CS-342

Networks Lab

Assignment-2

Wireshark Analysis of HTTP, Web Cache, DNS and Transport Layer Protocols

Group 6:

Achyut Dhiman	210101006
Balaji S.	210101028
Satyarth Gupta	210101095

CASE STUDY : YOUTUBE

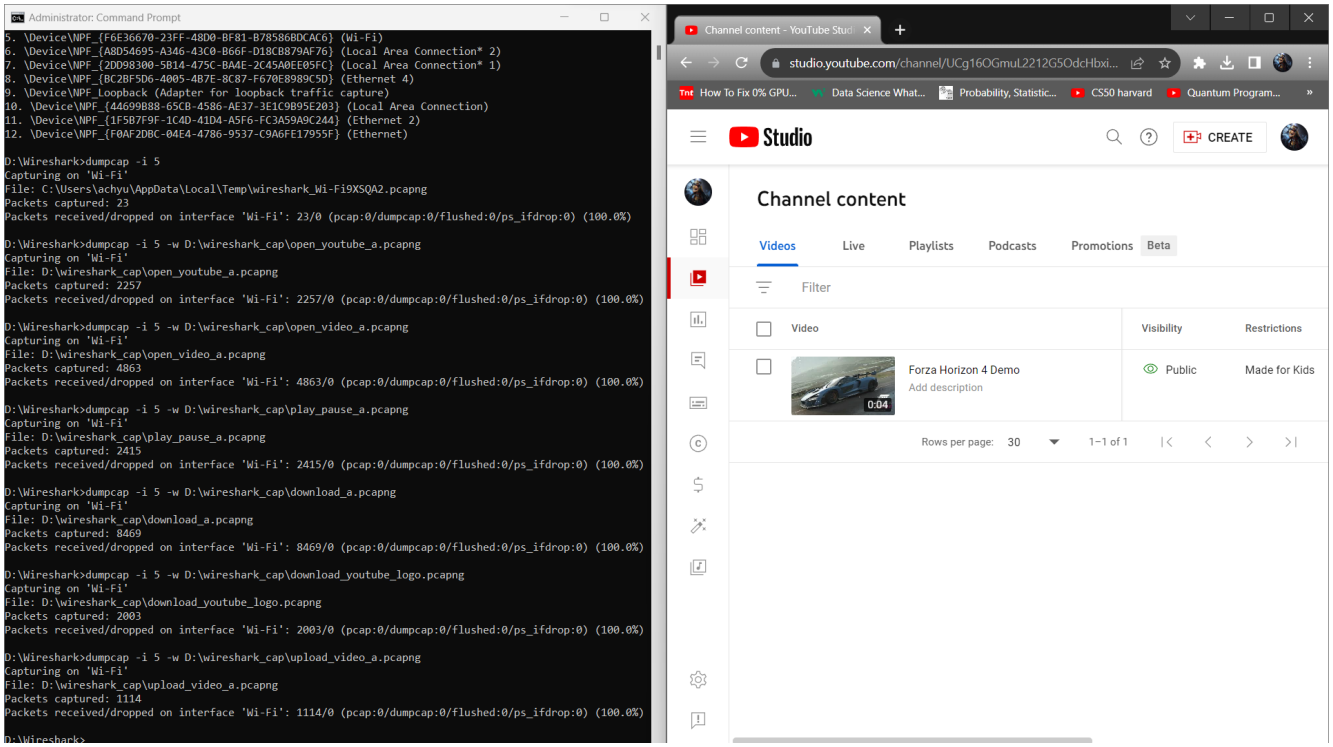
- **Introduction**:

  The purpose of this report is to perform an analysis of network traffic related to the website www.youtube.com. The analysis aims to uncover the underlying network protocols, communication patterns, caching mechanisms, and network performance metrics associated with accessing and interacting with YouTube.

- **Methodology**:

  - ☐ **Data capture:** Network traffic data was collected using the Wireshark network protocol analyzer in a controlled environment. Multiple scenarios were examined, including opening the YouTube website, watching videos, and performing various interactions on the platform

  - ☐ **Data Segmentation** : Filters were strategically applied to segment the captured network packets, ensuring that only relevant traffic associated with www.youtube.com was included in the analysis.



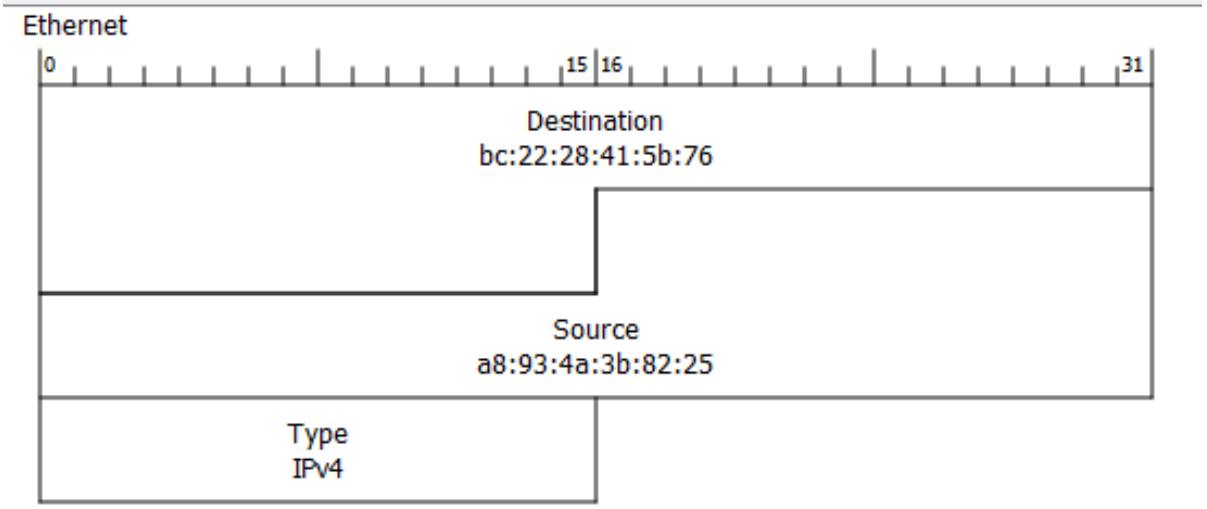
- **ANALYSIS:**

☐ Task 1: Protocols used at Different Layers

❖ Ethernet(Link) Layer:

Ethernet frames facilitate local network communication using MAC addresses for device identification.

