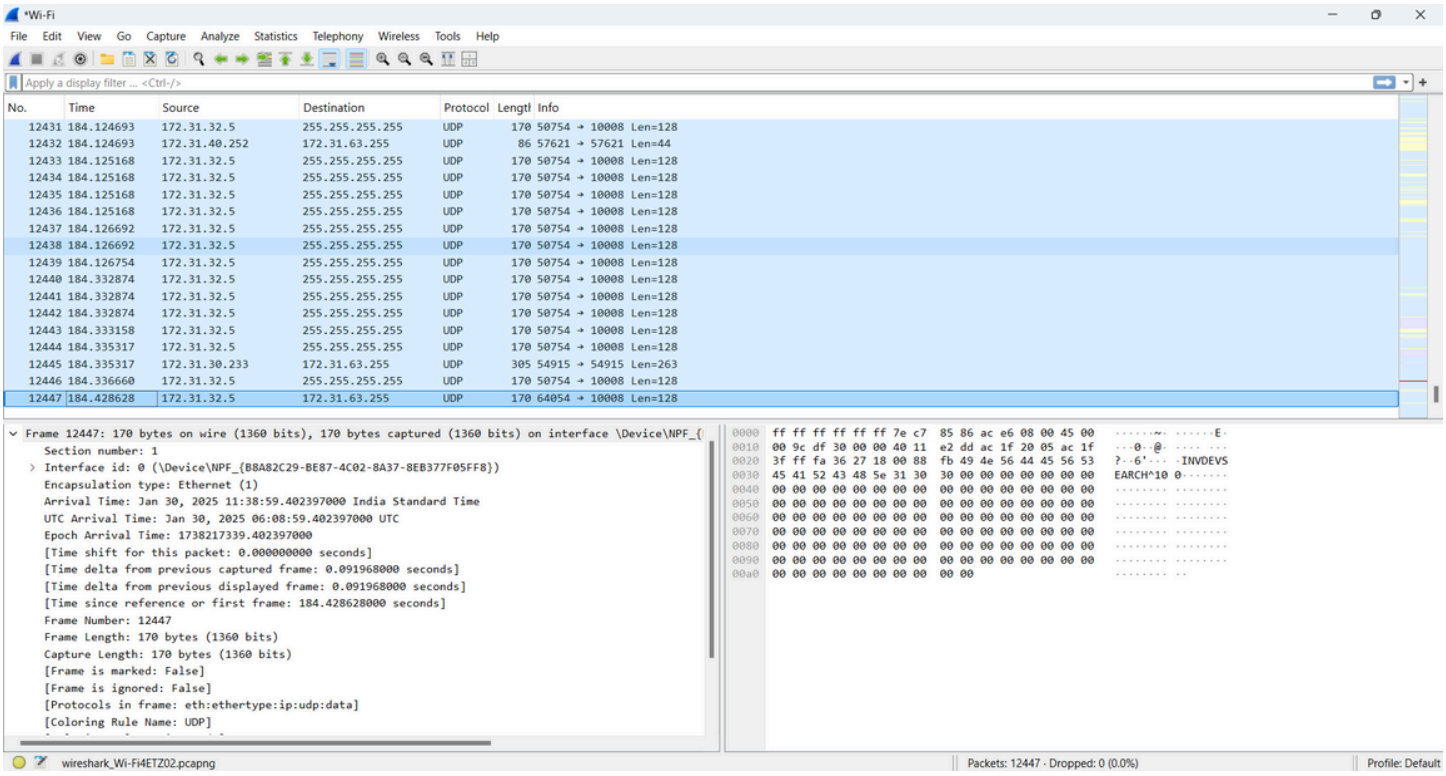
Cyber Security Lab 3

Name : Anubhav Tandon

Roll Number : B22CS013

1. When I started packet capture in Wireshark on my wireless interface, I observed a constant stream of network packets flowing through my network adapter, which is shown in the screenshot below:

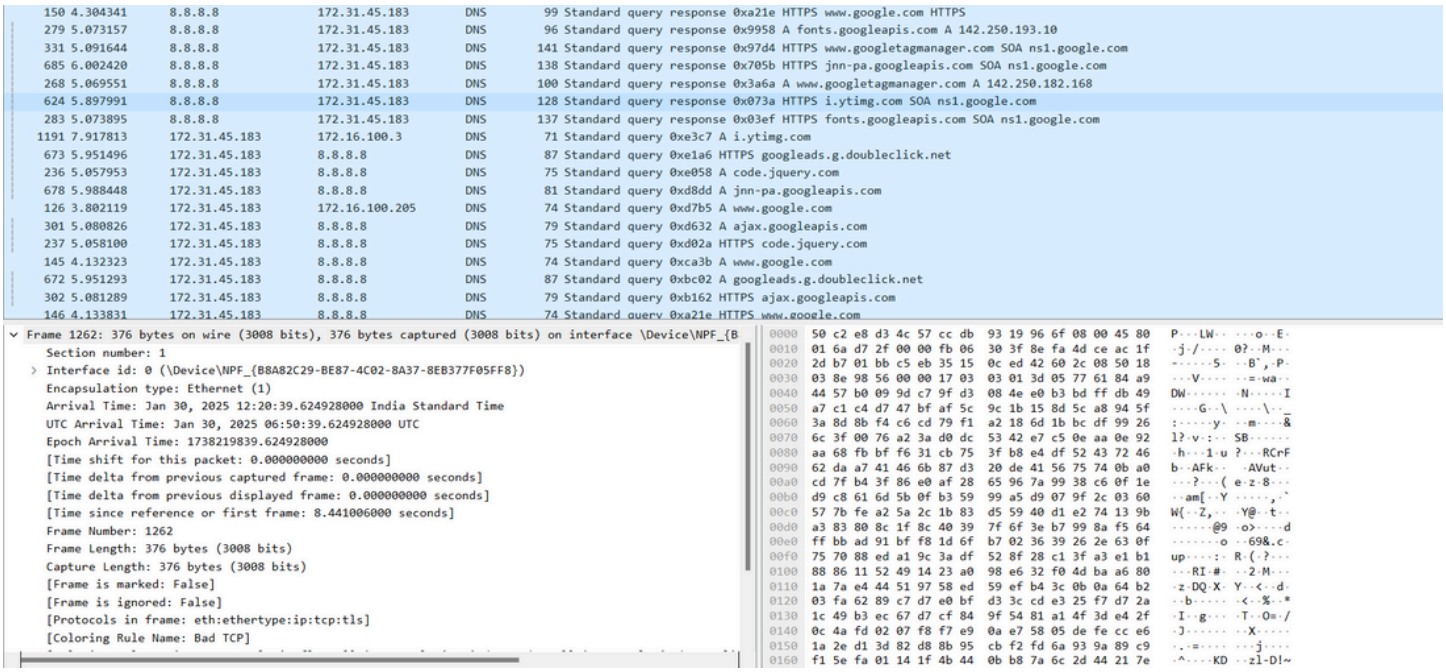


I analyzed the frame 12447, and my observations are as follows:

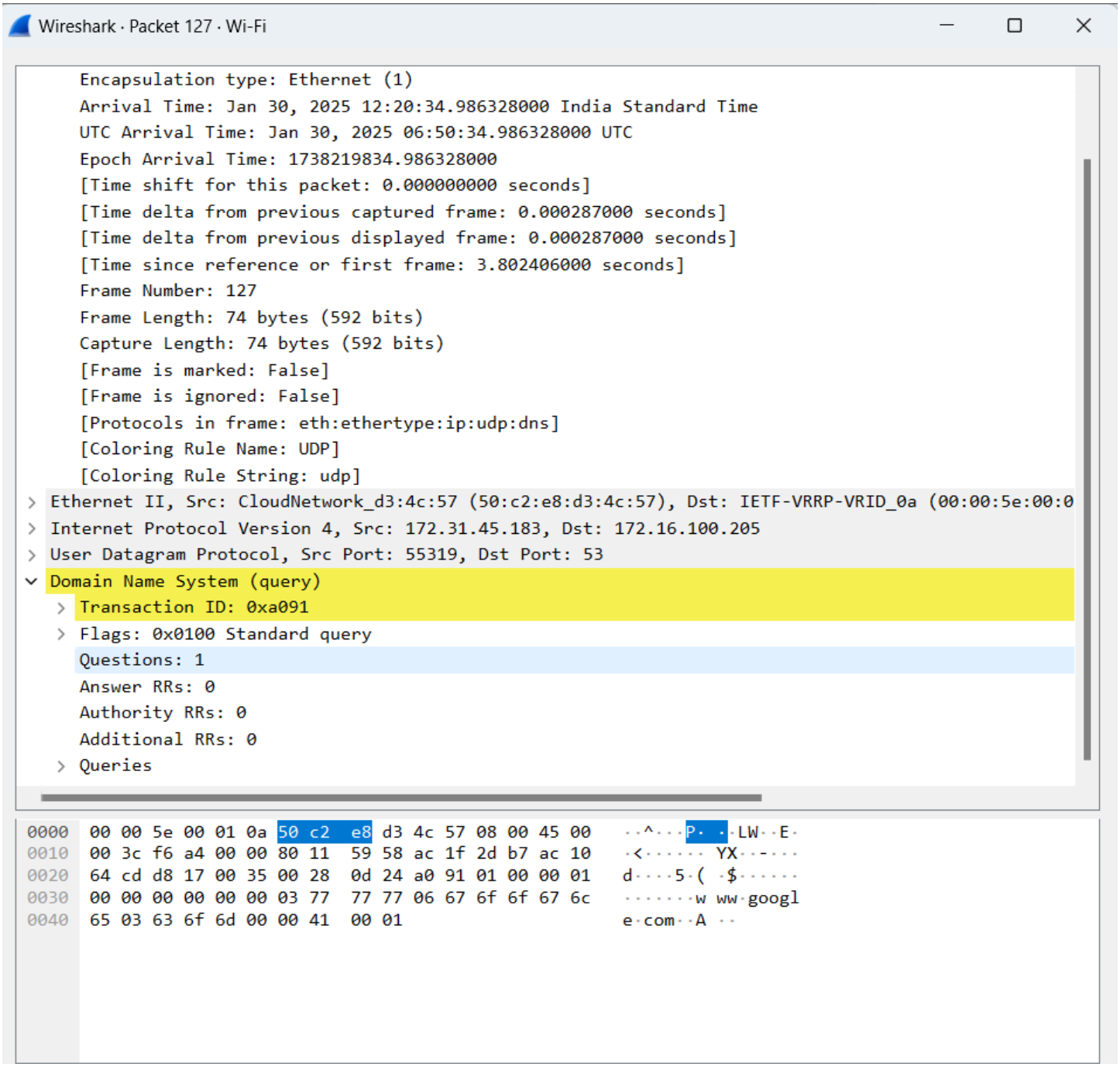
```
Frame 12447: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits) on interface \Device\NPF_{B8A82C29-BE87-4C02-8A37-8EB377F05FF8}, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_{B8A82C29-BE87-4C02-8A37-8EB377F05FF8})
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan 30, 2025 11:38:59.402397000 India Standard Time
    UTC Arrival Time: Jan 30, 2025 06:08:59.402397000 UTC
    Epoch Arrival Time: 1738217339.402397000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.091968000 seconds]
    [Time delta from previous displayed frame: 0.091968000 seconds]
    [Time since reference or first frame: 184.428628000 seconds]
    Frame Number: 12447
    Frame Length: 170 bytes (1360 bits)
    Capture Length: 170 bytes (1360 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:udp:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
> Ethernet II, Src: 7e:c7:85:86:ac:e6 (7e:c7:85:86:ac:e6), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 172.31.32.5, Dst: 172.31.63.255
> User Datagram Protocol, Src Port: 64054, Dst Port: 10008
> Data (128 bytes)
```

