ASSIGNMENT-2

Submitted by: Aranya Aryaman
Roll Number: 170101011

Question 1:

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{CDD74477-37E9-48C3-9023-FA473661E222}, id 0
> Ethernet II, Src: Dell_3b:7f:15 (28:f1:0e:3b:7f:15), Dst: Cisco_97:1e:ef (4c:4e:35:97:1e:ef)
> Internet Protocol Version 4, Src: 10.3.3.35, Dst: 195.8.215.136
> Transmission Control Protocol, Src Port: 60300, Dst Port: 443, Seq: 0, Len: 0
```

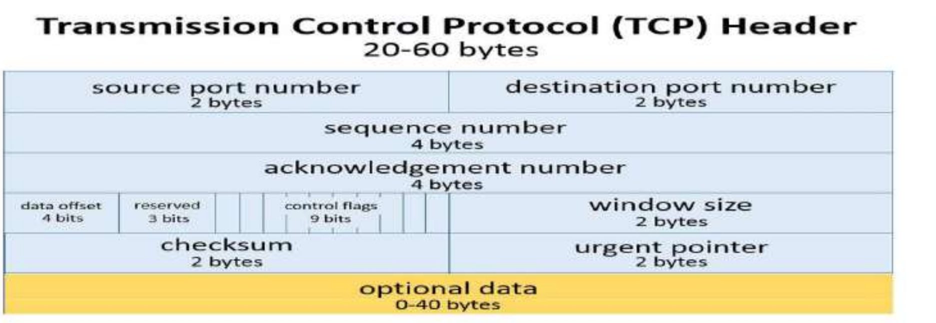
The different protocols used at different layers by the application www.dailymotion.com are as follows:

1) Transport Layer: **TCP**
2) Network Layer: **IPv4**
3) Application Layer: **HTTP**
4) Physical Layer: **Ethernet II**
5) Secure Socket Layer: **TLSv1.2 Protocol**

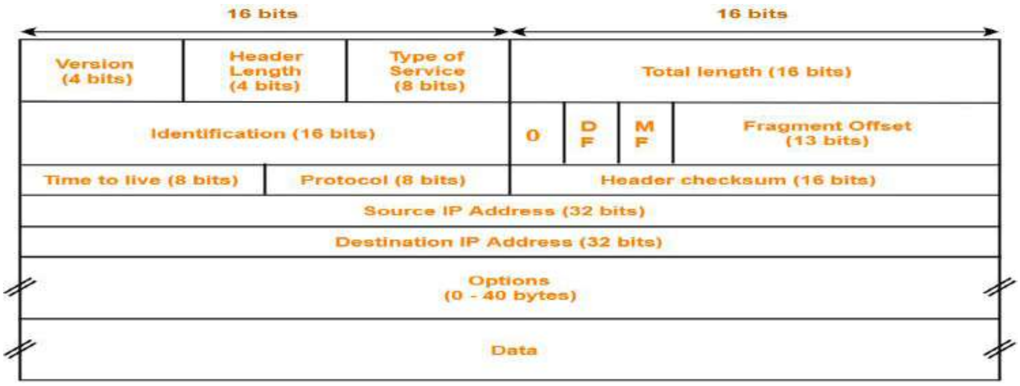
Note: Some of these protocols are found to be used in the application from traces but are not shown in the above figure by wireshark. Hence, they have been mentioned in the answer.

Packet Formats for various protocols are as follows:

Transport Layer: TCP (*Details about TCP Frame Format has been mentioned in Question 2*)



Network Layer: IPv4 (*Details about IPv4 Frame Format has been mentioned in Question 2*)



IPv4 Header

Application Layer: HTTP

HTTP Messages consist of requests being sent from a client to the server and responses from the server to the client.

HTTP-message = **<*Request*>|<*Response*>; HTTP/1.1 messages**

Request and Response messages use the generic message format of **RFC 822** for transferring entities (the payload of the message). This generic message format consists of the following four items:

- **a)** **Start-line**: Request Line or Status Line.
- **b)** **Zero or more header fields followed by CRLF**: The headers are as follows: General Header, Response Header, Request Header and Entity Header.
- **c)** **An empty line**: indicating the end of the header fields
- **d)** **message-body**: It is used to carry the entity-body associated with the request or response.

SSL (Secure Socket Layer): TLSv1.2

This acts as an intermediate b/w transport layer and application layer. It deals with session and connection coordination. **TLSv1.2** was defined in RFC 5246. The packet format is as follows:

- **a)** **Content type**: The type field is identical to TLSCompressed.type
- **b)** **Version**: The version field is identical to TLSCompressed.version
- **c)** **Length**: The length (in bytes) of the following TLSCiphertext.fragment. The length MUST NOT exceed 2^14 + 2048.
- **d)** **Fragment**: The encrypted form of TLSCompressed.fragment with the MAC.
- **e)** **Message Authentication Code (MAC)**: It is a one-way hash computed from a message and some secret data. It is difficult to forge without knowing the secret data. Its purpose is to detect if the message has been altered.

Physical Layer: Ethernet II

DA	SA	Type	DSAP	SSAP	Ctrl	Data	FCS
6 Bytes	6 Bytes	2 Bytes	1 Byte	1 Byte	1 Byte	> 46 Bytes	4 Bytes

The fields in the frame are as follows

- **a)** **DA** – Destination Address
- **b)** **SA** – Source Address
- **c)** **Type** – 0x8870 Ether Type
- **d)** **DSAP** – 802.2 Destination Service Access Point
- **e)** **SSAP** – 802.2 Source Service Access Point
- **f)** **Ctrl** – 802.2 Control Field
- **g)** **Data** – Protocol Data
- **h)** **FCS** – Frame Checksum

Question 2:

The following figures describe the various fields of the different protocols mentioned above.

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{CDD74477-37E9-48C3-9023-FA473661E222}, id 0
> Ethernet II, Src: Dell_3b:7f:15 (28:f1:0e:3b:7f:15), Dst: Cisco_97:1e:ef (4c:4e:35:97:1e:ef)
v Internet Protocol Version 4, Src: 10.3.3.35, Dst: 195.8.215.136
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x49b6 (18870)
  > Flags: 0x4000, Don't fragment
    ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.3.3.35
    Destination: 195.8.215.136
```