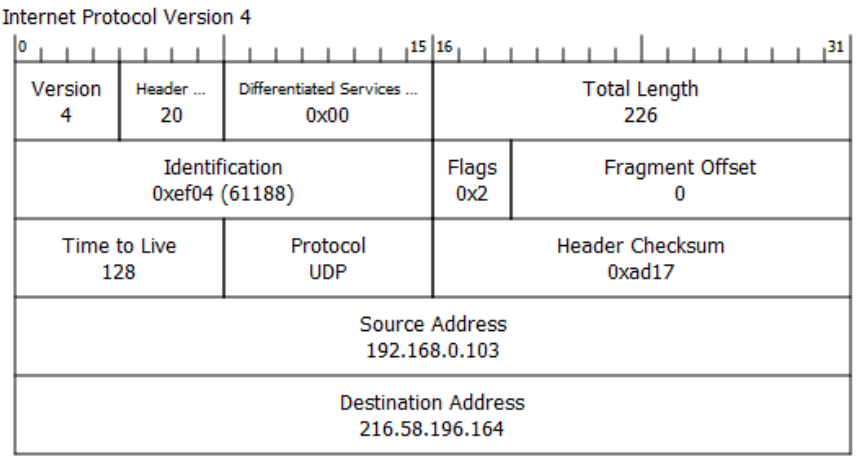
```
Ethernet II, Src: Chongqin_3b:82:25 (a8:93:4a:3b:82:25), Dst: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
  ▼ Destination: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
        Address: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▼ Source: Chongqin_3b:82:25 (a8:93:4a:3b:82:25)
        Address: Chongqin_3b:82:25 (a8:93:4a:3b:82:25)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```



❖ IP(Network) Layer:

IP protocol handles routing and addressing, facilitating communication between local devices and YouTube servers.

```
Internet Protocol Version 4, Src: 192.168.0.103 (192.168.0.103), Dst: www.google.com (216.58.196.164)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1157
    Identification: 0xeefd (61181)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0xa97b [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.0.103 (192.168.0.103)
    Destination Address: www.google.com (216.58.196.164)
```



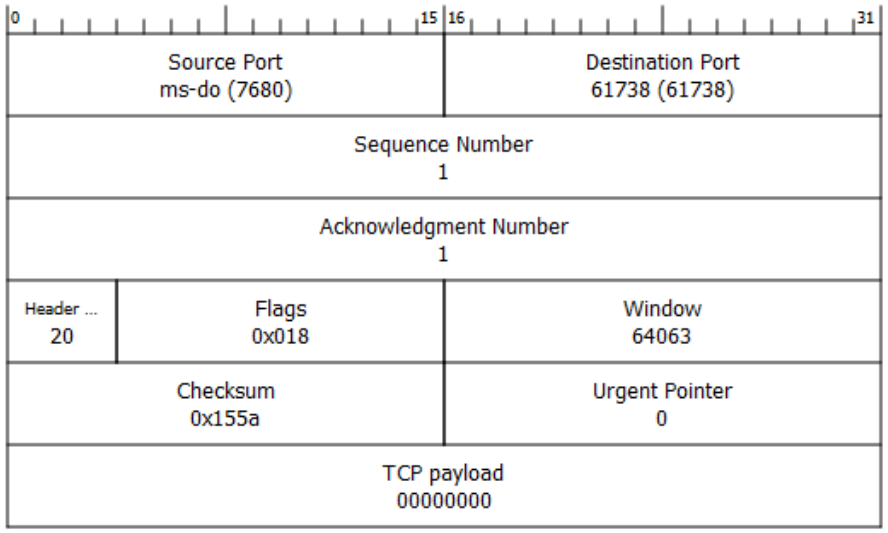
❖ Transport Layer:

TCP and UDP serve as transport layer protocols, responsible for end-to-end data delivery.

TCP:

```
Transmission Control Protocol, Src Port: 64551 (64551), Dst Port: http (80), Seq: 0, Len: 0
    Source Port: 64551 (64551)
    Destination Port: http (80)
    [Stream index: 2]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1347528746
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0x0895 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  > [Timestamps]
```

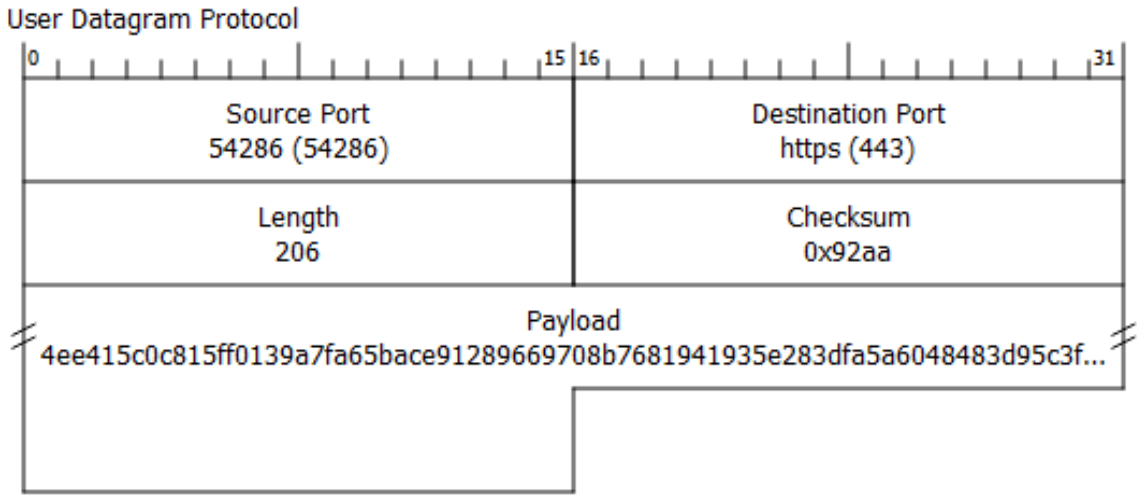
Transmission Control Protocol



UDP:

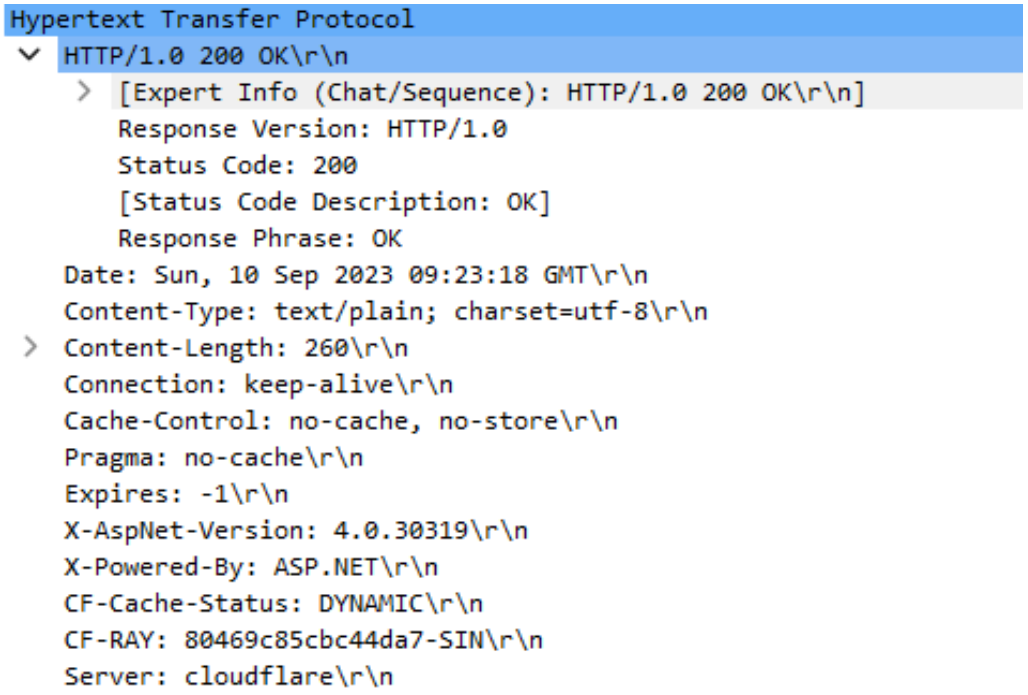
```
User Datagram Protocol, Src Port: 54286 (54286), Dst Port: https (443)
    Source Port: 54286 (54286)
    Destination Port: https (443)
    Length: 1137
    Checksum: 0xcda7 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  > [Timestamps]
    UDP payload (1129 bytes)
```

❖ Application Layer:

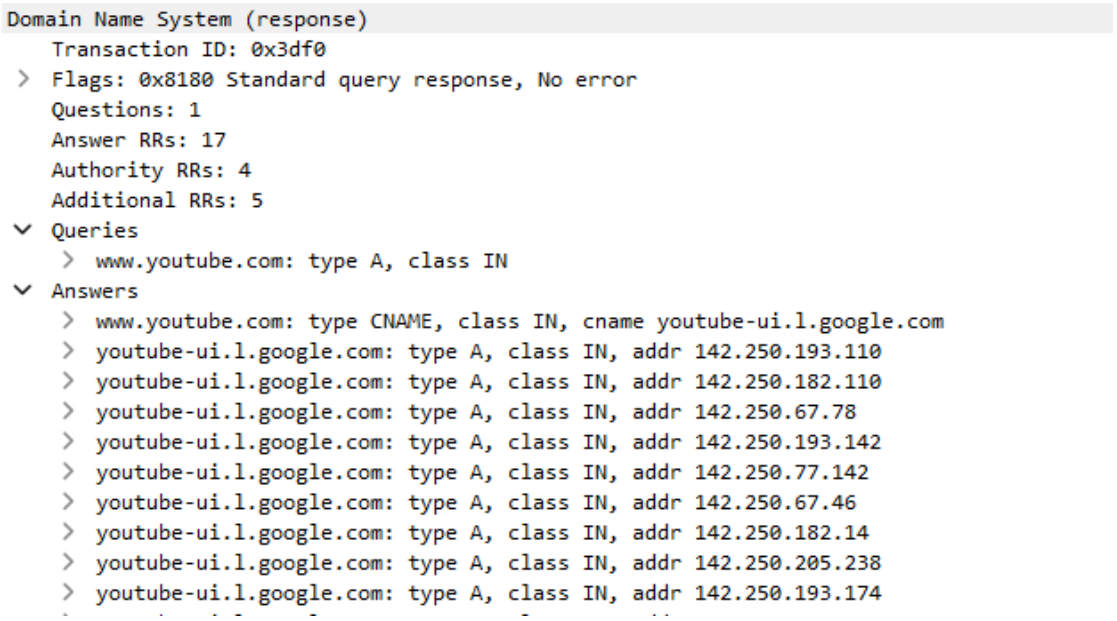


HTTP, DNS, QUIC (Quick UDP Internet Connections), TLSv1.3 (Transport Level Security) are being utilized for web communication with YouTube servers.

HTTP:


Hypertext Transfer Protocol
HTTP/1.0 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.0 200 OK\r\n]
Response Version: HTTP/1.0
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK
Date: Sun, 10 Sep 2023 09:23:18 GMT\r\n
Content-Type: text/plain; charset=utf-8\r\n
Content-Length: 260\r\n
Connection: keep-alive\r\n
Cache-Control: no-cache, no-store\r\n
Pragma: no-cache\r\n
Expires: -1\r\n
X-AspNet-Version: 4.0.30319\r\n
X-Powered-By: ASP.NET\r\n
CF-Cache-Status: DYNAMIC\r\n
CF-RAY: 80469c85cbc44da7-SIN\r\n
Server: cloudflare\r\n

DNS:


Domain Name System (response)
Transaction ID: 0x3df0
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 17
Authority RRs: 4
Additional RRs: 5
Queries
www.youtube.com: type A, class IN
Answers
www.youtube.com: type CNAME, class IN, cname youtube-ui.l.google.com
youtube-ui.l.google.com: type A, class IN, addr 142.250.193.110
youtube-ui.l.google.com: type A, class IN, addr 142.250.182.110
youtube-ui.l.google.com: type A, class IN, addr 142.250.67.78
youtube-ui.l.google.com: type A, class IN, addr 142.250.193.142
youtube-ui.l.google.com: type A, class IN, addr 142.250.77.142
youtube-ui.l.google.com: type A, class IN, addr 142.250.67.46
youtube-ui.l.google.com: type A, class IN, addr 142.250.182.14
youtube-ui.l.google.com: type A, class IN, addr 142.250.205.238
youtube-ui.l.google.com: type A, class IN, addr 142.250.193.174

QUIC:

```
QUIC IETF
 ⌄ QUIC Connection information
       [Connection Number: 1]
   [Packet Length: 935]
 ⌄ QUIC Short Header
       0... .... = Header Form: Short Header (0)
       .1.. .... = Fixed Bit: True
       ..0. .... = Spin Bit: False
   Remaining Payload: dc5fb46530c68cd32af657298544ef26a4bd2e7f489be9d53398e9d771187183a0405c40…
```

TLSv1.3:

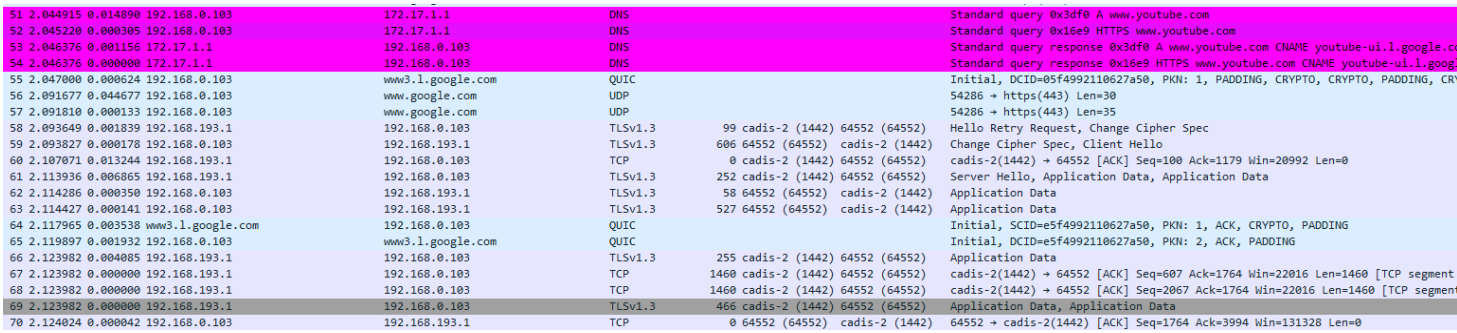
```
Transport Layer Security
 ⌄ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
       Content Type: Handshake (22)
       Version: TLS 1.0 (0x0301)
       Length: 512
    ⌄ Handshake Protocol: Client Hello
          Handshake Type: Client Hello (1)
          Length: 508
          Version: TLS 1.2 (0x0303)
          Random: c5e7cf152f86f81066e98b38d07b1e4cea82b993976a6aa67098a4ea23f06303
          Session ID Length: 32
          Session ID: 82d1b3a29c05554530afec43956d2483e6bb808edffe2b9199626461671bd591
          Cipher Suites Length: 32
        › Cipher Suites (16 suites)
          Compression Methods Length: 1
        › Compression Methods (1 method)
          Extensions Length: 403
        › Extension: Reserved (GREASE) (len=0)
        › Extension: ec_point_formats (len=2)
        › Extension: server_name (len=16)
        › Extension: application_settings (len=5)
        › Extension: session ticket (len=0)
```

☐ Task 2: Observed Protocol Fields

26	1.360140	0.007381	192.168.0.103	172.67.9.68	TCP	0 64551 (64551)	http (80)	64551 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS
27	1.362224	0.002084	172.67.9.68	192.168.0.103	TCP	0 http (80)	64551 (64551)	http(80) → 64551 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0
28	1.362288	0.000064	192.168.0.103	172.67.9.68	TCP	0 64551 (64551)	http (80)	64551 → http(80) [ACK] Seq=1 Ack=1 Win=131328 Len=0
29	1.362350	0.000062	192.168.0.103	172.67.9.68	TCP	501 64551 (64551)	http (80)	64551 → http(80) [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=5
30	1.364733	0.002383	172.67.9.68	192.168.0.103	TCP	0 http (80)	64551 (64551)	http(80) → 64551 [ACK] Seq=1 Ack=502 Win=19456 Len=0
31	1.364746	0.000013	192.168.0.103	172.67.9.68	HTTP	81 64551 (64551)	http (80)	POST /api.ashx?method=gSR&checksum=9d767546ceb253ae86f74
32	1.365891	0.001145	172.67.9.68	192.168.0.103	TCP	0 http (80)	64551 (64551)	http(80) → 64551 [ACK] Seq=1 Ack=583 Win=19456 Len=0
33	1.448326	0.082435	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=1246
34	1.448326	0.000000	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=971
35	1.448326	0.000000	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=25
36	1.448326	0.000000	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=77
37	1.448686	0.000360	192.168.0.103	www.google.com	UDP			54286 → https(443) Len=35
38	1.448859	0.000173	192.168.0.103	www.google.com	UDP			54286 → https(443) Len=31
39	1.454384	0.005525	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=1027
40	1.454384	0.000000	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=25
41	1.454521	0.000137	192.168.0.103	www.google.com	UDP			54286 → https(443) Len=35
42	1.455072	0.000551	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=261
43	1.455157	0.000085	192.168.0.103	www.google.com	UDP			54286 → https(443) Len=31
44	1.548860	0.093703	www.google.com	192.168.0.103	UDP			https(443) → 54286 Len=23
45	1.999510	0.450650	192.168.0.103	192.168.193.1	TCP	0 64552 (64552)	cadis-2 (1442)	64552 → cadis-2(1442) [SYN] Seq=0 Win=64240 Len=0 MSS=14
46	2.001092	0.001582	192.168.193.1	192.168.0.103	TCP	0 cadis-2 (1442)	64552 (64552)	cadis-2(1442) → 64552 [SYN, ACK] Seq=0 Ack=1 Win=18352 L
47	2.001170	0.000078	192.168.0.103	192.168.193.1	TCP	0 64552 (64552)	cadis-2 (1442)	64552 → cadis-2(1442) [ACK] Seq=1 Ack=1 Win=131328 Len=0
48	2.001400	0.000230	192.168.0.103	192.168.193.1	TLSv1.3	572 64552 (64552)	cadis-2 (1442)	Client Hello
49	2.002497	0.001097	192.168.193.1	192.168.0.103	TCP	0 cadis-2 (1442)	64552 (64552)	cadis-2(1442) → 64552 [ACK] Seq=1 Ack=573 Win=19968 Len=

Here, we see the host(our computer) trying to establish a TCP connection (highlighted in green) with Destination IP Address: 172.67.9.68 on port 80 which(HTTP). It is a part of the TCP three-way handshake process, where the sender is synchronizing with the receiver to establish a reliable connection for potential HTTP communication. We also see that the port number 443 is used for HTTPS communication.

A DNS packet:



Here, we see the packet number 51 sending a DNS query(highlighted in pink) to the DNS server at 172.17.1.1. The query is asking for the IPv4 address associated with domain name “www.youtube.com”.
And henceforth, packet number 53 is the DNS response which provides the IPv4 address and additional information including listing of multiple IP addresses that can be used to reach YouTube servers.

```
Ethernet II, Src: Chongqin_3b:82:25 (a8:93:4a:3b:82:25), Dst: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
  Destination: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
       Address: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
       .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
       .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: Chongqin_3b:82:25 (a8:93:4a:3b:82:25)
       Address: Chongqin_3b:82:25 (a8:93:4a:3b:82:25)
       .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
       .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

This is the ethernet layer (link) field with respect to the above DNS query packet. The given information is part of an Ethernet II frame’s header, which describes the source and destination MAC addresses.