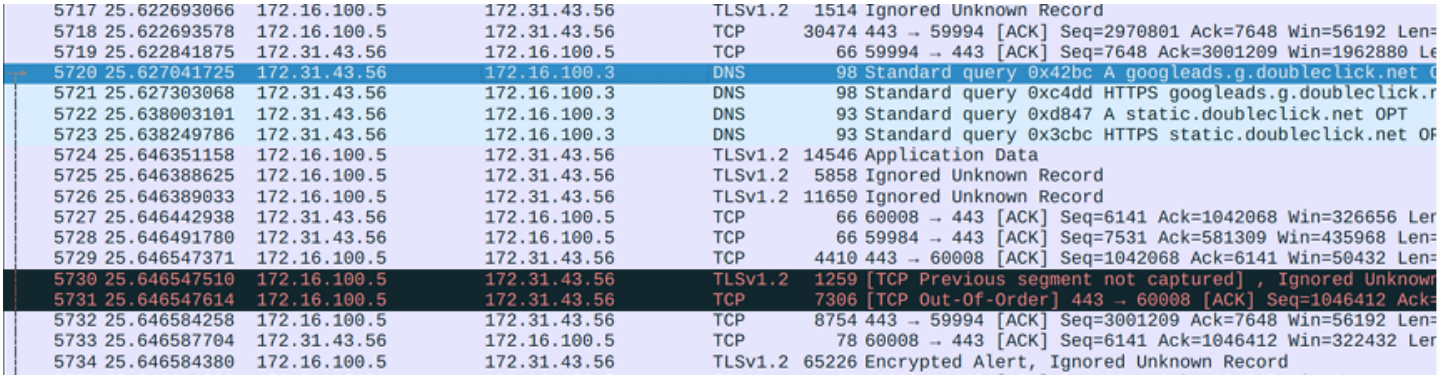
- There's a surprising amount of background network traffic even when I'm not actively browsing or using the internet. This includes:
- Various broadcast messages
- DNS queries
- ARP requests and responses
- The packets are color-coded in Wireshark:
- Light purple packets for TCP
- Light blue for UDP
- Black for TCP packets with problems
- Each packet shows important information in the columns
- Time
- Source IP address
- Destination IP address
- Protocol (TCP, UDP, DNS, etc.)
- Length
- Info about what the packet contains
- The traffic is continuous and updates in real-time, showing how my computer is constantly communicating with other devices on the network and the internet.

2 .



- Yes, I could see the DNS request! A DNS query request was sent to the DNS server for www.iitj.ac.in.
- A DNS query packet sent to my DNS server
- The response containing the IP address for www.iitj.ac.in
- The DNS protocol appears in light blue in Wireshark
- I noticed it used UDP port 53, which is standard for DNS





- I observed TCP 3-way handshake: SYN packet from my computer
- SYN-ACK from the server

- ACK from my computer

This established the TCP connection needed for HTTP

Yes, I could see multiple HTTP requests and responses:

http

No.	Time	Source	Destination	Protocol	Length	Info
1344	23.191219182	172.31.43.56	16.182.98.237	HTTP	342	GET /Entrust-OVTLS-I-R1.cer HTTP/1.1
1391	23.458981590	16.182.98.237	172.31.43.56	HTTP	248	HTTP/1.1 200 OK (application/pkix-cert)

The IP Address of the IITJ server is **172.16.100.5** As we can see in the screenshot below.

