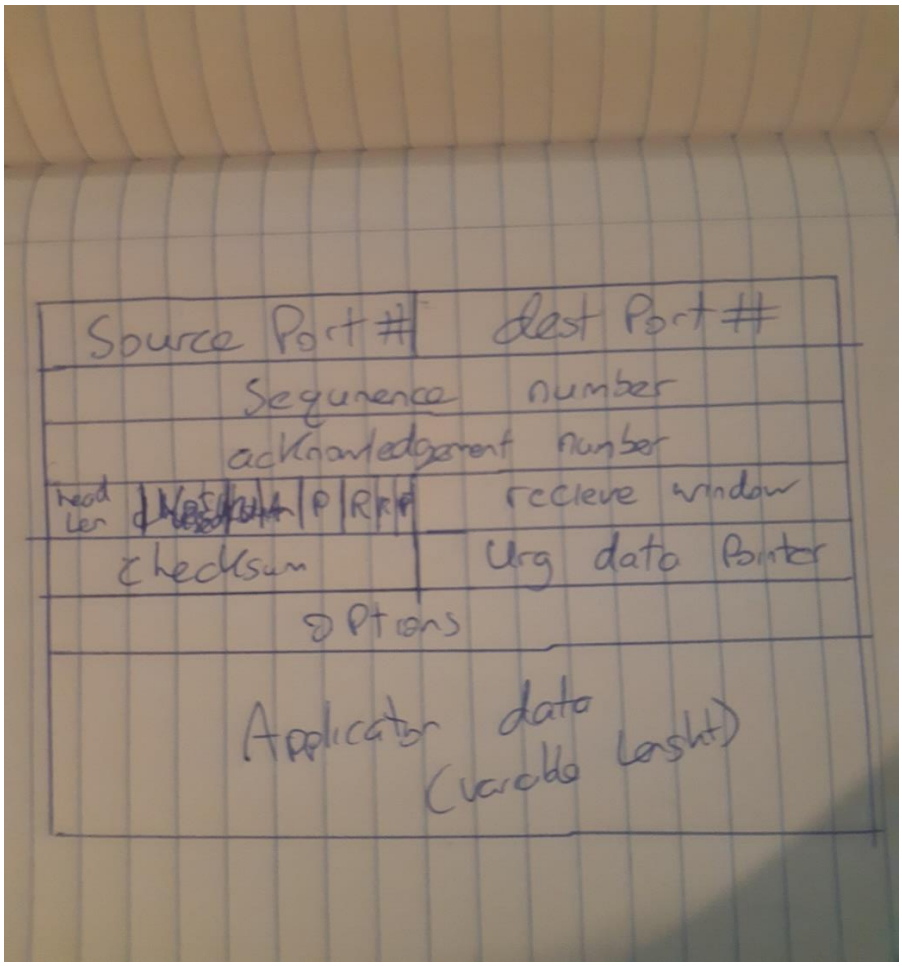
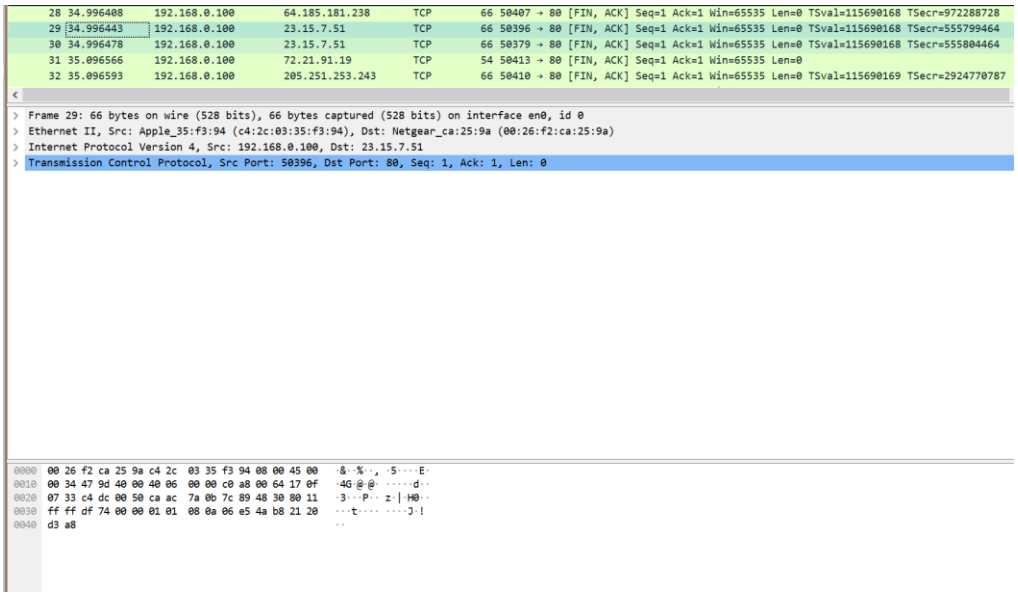
Lab 5

