CS 342 :: Assignment- 2

Name : **Naman Goyal**
Roll No: **180123029**

TOPIC : **YOUTUBE LIVE VIDEO**

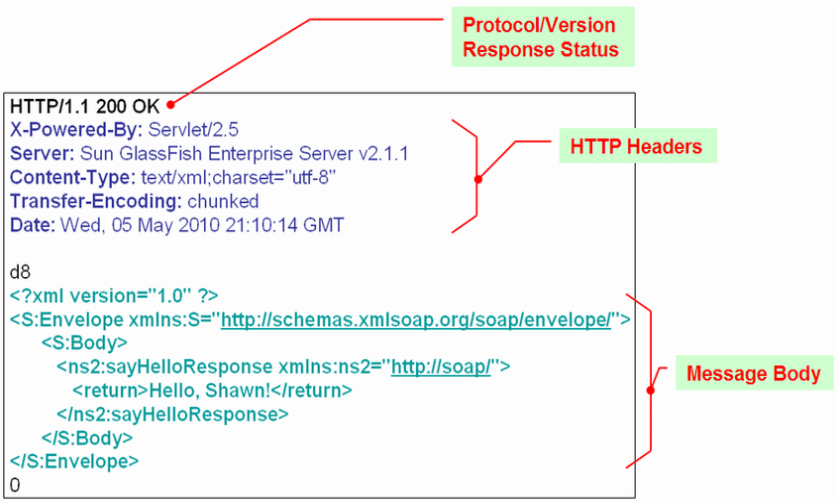
Drive Link : https://drive.google.com/drive/folders/1-9mkQHDjsBqqvNNlpx6zedoUu81fUVW1?usp=sharing
Drive link is given above for traces.

Ques.1)

-> Protocols used in different layers and their packet formats:

***a.) Application Layer*:**

-> HTTP stands for **HyperText Transfer Protocol** is used in Application layer. The exchange of information data between web server and web client happens through 2 ways namely **HTTP Request** and **HTTP Response.** An HTTP Request consists of request line containing the information about the type of request for what it is called namely **GET, POST, PUT, DELETE, HEAD** and the HTTP version as well as headers which may contain additional information about the languages. Addtional information could be send using the HTTP body. An HTTP response is been responded against any HTTP requests that the server receives. HTTP Responses contains the initial state lines like 404 means not found**,** header lines and the entity body.

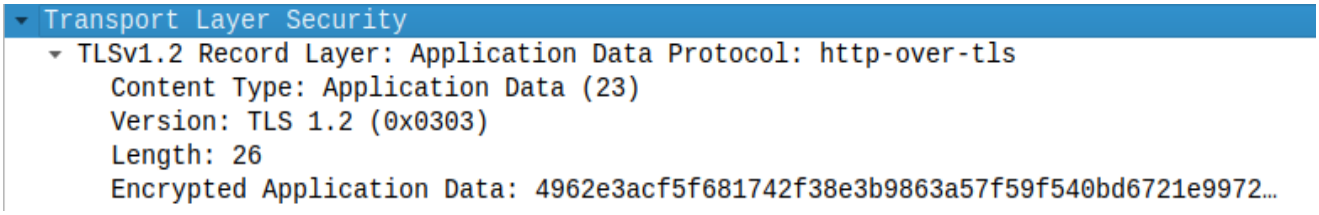


HTTP Response ***HTTP Request***

b.) Session Layer :

-> **TLS** layer lies between the Application and transport layer. It's responsible for the security during transmission which is been achieved by encrypting the data that has to be sent. **Record** is it's basic unit. Each record consists of a 5-byte record header, followed by data. The record format is Type (Handshake, Application Data, Alert and Change Cipher Spec), Version and length. It can be used by any protocol that used TCP as the transport layer.

-> ***Observed Values:***



1.) Application Data: Type of content being transferred. 23 is an identifier.

2.) Version : TLS 1.2, refers to version of protocol.

