Lab 5 Report

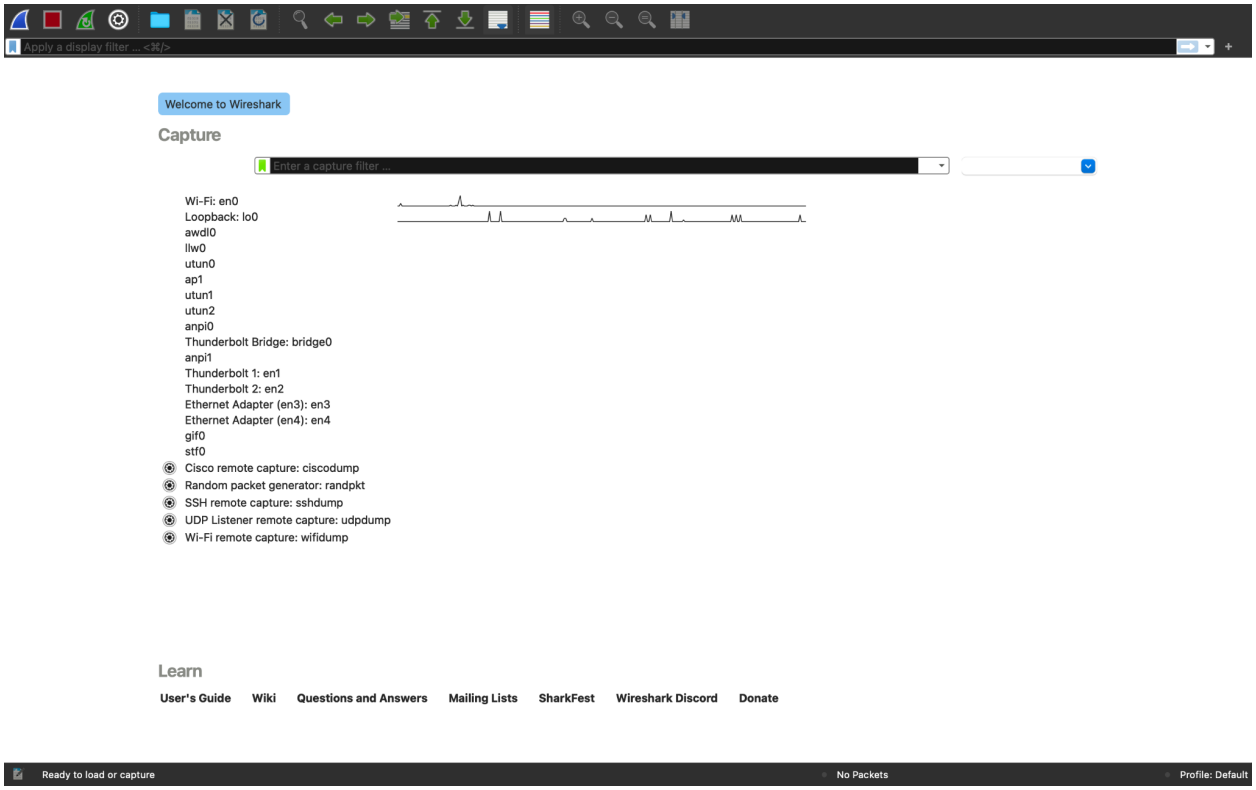
CSL 6010 - Cyber Security

Rahul Barodia

B20CS047

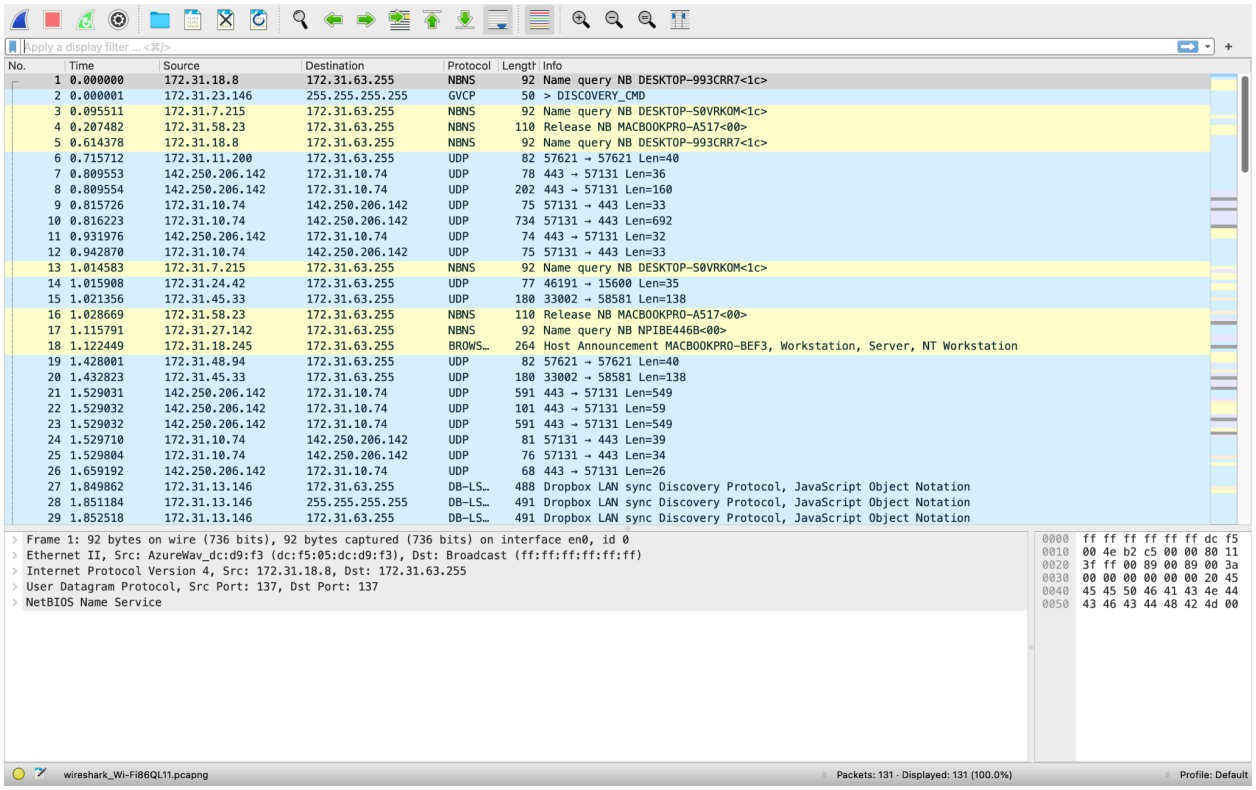
1) Packet Capturing

First, we need to select the wireless interface which we need to observe.



Upon clicking it packet capturing starts and based on our connected network.

I clicked on the Wi-Fi and the following packet were captured:

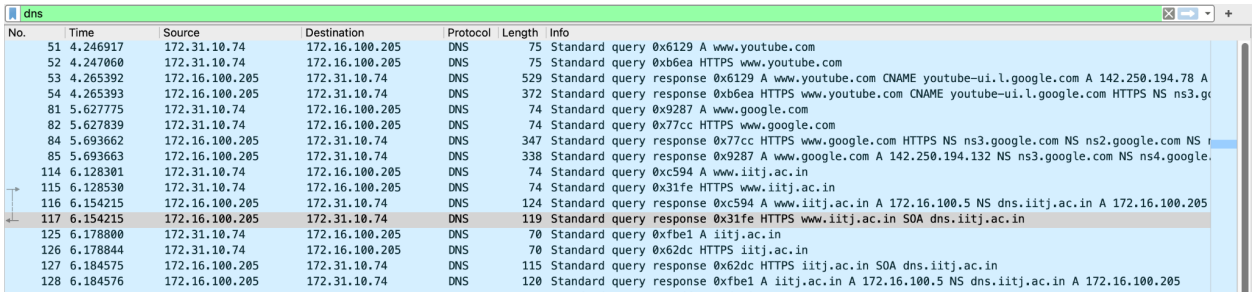


We can observe time , source and destination , type of protocol and info of each of these packets.

2) Packet capturing upon visiting IITJ website

I started packet capturing and then visited www.iitj.ac.in and then stopped packet capturing.

