

## Application Assigned : Hotstar Video Streaming

### Link for data : Data Collected

#### Question 1: Protocols Used

##### A) Application Layer

- **HTTP** : The protocol used in the application layer is **Hyper Text Transfer Protocol**. The HTTP messages are of two types, request and response. Request messages consist of a **request line**, followed by the **Header** and the **body** of the message. Response message format is similar but it has a **status line** instead of a request line. Request line holds the info regarding the **method** of the request, the server **URL** and the **version** of HTTP being used. Status line holds the **version**, the **status code** and the status **phrase**. Header consists of **field name** value pairs holding other meta information and finally the body consists of data that is sent or received.

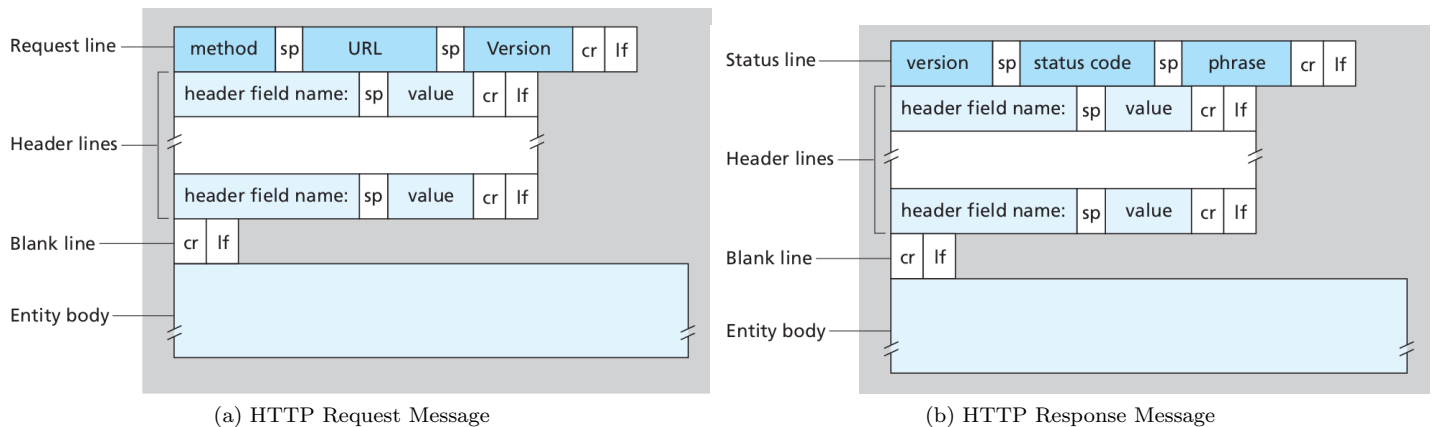


Figure 1: HTTP Message Format

- **TLSv1.2** : TLSv1.2 is the successor of **SSL** and it provides communications security over a computer network. Symmetric cryptography is used to encrypt the data transmitted. The packet contains the type of message (handshake, alert, or data) in the '**Content Type**' field. It also contains the **version**, **length** of data and **MAC** (Message Authentication Code).

Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	Payload			
n..m	MAC			
m..p	Padding (block ciphers only)			

Figure 2: TLSv1.2 Message Format

##### B) Transport Layer

**Transmission Control Protocol** is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. **Source Port** and **Destination Port** identify the hosts of the connection, source being the end point from where the segment is sent. **Sequence Number** specifies the number assigned to the first byte of data in the current message. If the ACK control bit is set, then **Acknowledgment number** refers to the next sequence number that the sender is expecting to receive. **Data offset** specifies the size of the variable sized TCP header. **Flags** are 1 bit values that specify the state of the connection and are used for control. **Window size** is the size of the buffer of the receiver. **Checksum** is used for error correction. Data field contains the payload of the segment.

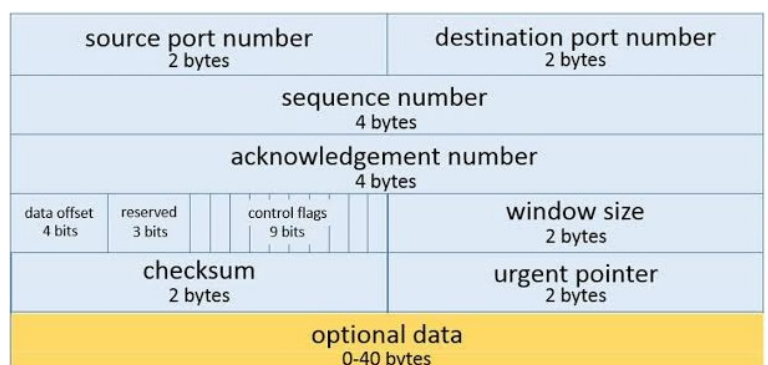


Figure 3: TCP Segment Format

## C) Network Layer

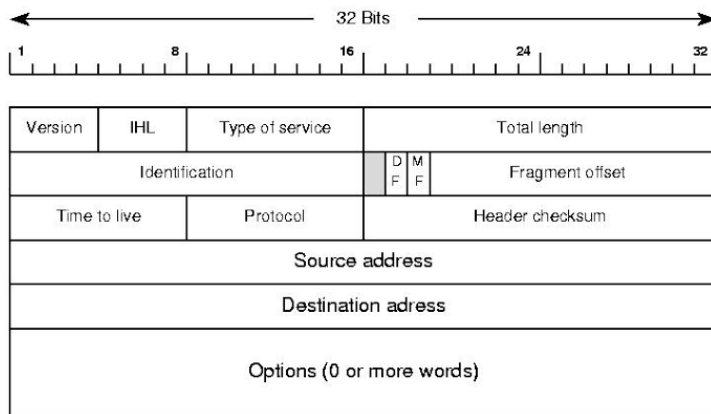


Figure 4: IP Datagram Header Format

**Protocol** indicates the next higher level protocol that is contained within the data portion of the packet. **Header checksum** is used for error detection. **Source** and **destination addresses** are the addresses of the source and destination of the packet respectively.

