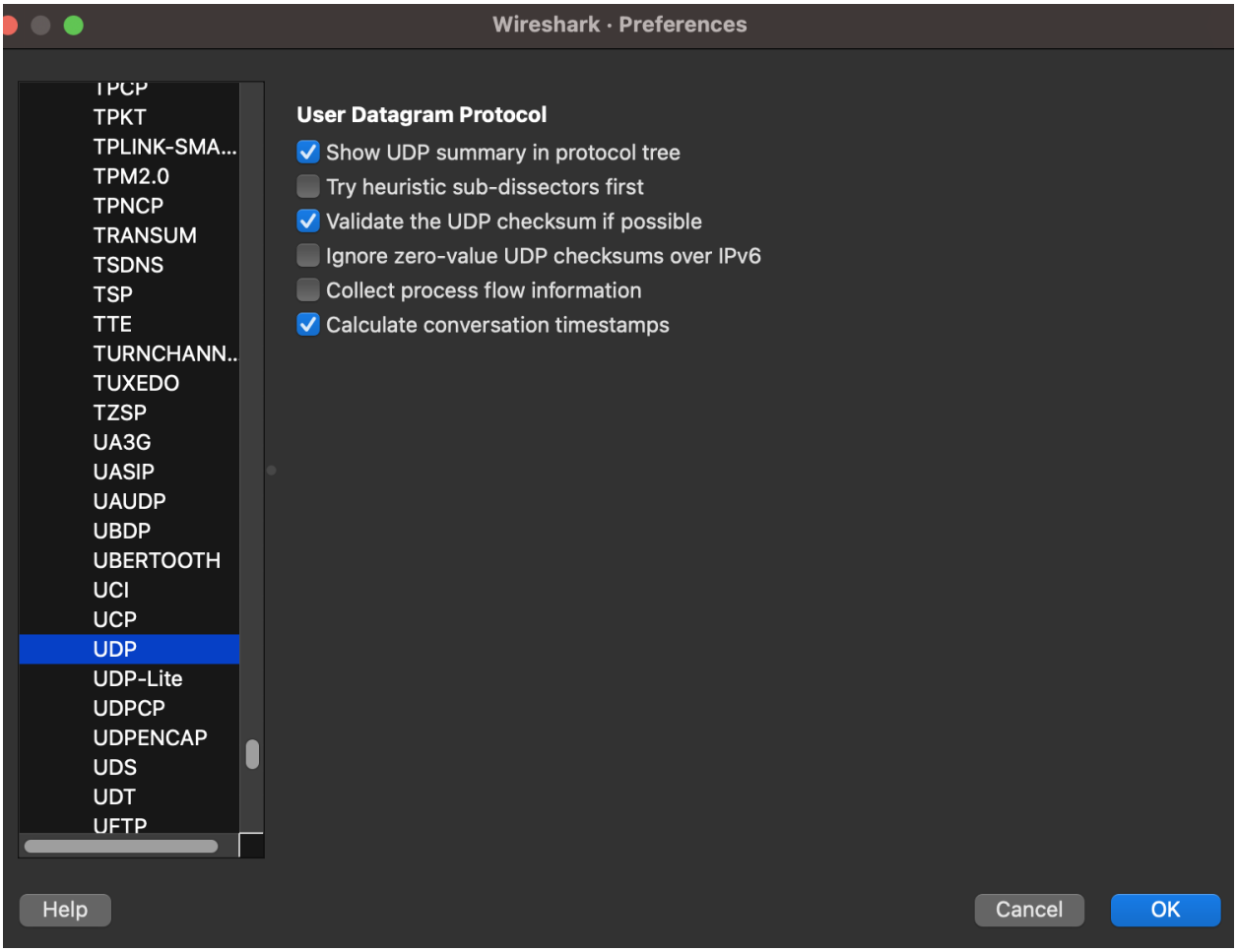
(Figure 2)

The checksum is a mathematical calculation used to ensure the integrity of the data in the packet. It's computed as the sum of all the 16-bit words in the header and data, with any overflow bits added back to the result. If the values was different than data is corrupted.

TCP also has a checksum mechanism. The difference between UDP and TCP in terms of data reliability is, UDP only says this data is corrupted, does not try to retransmit the corrupted packet, while TCP ensures the data transfer reliability therefore it retransmit the corrupted packet.

To activate the checksum validation i did this.





Part 4:

(Figure 2)

ip.flags.rb

0... .... = Reserved bit: Not set

This is the first bit and it is always set to 0. The purpose of this bit is reserved for future use. However, in practice, it has remained unused and is expected to be 0 in all IPv4 packets. If this bit is set to 1, it's typically considered a malformed packet.

Part 5:

(Figure 2)

The length field specifies the number of bytes in the UDP segment (header plus data).

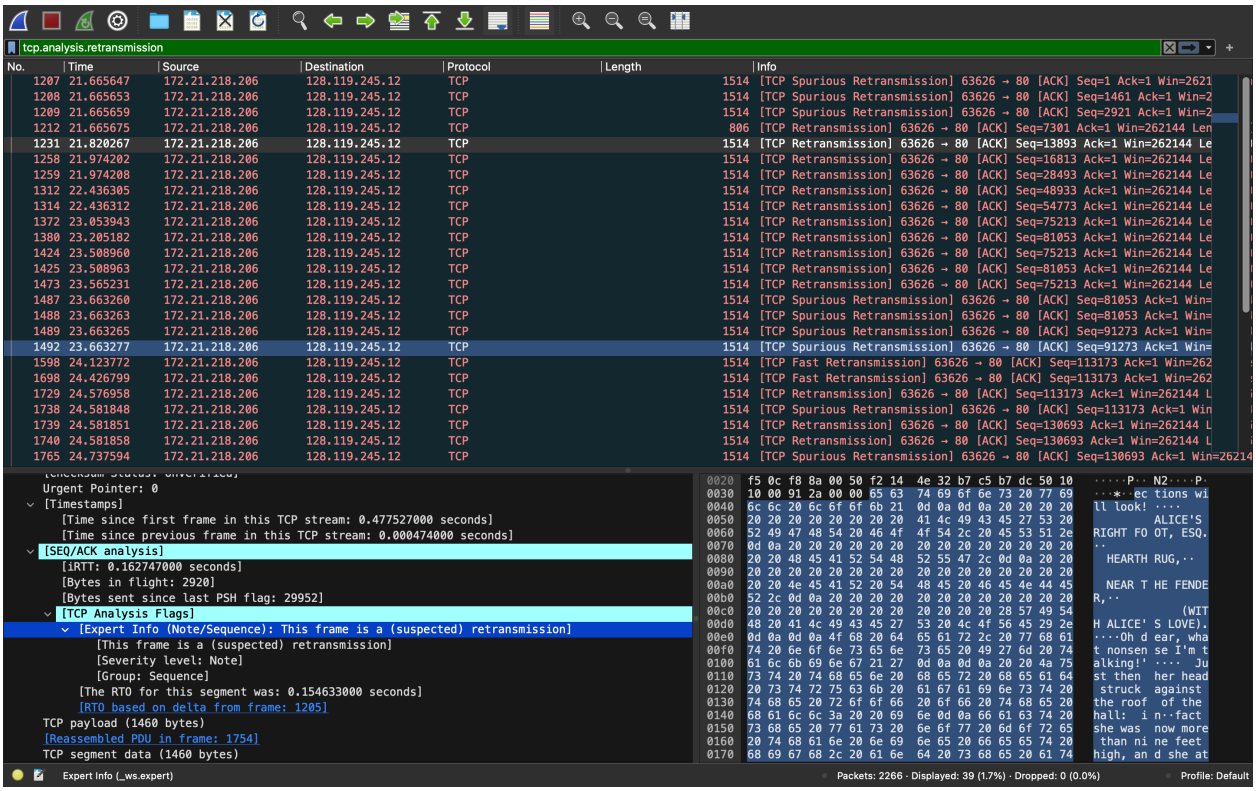
An explicit length value is needed since the size of the data field may differ from one UDP segment to the next.This length includes the UDP header (usually 8 bytes) and the UDP payload. The length of UDP payload for selected packet is 32 bytes. 1468 bytes - 8 bytes = 1460 bytes.