Q 1. TCP Header



WireShark of TCP:



Source Port: Port number used by the computer sending the TCP segment.

Destination Port: Port number used by the computer receiving the TCP segment.

Sequence Number: Used for segmenting data into TCP segments and reassembling on the other side.

Data Offset/head Len: Number of bytes into the TCP packet where the data can be found.

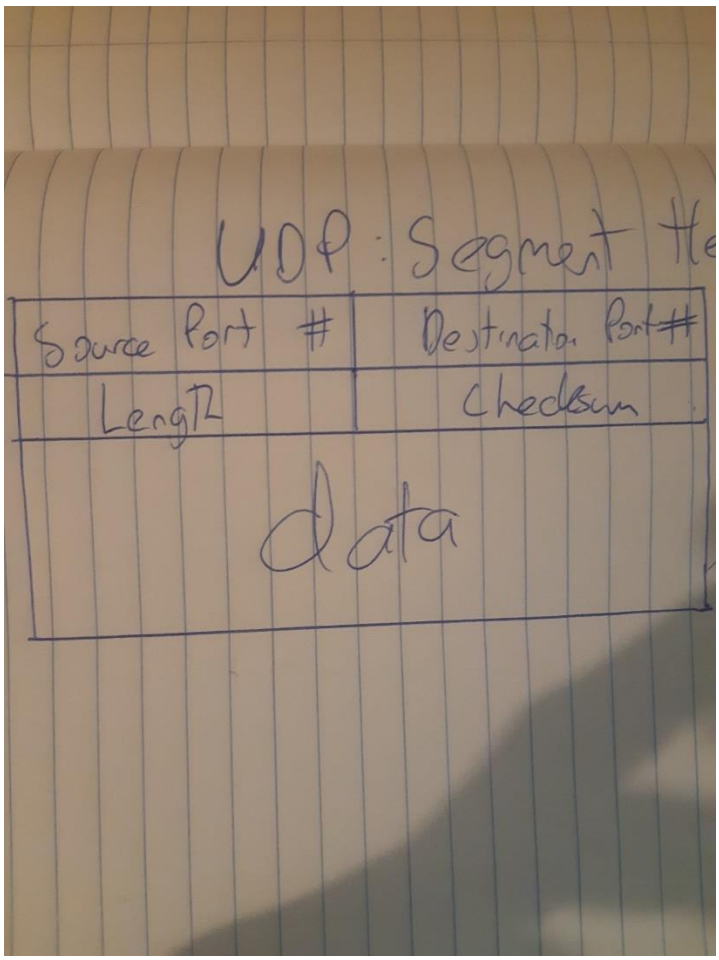
U/A/P/R/S/F: Are flags used to indicate parts of the 3 way handshake and the end of the TCP session.