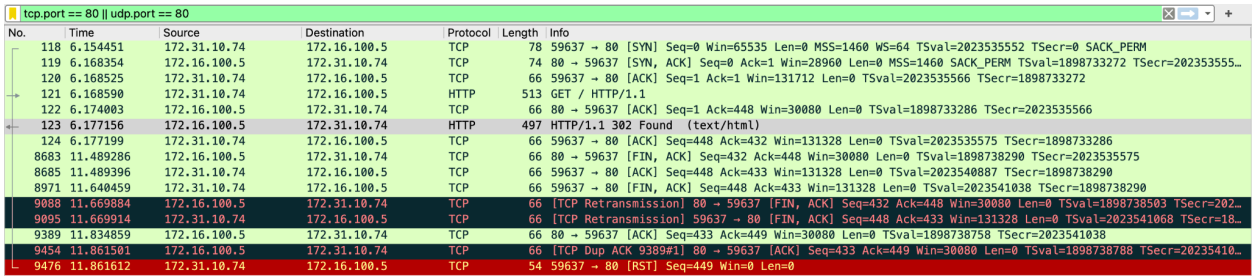
Upon entering the DNS filter I eas able to see the follwoing DNS requests



dns

No.	Time	Source	Destination	Protocol	Length	Info
51	4.246917	172.31.10.74	172.16.100.205	DNS	75	Standard query 0x6129 A www.youtube.com
52	4.247060	172.31.10.74	172.16.100.205	DNS	75	Standard query 0xb6ea HTTPS www.youtube.com
53	4.265392	172.16.100.205	172.31.10.74	DNS	529	Standard query response 0x6129 A www.youtube.com CNAME youtube-ui.l.google.com A 142.250.194.78 A
54	4.265393	172.16.100.205	172.31.10.74	DNS	372	Standard query response 0xb6ea HTTPS www.youtube.com CNAME youtube-ui.l.google.com HTTPS NS ns3.g
81	5.627775	172.31.10.74	172.16.100.205	DNS	74	Standard query 0x9287 A www.google.com
82	5.627839	172.31.10.74	172.16.100.205	DNS	74	Standard query 0x77cc HTTPS www.google.com
84	5.693662	172.16.100.205	172.31.10.74	DNS	347	Standard query response 0x77cc HTTPS www.google.com HTTPS NS ns3.google.com NS ns2.google.com NS
85	5.693663	172.16.100.205	172.31.10.74	DNS	338	Standard query response 0x9287 A www.google.com A 142.250.194.132 NS ns3.google.com NS ns4.google.
114	6.128301	172.31.10.74	172.16.100.205	DNS	74	Standard query 0xc594 A www.iitj.ac.in
115	6.128530	172.31.10.74	172.16.100.205	DNS	74	Standard query 0x31fe HTTPS www.iitj.ac.in
116	6.154215	172.16.100.205	172.31.10.74	DNS	124	Standard query response 0xc594 A www.iitj.ac.in A 172.16.100.5 NS dns.iitj.ac.in A 172.16.100.205
117	6.154215	172.16.100.205	172.31.10.74	DNS	119	Standard query response 0x31fe HTTPS www.iitj.ac.in SOA dns.iitj.ac.in
125	6.178800	172.31.10.74	172.16.100.205	DNS	70	Standard query 0xfbe1 A iitj.ac.in
126	6.178844	172.31.10.74	172.16.100.205	DNS	70	Standard query 0x62dc HTTPS iitj.ac.in
127	6.184575	172.16.100.205	172.31.10.74	DNS	115	Standard query response 0x62dc HTTPS iitj.ac.in SOA dns.iitj.ac.in
128	6.184576	172.16.100.205	172.31.10.74	DNS	120	Standard query response 0xfbe1 A iitj.ac.in A 172.16.100.5 NS dns.iitj.ac.in A 172.16.100.205

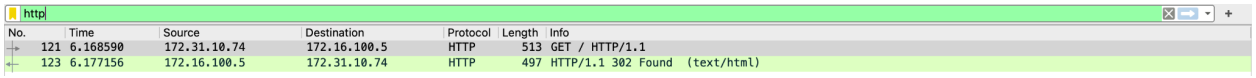
TCP :



tcp.port == 80 || udp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
118	6.154451	172.31.10.74	172.16.100.5	TCP	78	59637 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2023535552 TSecr=0 SACK_PERM
119	6.168354	172.16.100.5	172.31.10.74	TCP	74	80 → 59637 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM TSval=1898733272 TSecr=202353555…
120	6.168525	172.31.10.74	172.16.100.5	TCP	66	59637 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2023535566 TSecr=1898733272
121	6.168590	172.31.10.74	172.16.100.5	HTTP	513	GET / HTTP/1.1
122	6.174003	172.16.100.5	172.31.10.74	TCP	66	80 → 59637 [ACK] Seq=1 Ack=448 Win=30080 Len=0 TSval=1898733286 TSecr=2023535566
123	6.177156	172.16.100.5	172.31.10.74	HTTP	497	HTTP/1.1 302 Found (text/html)
124	6.177199	172.31.10.74	172.16.100.5	TCP	66	59637 → 80 [ACK] Seq=448 Ack=432 Win=131328 Len=0 TSval=2023535575 TSecr=1898733286
8683	11.489286	172.16.100.5	172.31.10.74	TCP	66	80 → 59637 [FIN, ACK] Seq=432 Ack=448 Win=30080 Len=0 TSval=1898738290 TSecr=2023535575
8685	11.489396	172.31.10.74	172.16.100.5	TCP	66	59637 → 80 [ACK] Seq=448 Ack=433 Win=131328 Len=0 TSval=2023540887 TSecr=1898738290
8971	11.640459	172.31.10.74	172.16.100.5	TCP	66	59637 → 80 [FIN, ACK] Seq=448 Ack=433 Win=131328 Len=0 TSval=2023541038 TSecr=1898738290
9088	11.669884	172.16.100.5	172.31.10.74	TCP	66	[TCP Retransmission] 80 → 59637 [FIN, ACK] Seq=432 Ack=448 Win=30080 Len=0 TSval=1898738503 TSecr=202…
9095	11.669914	172.31.10.74	172.16.100.5	TCP	66	[TCP Retransmission] 59637 → 80 [FIN, ACK] Seq=448 Ack=433 Win=131328 Len=0 TSval=2023541068 TSecr=18…
9389	11.834859	172.16.100.5	172.31.10.74	TCP	66	80 → 59637 [ACK] Seq=433 Ack=449 Win=30080 Len=0 TSval=1898738758 TSecr=2023541038
9454	11.861501	172.16.100.5	172.31.10.74	TCP	66	[TCP Dup ACK 9389#1] 80 → 59637 [ACK] Seq=433 Ack=449 Win=30080 Len=0 TSval=1898738788 TSecr=20235410…
9476	11.861612	172.31.10.74	172.16.100.5	TCP	54	59637 → 80 [RST] Seq=449 Win=0 Len=0

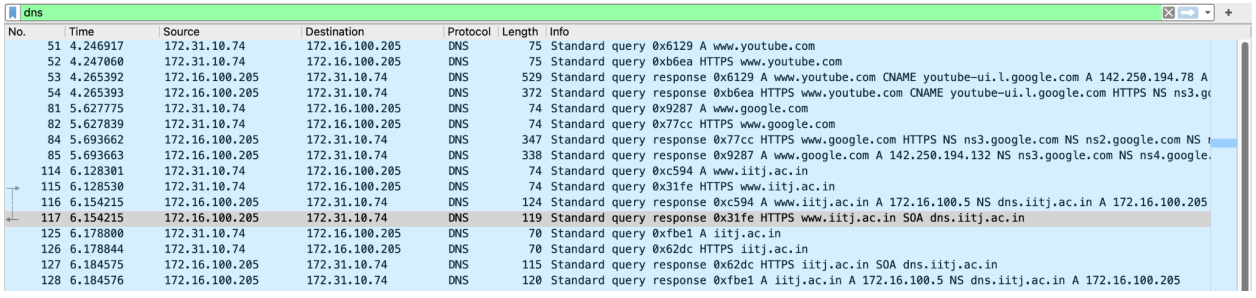
HTTP:



http

No.	Time	Source	Destination	Protocol	Length	Info
121	6.168590	172.31.10.74	172.16.100.5	HTTP	513	GET / HTTP/1.1
123	6.177156	172.16.100.5	172.31.10.74	HTTP	497	HTTP/1.1 302 Found (text/html)

We can get the IP address of the IITJ server by observing the DNS response packet which contains our IP address