## D) Link Layer

**Ethernet II** is used in the link layer.

**Preamble** is a 7 byte pattern of alternating 0's and 1's which indicates starting of the frame and allow sender and receiver to establish bit synchronization. **SFD** is the start frame delimiter and marks the start of frame.

**Destination** and **source addresses**

are the MAC addresses of the sending and receiving machines of the frame respectively. **Type** field is used to specify the protocol that is being used. **FCS (Frame Check Sequence)** is the error detecting code that is added.

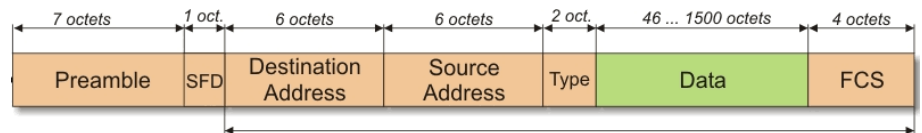


Figure 5: Ethernet Frame Format

## Question 2: Observed Values in Different Protocols

### A) Application Layer

```

Secure Sockets Layer
  TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 34
    Encrypted Application Data: 463c690c61e4a124ba8c6e399088848dc50f1bd9a39261ea...
    
```

Figure 6: TLSv1.2 Record Example

It is visible from the example that the application data protocol is **Http-over-tls** (aka HTTPS). The **content type** in this message is Application Data. **Version** of TLS is 1.2. **Length** of the data is 34 bytes. **Encrypted Application Data** can also be seen.

HTTPS encrypts all message contents, including the HTTP headers and the request/response data,

therefore no HTTP header or request/response can be seen explicitly.

### B) Transport Layer

It can be seen that the **source port** is 443 (This is to be expected because the default port for HTTPS connection is 443). The **destination port** is 55036. The **TCP Segment Length** is 39 bytes (payload). The **sequence number** is 40. **Acknowledgement number** is 218 which means that the sender of this segment is expecting a segment with sequence number 218 from the receiver. **Flags** field tells us that PSH and ACK flag is enabled. PSH flag is an option provided by TCP that allows the sending application to start sending the data even when the buffer is not full. **Window size** value is 270 (Number of packets sent before acknowledgment). The **checksum** value can also be seen that is used for error detection.

```

Transmission Control Protocol, Src Port: 443, Dst Port: 55036, Seq: 40, Ack: 218, Len: 39
  Source Port: 443
  Destination Port: 55036
  [Stream index: 17]
  [TCP Segment Len: 39]
  Sequence number: 40 (relative sequence number)
  [Next sequence number: 79 (relative sequence number)]
  Acknowledgment number: 218 (relative ack number)
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x018 (PSH, ACK)
  Window size value: 270
  [Calculated window size: 270]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x50be [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  [SEQ/ACK analysis]
  [Timestamps]
  TCP payload (39 bytes)
    
```

Figure 7: TCP Segment Example

## C) Network Layer

```
Internet Protocol Version 4, Src: 180.149.60.168, Dst: 10.19.4.77
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 91
    Identification: 0x772e (30510)
  Flags: 0x0000
    Time to live: 62
    Protocol: TCP (6)
    Header checksum: 0x05d2 [validation disabled]
    [Header checksum status: Unverified]
    Source: 180.149.60.168
    Destination: 10.19.4.77
```

Figure 8: IP Datagram Example

## D) Link Layer

The information about the **Destination** and **Source MAC** addresses can be seen. They are unique addresses assigned to the **Network Interface controllers** of the machines. The source of this frame is a Cisco device and the destination is my laptop. The **Type** of connection can also be seen.

```
Ethernet II, Src: Cisco_74:60:43 (ec:44:76:74:60:43), Dst: IntelCor_52:3e:06 (ac:ed:5c:52:3e:06)
  Destination: IntelCor_52:3e:06 (ac:ed:5c:52:3e:06)
    Address: IntelCor_52:3e:06 (ac:ed:5c:52:3e:06)
    .... 00000000 = LG bit: Globally unique address (factory default)
    .... 00000000 = IG bit: Individual address (unicast)
  Source: Cisco_74:60:43 (ec:44:76:74:60:43)
    Address: Cisco_74:60:43 (ec:44:76:74:60:43)
    .... 00000000 = LG bit: Globally unique address (factory default)
    .... 00000000 = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
```