3.) Length: 26, length of data being transferred. Includes MAC and padding trailers.
4.) Encrypted Application Data: The encrypted data's value in the above pic for security.
5.) Application Data Protocol : http-over-tls, it ensures secure communication over the network using encryption of data.

***c.) <u>Transport Layer</u>* :**
-> **TCP** full form : **Tranfer Control Protocol** is a sathndard that defines how a connection could be maintained between the applications programs over the network.
Contents of TCP header :
-> **Source Port** (16 bits), **Destination Port** (16 bits) for the identification of addresses of communicating hosts.
-> **Sequence Number** and **Acknowledgement Number** (32 bits) used for ordering the packets.
-> **Header Length** (4 bits) : total size of TCP header in multiple of 4 bytes.
-> **Reserved Bits** (3 bits): used in alignment of total header size so that it becomes multiple of 4.
-> **Window Size** (32 bits) Regulating the amount of data being exchanged.
-> **Check Sum** (2 bits): for error correction.
-> **Urgent Pointer :** used to point to data that is urgently required and oftenly set to zero.
-> **Data** : can be of variable length contains upper layer information.

-> ***Observed Values:***

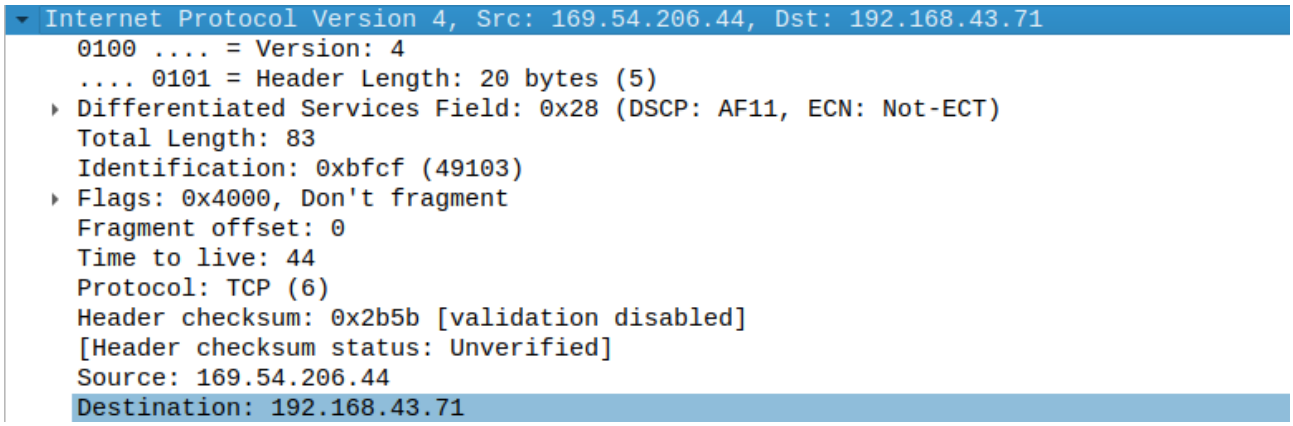
```
▾ Transmission Control Protocol, Src Port: 443, Dst Port: 46330, Seq: 63, Ack: 71,
    Source Port: 443
    Destination Port: 46330
    [Stream index: 0]
    [TCP Segment Len: 31]
    Sequence number: 63    (relative sequence number)
    Sequence number (raw): 465900019
    [Next sequence number: 94    (relative sequence number)]
    Acknowledgment number: 71    (relative ack number)
    Acknowledgment number (raw): 3018455664
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 244
    [Calculated window size: 244]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x8ea8 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (31 bytes)
```

1.) Source Port: 443, keeping track of new incoming connections, sended by host.
2.) Destination Port: 46330, Used by receiver to keep track of connections.
3.) Sequence Number: 63, assigned relative to the advent of the TCP connections. In all 62 packets have been sent till the connection is established.
4.) Acknowledgement Number: 71, squence number of the next byte the receiver expects to receive.
5.) Header Length: 8, represents the length of the header.