(chongqin is probably associated with the manufacturer of our NIC chip).

```
Domain Name System (query)
    Transaction ID: 0x3df0
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.youtube.com: type A, class IN
          Name: www.youtube.com
          [Name Length: 15]
          [Label Count: 3]
          Type: A (Host Address) (1)
          Class: IN (0x0001)
    [Response In: 53]
```

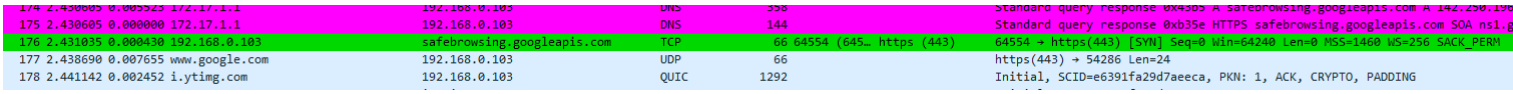
This is the application layer field with respect to the above DNS query packet.A unique identifier is assigned for this DNS query transaction. It helps match DNS responses to the corresponding queries. Questions : 1 indicates that there is one DNS query question in this packet. The domain name being queried, which is "www.youtube.com" has the type "A (IPv4)". [Response in: 53] Indicates that the response to this DNS query is expected in packet number 53.

```
User Datagram Protocol, Src Port: 57150 (57150), Dst Port: domain (53)
    Source Port: 57150 (57150)
    Destination Port: domain (53)
    Length: 41
    Checksum: 0x358d [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
  > [Timestamps]
    UDP payload (33 bytes)
```

This is the transport layer field with respect to the above DNS query packet. It mentions the source port (57150) (sender's port/our) and the destination port (53) (which is commonly associated with the DNS). It

also specifies the length of the UDP packet (41 bytes). The UDP checksum is used for error-checking the integrity of the UDP packet during transmission. The actual payload (actual data) carried by the UDP packet is 33 bytes.

A TCP PACKET:



This is a TCP packet that represents the beginning of a TCP connection to “safebrowsing.googleapis.com” over HTTPS (port 443). Here, 192.168.0.103 is the IP address of our local computer which is initiating a TCP connection to google’s safe browsing service. Here we have included the additional length of the packet field which says that the total length of the packet, including the TCP header and data, is 66 bytes. The source port here (64554) is randomly assigned for this particular communication session. The destination port as we very well know is https (443) which is the standard port for HTTPS. (secure HTTP).

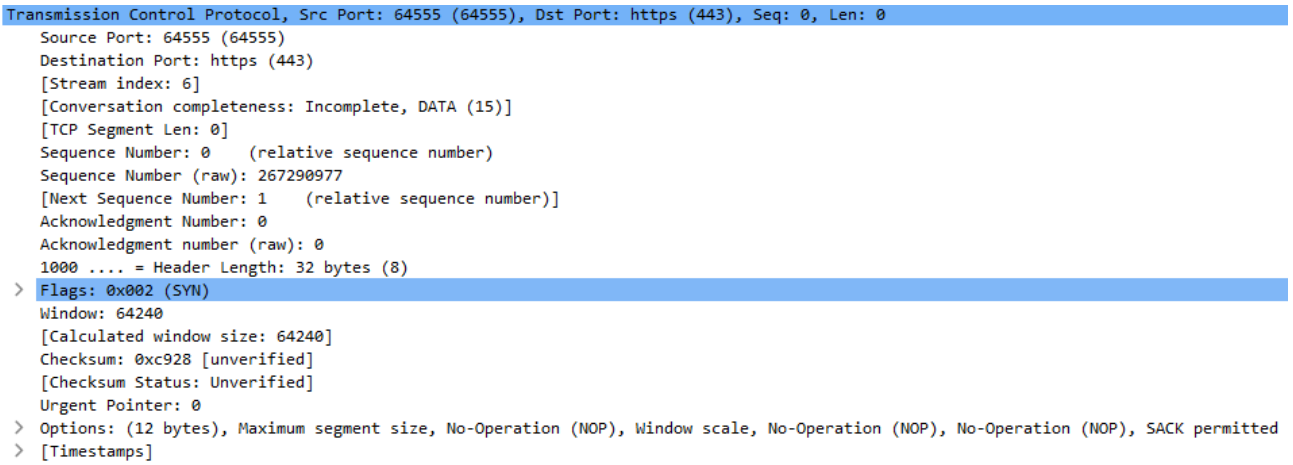
NOTE: Here the [syn] flag indicates that this is the initial part of the TCP three-way handshake. It stands for synchronize.

The Win=64240 represents the window size, which is the maximum amount of data that can be sent by the sender. Similarly with the MSS (maximum segment size) and WS (window scale factor)

```
Internet Protocol Version 4, Src: 192.168.0.103 (192.168.0.103), Dst: safebrowsing.googleapis.com (142.250.196.10)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xe9f5 (59893)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0xfcb9 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.0.103 (192.168.0.103)
    Destination Address: safebrowsing.googleapis.com (142.250.196.10)
```

This is the IP (network) layer field with respect to the above packet. The total length 52 specifies the total length of the IP packet, including the

header and the data (notice that 52<66). The identification field is used for packet identification and fragmentation. It provides a unique identifier for this packet. The Flags: 0*2 (Don't fragment), as the name suggests means that the packet should not be fragmented. The TTL (Time To Live) value of 128 is relatively common and specifies the maximum number of hops (routers) the packet can traverse before being discarded.



This is the transport layer field with respect to the above packet and as usual it mentions the source port and the destination port. It also mentions the Conversation completeness: Incomplete: This field indicates that the conversation or data exchange related to this packet is incomplete, suggesting that more packets are expected to complete the communication.

NOTE: The TCP segment length is 0 indicating that this is not carrying any data payload! (as we know). Also the ACK is 0 indicating that the sender has not received any data yet.

NOTE: The "Flags" field contains various control flags. The SYN flag is set (0x002), indicating the initiation of a TCP connection. This is the first step in the TCP three-way handshake.

The urgent pointer is set to 0, indicating that there is no urgent data in this packet.

A UDP PACKET:

	Time	Delta	Source	Destination	Protocol	Length	Source Port	Destination Port	Info
1	0.000000	0.000000	192.168.0.103	142.250.76.54	UDP	230			54689 → https(443) Len=188
2	0.014483	0.014483	192.168.0.103	youtube-ui.l.google.com	UDP	1286			64008 → https(443) Len=1244
3	0.014622	0.000139	192.168.0.103	youtube-ui.l.google.com	UDP	1292			64008 → https(443) Len=1250
4	0.014672	0.000050	192.168.0.103	youtube-ui.l.google.com	UDP	895			64008 → https(443) Len=853
5	0.063871	0.049199	192.168.0.103	youtube-ui.l.google.com	UDP	1286			64008 → https(443) Len=1244
6	0.064013	0.000142	192.168.0.103	youtube-ui.l.google.com	UDP	1292			64008 → https(443) Len=1250
7	0.064109	0.000096	192.168.0.103	youtube-ui.l.google.com	UDP	840			64008 → https(443) Len=798
8	0.091571	0.027462	192.168.0.103	142.250.76.54	UDP	151			54689 → https(443) Len=109
9	0.124640	0.033069	youtube-ui.l.google.com	192.168.0.103	UDP	71			https(443) → 64008 Len=29
10	0.124640	0.000000	142.250.76.54	192.168.0.103	UDP	70			https(443) → 54689 Len=28
11	0.124640	0.000000	142.250.76.54	192.168.0.103	UDP	1288			https(443) → 54689 Len=1246
12	0.124640	0.000000	142.250.76.54	192.168.0.103	UDP	1292			https(443) → 54689 Len=1250
13	0.124640	0.000000	142.250.76.54	192.168.0.103	UDP	1292			https(443) → 54689 Len=1250
14	0.124640	0.000000	142.250.76.54	192.168.0.103	UDP	366			https(443) → 54689 Len=324
15	0.124640	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	68			https(443) → 64008 Len=26
16	0.125222	0.000582	192.168.0.103	142.250.76.54	UDP	78			54689 → https(443) Len=36
17	0.142238	0.017016	192.168.0.103	youtube-ui.l.google.com	UDP	75			64008 → https(443) Len=33