Figure 9: Ethernet Frame Example

## Question 3: Protocols Used in Functioning of The Application

### A) Streaming

#### HTTPS

(HTTP-over-TLS) is used in the Application Layer as HTTP is used by World Wide Web and defines how messages should be formatted and transmitted. TLS (Transport Layer Security) is a **cryptographic protocol** designed to provide communications **security** over a computer network. It **encrypts** the data and prevents hackers from **snooping**. It also prevents against **Man in the Middle** attacks. TCP is used at the Transport Layer as it is a **connection-oriented reliable** data transfer protocol. It offers facilities like **flow control**, **congestion control** and **error handling** mechanisms. Since the protocol is reliable, all the messages from the server reach the browser without any losses. This error free connection is required by Hotstar because it offers very **high quality streaming** (HD is also possible) which would not be possible otherwise. Also, the users can go back and forth to different parts of a clip unlike other live streaming services. IP protocol is used in the Network Layer as it is a requirement for using the Internet. IPv4 is a connectionless protocol for use on packet-switched network. Ethernet(II) is used in the link layer as it is one of the most widely used and reliable link layer protocol. It has **error handling** capabilities and ensures reliable data transfer between two network devices.

```
632 10.831273602 172.217.167.130 10.19.4.77 TCP 66 443 -> 44404 [ACK] Seq=831 Ack=257 Win=224 Len=0 TSval=166842577 TSecr=315375348
633 10.854417192 10.19.4.77 172.217.167.130 TLSv1.2 283 Application Data
634 10.877901233 180.149.60.171 10.19.4.77 TLSv1.2 218 Server Hello, Change Cipher Spec, Encrypted Handshake Message
635 10.877920695 10.19.4.77 180.149.60.171 TCP 66 53492 -> 443 [ACK] Seq=593 Ack=153 Win=30336 Len=0 TSval=1394902112 TSecr=166842579
636 10.877937824 180.149.60.171 10.19.4.77 TLSv1.2 218 Server Hello, Change Cipher Spec, Encrypted Handshake Message
637 10.877942819 10.19.4.77 180.149.60.171 TCP 66 53490 -> 443 [ACK] Seq=593 Ack=153 Win=30336 Len=0 TSval=1394902112 TSecr=166842579
638 10.877944831 180.149.60.171 10.19.4.77 TLSv1.2 218 Server Hello, Change Cipher Spec, Encrypted Handshake Message
639 10.877951678 10.19.4.77 180.149.60.171 TCP 66 53494 -> 443 [ACK] Seq=593 Ack=153 Win=30336 Len=0 TSval=1394902112 TSecr=166842580
640 10.878238503 10.19.4.77 180.149.60.171 TLSv1.2 117 Change Cipher Spec, Encrypted Handshake Message
641 10.881012327 172.217.163.163 10.19.4.77 TCP 66 443 -> 32824 [ACK] Seq=1 Ack=593 Win=283 Len=0 TSval=1700761312 TSecr=969861328
642 10.881418421 10.19.4.77 180.149.60.171 TLSv1.2 117 Change Cipher Spec, Encrypted Handshake Message
643 10.884337535 10.19.4.77 180.149.60.171 TLSv1.2 117 Change Cipher Spec, Encrypted Handshake Message
644 10.886332414 216.58.197.74 10.19.4.77 TLSv1.2 195 Application Data
645 10.890671576 10.19.4.77 172.217.167.130 TLSv1.2 466 Application Data
646 10.892006412 10.19.4.77 180.149.60.171 TLSv1.2 544 Application Data
647 10.892613225 10.19.4.77 180.149.60.171 TLSv1.2 547 Application Data
648 10.892616826 10.19.4.77 180.149.60.171 TLSv1.2 543 Application Data
649 10.922742193 172.217.167.130 10.19.4.77 TCP 66 443 -> 44404 [ACK] Seq=831 Ack=474 Win=238 Len=0 TSval=166842583 TSecr=315375387
650 10.922761917 180.149.60.171 10.19.4.77 TCP 66 443 -> 53492 [ACK] Seq=153 Ack=644 Win=19584 Len=0 TSval=166842584 TSecr=1394902116
651 10.922765697 172.217.167.130 10.19.4.77 TCP 66 443 -> 44404 [ACK] Seq=831 Ack=974 Win=253 Len=0 TSval=166842584 TSecr=315375423
652 10.922769337 180.149.60.171 10.19.4.77 TCP 66 443 -> 53494 [ACK] Seq=153 Ack=644 Win=19584 Len=0 TSval=166842584 TSecr=1394902112
653 10.922772965 180.149.60.171 10.19.4.77 TCP 66 443 -> 53494 [ACK] Seq=153 Ack=644 Win=19584 Len=0 TSval=166842584 TSecr=1394902119
654 10.929278205 10.19.4.77 216.58.197.74 TCP 66 42914 -> 443 [ACK] Seq=40 Ack=40 Win=363 Len=0 TSval=359207322 TSecr=166842583
```

Figure 10: Example of Messages Transferred

### B) Pause

Just as Pause button was pressed, a small number of packets were transmitted from the client to the server and vice versa. The content of these messages can however not be seen as TLS encrypts the message.

```
17755 168.348988643 10.19.4.77 124.124.201.232 TLSv1.2 696 Application Data
17756 168.348173682 10.19.4.77 124.124.201.232 TCP 1514 35182 -> 443 [ACK] Seq=16147 Ack=3079 Win=515 Len=1448 TSval=3748876737 TSecr=186445844 [TCP segment of a reassembled PDU]
17757 168.348184651 10.19.4.77 124.124.201.232 TCP 17595 Ack=3079 Win=515 Len=1448 TSval=3748876737 TSecr=186445844 [TCP segment of a reassembled PDU]
17758 168.349713833 10.19.4.77 124.124.201.232 TLSv1.2 856 Application Data
17759 168.352276872 124.124.201.232 10.19.4.77 TCP 66 443 -> 35182 [ACK] Seq=3079 Ack=16147 Win=565 Len=0 TSval=186449793 TSecr=3748876737
17760 168.352290515 124.124.201.232 10.19.4.77 TCP 66 443 -> 35182 [ACK] Seq=3079 Ack=17595 Win=588 Len=0 TSval=186449793 TSecr=3748876737
17761 168.352294156 124.124.201.232 10.19.4.77 TCP 66 443 -> 35182 [ACK] Seq=3079 Ack=19043 Win=611 Len=0 TSval=186449793 TSecr=3748876737
17762 168.352297485 124.124.201.232 10.19.4.77 TCP 66 443 -> 35182 [ACK] Seq=3079 Ack=19833 Win=633 Len=0 TSval=186449793 TSecr=3748876737
17763 168.378002686 10.19.4.77 124.124.201.232 TLSv1.2 785 Application Data
17764 168.378496625 10.19.4.77 124.124.201.232 TLSv1.2 866 Application Data
17765 168.380716071 124.124.201.232 10.19.4.77 TCP 66 443 -> 35188 [ACK] Seq=1027 Ack=6531 Win=372 Len=0 TSval=186449796 TSecr=3748876767
17766 168.380805252 124.124.201.232 10.19.4.77 TCP 66 443 -> 35188 [ACK] Seq=1027 Ack=7331 Win=395 Len=0 TSval=186449796 TSecr=3748876767
17767 168.424974468 10.19.5.1 10.19.7.255 UDP 305 54915 -> 54915 Len=263
17768 168.459305088 124.124.201.232 10.19.4.77 TLSv1.2 579 Application Data
17769 168.459377626 10.19.4.77 124.124.201.232 TCP 66 35182 -> 443 [ACK] Seq=19833 Ack=3592 Win=538 Len=0 TSval=3748876848 TSecr=186449803
17770 168.491670804 124.124.201.232 10.19.4.77 TLSv1.2 579 Application Data
17771 168.491749268 10.19.4.77 124.124.201.232 TCP 66 65185 -> 443 [ACK] Seq=7331 Ack=1540 Win=455 Len=0 TSval=3748876881 TSecr=186449807
```