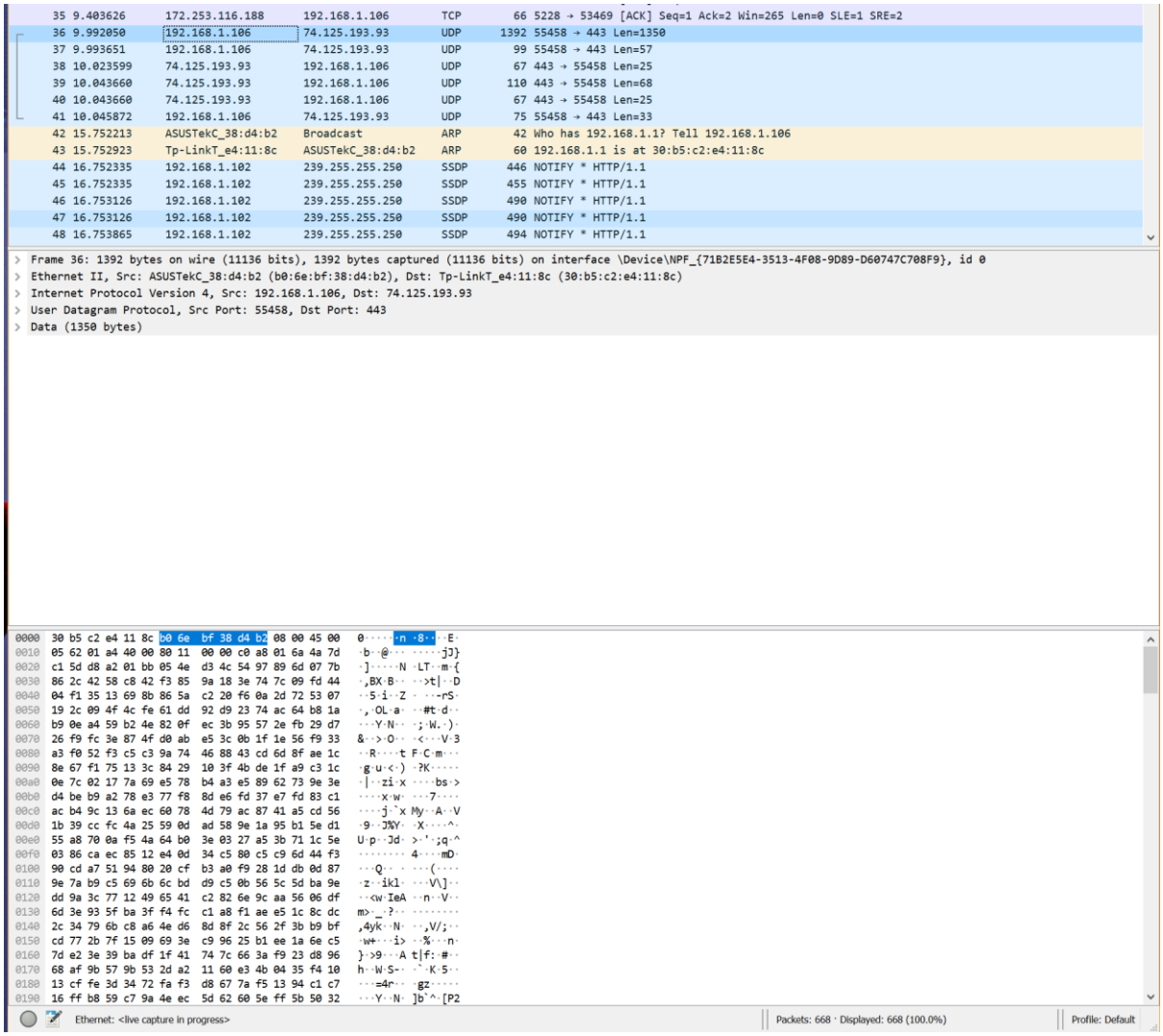
Window: Number of octets in the TCP header

Checksum: A Cyclic Redundancy Check, used to verify the integrity of the data in the TCP

Urgent Pointer: Points to the end of the "Urgent" data in the packet (if URG flag is set)

Q2. UDP Header





Source Port: Port number used by the computer sending the TCP segment.