1) **Frame 1**: The frame number of the trace I took for the observation is 1. The frame protocol is not a real protocol itself, but used by Wireshark as a base for all protocols on top of it. 66 bytes were used by the frame.
2) **Ethernet II**:

```
v Ethernet II, Src: Dell_3b:7f:15 (28:f1:0e:3b:7f:15), Dst: Cisco_97:1e:ef (4c:4e:35:97:1e:ef)
  > Destination: Cisco_97:1e:ef (4c:4e:35:97:1e:ef)
  > Source: Dell_3b:7f:15 (28:f1:0e:3b:7f:15)
    Type: IPv4 (0x0800)
```

Src: *Dell_3b:7f:15 (28:f1:0e:3b:7f:15)*: MAC Address of my PC.
Dst: *Cisco_97:1e:ef (4c:4e:35:97:1e:ef)*: MAC Address of Destination.

3) **Internet Protocol**:

Version 4: Field indicates the format of the internet header.
Source-10.3.3.35: My PC's IP Address
Destination-195.8.215.136: Destination's IP Address
Header Length-20 bytes: Number of 32bit words in the TCP Header
Differentiated Services Field 0x00: Indicates particular quality of service needs from the network, the DSF defines the way the routers should queue packets while they are waiting to be forwarded.
Total Length-52: Length of the datagram, measured in octets, including internet header and data.

4) **Transmission Control Protocol:**

```
v Transmission Control Protocol, Src Port: 60300, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 60300
    Destination Port: 443
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 2309263029
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 64240
    [Calculated window size: 64240]
    Checksum: 0xa7dd [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]
```

Source Port- 60300: My computer port in this case, which is sending packets.
Destination Port-443: Destination Port, which is receiving packets.
TCP Segment Length-0: This is TCP packet segment length.
Sequence Number 1: If the SYN flag is set to 1, this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1.
Acknowledgement Number 0: The acknowledgement flag is set to 0.
Window size value- 64240: This is the space for incoming data.

Checksum- 0xa7dd: The 16-bit Checksum field is used for error checking of the header and data.

Urgent Pointer-0: If the URG Flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte.