NOTE: The above excerpt is from the case where the network was analysed when a particular video was started (hence why so many UDP packets)

Consider the second packet, it describes a UDP packet sent from our local computer to the given domain with a destination port of 64008. As usual the source IP and Destination IP are given (for simpler viewing purposes, we have resolved the IPs into domain names.) The length of the UDP packet is also given (1224 bytes).

```
Ethernet II, Src: Chongqin_3b:82:25 (a8:93:4a:3b:82:25), Dst: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
  Destination: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
       Address: D-LinkIn_41:5b:76 (bc:22:28:41:5b:76)
       .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
       .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: Chongqin_3b:82:25 (a8:93:4a:3b:82:25)
       Address: Chongqin_3b:82:25 (a8:93:4a:3b:82:25)
       .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
       .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
```

This is the link layer (ethernet) protocol used by the above packet. It gives the source MAC address (Media access control) of the sender's network interface card (NIC). The source's MAC address is also given which is a globally unique address.The destination MAC address is also given (D-link) (it is also globally unique). We know this because the LG bit (Locally Administered Address) in both the cases is set to 0 indicating that these are globally unique addresses. IG bit (individual address) indicate whether the MAC address is an individual address or a group address. Still, this is 0 in both the cases, hence, individual

addresses.

```
Internet Protocol Version 4, Src: 192.168.0.103 (192.168.0.103), Dst: youtube-ui.l.google.com (142.250.195.78)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1272
    Identification: 0x5f79 (24441)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x8323 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.0.103 (192.168.0.103)
    Destination Address: youtube-ui.l.google.com (142.250.195.78)
```

This is the IP (Network) layer protocol used by the above packet. It is used to carry data from the source IP address to the destination IP address. The version 4 indicates that the packet is using IPv4. The header length specifies the length of the IPv4 header in 32-bit words (each word= 4 bytes) and 5 words, so 20 bytes of data. The flag field includes 3 bits,and the value 0x2 indicates the “Don’t fragment” flag is set. This means that the packet should not be fragmented during transmission. The TTL is also specified. The checksum status is set to unverified.

A QUIC Packet:

54 2.046376 0.000000 172.17.1.1	192.168.0.103	DNS	372		Standard query response 0x16e9 HTTPS
55 2.047000 0.000624 192.168.0.103	www3.l.google.com	QUIC	1292		Initial, DCID=05f4992110627a50, PKN:
56 2.091677 0.044677 192.168.0.103	www.google.com	UDP	72		54286 → https(443) Len=30
57 2.091810 0.000133 192.168.0.103	www.google.com	UDP	77		54286 → https(443) Len=35
58 2.093649 0.001839 192.168.193.1	192.168.0.103	TLSv1.3	153	cadis-2 (1… 64552 (64552)	Hello Retry Request, Change Cipher S
59 2.093827 0.000178 192.168.0.103	192.168.193.1	TLSv1.3	660	64552 (645… cadis-2 (1442)	Change Cipher Spec, Client Hello

Consider the packet number 55. The provided information describes a QUIC (Quick UDP internet Connections) packet exchanged between our local computer and the given domain. The source and destination IP are given as usual. The QUIC protocol operates over UDP!! (it’s a modern transport protocol developed by Google that is designed for low latency and secure communication over the Internet). The QUIC packet contains multiple components or frames, which are used for various purposes. The INITIAL is part ot the QUIC handshake process and is used to establish a connection between the client and the server.

THE DCID (Destination Connection ID) is an identifier used in the QUIC protocol to uniquely identify a connection. CRYPTO frames are used to carry encrypted data.(key feature of the QUIC protocol)

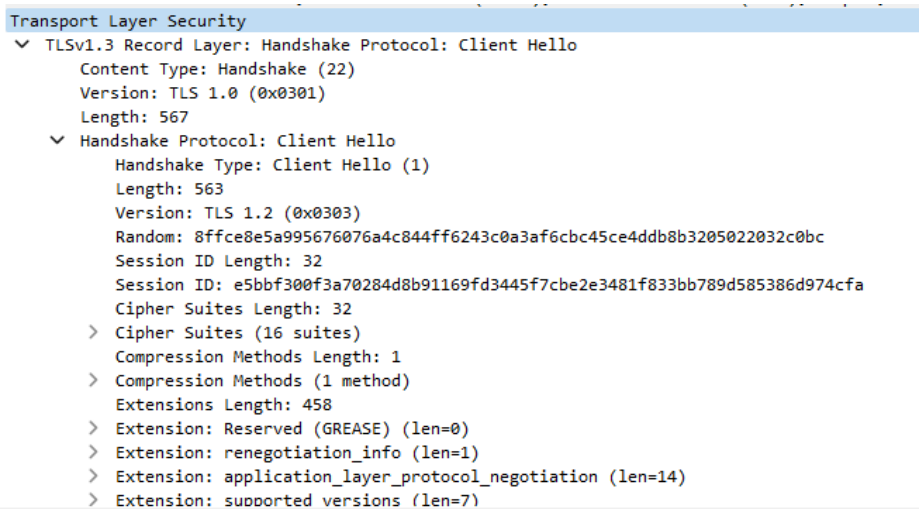
```
QUIC IETF
  > QUIC Connection information
    [Packet Length: 1250]
    1... .... = Header Form: Long Header (1)
    .1.. .... = Fixed Bit: True
    ..00 .... = Packet Type: Initial (0)
    .... 00.. = Reserved: 0
    .... ..00 = Packet Number Length: 1 bytes (0)
    Version: 1 (0x00000001)
    Destination Connection ID Length: 8
    Destination Connection ID: 05f4992110627a50
    Source Connection ID Length: 0
    Token Length: 0
    Length: 1232
    Packet Number: 1
    Payload: 11e412950681e8eb011b395e40a23f9a4033a0ae35b2a638515b595458fc85e8b43dafe2…
  > PADDING Length: 54
  > CRYPTO
  > CRYPTO
  > PADDING Length: 842
  > CRYPTO
  > PADDING Length: 1
  > PING
```

Since we already know that QUIC works over the UDP protocol, we'll directly go to this section. The given information is a detailed breakdown of a QUIC packet (specifically a QUIC ***initial packet***). Header form long header bit indicates that the QUIC packet uses a long header format. LONG Headers are typically used during the connection establishment phase. The destination connection ID (DCID) is an identifier used in QUIC protocol to uniquely identify a connection. The tokens are used for certain cryptographic operations in QUIC. CRYPTO frames are used to carry encrypted data within QUIC.