Figure 11: Segments Transferred on Pause



Following this **Keep Alive** segments were sent from the client to the server telling the server **not to close** the TCP connection even though no data is being currently transferred. **Acknowledgement** for those Keep Alive messages can also be seen. These Keep Alive messages are very small in size and therefore do not use much bandwidth while the video is paused. **These segments are essential otherwise the connection would be closed and no further data could be transferred even if the video is unpaused.**

18019	192.253296404	10.19.4.77	180.149.60.171	TCP	66 [TCP Keep-Alive]	35490 → 443 [ACK]	Seq=60332 Ack=31027470 Win=550656 Len=0 TSval=1437485786 TSecr=186447679
18020	192.253333318	10.19.4.77	180.149.60.171	TCP	66 [TCP Keep-Alive]	35500 → 443 [ACK]	Seq=1166 Ack=4978 Win=45696 Len=0 TSval=1437485786 TSecr=186447679
18021	192.253347262	10.19.4.77	216.58.196.194	TCP	66 [TCP Keep-Alive]	58994 → 443 [ACK]	Seq=1980 Ack=3984 Win=420 Len=0 TSval=2490148958 TSecr=186447679
18022	192.253358587	10.19.4.77	104.244.42.131	TCP	66 [TCP Keep-Alive]	45116 → 443 [ACK]	Seq=497 Ack=264 Win=348 Len=0 TSval=1888175320 TSecr=186447679
18023	192.253368501	10.19.4.77	104.244.42.133	TCP	66 [TCP Keep-Alive]	49694 → 443 [ACK]	Seq=324 Ack=276 Win=348 Len=0 TSval=3108977748 TSecr=186447679
18024	192.253379331	10.19.4.77	151.101.8.157	TCP	66 [TCP Keep-Alive]	46738 → 443 [ACK]	Seq=268 Ack=154 Win=334 Len=0 TSval=472258145 TSecr=186447679
18025	192.253389220	10.19.4.77	104.18.100.194	TCP	66 [TCP Keep-Alive]	60440 → 443 [ACK]	Seq=294 Ack=1346 Win=342 Len=0 TSval=11649493619 TSecr=186447678
18026	192.253399610	10.19.4.77	172.217.167.230	TCP	66 [TCP Keep-Alive]	50921 → 443 [ACK]	Seq=1886 Ack=3016 Win=519 Len=0 TSval=238895622 TSecr=186447678
18027	192.253418023	10.19.4.77	216.58.221.34	TCP	66 [TCP Keep-Alive]	56856 → 443 [ACK]	Seq=749 Ack=2010 Win=376 Len=0 TSval=339842417 TSecr=186447678
18028	192.253420725	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35190 → 443 [ACK]	Seq=2675 Ack=514 Win=402 Len=0 TSval=3748906643 TSecr=186447678
18029	192.253439903	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35180 → 443 [ACK]	Seq=5297 Ack=1027 Win=432 Len=0 TSval=3748906643 TSecr=186447678
18030	192.253442236	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35186 → 443 [ACK]	Seq=9735 Ack=2476 Win=500 Len=0 TSval=3748906643 TSecr=186447678
18031	192.253452452	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35184 → 443 [ACK]	Seq=3151 Ack=514 Win=402 Len=0 TSval=3748906643 TSecr=186447678
18032	192.256217913	180.149.60.171	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35490 [ACK]	Seq=31027470 Ack=60333 Win=138880 Len=0 TSval=186452183 TSecr=1437395573
18033	192.259074019	216.58.196.194	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 58994 [ACK]	Seq=3984 Ack=1981 Win=221 Len=0 TSval=186452183 TSecr=2489968558
18034	192.259109156	104.244.42.131	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 45116 [ACK]	Seq=264 Ack=498 Win=169 Len=0 TSval=186452183 TSecr=1887994572
18035	192.259116975	172.217.167.230	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 50921 [ACK]	Seq=3816 Ack=1887 Win=260 Len=0 TSval=186452183 TSecr=238715372
18036	192.259123097	180.149.60.171	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35500 [ACK]	Seq=4978 Ack=1167 Win=20600 Len=0 TSval=186452183 TSecr=1437395556
18037	192.259134885	104.244.42.133	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 49694 [ACK]	Seq=276 Ack=325 Win=169 Len=0 TSval=186452183 TSecr=3108706097
18038	192.259141726	151.101.8.157	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 46738 [ACK]	Seq=154 Ack=269 Win=169 Len=0 TSval=186452183 TSecr=472076635
18039	192.259148513	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35188 [ACK]	Seq=1027 Ack=5298 Win=327 Len=0 TSval=186452183 TSecr=3748719901
18040	192.259155055	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35186 [ACK]	Seq=2476 Ack=9736 Win=463 Len=0 TSval=186452183 TSecr=3748719873
18041	192.259161325	104.18.100.194	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 60440 [ACK]	Seq=1346 Ack=295 Win=169 Len=0 TSval=186452183 TSecr=1164738078
18042	192.259167831	216.58.221.34	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 56856 [ACK]	Seq=2010 Ack=750 Win=182 Len=0 TSval=186452183 TSecr=339661668
18043	192.259174442	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35190 [ACK]	Seq=514 Ack=2676 Win=249 Len=0 TSval=186452183 TSecr=3748719806
18044	192.259181007	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35184 [ACK]	Seq=514 Ack=3152 Win=282 Len=0 TSval=186452183 TSecr=3748719795