Question 3:

The different protocols used while streaming/downloading/uploading videos on/from Dailymotion were **TCP**, **HTTP**, **TLSv1.2**. Their functions are as follows:

a) **TCP**: Its responsibility includes end-to-end message transfer independent of the underlying network and structure of user data, along with **error control**, **segmentation**, **flow control**, and helps to minimize **traffic congestion control**. It is a connection-oriented protocol that addresses numerous **reliability** issues in providing a reliable byte stream: data arrives in-order, data has minimal error (i.e., correctness), duplicate data is discarded and lost or discarded packets are resent. TCP is optimized for **accurate delivery** rather than timely delivery, as correct sequence of buffer to be fetched. TCP's bandwidth probing and congestion control will attempt to use all of the available bandwidth between server and client.

b) **HTTP:** TCP works in the **Transport layer** while HTTP works in **Application layer** of TCP/IP model. TCP is in charge of setting up a reliable connection between two machines and HTTP uses this connection to transfer data between the web servers and the client in the communication process, we can say connection is fundamentally out of scope of HTTP as it is controlled at Transport Layer. HTTP is a **stateless protocol** though not session-less, meaning that the server does not keep any data (state) between two requests/each request message can be understood in isolation. HTTP follows a classical **client-server model**, with a client opening a connection to make a request, then waiting until it receives a response. HTTP is an **extensible protocol** that is easy to use. The client-server structure, combined with the ability to simply add headers, allows HTTP to **advance** along with the extended capabilities of the Web. Usually, HTTP responses are **buffered** rather than streamed. **HTTP 1.1** added supports for streaming through keep-alive header so data could be streamed. HTTP is Media independent i.e. any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content.

c) **TLSv1.2:** TLS was designed to operate on top of a reliable transport protocol such as TCP. Web servers use cookies to identify the web user. They are small piece of data stored into the web user’s disk. TLS is used to protect **session cookies** on the rest of the sites from being intercepted to protect user accounts. The TLS protocol aims primarily to provide **privacy** and **data integrity** between two communicating computer applications. SSL and TLS are both cryptographic protocols utilizing X.509 certificates, public/private key encryption that provide **authentication** and **data encryption** between servers, machines and applications operating over a network. TLS provides verification of **identity** of server, which is as important as encryption. The goals of the TLS protocol are cryptographic **security, extensibility, and relative efficiency**. These goals are achieved through implementation of the TLS protocol on two levels: the TLS Record protocol and the TLS Handshake protocol. TLS allows the peers to negotiate **a shared secret key** without having to establish any prior knowledge of each other, and to do so over an unencrypted channel, Client-server applications use the TLS protocol to communicate across a network in a way designed to prevent eavesdropping and tampering.

Question 4:

Some of the important functionalities of the application are:

a) **3-way TCP Handshake Message Sequence**: Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. Establishing a normal TCP connection requires three separate steps:

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.3.3.35	195.8.215.136	TCP	66	60300 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.003394	195.8.215.136	10.3.3.35	TCP	66	443 → 60300 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=128
3	0.003459	10.3.3.35	195.8.215.136	TCP	54	60300 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0

1) **SYN**: The client sending a SYN to the server performs the active open. The client sets the segment's sequence number to a random value A.
2) **SYN,ACK**: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.
3) **ACK**: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1.

b) TLS Handshaking Message Sequence: The record encapsulates a "control" protocol (the handshake messaging protocol) when the TLS connection starts. This protocol is used to exchange all the information required by both sides for the exchange of the actual application data by TLS and to negotiate the **secure attributes of a session**. It defines the messages formatting or containing this information and the order of their exchange. These may vary according to the demands of the client and server. The TLS Handshake protocol allows **authenticated communication** to commence between the server and client. This protocol allows the client and server to speak the same language, allowing them to agree upon an **encryption algorithm and encryption keys** before the selected application protocol begins to send data.