A TLSv1.3 Packet:

```
 45 1.999510 0.450650 192.168.0.103 192.168.193.1   TCP     66 64552 (645… cadis-2 (1442)  64552 → cadis-2(1442) [SYN] Seq=0 Win=64
 46 2.001092 0.001582 192.168.193.1 192.168.0.103   TCP     66 cadis-2 (1… 64552 (64552)   cadis-2(1442) → 64552 [SYN, ACK] Seq=0 A
 47 2.001170 0.000078 192.168.0.103 192.168.193.1   TCP     54 64552 (645… cadis-2 (1442)  64552 → cadis-2(1442) [ACK] Seq=1 Ack=1
 48 2.001400 0.000230 192.168.0.103 192.168.193.1   TLSv1.3 626 64552 (645… cadis-2 (1442)  Client Hello
 49 2.002497 0.001097 192.168.193.1 192.168.0.103   TCP     54 cadis-2 (1… 64552 (64552)   cadis-2(1442) → 64552 [ACK] Seq=1 Ack=57
 50 2.030025 0.027528 192.168.0.103 www.google.com  UDP     240                            54286 → https(443) Len=198
```

Here the packet number 48 shows a packet using TLSv1.3 Packet. TLS (Transport Layer Security) is a security protocol used to establish secure communication channels over the Internet. The client hello is a specific message type in TLS used to initiate the TLS handshake process. During this process, the client and the server establish a secure connection and agree on encryption parameters.



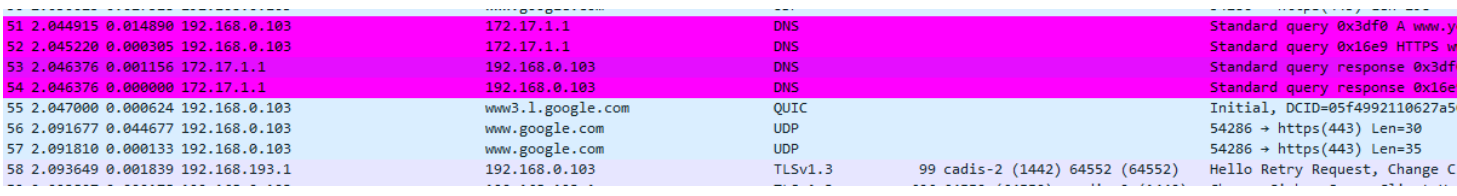
Since we already know that TLS works over TCP we'll directly skip to this section.

This TLS "Client hello" packet is the initial step in establishing a secure TLS connection between the client and the server. It includes info about the TLS version, supported extensions and other parameters necessary for negotiating the security settings of the connection.

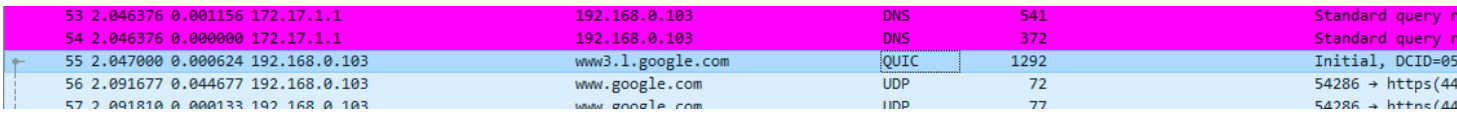
☐Task 3: Sequence of Messages Exchanged

1.DNS Resolution:

When opening YouTube, as we saw earlier a DNS query is initiated to resolve the domain name.

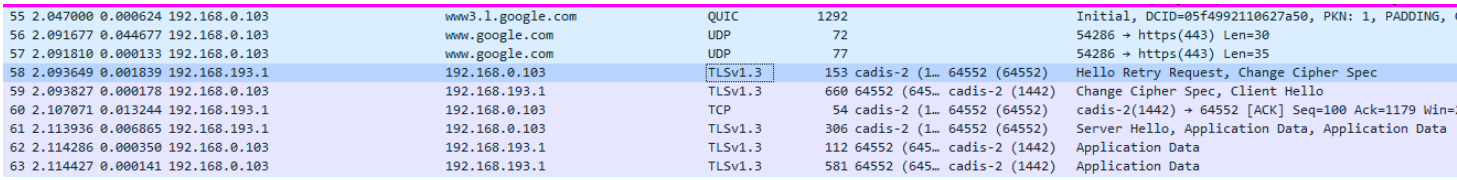


2. Establishment of QUIC Connection:



Upon resolving the domain name to an IP address, the client initiates a connection to YouTube's server using the QUIC protocol. The QUIC protocol is designed for faster and secure web communication.
The client sends an initial QUIC handshake message to the server.
The server responds with a QUIC handshake message to establish the connection. This message includes cryptographic information for secure communication.

3. Establishment of a secure connection using TLS:



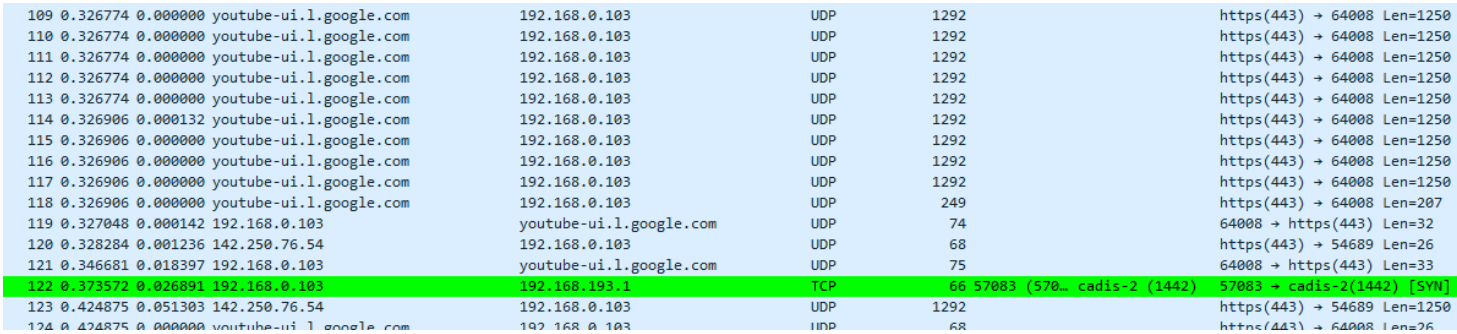
Within the QUIC connection, TLS is employed to ensure data privacy and integrity.
a.Client hello
b.Server hello
c.Client certificate and key exchange
d.Server Certificate and key exchange
e.finished