Figure 12: Keep Alive Messages

## C) Play

Again a small number of segments are exchanged. Now the Keep Alive Messages stop though.

8107	203.486852844	10.19.4.77	124.124.201.232	TLSv1.2	785 Application Data		
8108	203.487846644	10.19.4.77	124.124.201.232	TLSv1.2	867 Application Data		
8109	203.492554800	124.124.201.232	10.19.4.77	TCP	66 443 → 35188 [ACK]	Seq=1540 Ack=8050 Win=418 Len=0 TSval=1864533306 TSecr=3748911876	
8110	203.492573399	124.124.201.232	10.19.4.77	TCP	66 443 → 35188 [ACK]	Seq=1540 Ack=8851 Win=440 Len=0 TSval=1864533306 TSecr=3748911877	
8111	203.648819033	124.124.201.232	10.19.4.77	TLSv1.2	579 Application Data		
8112	203.648846804	10.19.4.77	124.124.201.232	TCP	66 35188 → 443 [ACK]	Seq=8851 Ack=2053 Win=477 Len=0 TSval=3748912038 TSecr=186453317	
8113	203.811780651	10.19.4.77	172.217.166.238	TLSv1.2	355 Application Data		
8114	203.812767423	10.19.4.77	172.217.166.238	TLSv1.2	105 Application Data		
8115	203.814744827	172.217.166.238	10.19.4.77	TCP	66 443 → 56282 [ACK]	Seq=824 Ack=2242 Win=227 Len=0 TSval=186453339 TSecr=1461686994	
8116	203.815779379	172.217.166.238	10.19.4.77	TCP	66 443 → 56282 [ACK]	Seq=824 Ack=2281 Win=227 Len=0 TSval=186453339 TSecr=1461686995	
8117	203.874724637	172.217.166.238	10.19.4.77	TLSv1.2	143 Application Data		
8118	203.874745911	172.217.166.238	10.19.4.77	TLSv1.2	132 Application Data		
8119	203.874749969	172.217.166.238	10.19.4.77	TLSv1.2	105 Application Data		
8120	203.874752662	172.217.166.238	10.19.4.77	TLSv1.2	105 Application Data		
8121	203.874816115	10.19.4.77	172.217.166.238	TCP	66 56282 → 443 [ACK]	Seq=2281 Ack=1045 Win=273 Len=0 TSval=1461687057 TSecr=186453345	
8122	203.875468975	10.19.4.77	172.217.166.238	TLSv1.2	105 Application Data		
8123	203.876873938	172.217.166.238	10.19.4.77	TCP	66 443 → 56282 [ACK]	Seq=1045 Ack=2320 Win=227 Len=0 TSval=186453345 TSecr=1461687058	

Figure 13: Segments Transferred on Play

## D) Skip 10 Seconds

Similar transfer of packets takes place between the client and the server.

# Question 4: Protocols Used in Functioning of The Application

## A) TCP Handshake

TCP connection is established via a **3 way handshake**. First the client sends a **SYN** segment to the server. The server responds with **ACK** for the request. It also sends a SYN in the same segment. Finally, the client responds with ACK and the connection is setup.

137	7.132079633	10.19.4.77	103.254.155.35	TCP	74 47232 → 443 [SYN]	Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3059431405 TSecr=0 WS=128
138	7.132650602	103.254.155.35	10.19.4.77	TCP	74 443 → 47232 [SYN, ACK]	Seq=0 Ack=1 Win=18328 Len=0 MSS=9176 SACK_PERM=1 TSval=161581868 TSecr=3059431404 WS=128
139	7.132699938	10.19.4.77	103.254.155.35	TCP	66 47230 → 443 [ACK]	Seq=1 Ack=1 Win=29312 Len=0 TSval=3059431406 TSecr=161581868

Figure 14: TCP Handshake

## B) Application Layer Handshake

Before the client and the server can begin exchanging application data over TLS, the encrypted tunnel must be negotiated. The client and the server must agree on the version of the TLS protocol. They must choose the ciphersuite, and verify certificates if necessary.

The first step of the handshake is **Client Hello**. Within this, the client sends a number of specifications in plain text, such as the version of the TLS protocol it is running, the list of supported **ciphersuites**, and other TLS options it may want to use.

The second step is **Server Hello**. The server picks the TLS protocol version for further communication. It decides on a ciphersuite from the list provided by the client, attaches its certificate, and sends the response back to the client. Optionally, the server can also send a request for the clients certificate and parameters for other TLS extensions.

The third step is **Client and Server Key Exchange**. Assuming both sides are able to negotiate a common version and cipher, and the client is happy with the certificate provided by the server, the client initiates either the RSA or the Diffie-Hellman key exchange, which is used to establish the symmetric key for the ensuing session.