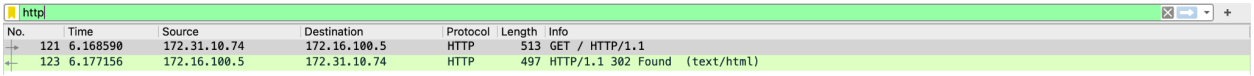


dns

No.	Time	Source	Destination	Protocol	Length	Info
51	4.246917	172.31.10.74	172.16.100.205	DNS	75	Standard query 0x6129 A www.youtube.com
52	4.247060	172.31.10.74	172.16.100.205	DNS	75	Standard query 0xb6ea HTTPS www.youtube.com
53	4.265392	172.16.100.205	172.31.10.74	DNS	529	Standard query response 0x6129 A www.youtube.com CNAME youtube-ui.l.google.com A 142.250.194.78 A
54	4.265393	172.16.100.205	172.31.10.74	DNS	372	Standard query response 0xb6ea HTTPS www.youtube.com CNAME youtube-ui.l.google.com HTTPS NS ns3.g
81	5.627775	172.31.10.74	172.16.100.205	DNS	74	Standard query 0x9287 A www.google.com
82	5.627839	172.31.10.74	172.16.100.205	DNS	74	Standard query 0x77cc HTTPS www.google.com
84	5.693662	172.16.100.205	172.31.10.74	DNS	347	Standard query response 0x77cc HTTPS www.google.com HTTPS NS ns3.google.com NS ns2.google.com NS
85	5.693663	172.16.100.205	172.31.10.74	DNS	338	Standard query response 0x9287 A www.google.com A 142.250.194.132 NS ns3.google.com NS ns4.google.
114	6.128301	172.31.10.74	172.16.100.205	DNS	74	Standard query 0xc594 A www.iitj.ac.in
115	6.128530	172.31.10.74	172.16.100.205	DNS	74	Standard query 0x31fe HTTPS www.iitj.ac.in
116	6.154215	172.16.100.205	172.31.10.74	DNS	124	Standard query response 0xc594 A www.iitj.ac.in A 172.16.100.5 NS dns.iitj.ac.in A 172.16.100.205
117	6.154215	172.16.100.205	172.31.10.74	DNS	119	Standard query response 0x31fe HTTPS www.iitj.ac.in SOA dns.iitj.ac.in
125	6.178800	172.31.10.74	172.16.100.205	DNS	70	Standard query 0xfbe1 A iitj.ac.in
126	6.178844	172.31.10.74	172.16.100.205	DNS	70	Standard query 0x62dc HTTPS iitj.ac.in
127	6.184575	172.16.100.205	172.31.10.74	DNS	115	Standard query response 0x62dc HTTPS iitj.ac.in SOA dns.iitj.ac.in
128	6.184576	172.16.100.205	172.31.10.74	DNS	120	Standard query response 0xfbe1 A iitj.ac.in A 172.16.100.5 NS dns.iitj.ac.in A 172.16.100.205

From No. 116 we can see that IP address of IITJ server is 172.16.100.5

Yes, I can see different HTTP requests/response



Upon browsing on google , I learned that HTTP GET request is being sent to the server if the user scrolls the homepage of the website. I scrolled the IITJ website so I got HTTP GET request followed by HTTP response containing HTML code for the homepage.

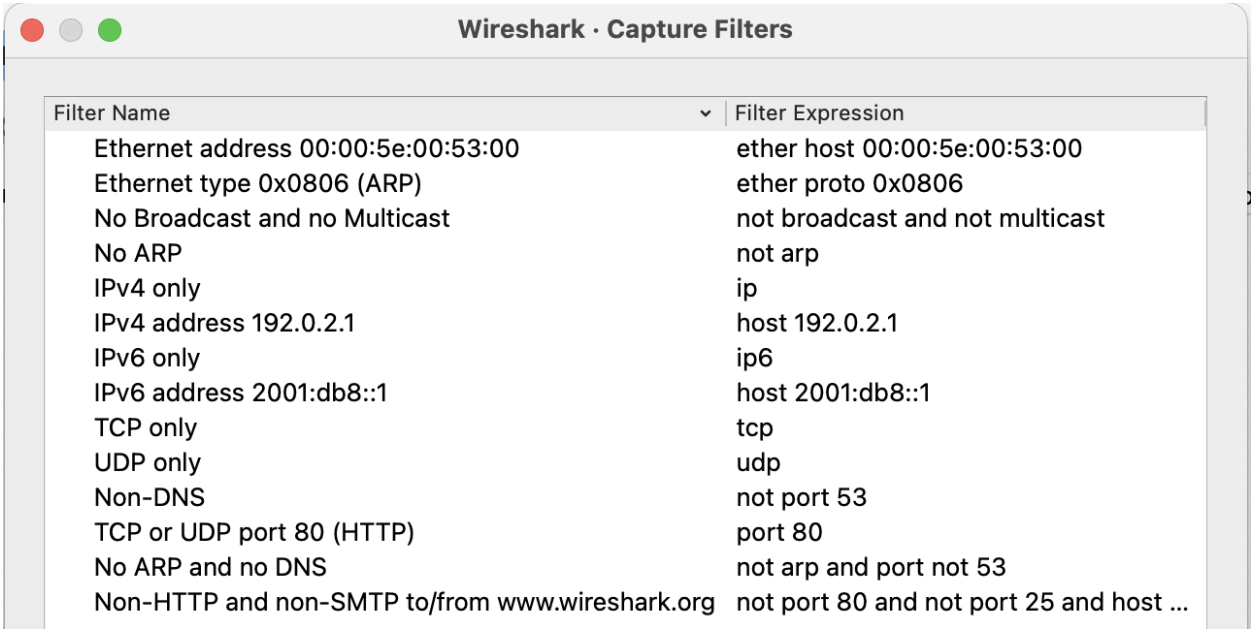
3)Packet highlighted in Black Colour

Packet highlighted in Black colour signifies that the packet contains some error.

4) Filters in Wireshark

Filters are used so that the users can see particular network traffic.

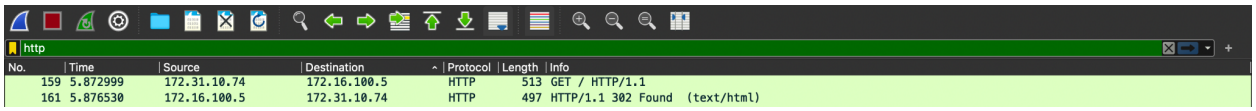
I can see the following filters in Wireshark



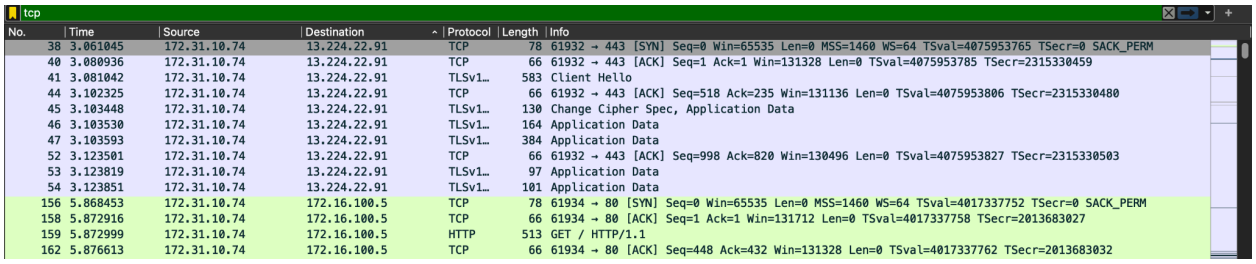
We can also add Filters by specifying the filter name and the filter expression

The 5 filters that I explored are:

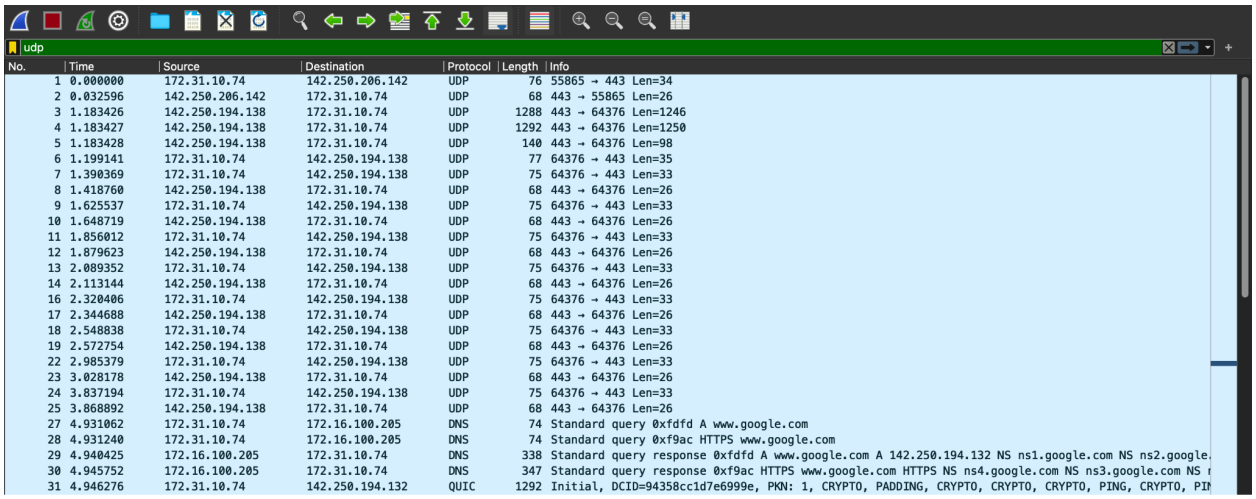
1) http filter



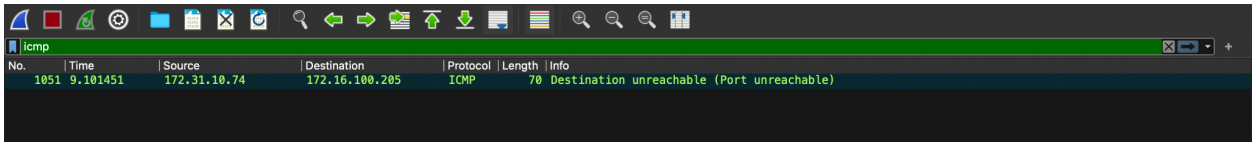
2) tcp filter



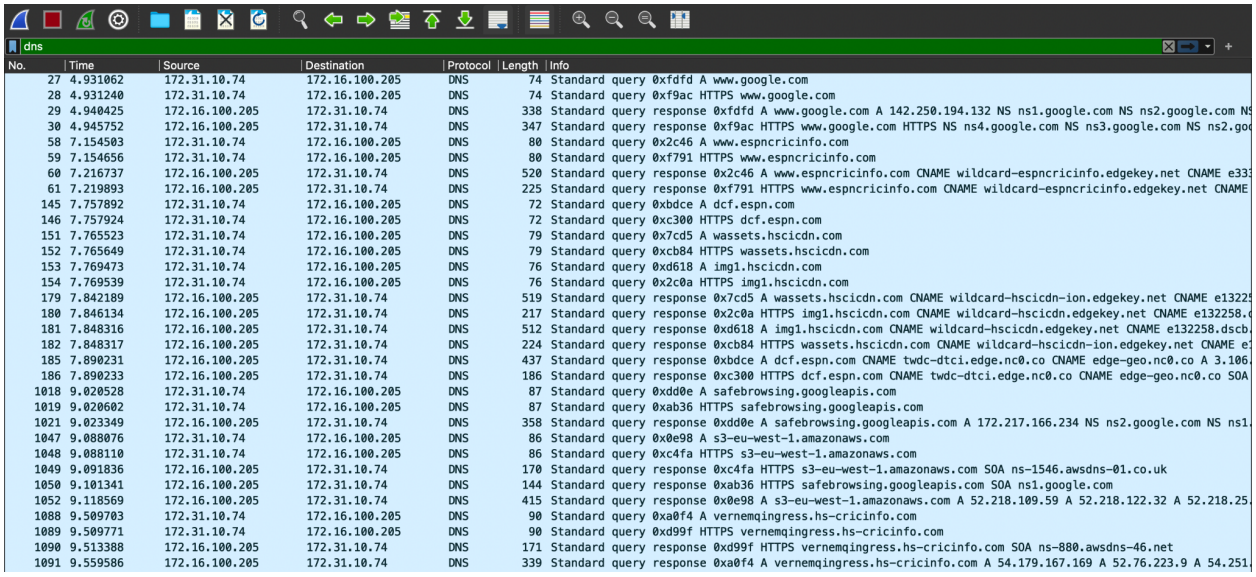
3) udp filter



4) ICMP filter



5) DNS Filter



dns

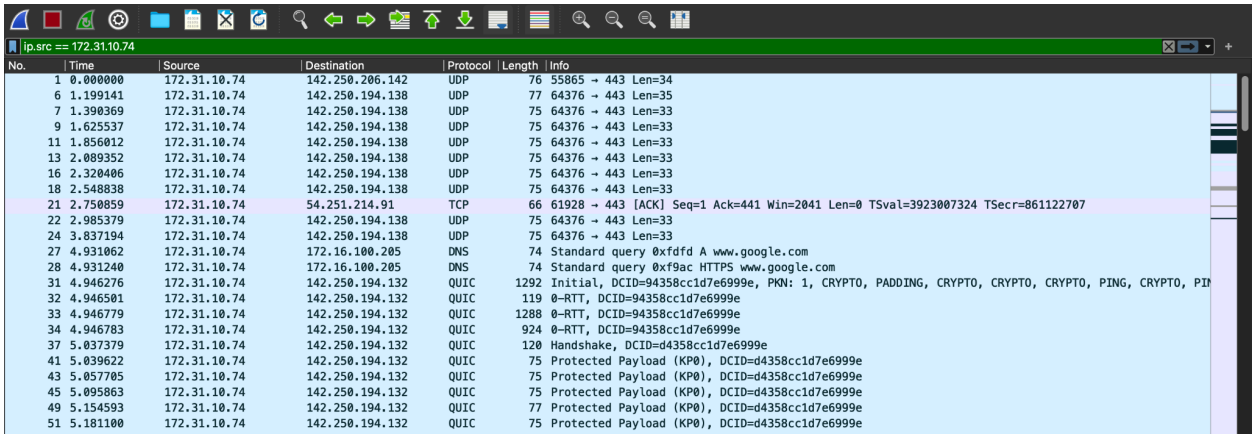
No.	Time	Source	Destination	Protocol	Length	Info
27	4.931062	172.31.10.74	172.16.100.205	DNS	74	Standard query 0xfdfd A www.google.com
28	4.931240	172.31.10.74	172.16.100.205	DNS	74	Standard query 0xf9ac HTTPS www.google.com
29	4.940425	172.16.100.205	172.31.10.74	DNS	338	Standard query response 0xfdfd A www.google.com A 142.250.194.132 NS ns1.google.com NS ns2.google.com NS
30	4.945752	172.16.100.205	172.31.10.74	DNS	347	Standard query response 0xf9ac HTTPS www.google.com HTTPS NS ns4.google.com NS ns3.google.com NS ns2.goo
58	7.154503	172.31.10.74	172.16.100.205	DNS	80	Standard query 0x2c46 A www.espncricinfo.com
59	7.154656	172.31.10.74	172.16.100.205	DNS	80	Standard query 0xf791 HTTPS www.espncricinfo.com
60	7.216737	172.16.100.205	172.31.10.74	DNS	520	Standard query response 0x2c46 A www.espncricinfo.com CNAME wildcard-espncricinfo.edgekey.net CNAME e33
61	7.219893	172.16.100.205	172.31.10.74	DNS	225	Standard query response 0xf791 HTTPS www.espncricinfo.com CNAME wildcard-espncricinfo.edgekey.net CNAME
145	7.757892	172.31.10.74	172.16.100.205	DNS	72	Standard query 0xbdce A dcf.espn.com
146	7.757924	172.31.10.74	172.16.100.205	DNS	72	Standard query 0xc300 HTTPS dcf.espn.com
151	7.765523	172.31.10.74	172.16.100.205	DNS	79	Standard query 0x7cd5 A wassets.hscicdn.com
152	7.765649	172.31.10.74	172.16.100.205	DNS	79	Standard query 0xcb84 HTTPS wassets.hscicdn.com
153	7.769473	172.31.10.74	172.16.100.205	DNS	76	Standard query 0xd618 A img1.hscicdn.com
154	7.769539	172.31.10.74	172.16.100.205	DNS	76	Standard query 0x2c0a HTTPS img1.hscicdn.com
179	7.842189	172.16.100.205	172.31.10.74	DNS	519	Standard query response 0x7cd5 A wassets.hscicdn.com CNAME wildcard-hscicdn-ion.edgekey.net CNAME e13225
180	7.846134	172.16.100.205	172.31.10.74	DNS	217	Standard query response 0x2c0a HTTPS img1.hscicdn.com CNAME wildcard-hscicdn.edgekey.net CNAME e132258.c
181	7.848316	172.16.100.205	172.31.10.74	DNS	512	Standard query response 0xd618 A img1.hscicdn.com CNAME wildcard-hscicdn.edgekey.net CNAME e132258.dscb
182	7.848317	172.16.100.205	172.31.10.74	DNS	224	Standard query response 0xcb84 HTTPS wassets.hscicdn.com CNAME wildcard-hscicdn-ion.edgekey.net CNAME e
185	7.890231	172.16.100.205	172.31.10.74	DNS	437	Standard query response 0xbdce A dcf.espn.com CNAME twdc-dtci.edge.nc0.co CNAME edge-geo.nc0.co A 3.106
186	7.890233	172.16.100.205	172.31.10.74	DNS	186	Standard query response 0xc300 HTTPS dcf.espn.com CNAME twdc-dtci.edge.nc0.co CNAME edge-geo.nc0.co SOA
1018	9.020528	172.31.10.74	172.16.100.205	DNS	87	Standard query 0xdd0e A safebrowsing.googleapis.com
1019	9.020602	172.31.10.74	172.16.100.205	DNS	87	Standard query 0xab36 HTTPS safebrowsing.googleapis.com
1021	9.023349	172.16.100.205	172.31.10.74	DNS	358	Standard query response 0xdd0e A safebrowsing.googleapis.com A 172.217.166.234 NS ns2.google.com NS ns1
1047	9.088076	172.31.10.74	172.16.100.205	DNS	86	Standard query 0x0e98 A s3-eu-west-1.amazonaws.com
1048	9.088110	172.31.10.74	172.16.100.205	DNS	86	Standard query 0xc4fa HTTPS s3-eu-west-1.amazonaws.com
1049	9.091836	172.16.100.205	172.31.10.74	DNS	170	Standard query response 0xc4fa HTTPS s3-eu-west-1.amazonaws.com SOA ns-1546.awsdns-01.co.uk
1050	9.101341	172.16.100.205	172.31.10.74	DNS	144	Standard query response 0xab36 HTTPS safebrowsing.googleapis.com SOA ns1.google.com
1052	9.118569	172.16.100.205	172.31.10.74	DNS	415	Standard query response 0x0e98 A s3-eu-west-1.amazonaws.com A 52.218.109.59 A 52.218.122.32 A 52.218.25
1088	9.509703	172.31.10.74	172.16.100.205	DNS	90	Standard query 0xa0f4 A vernemqingress.hs-cricinfo.com
1089	9.509771	172.31.10.74	172.16.100.205	DNS	90	Standard query 0xd99f HTTPS vernemqingress.hs-cricinfo.com
1090	9.513388	172.16.100.205	172.31.10.74	DNS	171	Standard query response 0xd99f HTTPS vernemqingress.hs-cricinfo.com SOA ns-880.awsdns-46.net
1091	9.559586	172.16.100.205	172.31.10.74	DNS	339	Standard query response 0xa0f4 A vernemqingress.hs-cricinfo.com A 54.179.167.169 A 52.76.223.9 A 54.251