4	0.003743	10.3.3.35	195.8.215.136	TLSv1.2	571	Client Hello
5	0.010170	195.8.215.136	10.3.3.35	TCP	60	443 → 60300 [ACK] Seq=1 Ack=518 Win=19456 Len=0
6	0.867162	195.8.215.136	10.3.3.35	TLSv1.2	1514	Server Hello
7	0.867162	195.8.215.136	10.3.3.35	TCP	1490	443 → 60300 [PSH, ACK] Seq=1461 Ack=518 Win=19456 Len=1436 [TCP segment of a reassembled PDU]
8	0.867215	10.3.3.35	195.8.215.136	TCP	54	60300 → 443 [ACK] Seq=518 Ack=2897 Win=131328 Len=0
9	0.867532	195.8.215.136	10.3.3.35	TLSv1.2	934	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
10	0.868498	10.3.3.35	195.8.215.136	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
11	0.869081	195.8.215.136	10.3.3.35	TCP	60	443 → 60300 [ACK] Seq=3777 Ack=644 Win=19456 Len=0
12	1.223567	195.8.215.136	10.3.3.35	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
13	1.224930	10.3.3.35	195.8.215.136	TLSv1.2	2912	Application Data

We can see various messages like '**Client Hello**', '**Server Hello**', '**Certificate, Server Key Exchange, Server Hello Done**', '**Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message**' b/w client (10.3.3.35) and server (195.8.215.136).

a) **Streaming Function:** This function shows a sequence of TCP connection establishment through 3-way handshake, then a HTTP request to the host. This got response from TCP after a TLS handshake to ensure security.

No.	Time	Source	Destination	Protocol	Length	Info
27291	45.411468	10.3.3.35	216.58.197.74	TCP	54	49993 → 443 [ACK] Seq=1597 Ack=1547 Win=131328 Len=0
27292	45.412510	10.3.3.35	216.58.197.74	TLSv1.3	93	Application Data
27302	45.449065	216.58.197.74	10.3.3.35	TCP	60	443 → 49993 [ACK] Seq=1547 Ack=1636 Win=21248 Len=0
27388	45.964569	10.3.3.35	192.168.193.1	TCP	66	49994 → 1442 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
27389	45.965234	192.168.193.1	10.3.3.35	TCP	66	1442 → 49994 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=512
27390	45.965285	10.3.3.35	192.168.193.1	TCP	54	49994 → 1442 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
27391	45.965759	10.3.3.35	192.168.193.1	TLSv1.3	625	Client Hello
27392	45.966550	192.168.193.1	10.3.3.35	TCP	60	1442 → 49994 [ACK] Seq=1 Ack=572 Win=19968 Len=0
27393	45.968711	192.168.193.1	10.3.3.35	TLSv1.3	153	Hello Retry Request, Change Cipher Spec
27394	45.969001	10.3.3.35	192.168.193.1	TLSv1.3	659	Change Cipher Spec, Client Hello
27396	45.974408	192.168.193.1	10.3.3.35	TLSv1.3	306	Server Hello, Application Data, Application Data
27397	45.976380	10.3.3.35	192.168.193.1	TLSv1.3	112	Application Data
27398	45.976628	10.3.3.35	192.168.193.1	TLSv1.3	499	Application Data
27399	45.977188	192.168.193.1	10.3.3.35	TLSv1.3	309	Application Data
27402	45.985247	192.168.193.1	10.3.3.35	TCP	1514	1442 → 49994 [ACK] Seq=607 Ack=1680 Win=22016 Len=1460 [TCP segment of a reassembled PDU]
27403	45.985251	192.168.193.1	10.3.3.35	TCP	1514	1442 → 49994 [ACK] Seq=2067 Ack=1680 Win=22016 Len=1460 [TCP segment of a reassembled PDU]

b) Downloading/Uploading Function: After clicking on the download button, initially connection is established through 3-way handshake, then a HTTP request to the host. Then followed by TLS handshake. Finally, TCP segments started being transferred from server to our computer for download function while for upload, TCP Segments are being transferred from our computer to the server.