Destination Port: Port number used by the computer receiving the TCP segment.

Length: Length in bytes the UDP header and UDP data. The type of IPv can determine this size.

Checksum: Error checking of the Header and data.

Q3. Checksum Example UDP, image below

C0a8 = 1100 0000 1010 1000 +

0065 = 0000 0000 0110 0101

= **1100 0001 0000 1101 +**

C0a8 = 1100 0000 1010 1000

= **1000 0001 1011 0101**

+ 1

= 1000 0001 1011 0110 +

000a = 0000 0000 0000 1010

= **1000 0001 1100 0000 +**

0011 = 0000 0000 0001 0001

= **1000 0001 1101 0001 +**

C2f1 = 1100 0010 1111 0001

= **0100 0100 1100 0010**

+ 1

= **0100 0100 1100 0011 +**

13e7 = 0001 0011 1110 0111

= **0101 1000 1010 1010 +**

0019 = 0000 0000 0001 1001

= **0101 1000 1100 0011 +**

0100 = 0000 0001 0000 0000

= **0101 1001 1100 0011 +**

0300 = 0000 0011 0000 0000

= **0101 1100 1100 0011 +**

0000 = 0000 0000 0000 0000

0800 = 0000 1000 0000 0000

= **0110 0100 1100 0011 +**

1182 = 0001 0001 1000 0010

= **0111 0110 0100 0101 +**

264e = 0010 0110 0100 1110

= **1001 1100 1001 0011 +**

0000 = 0000 0000 0000 0000

D20c = 1101 0010 0000 1100

= **0110 1110 1001 1111**

+ 1

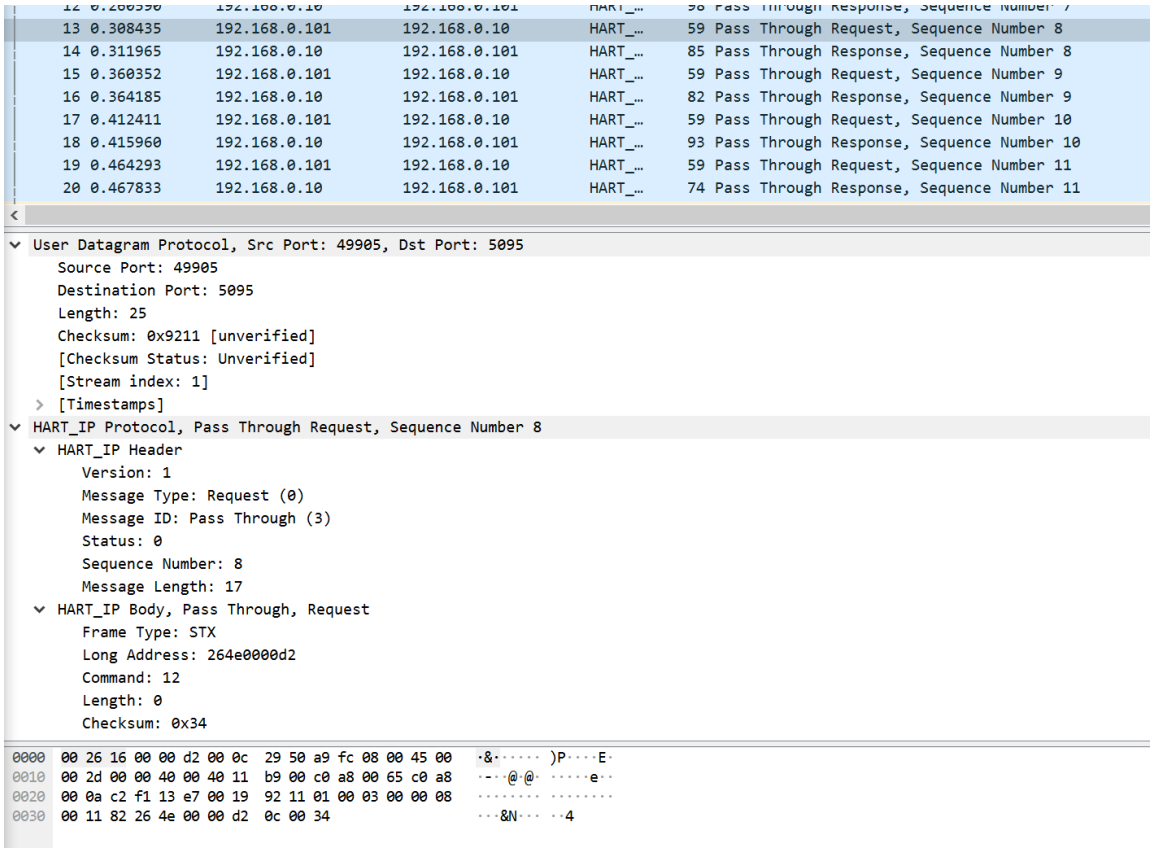
= 0110 1110 1010 0000 +

0034 = 0000 0000 0011 0100

= 0110 1110 1101 0100

Flip: 1001 0001 0010 1011

In Hexa: 912B