5) Filter command for all outgoing traffic

We can use the following command to list all the outgoing traffic :

ip.src == IP address

In my case, the command is , ip.src == 172.31.10.74

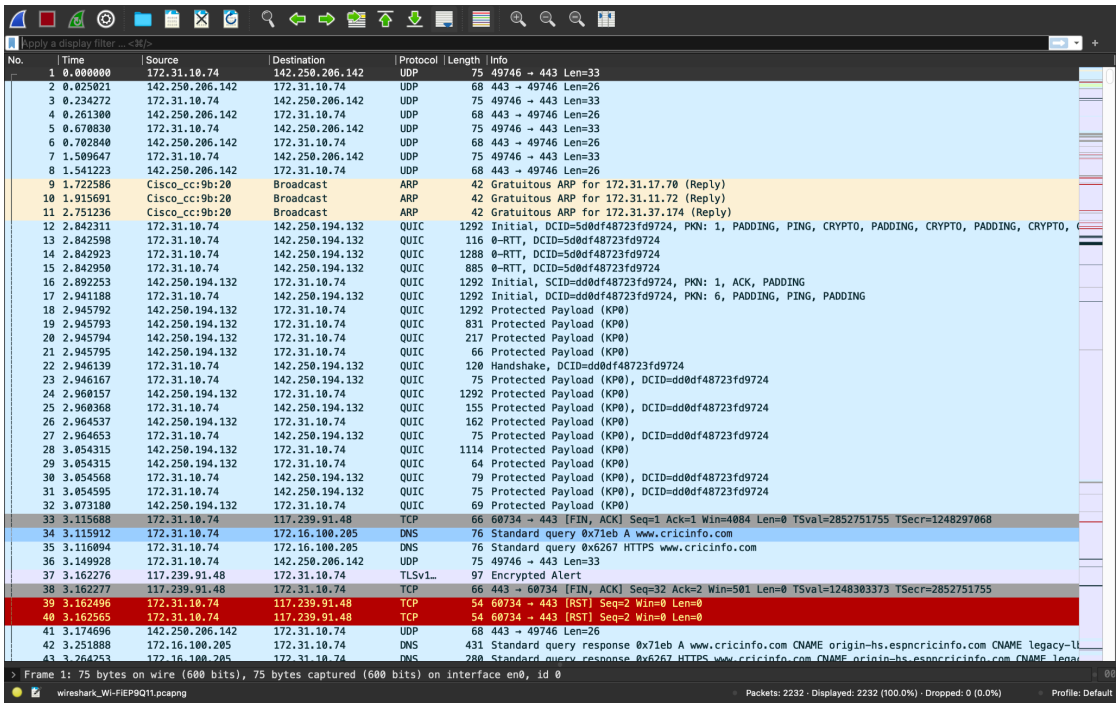


ip.src == 172.31.10.74

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.31.10.74	142.250.206.142	UDP	76	55865 → 443 Len=34
6	1.199141	172.31.10.74	142.250.194.138	UDP	77	64376 → 443 Len=35
7	1.390369	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
9	1.625537	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
11	1.856012	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
13	2.089352	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
16	2.320406	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
18	2.548838	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
21	2.750859	172.31.10.74	54.251.214.91	TCP	66	61928 → 443 [ACK] Seq=1 Ack=441 Win=2041 Len=0 TSval=3923007324 TSecr=861122707
22	2.985379	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
24	3.837194	172.31.10.74	142.250.194.138	UDP	75	64376 → 443 Len=33
27	4.931062	172.31.10.74	172.16.100.205	DNS	74	Standard query 0xfdfd A www.google.com
28	4.931240	172.31.10.74	172.16.100.205	DNS	74	Standard query 0xf9ac HTTPS www.google.com
31	4.946276	172.31.10.74	142.250.194.132	QUIC	1292	Initial, DCID=94358cc1d7e6999e, PKN: 1, CRYPTO, PADDING, CRYPTO, CRYPTO, CRYPTO, PING, CRYPTO, PIN
32	4.946501	172.31.10.74	142.250.194.132	QUIC	119	0-RTT, DCID=94358cc1d7e6999e
33	4.946779	172.31.10.74	142.250.194.132	QUIC	1288	0-RTT, DCID=94358cc1d7e6999e
34	4.946783	172.31.10.74	142.250.194.132	QUIC	924	0-RTT, DCID=94358cc1d7e6999e
37	5.037379	172.31.10.74	142.250.194.132	QUIC	120	Handshake, DCID=d4358cc1d7e6999e
41	5.039622	172.31.10.74	142.250.194.132	QUIC	75	Protected Payload (KP0), DCID=d4358cc1d7e6999e
43	5.057705	172.31.10.74	142.250.194.132	QUIC	75	Protected Payload (KP0), DCID=d4358cc1d7e6999e
45	5.095863	172.31.10.74	142.250.194.132	QUIC	75	Protected Payload (KP0), DCID=d4358cc1d7e6999e
49	5.154593	172.31.10.74	142.250.194.132	QUIC	77	Protected Payload (KP0), DCID=d4358cc1d7e6999e
51	5.181100	172.31.10.74	142.250.194.132	QUIC	75	Protected Payload (KP0), DCID=d4358cc1d7e6999e

6) 3-way TCP handshaking

I visited the website www.cricinfo.com and captured packets in wireshark.



TCP uses the 3-way handshaking protocol to establish a connection between two devices in a network, and after these steps, both devices begin to communicate (transmit data).

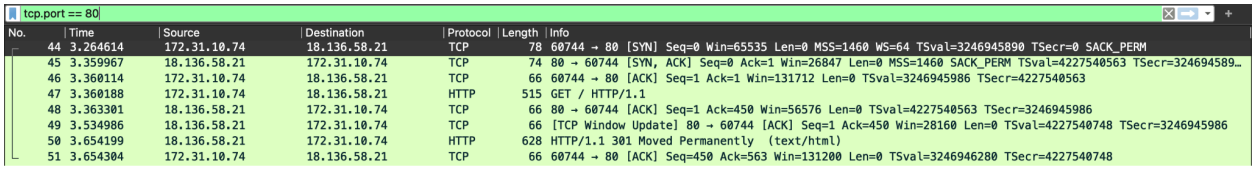
The three steps are

1) SYN (synchronize)

2) SYN-ACK (synchronize - acknowledge)

3) ACK (acknowledge)

To show the 3-way TCP handshaking we need to use the filter " tcp.port == 80 ". This filter will show us only the packets on port 80 i.e. HTTP port. The following is the screenshot of the same:



We can clearly see all the 3 ways SYN, SYN-ACK, and ACK in the info section of the above screenshot.

7) Why DNS uses UDP but HTTP uses TCP

UDP is a connectionless transfer protocol. It has no reliability. DNS responses being small in size does not require reliability. Therefore DNS uses UDP stream.