6.) Flags: PSH,ACK, The PSH or PUSH Flag allows sending application to start sending the data even when the buffer is not full. ACK or Acknowledgment used to see the completed receipt of packer.
7.) Check Sum: 0x8ea8, Error detection.
8.) Urgent Pointer: 0, Point to data required urgently.

d.) Networks Layer *:*
-> **IPV4** is a connectionless protocal or been used on packet switched networks, It's packet can contain upto 60 bytes of IP header data. Components of a IP:
-> **Version** specifies the version no of protocol used, **Internet Header Length** represents the length of header.
-> **Type of Services** namely **Differentiated Service Code Point** helps in defining differential services and **Explicit Congestion Notification** that allows end-to-end secure communication without dropping packets.
-> **Identification** is used to identify the unique group of fragments.
-> **Total length** defines entire length of IP Packet.
-> **TTL or Time to Live** tells the count of routers at maximum the packet can hop.
-> **Protocol** specifies the protocol itself.
-> **Header CheckSum** Error detection.
-> **Source and Destination IP Address** IP address of client and the server.
-> **Options** field is used for getting some additional information.

-> ***Observed Values:***



-> Version: 4, version of protocol used.
-> **Header Length:** 5, total length is 20 but the Header Length is counted in multiple of 4.
-> **Src:** 169.54.206.44, IP Address of the sender.
-> **Dst:** 192.168.43.71, IP Address of the receiver.
-> **Flags:** Don't fragment, if it were to be set, then the packet would have dropped because of fragmentaion.
-> **TTL:** 44, tells the count of routers at maximum the packet can hop.
-> **DSCP:** AF11, means that standard serbice is applied to the network.
-> **ECN:** Not-ECT, ECN is not used in the connection.

***e.) Link Layer*:**
-> Ethernet-II is used in this layer.
-> Ethernet frame is been transported as its payload. The components are as follows:
-> **Preamble** and a **Start Drame Delimiter (SFD)** marks the start of frame.

-> **Destination Address** and **Source Address:** specifies the MAC Address of the client and the host.
-> **Data:** also called **Payload** is the text itself.
-> **Frame Check Sequence:** detect errors.

-> ***Observed Values:***

```
▾ Ethernet II, Src: 66:98:27:70:70:7d (66:98:27:70:70:7d), Dst: HonHaiPr_87:33:7f (90:32:4b:87:33:7f)
  ▸ Destination: HonHaiPr_87:33:7f (90:32:4b:87:33:7f)
  ▸ Source: 66:98:27:70:70:7d (66:98:27:70:70:7d)
    Type: IPv4 (0x0800)
```

1.) Destination: 90:32 :4b:87:33:7f, MAC Address of the destination server.
2.) Source: 66:98:27:70:70:7d, MAC Address of the source.
3.) Type: IPv4, The upper layer used is IPv4.

-> ***Observed Values:***

```
▾ Frame 627: 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface wlp3s0, id 0
  ▸ Interface id: 0 (wlp3s0)
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 29, 2020 13:40:18.656102228 IST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1601367018.656102228 seconds
    [Time delta from previous captured frame: 4.580189597 seconds]
    [Time delta from previous displayed frame: 4.580189597 seconds]
    [Time since reference or first frame: 22.867372212 seconds]
    Frame Number: 627
    Frame Length: 97 bytes (776 bits)
    Capture Length: 97 bytes (776 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:tls]
    [Coloring Rule Name: TCP]
    [Coloring Rule String: tcp]
```

-> The interface id, Encapsulation type as well as Frame capturing and its attributes is been observed in the Frame Value.