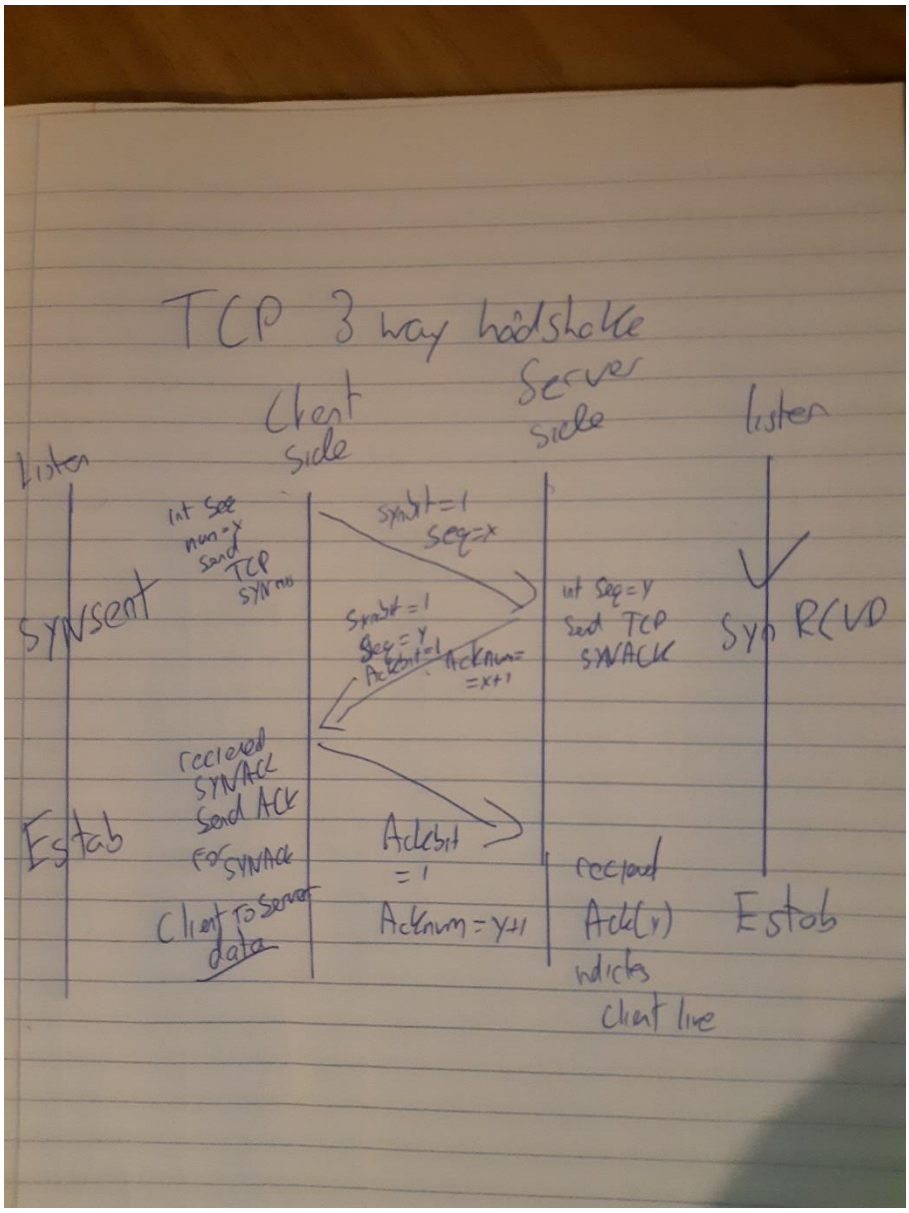
Actual checksum: 9211

The result I got was only marginally off(912B – 9211) so I assume it was a calculating error but the method I used was correct.

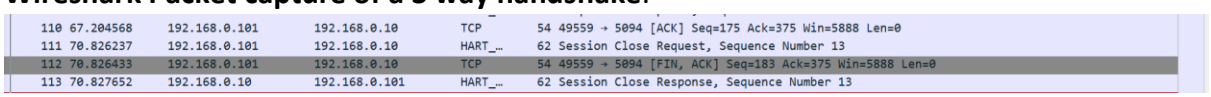
Q4. Checksum Example UDP, image below

After capturing some streaming packets, I found that the majority of the packets were UDP. The packets were used for transferring data for the stream to the computer, while UDP may lose information or possibly display it out of order the brain and can male up for this error when looking at the frames as long as it is not that bad. UDP is used for streaming because it allows faster transfer of data.