Whereas, TCP is a connection oriented transport protocol. It provides reliability in case of data transfer. HTTP responses being large in size requires reliability in data transfer. Therefore HTTP uses TCP stream.

8)TCP communication in socket program

I ran the following server and client programs on VS code.

server2.py > main

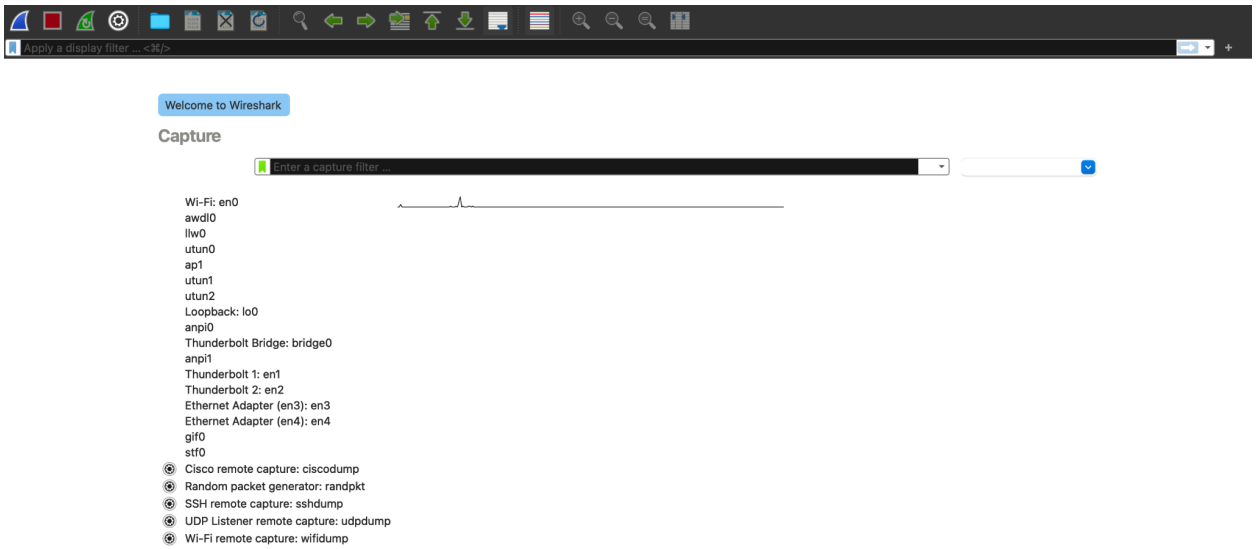
```python
import socket
import os

def handle_client(conn, addr):
    print("Connection from: " + str(addr))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break
        try:
            result = eval(data)
            conn.send(str(result).encode())
        except:
            conn.send("Invalid Expression".encode())
    conn.close()

def main():
    host = 'localhost'
    port = 5000

    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(5)

    while True:
        conn, addr = server_socket.accept()
        pid = os.fork()
        if pid == 0:
            server_socket.close()
            handle_client(conn, addr)
            exit()
        else:
            conn.close()

if __name__ == '__main__':
    main()
```

client.py > ...

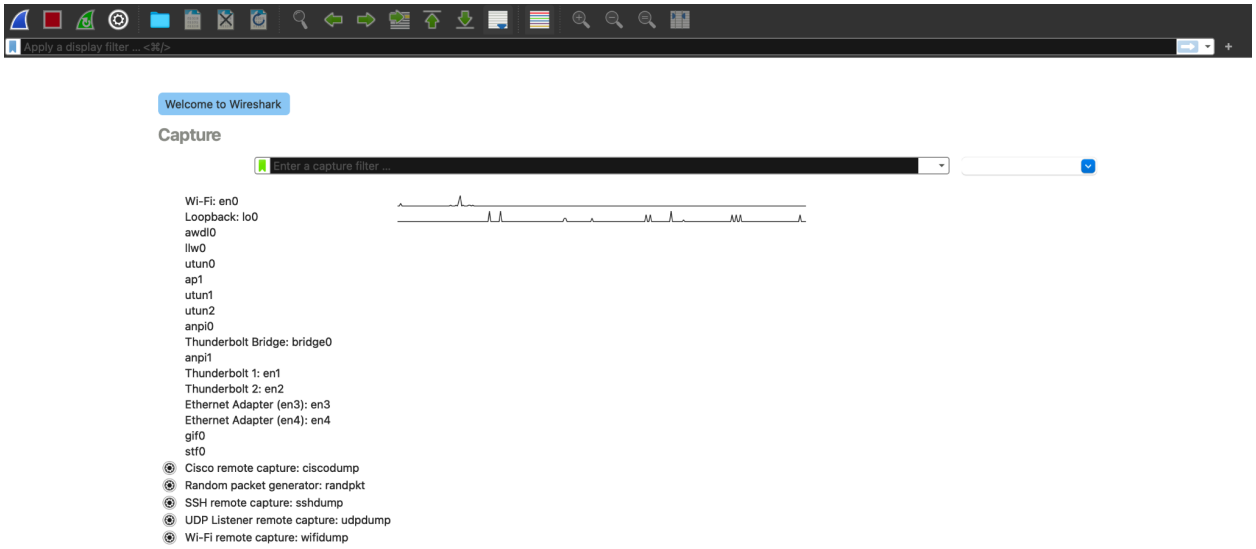
```python
import socket

def main():
    host = 'localhost'
    port = 5000

    client_socket = socket.socket()
    client_socket.connect((host, port))

    message = input("Enter a math expression: ")
    while message != 'q':
        client_socket.send(message.encode())
        data = client_socket.recv(1024).decode()
        print("Result: " + data)
        message = input("Enter a math expression: ")
    client_socket.close()

if __name__ == '__main__':
    main()

```

It is a simple client-server system where the client chats with a math server.

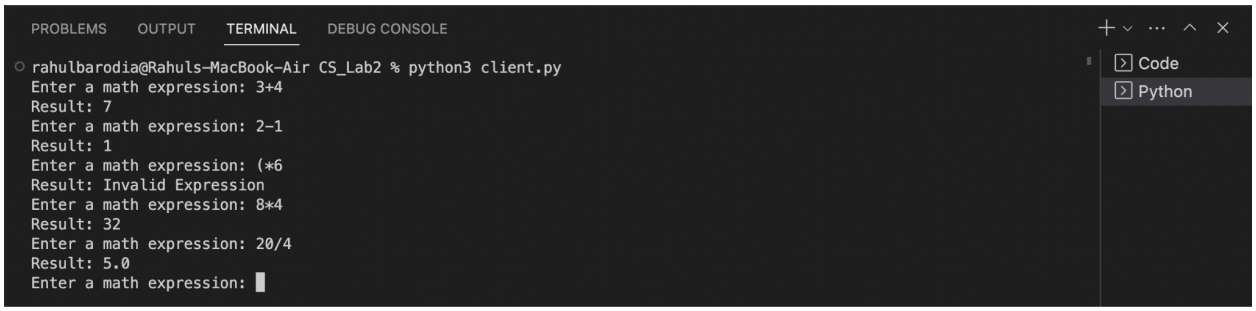
Before the client-server connection was established the interface of wireshark was looking like this:



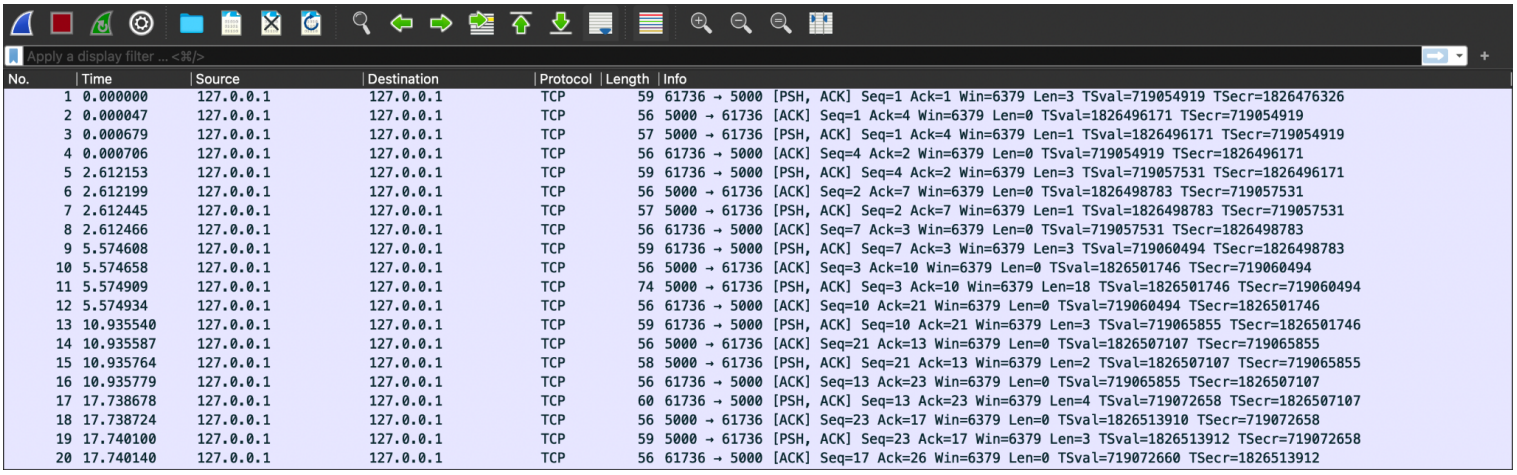
Upon establishment of client-server connection the interface looked like :