There is another length information which is written in the packet window. That can be seen in Figure 1. It is set to 1502. This is the total size of the packet as it was captured by Wireshark, including all headers (Ethernet, IP, TCP/UDP) and the data payload.It represents the entire packet size as it exists on the network.

Part 1.2. TCP Experiment

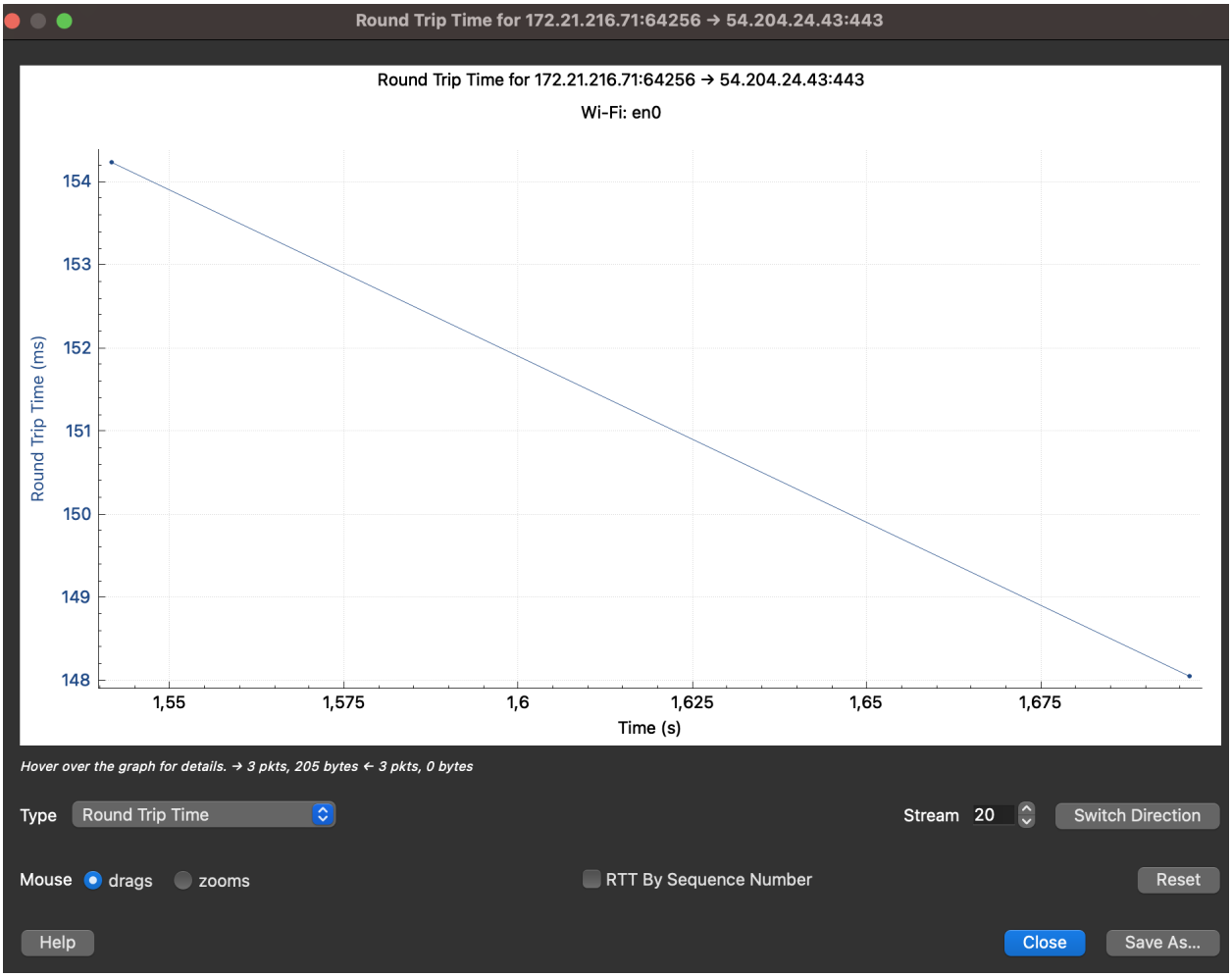
Part 1:

Using tcp.analysis.retransmission query on the filter section we can see the TCP Retransmission from the info.This is a post filter, because it is executed after the capture. In the packet details There is a section of [Seq/ack analysis] → [Tcp analysis flags] →[Expert Info (Note/Sequence)]. In this header it says This frame is a [suspected] retransmission. This is a clear indication of segment loss.



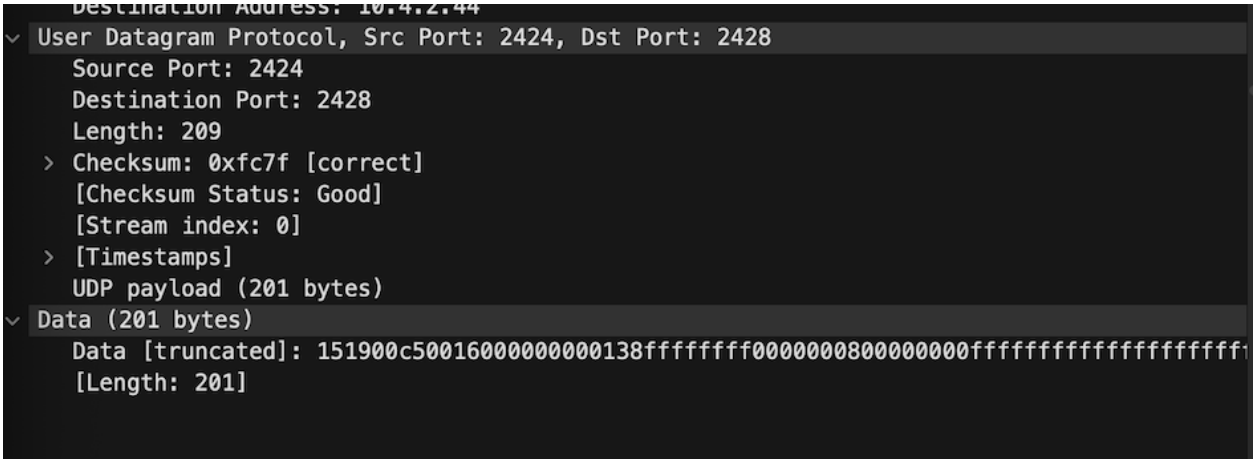
Part 2:

Round trip times are differ between 148- 154 ms as we can see the graph from below. There are variations present.