The final step is **ChangeCipherSpec**. The server processes the key exchange parameters sent by the client, checks

message integrity by verifying the MAC (Message Authentication Code), and returns an encrypted Finished message back to the client. On the client side, the message is decrypted with the negotiated symmetric key and the MAC is verified. If all is well, then the connection is established.

159	7.311958403	10.19.4.77	103.254.155.35	TLSv1.2	679 Client Hello
160	7.313175589	103.254.155.35	10.19.4.77	TCP	66 443 → 47236 [ACK] Seq=1 Ack=614 Win=19584 Len=0 TSval=161581887 TSecr=3059431585
209	7.578291699	103.254.155.35	10.19.4.77	TLSv1.2	1514 Server Hello
210	7.578318209	10.19.4.77	103.254.155.35	TCP	66 47232 → 443 [ACK] Seq=614 Ack=1449 Win=32128 Len=0 TSval=3059431851 TSecr=161581913
211	7.589801141	103.254.155.35	10.19.4.77	TLSv1.2	1823 Certificate, Server Key Exchange, Server Hello Done
212	7.589829513	10.19.4.77	103.254.155.35	TCP	66 47232 → 443 [ACK] Seq=614 Ack=3206 Win=35712 Len=0 TSval=3059431863 TSecr=161581913
213	7.590766125	10.19.4.77	103.254.155.35	TLSv1.2	159 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

Figure 15: TLS Handshake

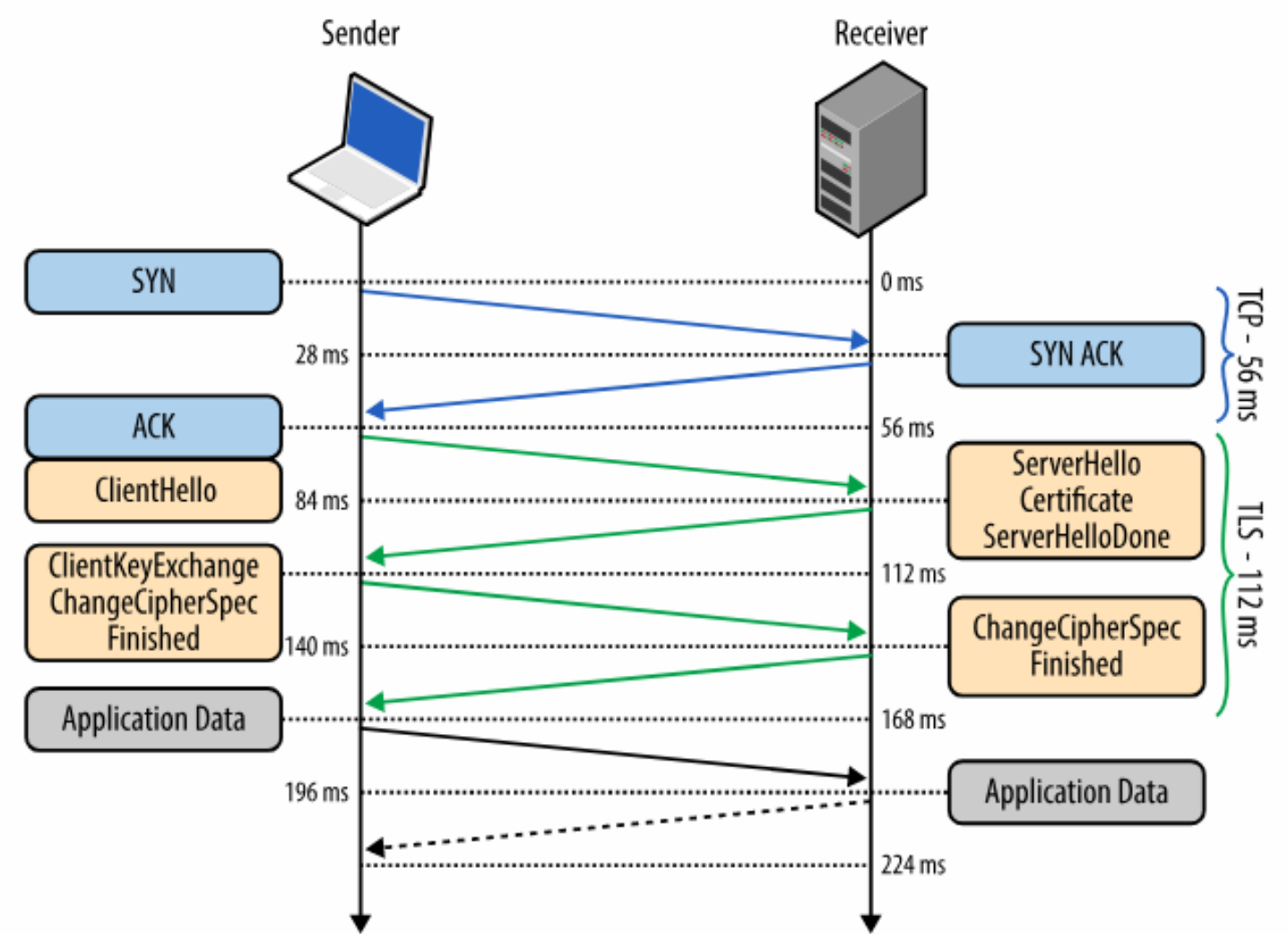


Figure 16: The Complete Picture

### C) Streaming

While streaming video, messages are sent from the server to the client containing the information about the video. This can be seen in the image as TLS segments are labeled Application data. Some segments might be fragmented that are then joined again to form the complete message. Once the segments arrive, they are re-assembled and the payload is delivered to the application layer.