Ques.2)
-> The various functionalities observed are:
1.) *Opening the Youtube website or an app*
2.) *Watching a Video*
3.) *Going live on Youtube*
4.) *Streaming a live video*
5.) *Pausing a video*
6.) *Downloading a Video*
7.) *Uploading a Video*
8.) *Subcribing a Youtuber*
9.) *Commenting on a video*
10.) *Sharing the video using URL*
11.) *Channel of a youtuber and the respective Channel Statistics*
12.) *Closing up the video*

The protocols used by the functionalities:

1.) TLS:

-> Used in prevention of eavesdropping and for modification on internet traffic.
-> The login credentials like in Youtube app or a Youtube Channel are safely transmiited from the client side to server side using TLS only, the security of personal data is secured.
-> It imparts security to the website from external hackers, it is been achieved beacuse of encrypting data in form of a key that is being decrypted onto client and server side only and man in the middle is a very hard to perform.

2.) TCP:

-> It is used for transmission of data. It provides reliability of data that means it will be almost sure that the packet will reach the respective destination without being lost during the the process but UDP doesn't have this functionality.
-> TCP connection besides being reliable is secure also and TLS also provied security as well so, this type of connection ensures maximum security as well as the reliability, Data security being very important nowadays.
-> Through the process of handshaking done in the beginning we can see that no information loss is been seen, if we do someting like rewinding of a video or not.
-> It's a stream-oriented protocol. The TCP/IP Protocol is responsible for breaking the stream of data into packets and sending those packets across the server and the clients.

3.) UDP:

-> It's one of the core member of Internet protocol suite. Simple connectionless communication. Provides checksum for data integrity, port numbers for addressing different functions at client and server sides.
-> Simple message oriented transport layer and transaction oriented.
-> No handshaking, hence no guarantee of tranfer of packets.
-> Suitable for error checking and correction. For eg some time sensitive applications uses UDP beacuse dropping packets is preferable to waiting for packets delayed due to retransmission.
-> Supports multicast and suitable for broadcasting information.

4.) IPv4:

-> It's a connection less protocol used in packet switching networks like in the case of Internet, browsing.
-> The packets are being delivered using the IP headers in the connection.
-> Used in networks layer.

5.) Ethernet II:

-> Used in the data link layer.
-> Its responsible for error detection and its rectification. Like for eg CRC (Cyclic Redundancy Check).
-> Stores the actual physical MAC address of source as well as destination

Ques.3)

-> The functionalities I will be discussing here are Streaming a live video, Pausing a video. The steps are as follows:

1.) DNS querying:

-> DNS querying is done by the browser as soon as the site is loaded. It's the information sent by the client side (DNS client) to a DNS server for the IP addresses associated woth the youtube web app.

10738	94.422463110	192.168.43.71	192.168.43.114	DNS	83	Standard query 0xb983 A youtube-ui.l.google.com
10739	94.422979728	192.168.43.71	192.168.43.114	DNS	83	Standard query 0x1971 AAAA youtube-ui.l.google.com
10740	94.803527229	192.168.43.114	192.168.43.71	DNS	323	Standard query response 0xb983 A youtube-ui.l.google.com A 14…
10741	94.803583540	192.168.43.114	192.168.43.71	DNS	195	Standard query response 0x1971 AAAA youtube-ui.l.google.com A…

The querying is for 'A' record type, used in relating IP adddresses with domain names.

2.) TCP Handshaking:

-> After DNS querying, a TCP/IP network connection is being established here between the client and the server side.

```
▾ Transmission Control Protocol, Src Port: 58530, Dst Port: 443, Seq: 1113, Ack: 4406, Len: 0
    Source Port: 58530
    Destination Port: 443
```

As we can see above my client port is **58530** and our destination port is **443**.