702	6.080102	172.31.45.183	142.250.206.162	TLSv1.2	128	Change Cipher Spec, Application Data
716	6.106543	172.31.45.183	142.250.194.234	TLSv1.2	128	Change Cipher Spec, Application Data
257	5.064312	172.31.45.183	172.16.100.5	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
261	5.064846	172.31.45.183	172.16.100.5	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
263	5.065763	172.31.45.183	172.16.100.5	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
633	5.916865	172.31.45.183	172.16.100.5	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
554	5.455641	172.31.45.183	172.16.100.115	TLSv1.3	670	Client Hello (SNI=aide.iitj.ac.in)
415	5.146203	172.31.45.183	151.101.66.137	TLSv1.3	542	Client Hello (SNI=code.jquery.com)
335	5.092669	172.31.45.183	142.250.193.10	TLSv1.2	832	Client Hello (SNI=fonts.googleapis.com)
683	5.995392	172.31.45.183	142.250.206.162	TLSv1.2	878	Client Hello (SNI=googleads.g.doubleclick.net)
251	5.061913	172.31.45.183	172.16.100.5	TLSv1.2	618	Client Hello (SNI=iitj.ac.in)
629	5.908999	172.31.45.183	172.16.100.5	TLSv1.2	522	Client Hello (SNI=iitj.ac.in)
693	6.017968	172.31.45.183	142.250.194.234	TLSv1.2	898	Client Hello (SNI=jnn-pa.googleapis.com)
158	4.321035	172.31.45.183	142.250.194.4	TLSv1.3	430	Client Hello (SNI=www.google.com)
339	5.095413	172.31.45.183	142.250.182.168	TLSv1.2	872	Client Hello (SNI=www.googletagmanager.com)
247	5.060722	172.31.45.183	172.16.100.5	TLSv1.2	526	Client Hello (SNI=www.iitj.ac.in)
249	5.061165	172.31.45.183	172.16.100.5	TLSv1.2	558	Client Hello (SNI=www.iitj.ac.in)

3.