3510	24.583799324	180.149.60.171	10.19.4.77	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
3511	24.583807241	180.149.60.171	10.19.4.77	TLSv1.2	1427 Application Data
3512	24.583949637	10.19.4.77	180.149.60.171	TCP	66 45726 → 443 [ACK] Seq=42023 Ack=4088655 Win=1885 Len=0 TSval=1368579662 TSecr=161583613
3513	24.602495970	34.201.234.134	10.19.4.77	TLSv1.2	267 Application Data
3514	24.602561687	10.19.4.77	34.201.234.134	TCP	66 49696 → 443 [ACK] Seq=45290 Ack=1081 Win=749 Len=0 TSval=2303178708 TSecr=161583615

Figure 17: Application Data Example

### D) Pause

The message transfered on pressing pause is shown below. A small number of segments are exchanged though their content cannot be understood as TLS encrypts them.

17755	168.348088643	10.19.4.77	124.124.201.232	TLSv1.2	696 Application Data				
17756	168.348178682	10.19.4.77	124.124.201.232	TCP	1514 35182 → 443 [ACK]	Seq=16147 Ack=3079 Win=515 Len=1448 TSval=3748876737 TSecr=186445844	[TCP segment of a reassembled PDU]		
17757	168.348184651	10.19.4.77	124.124.201.232	TCP	1514 35182 → 443 [ACK]	Seq=17595 Ack=3079 Win=515 Len=1448 TSval=3748876737 TSecr=186445844	[TCP segment of a reassembled PDU]		
17758	168.349713833	10.19.4.77	124.124.201.232	TLSv1.2	856 Application Data				
17759	168.352276872	124.124.201.232	10.19.4.77	TCP	66 443 → 35182 [ACK]	Seq=3079 Ack=16147 Win=565 Len=0 TSval=186449793 TSecr=3748876737			
17760	168.352290951	124.124.201.232	10.19.4.77	TCP	66 443 → 35182 [ACK]	Seq=3079 Ack=17595 Win=588 Len=0 TSval=186449793 TSecr=3748876737			
17761	168.352294156	124.124.201.232	10.19.4.77	TCP	66 443 → 35182 [ACK]	Seq=3079 Ack=19843 Win=611 Len=0 TSval=186449793 TSecr=3748876737			
17762	168.352297485	124.124.201.232	10.19.4.77	TCP	66 443 → 35182 [ACK]	Seq=3079 Ack=19833 Win=633 Len=0 TSval=186449793 TSecr=3748876739			
17763	168.378002586	10.19.4.77	124.124.201.232	TLSv1.2	785 Application Data				
17764	168.378496625	10.19.4.77	124.124.201.232	TLSv1.2	866 Application Data				
17765	168.380716071	124.124.201.232	10.19.4.77	TCP	66 443 → 35188 [ACK]	Seq=1027 Ack=6531 Win=372 Len=0 TSval=186449796 TSecr=3748876767			
17766	168.380805252	124.124.201.232	10.19.4.77	TCP	66 443 → 35188 [ACK]	Seq=1027 Ack=7331 Win=395 Len=0 TSval=186449796 TSecr=3748876767			
17767	168.424974404	10.19.4.77	10.19.7.255	UDP	305 54915 → 54925 Len=263				
17768	168.459305088	124.124.201.232	10.19.4.77	TLSv1.2	579 Application Data				
17769	168.459377626	10.19.4.77	124.124.201.232	TCP	66 35182 → 443 [ACK]	Seq=19833 Ack=3592 Win=538 Len=0 TSval=3748876848 TSecr=186449803			
17770	168.491678804	124.124.201.232	10.19.4.77	TLSv1.2	579 Application Data				
17771	168.491746268	10.19.4.77	124.124.201.232	TCP	66 35188 → 443 [ACK]	Seq=7331 Ack=1549 Win=455 Len=0 TSval=3748876881 TSecr=186449807			

Figure 18: Segments Transferred on Pause

**Keep Alive** segments are also sent from the client to the server as explained in the previous question. These segments and also their acknowledgements can be seen from the screenshot attached.