No.	Time	Source	Destination	Protocol	Length	Info
317	1.941667	10.3.3.35	172.17.1.1	DNS	79	Standard query 0x67fe A www.dailymotion.com
318	1.942340	172.17.1.1	10.3.3.35	DNS	292	Standard query response 0x67fe A www.dailymotion.com CNAME dmwww.api-aws.dailymotion.com A 195.8.215.136 NS ns-137.awsdns-17.c
319	1.942848	10.3.3.35	195.8.215.136	TCP	66	49434 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
320	1.943672	195.8.215.136	10.3.3.35	TCP	66	443 → 49434 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=128
321	1.943743	10.3.3.35	195.8.215.136	TCP	54	49434 → 443 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
322	1.944227	10.3.3.35	195.8.215.136	TLSv1.2	229	Client Hello
323	1.944809	195.8.215.136	10.3.3.35	TCP	60	443 → 49434 [ACK] Seq=1 Ack=176 Win=19456 Len=0
677	2.840997	195.8.215.136	10.3.3.35	TLSv1.2	1514	Server Hello
678	2.841410	195.8.215.136	10.3.3.35	TCP	1514	443 → 49434 [ACK] Seq=1461 Ack=176 Win=19456 Len=1460 [TCP segment of a reassembled PDU]
679	2.841412	195.8.215.136	10.3.3.35	TLSv1.2	422	Certificate, Server Key Exchange, Server Hello Done
680	2.841585	10.3.3.35	195.8.215.136	TCP	54	49434 → 443 [ACK] Seq=176 Ack=3289 Win=1051136 Len=0
689	2.877039	10.3.3.35	195.8.215.136	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
690	2.877473	195.8.215.136	10.3.3.35	TCP	60	443 → 49434 [ACK] Seq=3289 Ack=302 Win=19456 Len=0
740	3.161919	10.3.3.35	192.168.193.1	TCP	66	49436 → 1442 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
741	3.162330	192.168.193.1	10.3.3.35	TCP	66	1442 → 49436 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=512
742	3.162402	10.3.3.35	192.168.193.1	TCP	54	49436 → 1442 [ACK] Seq=1 Ack=1 Win=1051136 Len=0

1617	8.359054	10.3.3.35	103.195.32.183	TLSv1.2	1514	Application Data [TCP segment of a reassembled PDU]
1618	8.359331	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=157177 Win=93568 Len=0
1619	8.359613	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=158637 Win=92160 Len=0
1620	8.360336	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=160097 Win=90752 Len=0
1622	8.360585	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=161557 Win=89344 Len=0
1624	8.360841	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=171777 Win=79232 Len=0
1625	8.360841	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=173237 Win=77824 Len=0
1626	8.360841	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=174697 Win=76416 Len=0
1627	8.361095	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=176157 Win=75008 Len=0
1628	8.361095	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=177617 Win=73600 Len=0
1629	8.361096	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=179077 Win=72192 Len=0
1630	8.361580	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=180537 Win=70784 Len=0
1631	8.361580	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=181997 Win=69376 Len=0
1632	8.362537	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=190757 Win=60672 Len=0
1633	8.362538	103.195.32.183	10.3.3.35	TCP	60	443 → 49725 [ACK] Seq=3682 Ack=192217 Win=59264 Len=0

A TLS Handshake was used before transferring of TCP Segments to ensure security.

Question 5:

The following statistics were obtained using wireshark on 03-02-2020 for www.dailymotion.com at different times of the day. The activities for which the data has been taken has also been mentioned.

Activity	Time	Throughput (Packets/sec)	RTT(ms)	Avg. Packet Size(bytes)	Packets Lost	UDP Packets	TCP Packets	Response per request
Download	10 AM	221.1	0.75	913	0	2563	313	1.47
Upload	2 PM	268.3	1.15	1391	0	15141	8534	1.91
Streaming	8 PM	587.9	0.78	987	0	7099	19021	2.02

Question 6:

The video content is being sent by the application from different sources/servers during the three different times of the day. There can be multiple reasons behind the same. The most probable reason is due to load balancing issues or high traffic in one area. Since Dailymotion keeps multiple servers across different geographical locations to reduce latency & network congestion for various clients, this can also be a reason for different servers responding at different times of the day. This also provides redundancy for the video content in case one of these servers go down for maintenance.

Time of the day	IP Addresses of different hosts
10 AM	192.168.193.1,195.8.215.136
2 PM	172.217.31.196, 192.168.193.1,195.8.215.129, 103.195.32.183
8 PM	172.217.163.131,192.168.193.1, 172.217.31.196,188.65.124.58

* There were many more IPs except the above-mentioned IP Addresses. But the number of requests being sent/received to these hosts were less than 10. These IPs can be of some other websites which might be running parallelly while doing the experiment of Wireshark or might be due to some local update of the OS.

LINK FOR THE TRACES

All the traces can be found from the following google drive link:
https://drive.google.com/open?id=1-94RkljVV5lxLExSilURi3e5-u5cyN0l