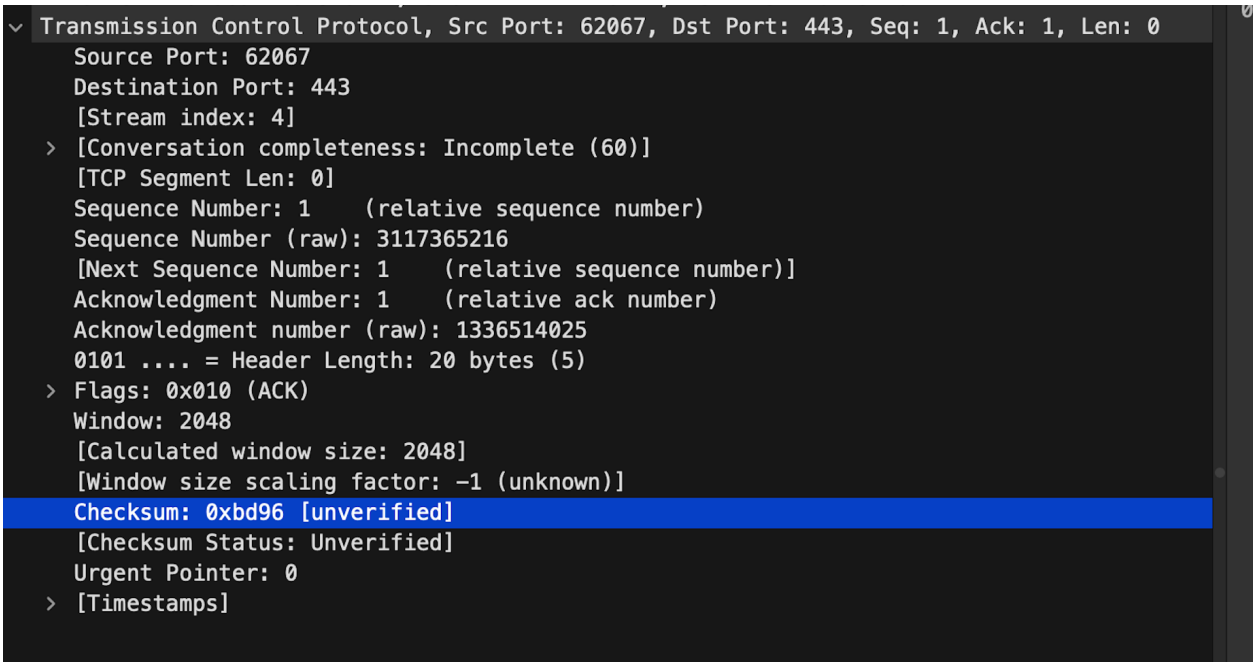


Part 3:

This is udp



This is tcp

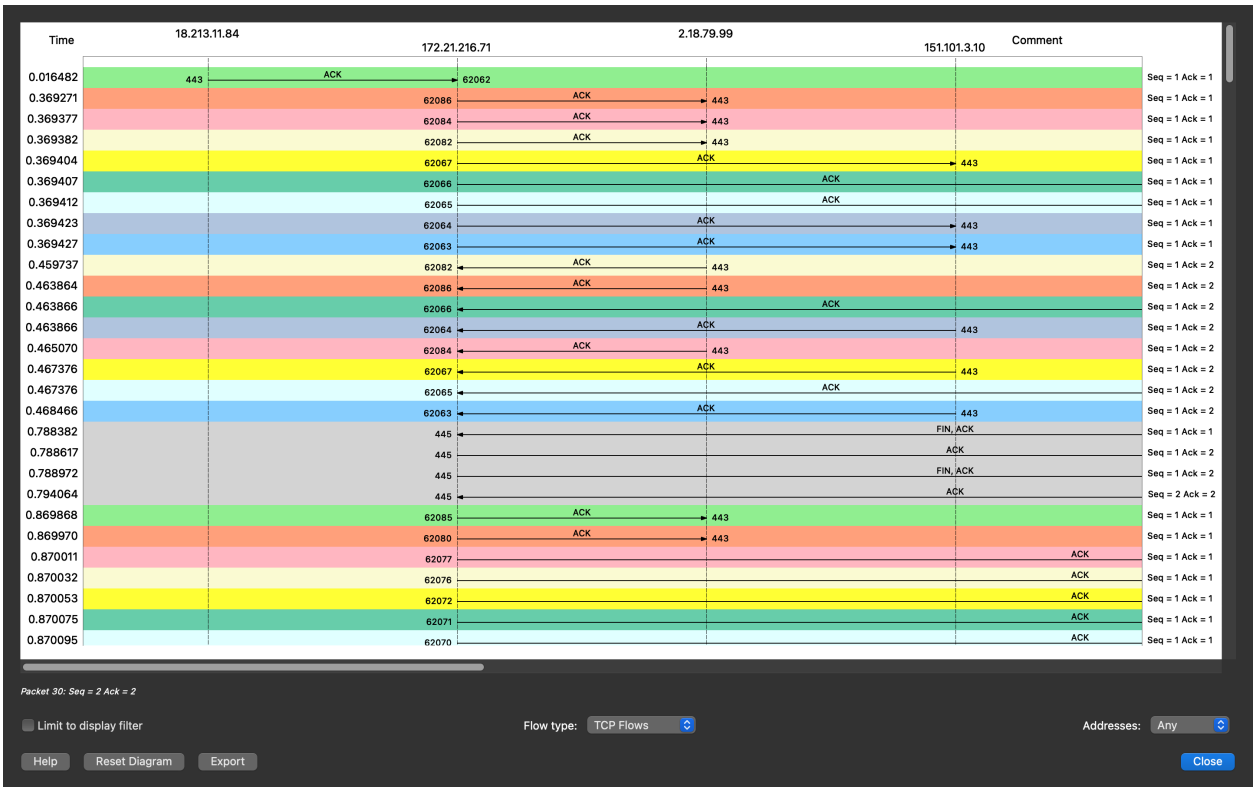


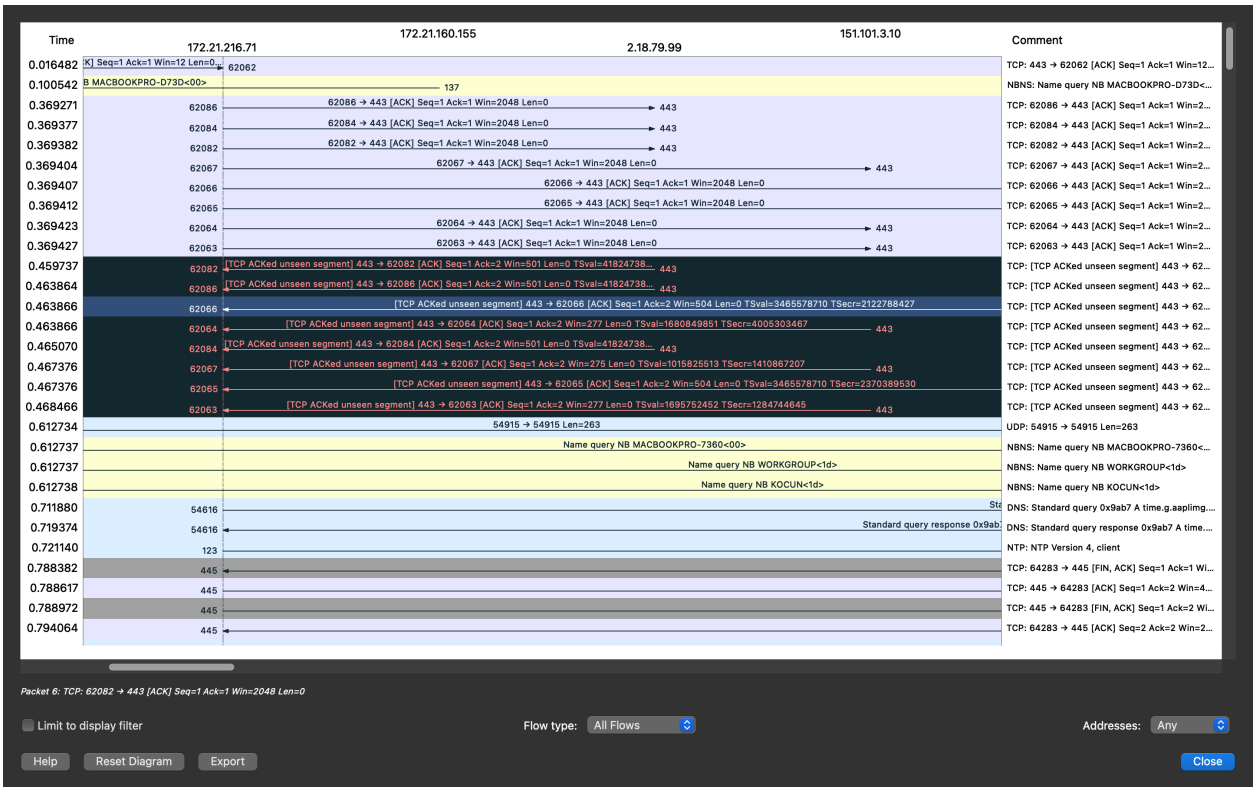
Window, acknowledgment and sequence numbers are present in the tcp which is not present in the udp. I explain what they signify and their purpose at the next part.

Part 4:

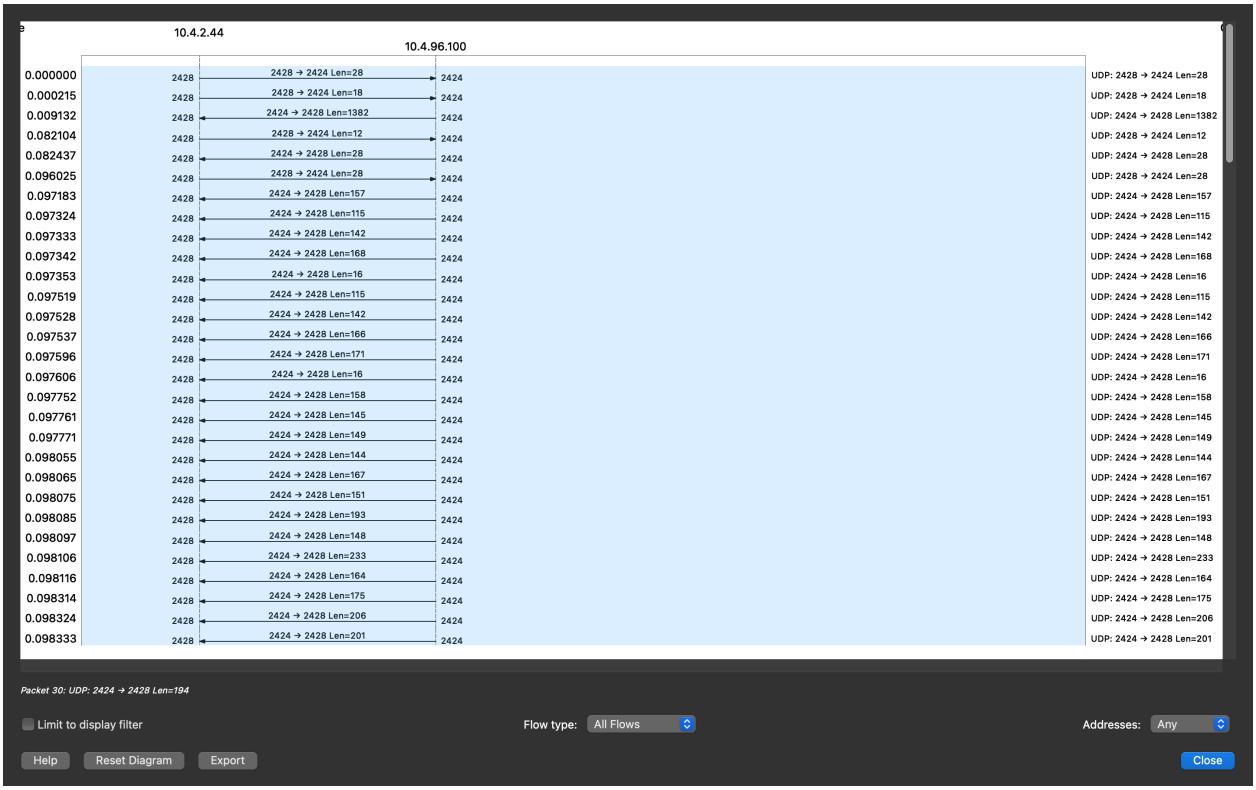
Those are the Flow Graphs from the Statistic Menu. First two of them is taken from the alice example, the last one is from udp_sample example.