I clicked on the Loopback. There was no packet capturing happening until this point. Now I supplied the sever following mathematical expressions from the client:



After this, the packet capturing interface at the Wireshark looked like this:



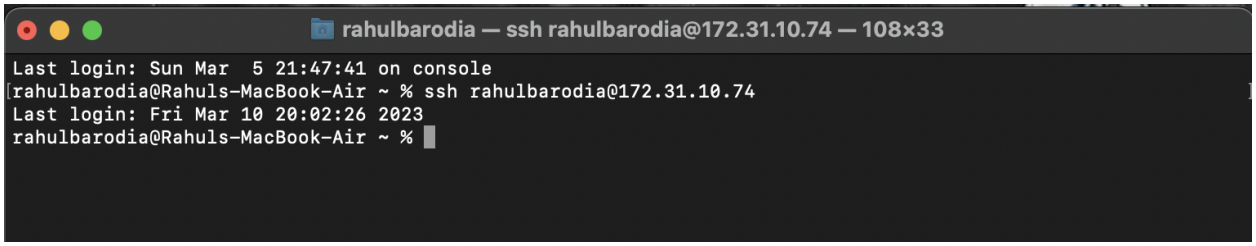
We can clearly observe that TCP communication is happening during client-server communication. The source and destination IP addresses are the same since both the server and client are running on the same machine.

The IP address is 127.0.0.1

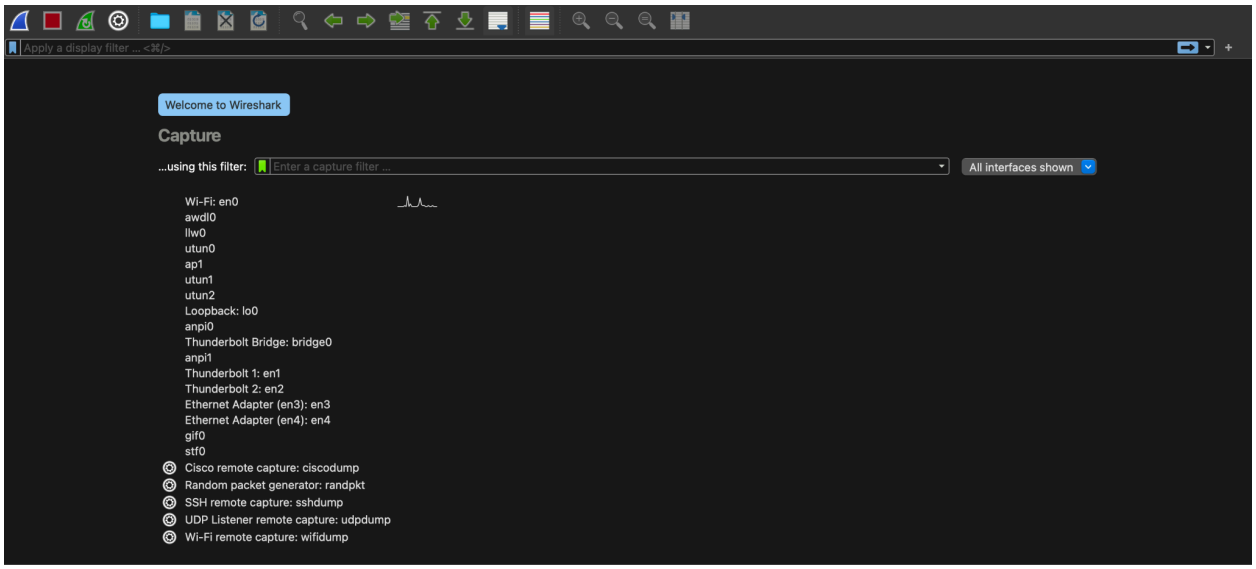
We can also observe from the above screenshot that the TCP communication is happening at two different ports. Port no. 5000 and Port no. 61736

9) Observing packet capturing in SSH connection

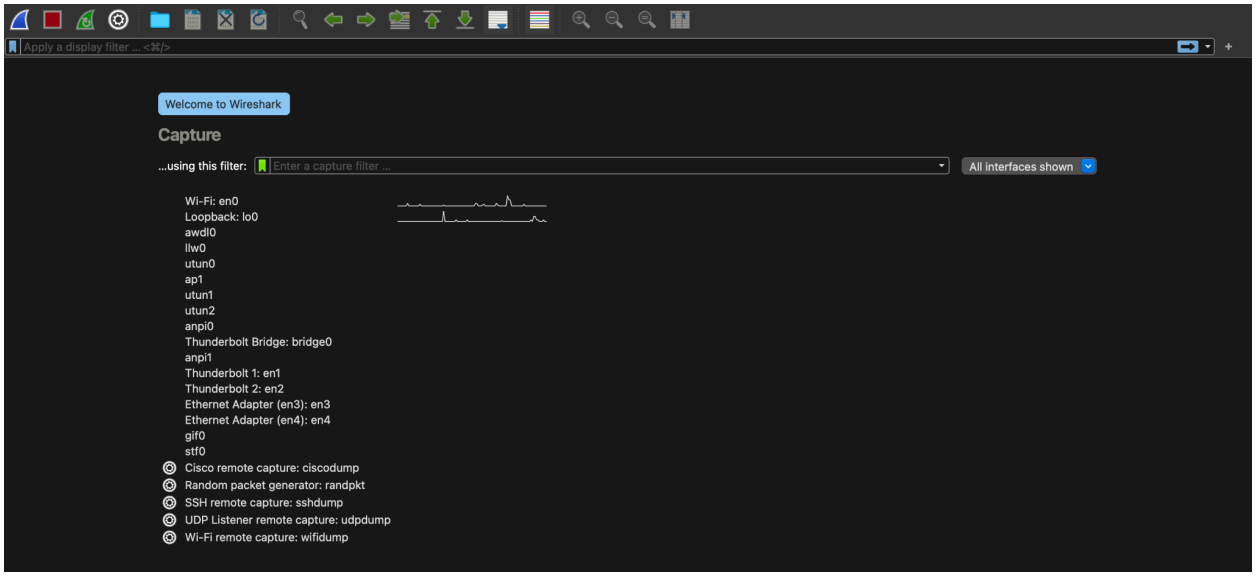
I established SSH connection with IITJ home folder as follows:



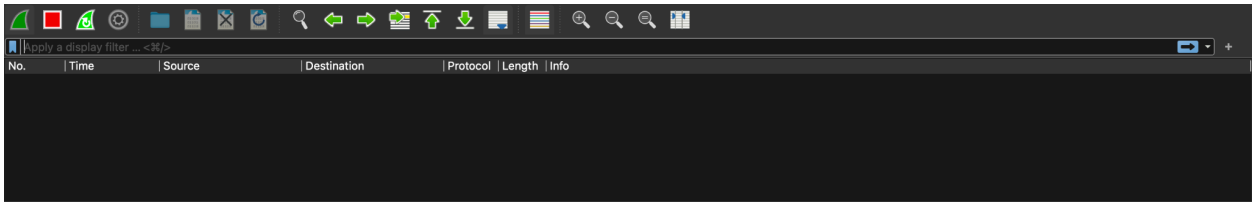
Prior to this the interface of wireshark looked like :