No.	Time	Source	Destination	Protocol	Lengtl	Info
1016	7.265958	142.250.77.206	172.31.45.183	TLSv1.2	737	[TCP Previous segment not captured] , Application Data
1037	7.268881	142.250.77.206	172.31.45.183	TLSv1.2	1350	[TCP Previous segment not captured] , Application Data
1041	7.269065	142.250.77.206	172.31.45.183	TLSv1.2	617	[TCP Previous segment not captured] , Application Data
1262	8.441006	142.250.77.206	172.31.45.183	TLSv1.2	376	[TCP Previous segment not captured] , Application Data
496	5.175943	142.250.193.10	172.31.45.183	TLSv1.2	111	[TCP Previous segment not captured] , Ignored Unknown Record
507	5.183886	142.250.182.168	172.31.45.183	TLSv1.2	115	[TCP Previous segment not captured] , Ignored Unknown Record
698	6.079242	142.250.206.162	172.31.45.183	TLSv1.2	118	[TCP Previous segment not captured] , Ignored Unknown Record
712	6.105906	142.250.194.234	172.31.45.183	TLSv1.2	112	[TCP Previous segment not captured] , Ignored Unknown Record
764	6.267528	142.250.194.234	172.31.45.183	TLSv1.2	265	[TCP Previous segment not captured] , Ignored Unknown Record
1081	7.294793	142.250.77.206	172.31.45.183	TLSv1.2	439	[TCP Previous segment not captured] , Ignored Unknown Record
1140	7.777513	142.250.77.206	172.31.45.183	TLSv1.2	99	[TCP Previous segment not captured] , Ignored Unknown Record
1185	7.795959	142.250.77.206	172.31.45.183	TLSv1.2	778	[TCP Previous segment not captured] , Ignored Unknown Record
163	4.402101	142.250.194.4	172.31.45.183	TCP	1196	[TCP Previous segment not captured] 443 → 50716 [PSH, ACK] Seq=4237 Ack=1789 Win=268032 Len=1142 [TCP PDU reassembled in 164]
490	5.173887	172.16.100.5	172.31.45.183	TCP	1363	[TCP Previous segment not captured] 443 → 50717 [PSH, ACK] Seq=299813 Ack=3151 Win=36096 Len=1309 [TCP PDU reassembled in 494], A
270	5.071096	172.16.100.5	172.31.45.183	TCP	298	[TCP Previous segment not captured] 443 → 50717 [PSH, ACK] Seq=4518 Ack=3151 Win=36096 Len=244 [TCP PDU reassembled in 274]
327	5.091325	172.16.100.5	172.31.45.183	TCP	891	[TCP Previous segment not captured] 443 → 50717 [PSH, ACK] Seq=63144 Ack=3151 Win=36096 Len=837 [TCP PDU reassembled in 328]
362	5.128928	172.16.100.5	172.31.45.183	TCP	473	[TCP Previous segment not captured] 443 → 50717 [PSH, ACK] Seq=80041 Ack=3151 Win=36096 Len=419 [TCP PDU reassembled in 364]
370	5.129411	172.16.100.5	172.31.45.183	TCP	407	[TCP Previous segment not captured] 443 → 50717 [PSH, ACK] Seq=96520 Ack=3151 Win=36096 Len=353 [TCP PDU reassembled in 376]
782	6.280046	142.250.194.234	172.31.45.183	TCP	470	[TCP Previous segment not captured] 443 → 50726 [PSH, ACK] Seq=46808 Ack=3113 Win=267008 Len=416 [TCP PDU reassembled in 989]
1196	7.966257	142.250.77.206	172.31.45.183	TCP	93	[TCP Retransmission] 443 → 50667 [PSH, ACK] Seq=252483 Ack=21266 Win=911 Len=39
1265	8.441006	142.250.77.206	172.31.45.183	TCP	124	[TCP Retransmission] 443 → 50667 [PSH, ACK] Seq=340458 Ack=21672 Win=910 Len=70
1106	7.504361	142.250.194.46	172.31.45.183	TCP	85	[TCP Retransmission] 443 → 50680 [PSH, ACK] Seq=2953 Ack=5029 Win=1023 Len=31
1097	7.504216	142.250.194.234	172.31.45.183	TCP	93	[TCP Retransmission] 443 → 50726 [PSH, ACK] Seq=92778 Ack=5867 Win=264960 Len=39
390	5.135934	172.16.100.5	172.31.45.183	TCP	2974	[TCP Spurious Retransmission] 443 → 50717 [ACK] Seq=69821 Ack=3151 Win=36096 Len=2920 [TCP PDU reassembled in 364]

```
> Frame 686: 337 bytes on wire (2696 bits), 337 bytes captured (2696 bits) on interface \Device\NPF_{B8A8
> Ethernet II, Src: Cisco_19:96:6f (cc:db:93:19:96:6f), Dst: CloudNetwork_d3:4c:57 (50:c2:e8:d3:4c:57)
> Internet Protocol Version 4, Src: 8.8.8.8, Dst: 172.31.45.183
> User Datagram Protocol, Src Port: 53, Dst Port: 54555
> Domain Name System (response)
```

```
0000  50 c2 e8 d3 4c 57 cc db  93 19 96 6f 08 00 45 80   P···LW·· ···o··E·
0010  01 43 a1 de 00 00 fb 11  32 65 08 08 08 08 ac 1f   ·C····· 2e·····
0020  2d b7 00 35 d5 1b 01 2f  80 43 d8 dd 81 80 00 01   -··5···/ ·C·····
0030  00 10 00 00 00 00 06 6a  6e 6e 2d 70 61 0a 67 6f   ·······j nn-pa·go
0040  6f 67 6c 65 61 70 69 73  03 63 6f 6d 00 00 01 00   ogleapis ·com····
0050  01 c0 0c 00 01 00 01 00  00 01 0a 00 04 8e fa c2   ········ ········
0060  ea c0 0c 00 01 00 01 00  00 01 0a 00 04 d8 3a c8   ········ ·····:·
0070  aa c0 0c 00 01 00 01 00  00 01 0a 00 04 8e fa cf   ········ ········
0080  ea c0 0c 00 01 00 01 00  00 01 0a 00 04 8e fa c2   ········ ········
0090  2a c0 0c 00 01 00 01 00  00 01 0a 00 04 8e fa c2   *····· ·· ·····
00a0  6a c0 0c 00 01 00 01 00  00 01 0a 00 04 ac d9 a6   j····· ·· ·····
00b0  0a c0 0c 00 01 00 01 00  00 01 0a 00 04 8e fa c2   ····· ·· ·····
```