Q5. TCP – 3 way handshake

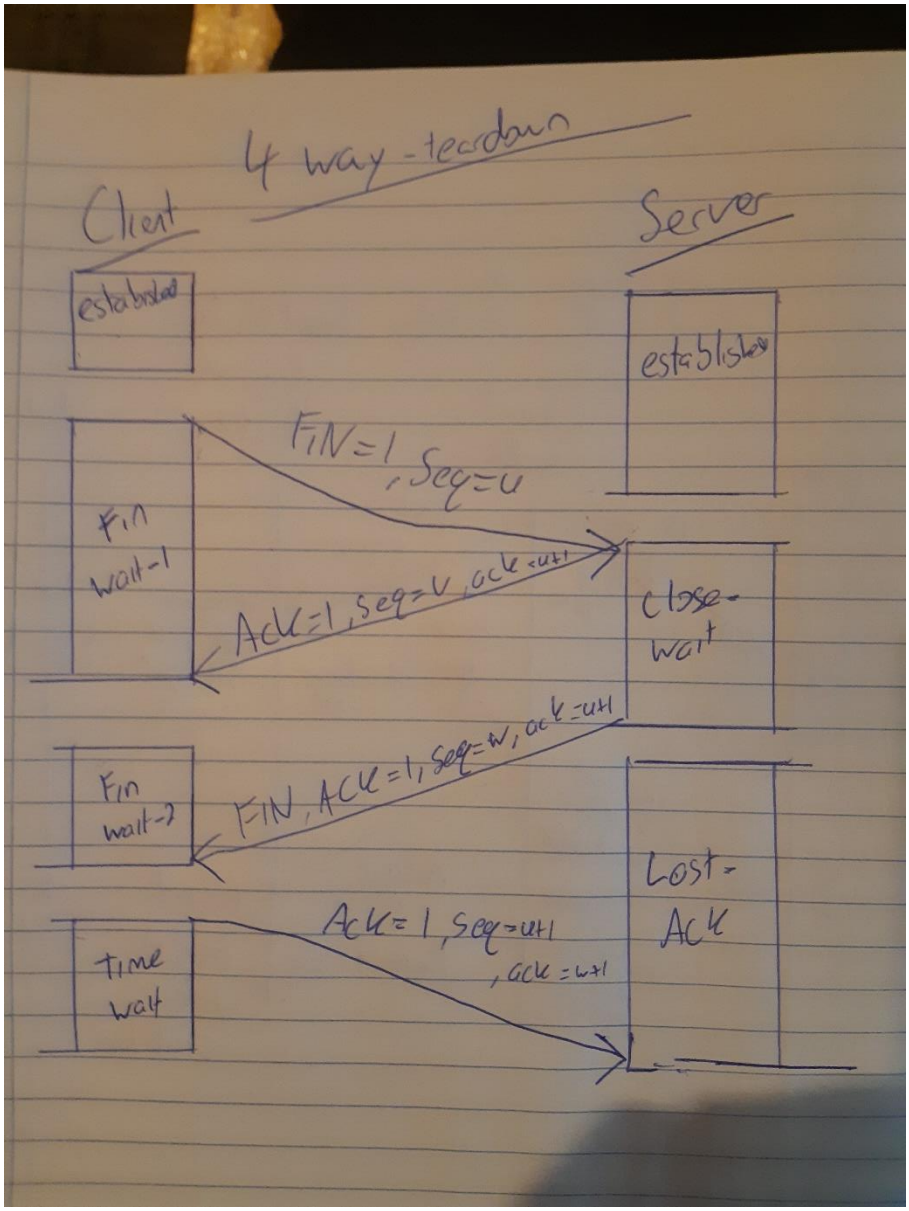


Wireshark Packet capture of a 3 way handshake:

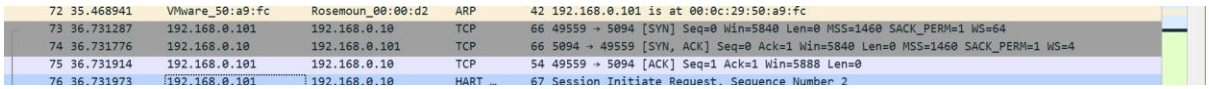


110	67.204568	192.168.0.101	192.168.0.10	TCP	54 49559 → 5094 [ACK] Seq=175 Ack=375 Win=5888 Len=0
111	70.826237	192.168.0.101	192.168.0.10	HART_…	62 Session Close Request, Sequence Number 13
112	70.826433	192.168.0.101	192.168.0.10	TCP	54 49559 → 5094 [FIN, ACK] Seq=183 Ack=375 Win=5888 Len=0
113	70.827652	192.168.0.10	192.168.0.101	HART_…	62 Session Close Response, Sequence Number 13

Q6. TCP – 4 way teardown:



Wireshark Packet capture of a 4 way teardown:



72	35.468941	VMware_50:a9:fc	Rosemoun_00:00:d2	ARP	42 192.168.0.101 is at 00:0c:29:50:a9:fc
73	36.731287	192.168.0.101	192.168.0.10	TCP	66 49559 → 5094 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=64
74	36.731776	192.168.0.10	192.168.0.101	TCP	66 5094 → 49559 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=4
75	36.731914	192.168.0.101	192.168.0.10	TCP	54 49559 → 5094 [ACK] Seq=1 Ack=1 Win=5888 Len=0
76	36.731973	192.168.0.101	192.168.0.10	HART …	67 Session Initiate Request, Sequence Number 2