10732	91.985003895	192.168.43.71	99.86.30.172	TCP	66	[TCP Dup ACK 356#2] 60690 → 443 [ACK] Seq=1 Ack=1 Win=501 Len…
10733	91.993012052	192.168.43.71	99.86.30.172	TCP	66	[TCP Dup ACK 355#2] 60692 → 443 [ACK] Seq=1 Ack=1 Win=501 Len…
10734	92.090339462	99.86.30.172	192.168.43.71	TCP	66	[TCP Dup ACK 359#2] [TCP ACKed unseen segment] 443 → 60690 [A…
10735	92.105401899	99.86.30.172	192.168.43.71	TCP	66	[TCP Dup ACK 360#2] [TCP ACKed unseen segment] 443 → 60692 [A…
10736	93.965015087	2401:4900:4639:531:…	2404:6800:4003:c00:…	TCP	86	[TCP Dup ACK 357#2] 45888 → 443 [ACK] Seq=1 Ack=1 Win=501 Len…
10737	94.189054558	2404:6800:4003:c00:…	2401:4900:4639:531:…	TCP	86	[TCP Dup ACK 469#2] [TCP ACKed unseen segment] 443 → 45888 [A…

-> The handshaking is done in 3 processes. The process starts when the hosts send a **SYN** packet to the destination which helps in synchronizing the sequence number. The destination respons by sending a ACK (Acknowledgement) and again a SYN packet to the client itself. Finally in respondence the client sends a ACK packet again, marking the end of this handshaking process.

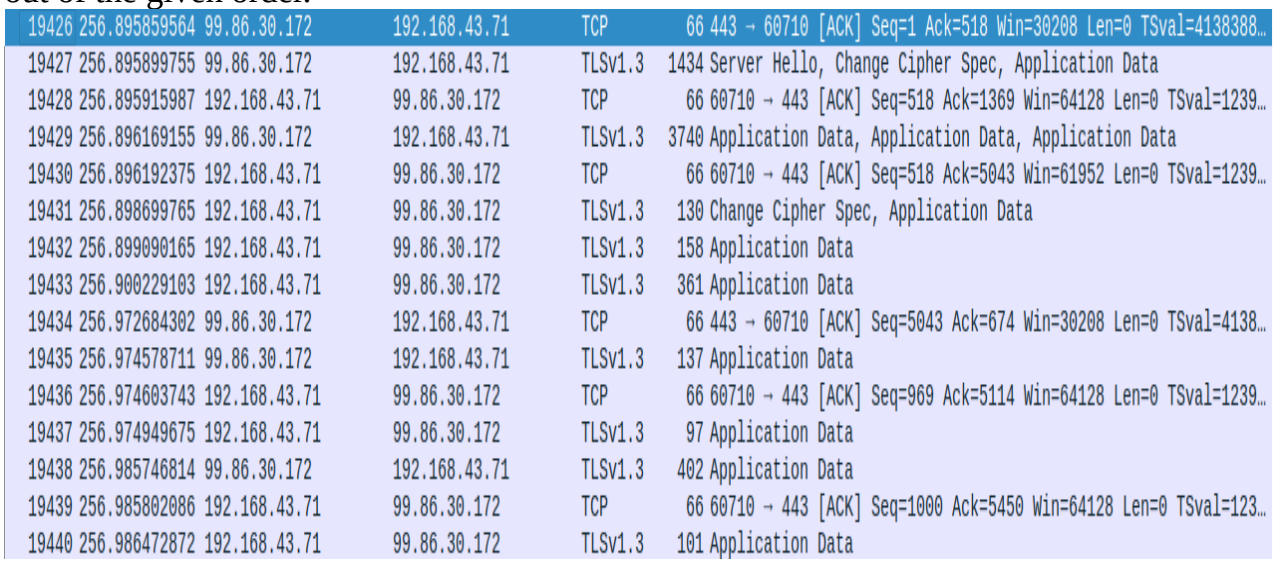
3.) TLS Handshaking:

481	10.016643406	2a00:1450:401b::7	2401:4900:4639:531:…	TLSv1.3	1294	Server Hello, Change Cipher Spec
482	10.016656892	2401:4900:4639:531:…	2a00:1450:401b::7	TCP	86	54152 → 443 [ACK] Seq=518 Ack=1209 Win=64128 Len=0 TSval=1795…
483	10.016722966	2401:4900:4639:531:…	2404:6800:4007:813:…	UDP	95	34399 → 443 Len=33
484	10.016884191	2a00:1450:401b::7	2401:4900:4639:531:…	TCP	3710	443 → 54152 [ACK] Seq=1209 Ack=518 Win=66816 Len=3624 TSval=2…
485	10.016896290	2401:4900:4639:531:…	2a00:1450:401b::7	TCP	86	54152 → 443 [ACK] Seq=518 Ack=4833 Win=61952 Len=0 TSval=1795…
486	10.016956712	2a00:1450:401b::7	2401:4900:4639:531:…	TLSv1.3	397	Application Data
487	10.016964467	2401:4900:4639:531:…	2a00:1450:401b::7	TCP	86	54152 → 443 [ACK] Seq=518 Ack=5144 Win=61696 Len=0 TSval=1795…
488	10.018834073	2401:4900:4639:531:…	2a00:1450:401b::7	TLSv1.3	150	Change Cipher Spec, Application Data

-> After TCP connection, TLS handshaking takes place. TLSv1.3 is been used for it. As we can see that the 1st msg is Client Hello, which is sent by client itself. The server responds with Server Hello. The Server key is sent to the client and client responds with the client key as well which marks the end of handshaking process.

Streaming Videos on youtube:

-> On starting the video on youtube app, the youtube's server sends the packets to the client. One case can be like different packets may take different paths to reach which depends on various factors such as load balancing. That may cause problems like the packets may arrive out of the given order.

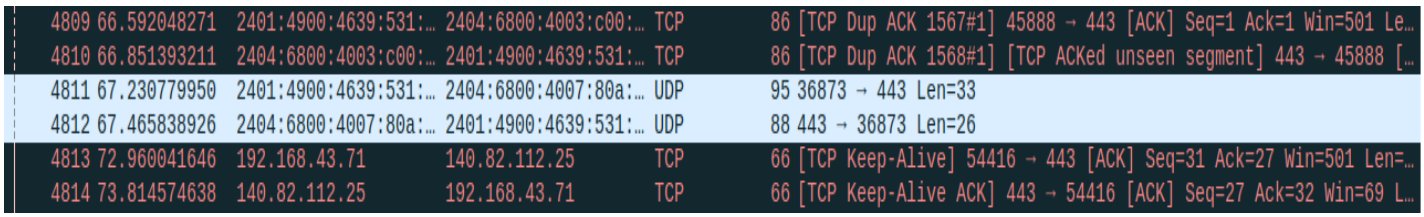


19426	256.895859564	99.86.30.172	192.168.43.71	TCP	66	443 → 60710 [ACK] Seq=1 Ack=518 Win=30208 Len=0 TSval=4138388…
19427	256.895899755	99.86.30.172	192.168.43.71	TLSv1.3	1434	Server Hello, Change Cipher Spec, Application Data
19428	256.895915987	192.168.43.71	99.86.30.172	TCP	66	60710 → 443 [ACK] Seq=518 Ack=1369 Win=64128 Len=0 TSval=1239…
19429	256.896169155	99.86.30.172	192.168.43.71	TLSv1.3	3740	Application Data, Application Data, Application Data
19430	256.896192375	192.168.43.71	99.86.30.172	TCP	66	60710 → 443 [ACK] Seq=518 Ack=5043 Win=61952 Len=0 TSval=1239…
19431	256.898699765	192.168.43.71	99.86.30.172	TLSv1.3	130	Change Cipher Spec, Application Data
19432	256.899090165	192.168.43.71	99.86.30.172	TLSv1.3	158	Application Data
19433	256.900229103	192.168.43.71	99.86.30.172	TLSv1.3	361	Application Data
19434	256.972684302	99.86.30.172	192.168.43.71	TCP	66	443 → 60710 [ACK] Seq=5043 Ack=674 Win=30208 Len=0 TSval=4138…
19435	256.974578711	99.86.30.172	192.168.43.71	TLSv1.3	137	Application Data
19436	256.974603743	192.168.43.71	99.86.30.172	TCP	66	60710 → 443 [ACK] Seq=969 Ack=5114 Win=64128 Len=0 TSval=1239…
19437	256.974949675	192.168.43.71	99.86.30.172	TLSv1.3	97	Application Data
19438	256.985746814	99.86.30.172	192.168.43.71	TLSv1.3	402	Application Data
19439	256.985802086	192.168.43.71	99.86.30.172	TCP	66	60710 → 443 [ACK] Seq=1000 Ack=5450 Win=64128 Len=0 TSval=123…
19440	256.986472872	192.168.43.71	99.86.30.172	TLSv1.3	101	Application Data