Those are TCP flow graphs


Time
18.213.11.84
172.21.216.71
2.18.79.99
151.101.3.10
Comment
0.016482 443 ACK 62062 Seq = 1 Ack = 1
0.369271 62086 ACK 443 Seq = 1 Ack = 1
0.369377 62084 ACK 443 Seq = 1 Ack = 1
0.369382 62082 ACK 443 Seq = 1 Ack = 1
0.369404 62067 ACK 443 Seq = 1 Ack = 1
0.369407 62066 ACK Seq = 1 Ack = 1
0.369412 62065 ACK Seq = 1 Ack = 1
0.369423 62064 ACK 443 Seq = 1 Ack = 1
0.369427 62063 ACK 443 Seq = 1 Ack = 1
0.459737 62082 ACK 443 Seq = 1 Ack = 2
0.463864 62086 ACK 443 Seq = 1 Ack = 2
0.463866 62066 ACK Seq = 1 Ack = 2
0.463866 62064 ACK 443 Seq = 1 Ack = 2
0.465070 62084 ACK 443 Seq = 1 Ack = 2
0.467376 62067 ACK 443 Seq = 1 Ack = 2
0.467376 62065 ACK Seq = 1 Ack = 2
0.468466 62063 ACK 443 Seq = 1 Ack = 2
0.788382 445 FIN, ACK Seq = 1 Ack = 1
0.788617 445 ACK Seq = 1 Ack = 2
0.788972 445 FIN, ACK Seq = 1 Ack = 2
0.794064 445 ACK Seq = 2 Ack = 2
0.869868 62085 ACK 443 Seq = 1 Ack = 1
0.869970 62080 ACK 443 Seq = 1 Ack = 1
0.870011 62077 ACK Seq = 1 Ack = 1
0.870032 62076 ACK Seq = 1 Ack = 1
0.870053 62072 ACK Seq = 1 Ack = 1
0.870075 62071 ACK Seq = 1 Ack = 1
0.870095 62070 ACK Seq = 1 Ack = 1
Packet 30: Seq = 2 Ack = 2
Limit to display filter
Flow type: TCP Flows
Addresses: Any
Help
Reset Diagram
Export
Close


Time
172.21.216.71
172.21.160.155
2.18.79.99
151.101.3.10
Comment
0.016482 62062 TCP: 443 → 62062 [ACK] Seq=1 Ack=1 Win=12...
0.100542 137 NBNS: Name query NB MACBOOKPRO-D73D<...
0.369271 62086 62086 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 443 TCP: 62086 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.369377 62084 62084 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 443 TCP: 62084 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.369382 62082 62082 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 443 TCP: 62082 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.369404 62067 62067 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 443 TCP: 62067 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.369407 62066 62066 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 TCP: 62066 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.369412 62065 62065 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 TCP: 62065 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.369423 62064 62064 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 443 TCP: 62064 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.369427 62063 62063 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0 443 TCP: 62063 → 443 [ACK] Seq=1 Ack=1 Win=2...
0.459737 62082 [TCP ACKed unseen segment] 443 → 62082 [ACK] Seq=1 Ack=2 Win=501 Len=0 TSval=41824738... 443 TCP: [TCP ACKed unseen segment] 443 → 62...
0.463864 62086 [TCP ACKed unseen segment] 443 → 62086 [ACK] Seq=1 Ack=2 Win=501 Len=0 TSval=41824738... 443 TCP: [TCP ACKed unseen segment] 443 → 62...
0.463866 62066 [TCP ACKed unseen segment] 443 → 62066 [ACK] Seq=1 Ack=2 Win=504 Len=0 TSval=3465578710 TSecr=2122788427 TCP: [TCP ACKed unseen segment] 443 → 62...
0.463866 62064 [TCP ACKed unseen segment] 443 → 62064 [ACK] Seq=1 Ack=2 Win=277 Len=0 TSval=1680849851 TSecr=4005303467 443 TCP: [TCP ACKed unseen segment] 443 → 62...
0.465070 62084 [TCP ACKed unseen segment] 443 → 62084 [ACK] Seq=1 Ack=2 Win=501 Len=0 TSval=41824738... 443 TCP: [TCP ACKed unseen segment] 443 → 62...
0.467376 62067 [TCP ACKed unseen segment] 443 → 62067 [ACK] Seq=1 Ack=2 Win=275 Len=0 TSval=1015825513 TSecr=1410867207 443 TCP: [TCP ACKed unseen segment] 443 → 62...
0.467376 62065 [TCP ACKed unseen segment] 443 → 62065 [ACK] Seq=1 Ack=2 Win=504 Len=0 TSval=3465578710 TSecr=2370389530 TCP: [TCP ACKed unseen segment] 443 → 62...
0.468466 62063 [TCP ACKed unseen segment] 443 → 62063 [ACK] Seq=1 Ack=2 Win=277 Len=0 TSval=1695752452 TSecr=1284744645 443 TCP: [TCP ACKed unseen segment] 443 → 62...
0.612734 54915 → 54915 Len=263 UDP: 54915 → 54915 Len=263
0.612737 Name query NB MACBOOKPRO-7360<00> NBNS: Name query NB MACBOOKPRO-7360<...
0.612737 Name query NB WORKGROUP<1d> NBNS: Name query NB WORKGROUP<1d>
0.612738 Name query NB KOCUN<1d> NBNS: Name query NB KOCUN<1d>
0.711880 54616 DNS: Standard query 0x9ab7 A time.g.aaplimg....
0.719374 54616 Standard query response 0x9ab7 DNS: Standard query response 0x9ab7 A time....
0.721140 123 NTP: NTP Version 4, client
0.788382 445 TCP: 64283 → 445 [FIN, ACK] Seq=1 Ack=1 Wi...
0.788617 445 TCP: 445 → 64283 [ACK] Seq=1 Ack=2 Win=4...
0.788972 445 TCP: 445 → 64283 [FIN, ACK] Seq=1 Ack=2 Wi...
0.794064 445 TCP: 64283 → 445 [ACK] Seq=2 Ack=2 Win=2...
Packet 6: TCP: 62082 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0
Limit to display filter
Flow type: All Flows
Addresses: Any
Help
Reset Diagram
Export
Close