Packets highlighted in black color signify TCP segments with problems or errors. Based on the packet capture analysis, these highlighted packets indicate several types of TCP issues:

1. **"TCP Previous segment not captured":**
- Seen in multiple packets from IPs 142.250.77.206 and 142.250.194.234 to 172.31.45.183
- Example lengths include 737, 1350, and 617 bytes
- These show missing segments in the TCP conversation
2. **"TCP Retransmission":**

- Found in frames 1196, 1265, and 1106
- All with small packet sizes (93, 124, and 85 bytes, respectively)
- Shows packets that needed to be sent again

3. **"TCP Spurious Retransmission":**

- Seen in frame 398 (5.135934)
- From 172.16.100.5 to 172.31.45.183
- Length of 2974 bytes
- Indicates an unnecessary retransmission occurred

4. Here are 5 different filters and their observations:

1. **tcp.flags.syn == 1**
- Displays packets that initiate TCP connections (SYN packets)
- Helped visualize new connection establishments between my computer and servers
2. **ip.addr == 172.31.45.183**
- Shows all traffic involving my computer's IP address
- Useful for isolating my network communication from other traffic
3. **dns**
- Revealed DNS queries and responses for domain name resolution
- Allowed me to track which domain names were being looked up during browsing
4. **http.request**
- Filtered to show only HTTP request packets
- It made it easy to see what resources websites were requesting
5. **tcp.window_size == 0**
- Shows packets where TCP flow control window is zero
- Helped identify potential network congestion points

5. The filter command to list all outgoing traffic would be:

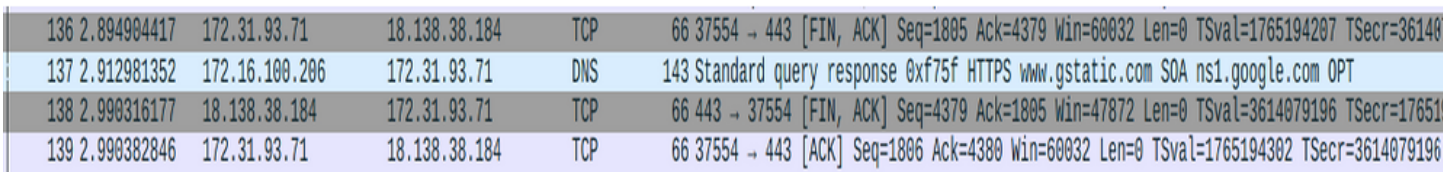
ip.src == 172.31.45.183

This displays:

- All packets where my IP (172.31.45.183) is the source
- Only outbound traffic, filtering out incoming packets

6. Start a new packet capture to now visit an external website, say www.cricinfo.com. Can you show the 3-way TCP handshake happening? Can you see your IITJ proxy in between? What is its IP address?

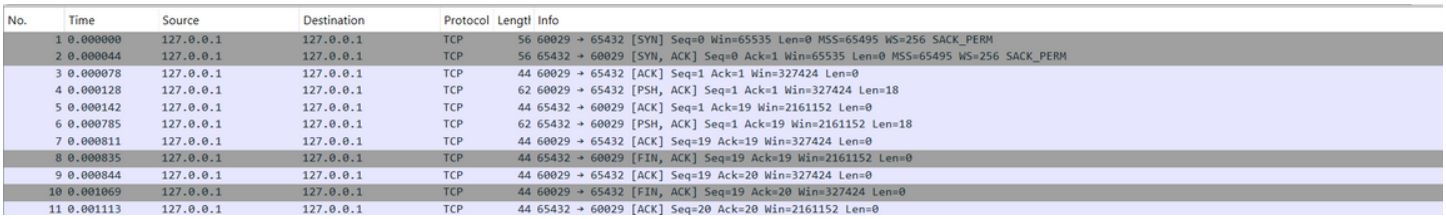
IP address of cricinfo.com : 18.138.38.184 Source IP Address of my machine : 172.31.93.71 IITJ Proxy : 172.16.100.206