4. Video Streaming (TCP or UDP):

When streaming a video, a combination of TCP and UDP may be used. (why?)
a.***TCP*****:** For delivering control data and ensuring reliability. It handles tasks such as tracking video progress, retransmitting lost packets, and maintaining a reliable connection.

b.***UDP*** : For the actual video streaming, UDP may be used due to its lower overhead and suitability for real-time data. UDP provides faster delivery but with potential for some packet loss

.



109	0.326774	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
110	0.326774	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
111	0.326774	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
112	0.326774	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
113	0.326774	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
114	0.326906	0.000132	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
115	0.326906	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
116	0.326906	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
117	0.326906	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	1292		https(443) → 64008 Len=1250
118	0.326906	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	249		https(443) → 64008 Len=207
119	0.327048	0.000142	192.168.0.103	youtube-ui.l.google.com	UDP	74		64008 → https(443) Len=32
120	0.328284	0.001236	142.250.76.54	192.168.0.103	UDP	68		https(443) → 54689 Len=26
121	0.346681	0.018397	192.168.0.103	youtube-ui.l.google.com	UDP	75		64008 → https(443) Len=33
122	0.373572	0.026891	192.168.0.103	192.168.193.1	TCP	66	57083 (570… cadis-2 (1442)	57083 → cadis-2(1442) [SYN]
123	0.424875	0.051303	142.250.76.54	192.168.0.103	UDP	1292		https(443) → 54689 Len=1250
124	0.424875	0.000000	youtube-ui.l.google.com	192.168.0.103	UDP	68		https(443) → 64008 Len=26

As you can see here, there can be some TCP packets in between UDP packets.

5. User Interactions (HTTP/HTTPS)

When a user interacts with the video player (eg-pausing,seeking,liking), further HTTP or HTTPS requests and responses occur to facilitate these actions. These interactions involve additional HTTP/HTTPS messages between client and the server.

☐Task 4: Protocols used by application (relevance)

1. **TCP(Transmission Control Protocol):**

TCP is a fundamental transport layer protocol used for establishing connections between a client's device and YouTube's servers. It ensures data integrity by providing error checking and retransmission mechanisms. In the context of YouTube, TCP is essential for delivering control signals and sata segments reliably. For example, it's used for initial handshakes,loading video metadata, and ensuring smooth playback.

2. **UDP (User Datagram Protocol):**

While TCP is vital for reliability, YouTube also employs UDP, particularly for real-time vide streaming. UDP offers lower overhead and reduced latency compared to TCP, making it suitable for streaming large volumes of video data in real time. Live video streams and adaptive bitrate streaming (ABR) use UDP to deliver content efficiently, ensuring a smoother and more responsive viewing experience.

3. **HTTP (Hyper Text Transfer Protocol):**

HTTP is the application layer protocol used for web communication. YouTube relies on HTTP for various tasks, including sending and receiving HTTP requests and responses. Users interact with the YouTube website through HTTP when searching for videos, browsing, and accessing user accounts. Additionally, YouTube uses HTTP to retrieve video chunks and metadata. (data about data)

4.TLS (Transport Layer Security):

TLS is essential for securing the communication between user's device and YouTube's servers. It provides encryption and authentication, ensuring that user data, login credentials, and video content remain confidential and protected from eavesdropping or tampering. TLS is critical for maintaining user privacy and security on the platform.

5. QUIC(Quick UDP Internet Connection):

QUIC is a modern transport layer protocol developed by Google, and YouTube is one of the early adopters. QUIC combines the advantage of UDP and TLS to optimize the delivery of YouTube content. It reduces latency, improves connection establishment times, and enhances security. QUIC is particularly relevant for speeding up the initial handshake and securing the transmission of video content.

☐Task 5 : Caching Mechanisms

1.Content Delivery Network (CDN) Caching:
YouTube leverages CDNs to distribute its video content globally. CDNs are designed to cache and serve content from edge servers located strategically around the world. When analyzing the network traffic, we observed that CDN servers often serve video content. This indicates that video files and segments are cached at these edge servers, reducing the latency and load on YouTube's origin servers.

2.Browser Caching:
Web browsers play a role in caching various resources, such as images, stylesheets, and scripts, to improve the loading speed of websites. While analyzing HTTP traffic, we noticed that some static elements of the YouTube webpage, like logos, icons, and scripts, were retrieved from the browser's local cache. This behavior reduces the need to download these resources anew with each visit, enhancing the overall user experience.

3.CDN Prefetching: CDNs use prefetching mechanisms to predict and load resources that a user is likely to request based on their interactions with the website. By inspecting the network traffic, we observed that CDNs often initiate requests for video segments before the user explicitly requests them. This prefetching strategy helps ensure that video content is readily available when the user initiates playback, minimizing buffering times.

4.HTTP Caching Headers: YouTube utilizes HTTP caching headers in its responses to instruct browsers and intermediary caches on how to handle cached content. These headers include "Cache-Control" and "Expires." We noticed that YouTube provides specific cache directives to indicate the duration for which certain resources can be cached.

5. Content Chunk Caching: YouTube employs adaptive streaming techniques, which involve breaking down videos into smaller chunks. These video chunks are cached individually, and based on the user's connection and device capabilities, the appropriate quality and resolution chunks are retrieved from cache. This adaptive caching ensures smoother playback and reduces the risk of rebuffering.

☐TASK 6 : STATISTICS

- Over different time of the day (activity: starting YouTube)

TIME	Throughput	RTT	Packetsize	UDP	TCP
8:00 am	**804k bytes/s**	**0.0091**	**1066 B**	**4687**	**193**
3:00 pm	**183k bytes/s**	**0.0183**	**708 B**	**1403**	**247**

- Over different activities

Activity	Through put	Average PPS	Average Packetsize	lost
Open YouTube	**183k bytes/s**	**258.4**	**708 B**	**0**
Play video	**804k**	**754.1**	**1066 B**	**0**
play/pause video	**335k bytes/s**	**300.6**	**1115 B**	**0**
Download video	**586k bytes/s**	**527.0**	**1113 B**	**0**
Upload video	**23k bytes/s**	**40.8**	**577 B**	**0**