This is UDP flow graph



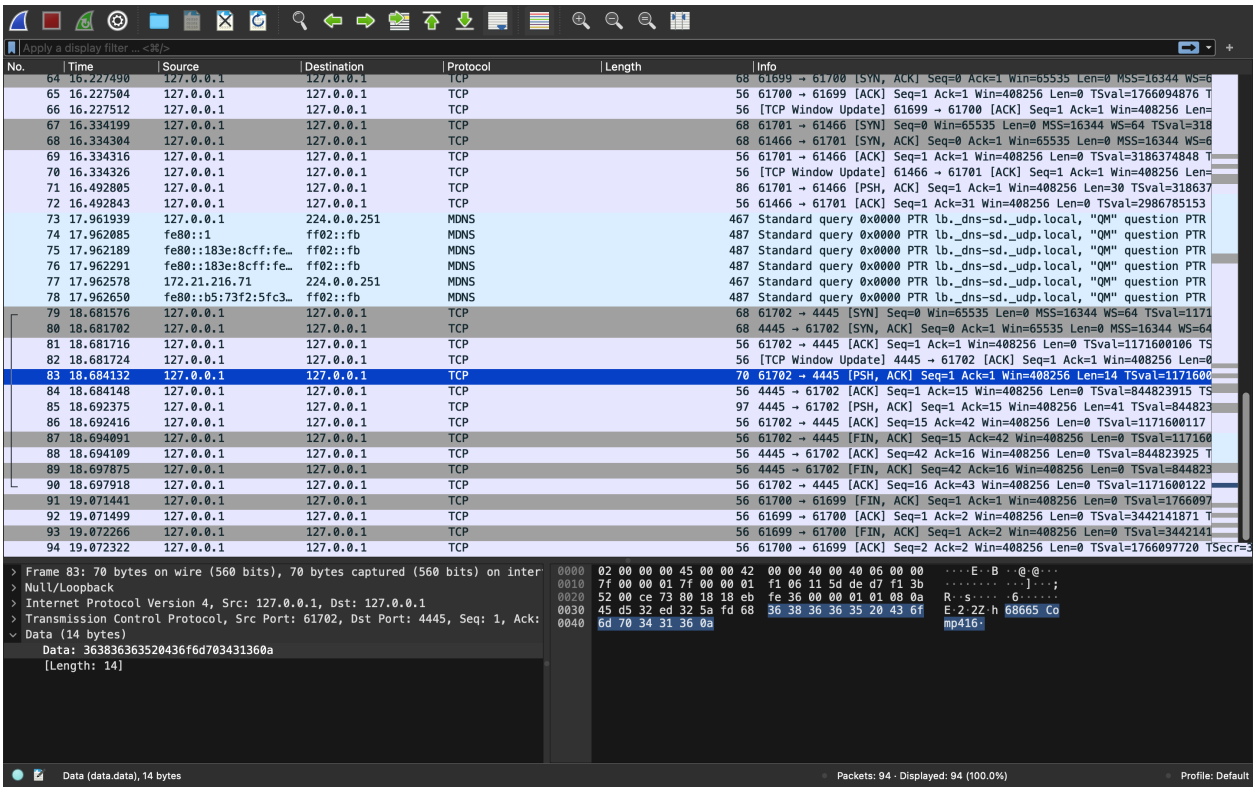
In TCP, we can see that ; seq , ack and win that is not present in the udp. Seq is sequence number which identifies the order of bytes sent over. Ack is the acknowledgment number which confirms the receipt of bytes by the receiver.With using those numbers, TCP knows which data has been received, and know which data needs to be resending.This is the mechanism that makes TCP reliable. Win is a TCP Window size, which determines the amount of data that cant be sent during the connection before receiving an acknowledgment that the data has been successfully received.Those are parts of TCP’s flow control mechanism which is not present in the UDP. Udp is connectionless protocol and does not have built-in mechanism for flow control, congestion avoidance and data reliabilty

Part 2.1 TCP vs SSL Experiments

Part1:

This is the screenshot from captured data of the insecured connection.

As we can see it, protocol is TCP.



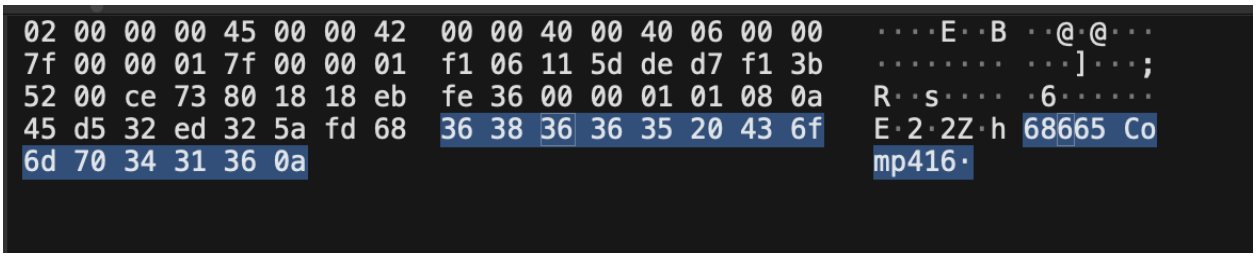
Part2:

When we look at the Packet 83 we can see the sent data within the sent packets.This is the TCP example

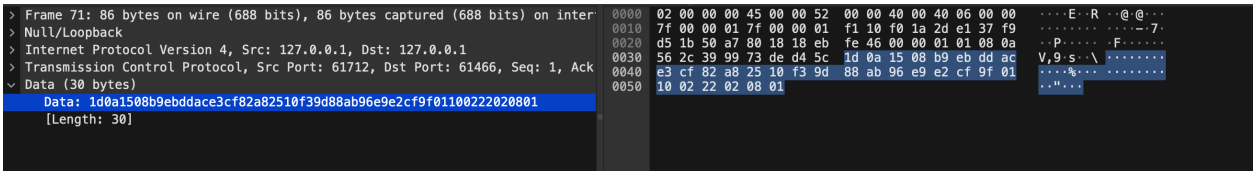
We can also see at the packet 71 that data is sent. This is the SSL example.

Part3:

In the TCP example, we can see the packet contents. Because it use insecure server which is not encrpyted.



But in the SSL example the data content is encrypted as a secure connection should be.



Part4:

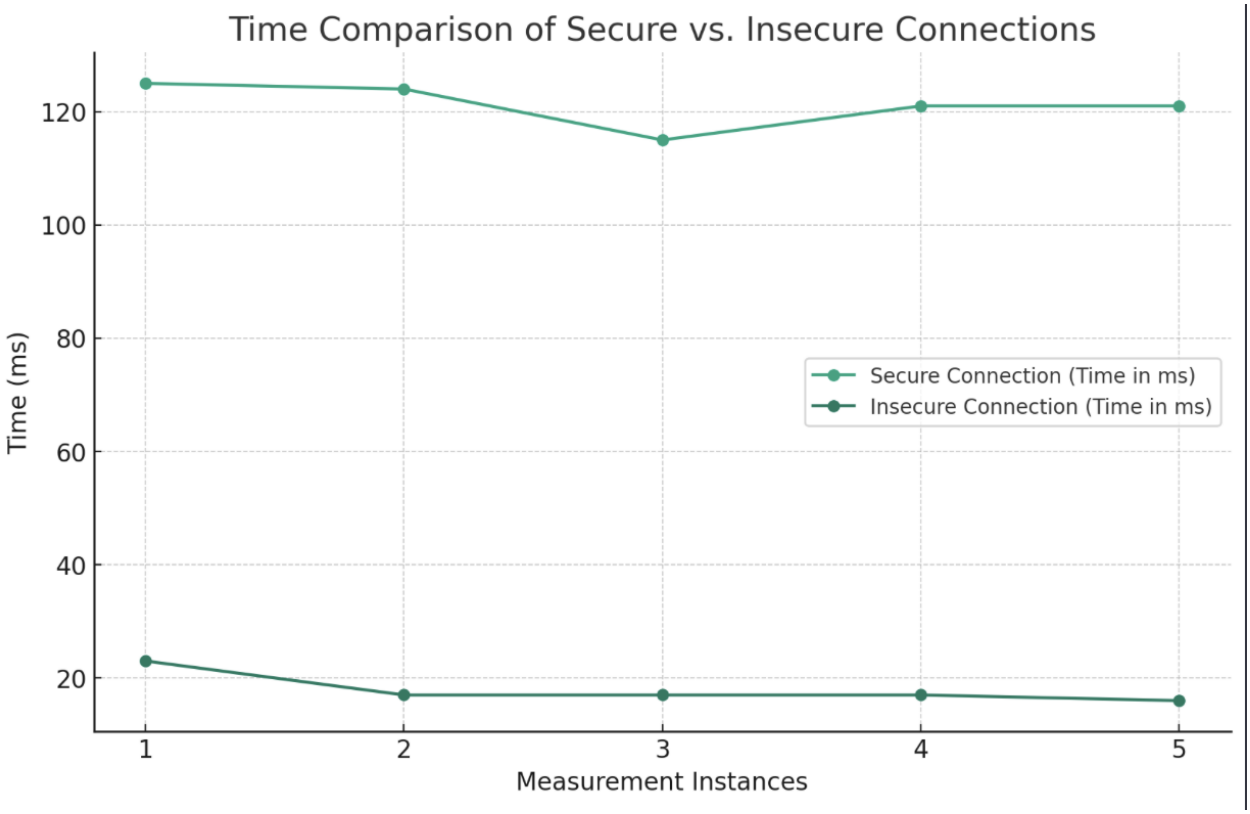
For the Secure Connection 5 different delay data in milliseconds are;

- 125
- 124
- 115
- 121
- 121

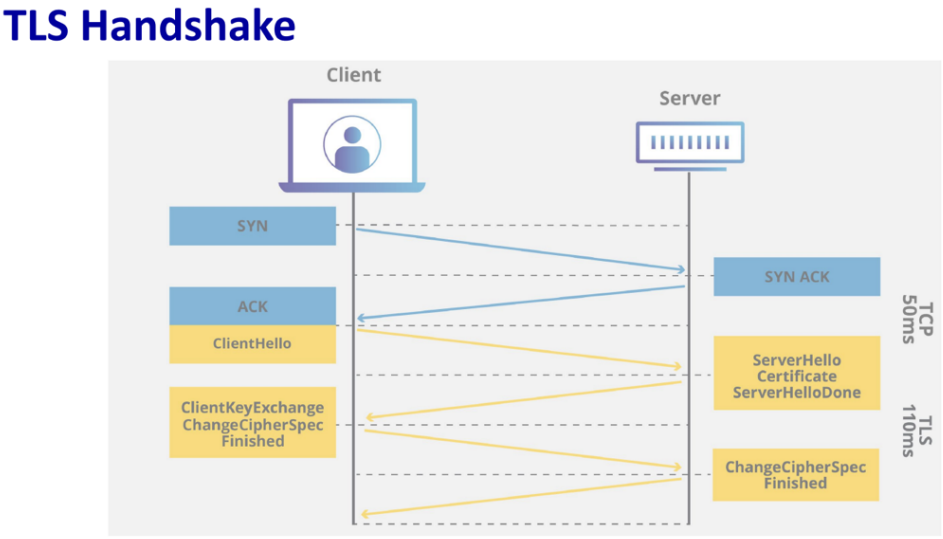
For the Insecure Connection 5 different delay data in milliseconds are;

- 23
- 17
- 17
- 17
- 16

Insecure connections is faster because it doesn't do any encryption on the data.



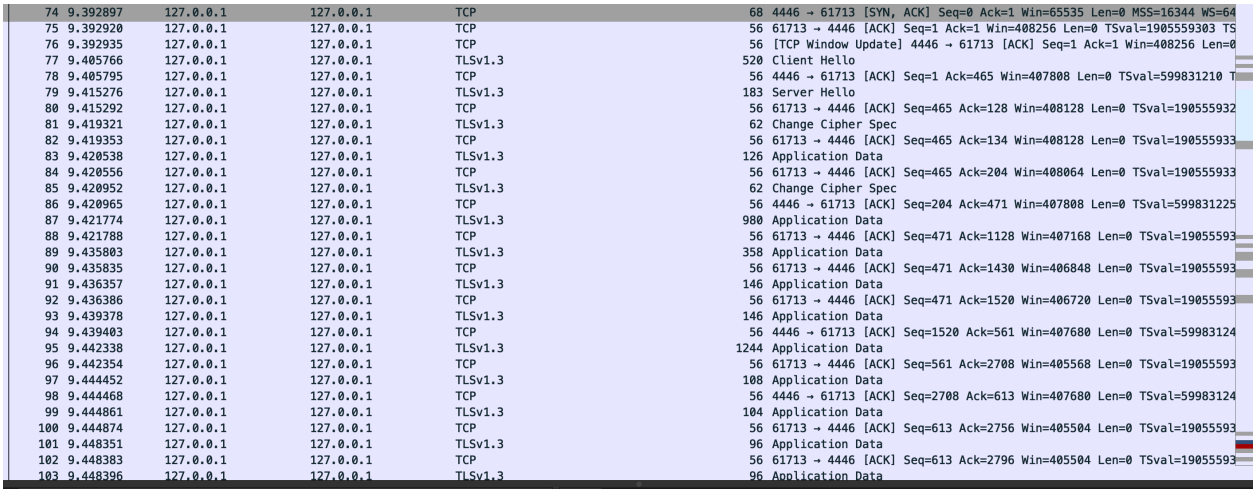
Part5:



The packets that are starting with Client Hello and Server Hello packets are from the TLS handshake. As we can see the diagram, initial phase is synchronization packet sent from the

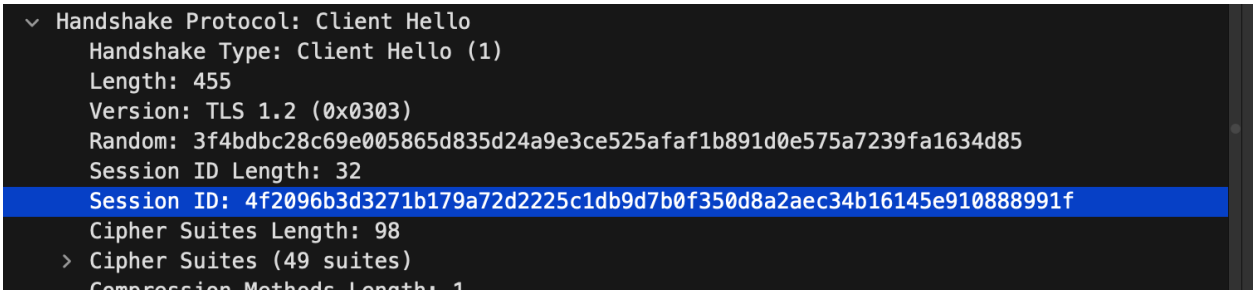
client to server, and acknowledgment of synchronization at the server, and returning to the client. This is the same for the all TCP connection.