7. DNS uses UDP because it is a **lightweight, connectionless protocol that allows for quick, simple, and efficient communication.** DNS queries are usually small, and the protocol does not require the overhead of establishing a connection or ensuring reliability through retransmissions, as failed queries are typically retried.

On the other hand, **HTTP uses TCP because it requires reliable, ordered delivery of data.** TCP's connection-oriented nature ensures that large files, such as webpages and their associated resources, are transferred correctly, with error correction, flow control, and retransmission capabilities. This makes TCP ideal for the more complex, multi-step communication needed for web browsing, where data integrity is crucial.

8.



No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	60029 → 65432 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
2	0.000044	127.0.0.1	127.0.0.1	TCP	56	65432 → 60029 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
3	0.000078	127.0.0.1	127.0.0.1	TCP	44	60029 → 65432 [ACK] Seq=1 Ack=1 Win=327424 Len=0
4	0.000128	127.0.0.1	127.0.0.1	TCP	62	60029 → 65432 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=18
5	0.000142	127.0.0.1	127.0.0.1	TCP	44	65432 → 60029 [ACK] Seq=1 Ack=19 Win=2161152 Len=0
6	0.000785	127.0.0.1	127.0.0.1	TCP	62	65432 → 60029 [PSH, ACK] Seq=1 Ack=19 Win=2161152 Len=18
7	0.000811	127.0.0.1	127.0.0.1	TCP	44	60029 → 65432 [ACK] Seq=19 Ack=19 Win=327424 Len=0
8	0.000835	127.0.0.1	127.0.0.1	TCP	44	65432 → 60029 [FIN, ACK] Seq=19 Ack=19 Win=2161152 Len=0
9	0.000844	127.0.0.1	127.0.0.1	TCP	44	60029 → 65432 [ACK] Seq=19 Ack=20 Win=327424 Len=0
10	0.001069	127.0.0.1	127.0.0.1	TCP	44	60029 → 65432 [FIN, ACK] Seq=19 Ack=20 Win=327424 Len=0
11	0.001113	127.0.0.1	127.0.0.1	TCP	44	65432 → 60029 [ACK] Seq=20 Ack=20 Win=2161152 Len=0

Server.py > ...

```python
# server.py
import socket

# Set up server details
host = '127.0.0.1'  # Localhost
port = 65432  # Port to listen on

# Create a TCP/IP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
server_socket.bind((host, port))

# Listen for incoming connections
server_socket.listen(1)
print(f"Server listening on {host}:{port}...")

# Accept a connection from a client
conn, addr = server_socket.accept()
print(f"Connected by {addr}")

# Receive and send data
data = conn.recv(1024)
print(f"Received from client: {data.decode()}")
conn.sendall(b"Hello from server!")

# Close the connection
conn.close()

```

```python
# client.py
import socket

# Set up server details
host = '127.0.0.1'  # Localhost
port = 65432  # Port to connect to

# Create a TCP/IP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server
client_socket.connect((host, port))

# Send data to the server
client_socket.sendall(b"Hello from client!")

# Receive data from the server
data = client_socket.recv(1024)
print(f"Received from server: {data.decode()}")

# Close the socket
client_socket.close()
```

- The communication is between a client and a server both running on the **localhost (127.0.0.1)**.
- The client and server are using different **ports (60029 and 65432)** for communication.
- The packets show a typical TCP data exchange where **data is sent and acknowledged between the client and server.**
- The use of **PSH flags** indicates that the data is being pushed to the receiving application immediately.
- The sequence and acknowledgment numbers are correctly incremented, showing a reliable data transfer process.