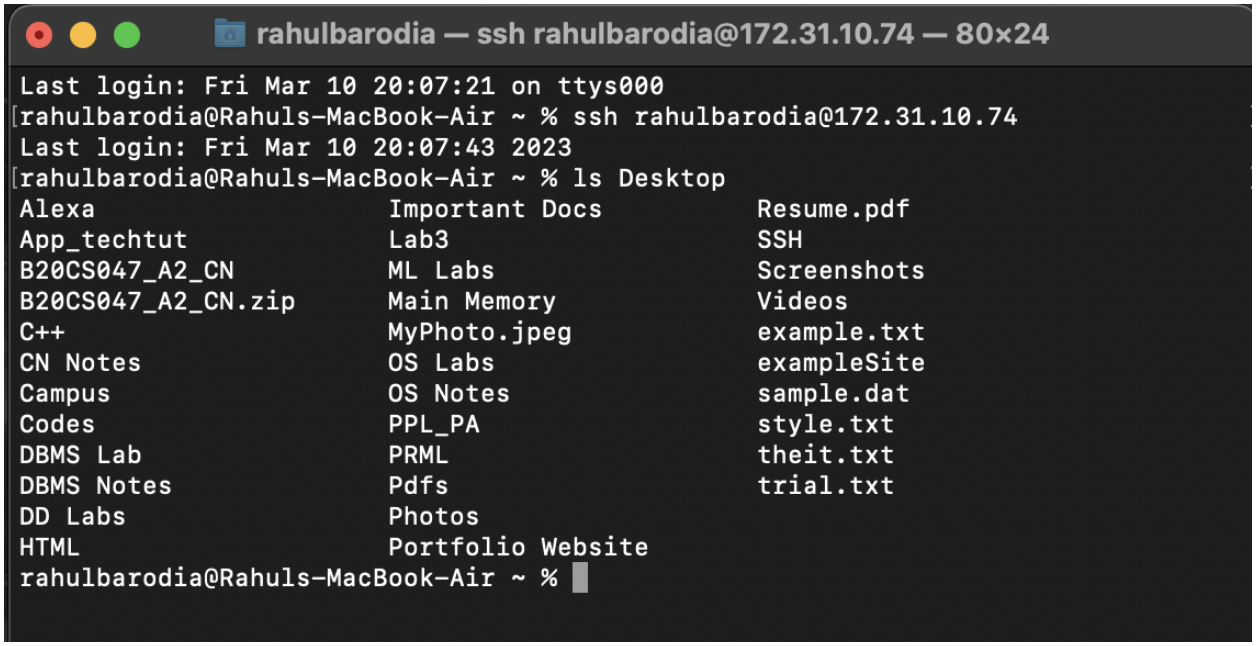
Now it looked like:



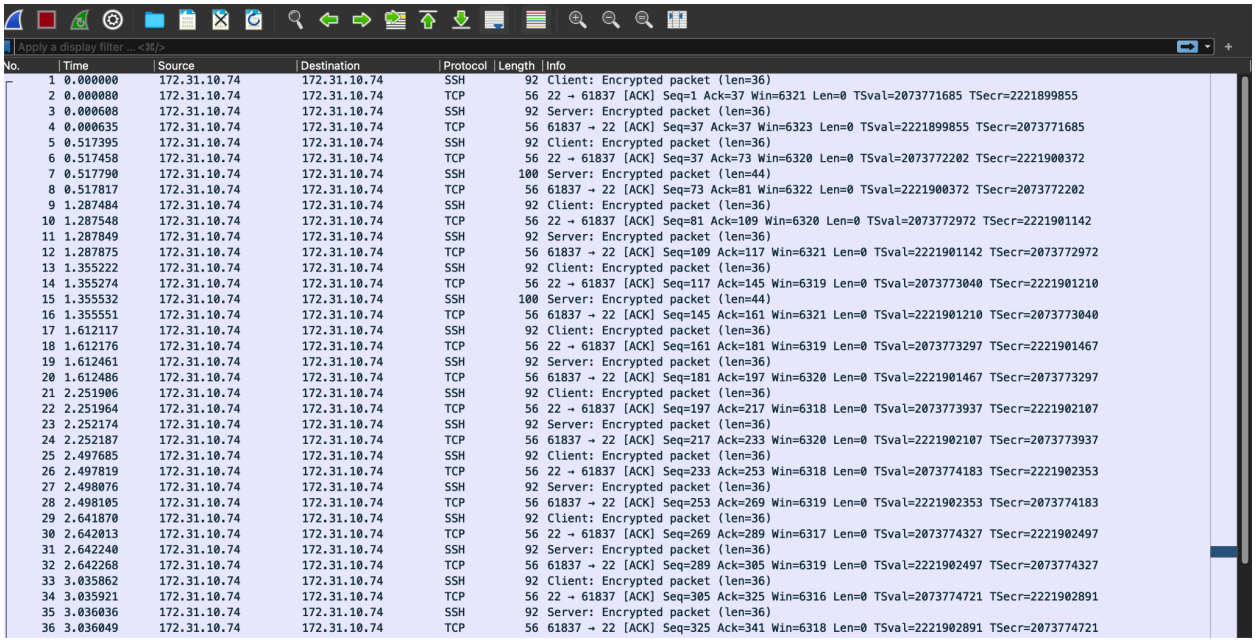
I clicked on the Loopback. There was no packet capturing happening until this point.



Then I executed the following command on terminal where SSH connection is established:



The Wireshark captured the packets and the results looked like:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
2	0.000080	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=1 Ack=37 Win=6321 Len=0 TSval=2073771685 TSecr=2221899855
3	0.000608	172.31.10.74	172.31.10.74	SSH	92	Server: Encrypted packet (len=36)
4	0.000635	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=37 Ack=37 Win=6323 Len=0 TSval=2221899855 TSecr=2073771685
5	0.517395	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
6	0.517458	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=37 Ack=73 Win=6320 Len=0 TSval=2073772202 TSecr=2221900372
7	0.517790	172.31.10.74	172.31.10.74	SSH	100	Server: Encrypted packet (len=44)
8	0.517817	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=73 Ack=81 Win=6322 Len=0 TSval=2221900372 TSecr=2073772202
9	1.287484	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
10	1.287548	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=81 Ack=109 Win=6320 Len=0 TSval=2073772972 TSecr=2221901142
11	1.287849	172.31.10.74	172.31.10.74	SSH	92	Server: Encrypted packet (len=36)
12	1.287875	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=109 Ack=117 Win=6321 Len=0 TSval=2221901142 TSecr=2073772972
13	1.355222	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
14	1.355274	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=117 Ack=145 Win=6319 Len=0 TSval=2073773040 TSecr=2221901210
15	1.355532	172.31.10.74	172.31.10.74	SSH	100	Server: Encrypted packet (len=44)
16	1.355551	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=145 Ack=161 Win=6321 Len=0 TSval=2221901210 TSecr=2073773040
17	1.612117	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
18	1.612176	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=161 Ack=181 Win=6319 Len=0 TSval=2073773297 TSecr=2221901467
19	1.612461	172.31.10.74	172.31.10.74	SSH	92	Server: Encrypted packet (len=36)
20	1.612486	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=181 Ack=197 Win=6320 Len=0 TSval=2221901467 TSecr=2073773297
21	2.251906	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
22	2.251964	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=197 Ack=217 Win=6318 Len=0 TSval=2073773937 TSecr=2221902107
23	2.252174	172.31.10.74	172.31.10.74	SSH	92	Server: Encrypted packet (len=36)
24	2.252187	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=217 Ack=233 Win=6320 Len=0 TSval=2221902107 TSecr=2073773937
25	2.497685	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
26	2.497819	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=233 Ack=253 Win=6318 Len=0 TSval=2073774183 TSecr=2221902353
27	2.498076	172.31.10.74	172.31.10.74	SSH	92	Server: Encrypted packet (len=36)
28	2.498105	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=253 Ack=269 Win=6319 Len=0 TSval=2221902353 TSecr=2073774183
29	2.641870	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
30	2.642013	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=269 Ack=289 Win=6317 Len=0 TSval=2073774327 TSecr=2221902497
31	2.642240	172.31.10.74	172.31.10.74	SSH	92	Server: Encrypted packet (len=36)
32	2.642268	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=289 Ack=305 Win=6319 Len=0 TSval=2221902497 TSecr=2073774327
33	3.035862	172.31.10.74	172.31.10.74	SSH	92	Client: Encrypted packet (len=36)
34	3.035921	172.31.10.74	172.31.10.74	TCP	56	22 → 61837 [ACK] Seq=305 Ack=325 Win=6316 Len=0 TSval=2073774721 TSecr=2221902891
35	3.036036	172.31.10.74	172.31.10.74	SSH	92	Server: Encrypted packet (len=36)
36	3.036049	172.31.10.74	172.31.10.74	TCP	56	61837 → 22 [ACK] Seq=325 Ack=341 Win=6318 Len=0 TSval=2221902891 TSecr=2073774721

We can analyze this SSH traffic.

Some observations are:

1) The source and destination IP addresses are the same because the SSH connection is established between the same machine.

2) SSH uses TCP as its transfer protocol.

3) The two connected ports have port no. 22 and 61837. SSH typically runs on port no. 22.

4) Since SSH connection is encrypted the data transmitted is not visible in the info section. Instead, we can see info such as Client: Encrypted packet (len=36)

5) Since it is a TCP connection, the way handshaking is happening. But SYN and SYN-ACK are hidden because the connection is secure. Wireshark can only detect the ACK step which the server sends to the client as the last step before establishing a connection.