When TCP connection is established, TLS handshake begins with ClientHello message to the server. We can see it from the packet's info as well.

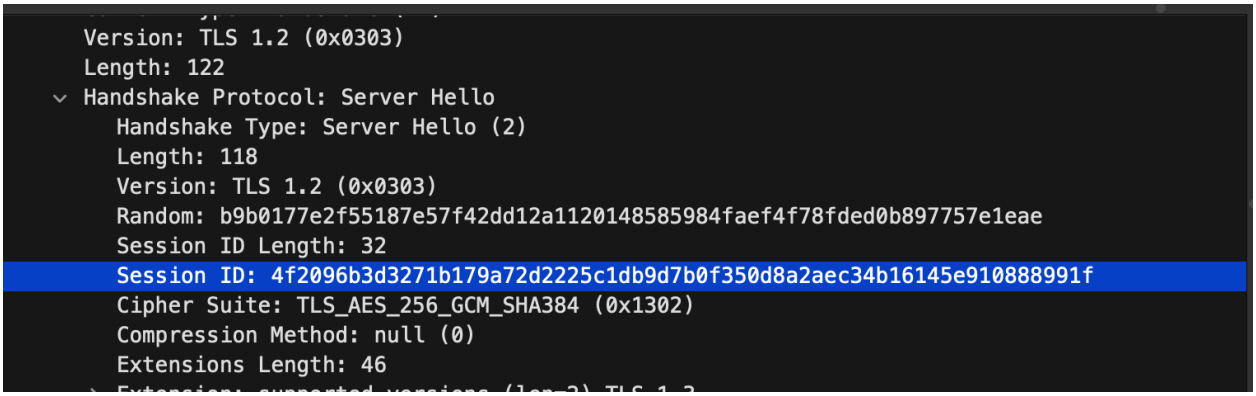


74	9.392897	127.0.0.1	127.0.0.1	TCP	68	4446 → 61713 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64
75	9.392920	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=1905559303 TS
76	9.392935	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 4446 → 61713 [ACK] Seq=1 Ack=1 Win=408256 Len=0
77	9.405766	127.0.0.1	127.0.0.1	TLSv1.3	520	Client Hello
78	9.405795	127.0.0.1	127.0.0.1	TCP	56	4446 → 61713 [ACK] Seq=1 Ack=465 Win=407808 Len=0 TSval=599831210 T
79	9.415276	127.0.0.1	127.0.0.1	TLSv1.3	183	Server Hello
80	9.415292	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=465 Ack=128 Win=408128 Len=0 TSval=190555932
81	9.419321	127.0.0.1	127.0.0.1	TLSv1.3	62	Change Cipher Spec
82	9.419353	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=465 Ack=134 Win=408128 Len=0 TSval=190555933
83	9.420538	127.0.0.1	127.0.0.1	TLSv1.3	126	Application Data
84	9.420556	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=465 Ack=204 Win=408064 Len=0 TSval=190555933
85	9.420952	127.0.0.1	127.0.0.1	TLSv1.3	62	Change Cipher Spec
86	9.420965	127.0.0.1	127.0.0.1	TCP	56	4446 → 61713 [ACK] Seq=204 Ack=471 Win=407808 Len=0 TSval=599831225
87	9.421774	127.0.0.1	127.0.0.1	TLSv1.3	980	Application Data
88	9.421788	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=471 Ack=1128 Win=407168 Len=0 TSval=19055593
89	9.435803	127.0.0.1	127.0.0.1	TLSv1.3	358	Application Data
90	9.435835	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=471 Ack=1430 Win=406848 Len=0 TSval=19055593
91	9.436357	127.0.0.1	127.0.0.1	TLSv1.3	146	Application Data
92	9.436386	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=471 Ack=1520 Win=406720 Len=0 TSval=19055593
93	9.439378	127.0.0.1	127.0.0.1	TLSv1.3	146	Application Data
94	9.439403	127.0.0.1	127.0.0.1	TCP	56	4446 → 61713 [ACK] Seq=1520 Ack=561 Win=407680 Len=0 TSval=59983124
95	9.442338	127.0.0.1	127.0.0.1	TLSv1.3	1244	Application Data
96	9.442354	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=561 Ack=2708 Win=405568 Len=0 TSval=19055593
97	9.444452	127.0.0.1	127.0.0.1	TLSv1.3	108	Application Data
98	9.444468	127.0.0.1	127.0.0.1	TCP	56	4446 → 61713 [ACK] Seq=2708 Ack=613 Win=407680 Len=0 TSval=59983124
99	9.444861	127.0.0.1	127.0.0.1	TLSv1.3	104	Application Data
100	9.444874	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=613 Ack=2756 Win=405504 Len=0 TSval=19055593
101	9.448351	127.0.0.1	127.0.0.1	TLSv1.3	96	Application Data
102	9.448383	127.0.0.1	127.0.0.1	TCP	56	61713 → 4446 [ACK] Seq=613 Ack=2796 Win=405504 Len=0 TSval=19055593
103	9.448396	127.0.0.1	127.0.0.1	TLSv1.3	96	Application Data

Part6:



```
▾ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 455
    Version: TLS 1.2 (0x0303)
    Random: 3f4bdbc28c69e005865d835d24a9e3ce525afaf1b891d0e575a7239fa1634d85
    Session ID Length: 32
    Session ID: 4f2096b3d3271b179a72d2225c1db9d7b0f350d8a2aec34b16145e910888991f
    Cipher Suites Length: 98
  › Cipher Suites (49 suites)
```



```
  Version: TLS 1.2 (0x0303)
  Length: 122
▾ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 118
    Version: TLS 1.2 (0x0303)
    Random: b9b0177e2f55187e57f42dd12a1120148585984faef4f78fded0b897757e1eae
    Session ID Length: 32
    Session ID: 4f2096b3d3271b179a72d2225c1db9d7b0f350d8a2aec34b16145e910888991f
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Compression Method: null (0)
    Extensions Length: 46
```

It uses session id’s. Server generates as session ID during the handshake and give it to the client. While server recognize the session id, whenever client wants to resume a session it can do it by the session ID.

At the Wireshark, when we click the Client Hello message we can see a session id. The same session id is also present in the Server Hello. That means that client wants to resume this session so it will avoid sending the certificate for each connection to optimization.

Part 2.2. Creating Keystore and Trust Store

Here is the clear screenshot of the information of the generated certificated.