-> The client acknowledges the packet by the ACK flag. The client then sends cumulative ACK back to the server. After all the segments arrives, they are assembled and data is then sent to the application layer like HTTP.

Pausing the Video:

-> When we pause the video, some of the data remains untill the buffer is full.



4809	66.592048271	2401:4900:4639:531:…	2404:6800:4003:c00:…	TCP	86	[TCP Dup ACK 1567#1] 45888 → 443 [ACK] Seq=1 Ack=1 Win=501 Le…
4810	66.851393211	2404:6800:4003:c00:…	2401:4900:4639:531:…	TCP	86	[TCP Dup ACK 1568#1] [TCP ACKed unseen segment] 443 → 45888 […
4811	67.230779950	2401:4900:4639:531:…	2404:6800:4007:80a:…	UDP	95	36873 → 443 Len=33
4812	67.465838926	2404:6800:4007:80a:…	2401:4900:4639:531:…	UDP	88	443 → 36873 Len=26
4813	72.960041646	192.168.43.71	140.82.112.25	TCP	66	[TCP Keep-Alive] 54416 → 443 [ACK] Seq=31 Ack=27 Win=501 Len=…
4814	73.814574638	140.82.112.25	192.168.43.71	TCP	66	[TCP Keep-Alive ACK] 443 → 54416 [ACK] Seq=27 Ack=32 Win=69 L…

Once the buffer is full, a FIN piggybacked by an ACK is send to the server by the client indicating to stop the flow of packets. But this is temporary as the client/user may start up the video once again hence to keep the process smooth client keeps on sending Keep-Alive packets to the server side itself. On tapping the play button again the flow of SYN packets continues.

Ques.4)

1.) For Streaming:

Time	Throughput	RTT	Packet Size	Lost Packets	UDP / TCP	Ratio of Responses
T1	149	0.00773	1159 bytes	0	169/8591	1
T2	540	0.01706	1152 bytes	0	129/20058	1.2
T3	632	0.01406	1113 bytes	0	2561/27841	1.29

2.) ***Pausing a Video:***

Time	Throughput	RTT	Packet Size	Lost Packets	UDP/TCP	Ratio of Responses
T1	1505	0.3703	557	0	202/8005	1
T2	722	0.0124	1117	0	237/9085	1.15
T3	448	0.0199	1120	0	164/7993	1.19

Ques 5.)

-> Youtube is a very well known app which supports video streaming, downloading (in case of mobile app), pausing, going live socially and many more features. Thus it has a huge amount of data in it's side that's why storing this huge amount of big data is a big concern, Hence a lot of servers are being planted across the world and hence these multiple servers are been used to respond to many incoming requests. Source IP depends on the location of the source as well as time of the connection also.

-> It was observed that there was a change in source IP while sending the data. We can see variable IP's many a times from the server side which is clearly visible that many servers exists and the response is coming from them only.

-> This Content Distribution Network has many benefits which includes decrease in server load time, ensuring faster delivery. Benefits also lies the reliabilty like if one server goes down or may damage there a whole lot team of servers to overcome that difficulty.

-> One advantage also lies in Load balancing which helps in distributing up the load on a server to many servers. Some requests could be handled by the other if one of the server is busy with some other request.