18019	192.253296404	10.19.4.77	180.149.60.171	TCP	66 [TCP Keep-Alive]	35490 → 443 [ACK]	Seq=60332 Ack=31027470 Win=550656 Len=0 TSval=1437485786 TSecr=186447679		
18020	192.253333318	10.19.4.77	180.149.60.171	TCP	66 [TCP Keep-Alive]	35580 → 443 [ACK]	Seq=1168 Ack=4878 Win=45696 Len=0 TSval=1437485786 TSecr=186447679		
18021	192.253347262	10.19.4.77	216.58.196.194	TCP	66 [TCP Keep-Alive]	58994 → 443 [ACK]	Seq=1980 Ack=3084 Win=420 Len=0 TSval=3748900643 TSecr=186447679		
18022	192.253358587	10.19.4.77	184.244.42.131	TCP	66 [TCP Keep-Alive]	45116 → 443 [ACK]	Seq=497 Ack=264 Win=348 Len=0 TSval=1868175320 TSecr=186447679		
18023	192.253368501	10.19.4.77	104.244.42.133	TCP	66 [TCP Keep-Alive]	49694 → 443 [ACK]	Seq=324 Ack=276 Win=348 Len=0 TSval=3708977748 TSecr=186447679		
18024	192.253379331	10.19.4.77	151.101.8.157	TCP	66 [TCP Keep-Alive]	46738 → 443 [ACK]	Seq=268 Ack=154 Win=334 Len=0 TSval=172258145 TSecr=186447679		
18025	192.253389220	10.19.4.77	104.18.100.194	TCP	66 [TCP Keep-Alive]	60440 → 443 [ACK]	Seq=294 Ack=1346 Win=342 Len=0 TSval=1164919619 TSecr=186447678		
18026	192.253399613	10.19.4.77	172.217.167.230	TCP	66 [TCP Keep-Alive]	50924 → 443 [ACK]	Seq=1886 Ack=3016 Win=619 Len=0 TSval=238895622 TSecr=186447678		
18027	192.253410023	10.19.4.77	216.58.221.34	TCP	66 [TCP Keep-Alive]	56656 → 443 [ACK]	Seq=749 Ack=2010 Win=376 Len=0 TSval=339842417 TSecr=186447678		
18028	192.253420725	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35190 → 443 [ACK]	Seq=2675 Ack=514 Win=402 Len=0 TSval=3748900643 TSecr=186447678		
18029	192.253430903	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35180 → 443 [ACK]	Seq=5297 Ack=1027 Win=432 Len=0 TSval=3748900643 TSecr=186447678		
18030	192.253442236	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35186 → 443 [ACK]	Seq=9735 Ack=2476 Win=500 Len=0 TSval=3748900643 TSecr=186447678		
18031	192.253452452	10.19.4.77	124.124.201.232	TCP	66 [TCP Keep-Alive]	35184 → 443 [ACK]	Seq=3151 Ack=514 Win=402 Len=0 TSval=3748900643 TSecr=186447678		
18032	192.256217013	180.149.60.171	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35490 [ACK]	Seq=31027470 Ack=60333 Win=138800 Len=0 TSval=186452183 TSecr=1437395573		
18033	192.259074019	216.58.196.194	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 58994 [ACK]	Seq=3984 Ack=1981 Win=221 Len=0 TSval=186452183 TSecr=2489968558		
18034	192.259109156	104.244.42.131	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 45116 [ACK]	Seq=264 Ack=498 Win=169 Len=0 TSval=186452183 TSecr=1887994572		
18035	192.259116975	172.217.167.230	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 50924 [ACK]	Seq=3016 Ack=1887 Win=260 Len=0 TSval=186452183 TSecr=238715372		
18036	192.259123987	180.149.60.171	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35509 [ACK]	Seq=4978 Ack=1167 Win=20608 Len=0 TSval=186452183 TSecr=1437305556		
18037	192.259134885	104.244.42.133	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 46964 [ACK]	Seq=276 Ack=325 Win=169 Len=0 TSval=186452183 TSecr=3108796997		
18038	192.259141726	151.101.8.157	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 46738 [ACK]	Seq=154 Ack=269 Win=169 Len=0 TSval=186452183 TSecr=472076635		
18039	192.259148513	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35180 [ACK]	Seq=1027 Ack=5298 Win=327 Len=0 TSval=186452183 TSecr=3748719901		
18040	192.259155055	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35186 [ACK]	Seq=2476 Ack=9736 Win=463 Len=0 TSval=186452183 TSecr=3748719873		
18041	192.259161325	180.149.60.171	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 60440 [ACK]	Seq=1346 Ack=295 Win=169 Len=0 TSval=186452183 TSecr=1164738078		
18042	192.259167031	216.58.221.34	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 50924 [ACK]	Seq=2010 Ack=760 Win=192 Len=0 TSval=186452183 TSecr=33061660		
18043	192.259174442	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35190 [ACK]	Seq=514 Ack=2676 Win=249 Len=0 TSval=186452183 TSecr=3748719806		
18044	192.259181007	124.124.201.232	10.19.4.77	TCP	66 [TCP Keep-Alive ACK]	443 → 35184 [ACK]	Seq=514 Ack=3152 Win=282 Len=0 TSval=186452183 TSecr=3748719795		

Figure 19: Keep Alive Messages

## E) Play

Again a small number of segments are exchanged. Now the Keep Alive Messages stop though.

8107	203.486852844	10.19.4.77	124.124.201.232	TLSv1.2	785 Application Data				
8108	203.487846644	10.19.4.77	124.124.201.232	TLSv1.2	867 Application Data				
8109	203.492554800	124.124.201.232	10.19.4.77	TCP	66 443 → 35188 [ACK]	Seq=1540 Ack=8050 Win=418 Len=0 TSval=186453306 TSecr=3748911876			
8110	203.492573399	124.124.201.232	10.19.4.77	TCP	66 443 → 35188 [ACK]	Seq=1540 Ack=8851 Win=440 Len=0 TSval=186453306 TSecr=3748911877			
8111	203.648819033	124.124.201.232	10.19.4.77	TLSv1.2	579 Application Data				
8112	203.648846804	10.19.4.77	124.124.201.232	TCP	66 35188 → 443 [ACK]	Seq=8851 Ack=2053 Win=477 Len=0 TSval=3748912038 TSecr=186453317			
8113	203.811780651	10.19.4.77	172.217.166.238	TLSv1.2	355 Application Data				
8114	203.812767423	10.19.4.77	172.217.166.238	TLSv1.2	105 Application Data				
8115	203.814744827	172.217.166.238	10.19.4.77	TCP	66 443 → 56282 [ACK]	Seq=824 Ack=2242 Win=227 Len=0 TSval=186453339 TSecr=1461686994			
8116	203.815779379	172.217.166.238	10.19.4.77	TCP	66 443 → 56282 [ACK]	Seq=824 Ack=2281 Win=227 Len=0 TSval=186453339 TSecr=1461686995			
8117	203.874724637	172.217.166.238	10.19.4.77	TLSv1.2	143 Application Data				
8118	203.874745911	172.217.166.238	10.19.4.77	TLSv1.2	132 Application Data				
8119	203.874749969	172.217.166.238	10.19.4.77	TLSv1.2	105 Application Data				
8120	203.874752662	172.217.166.238	10.19.4.77	TLSv1.2	105 Application Data				
8121	203.874816115	10.19.4.77	172.217.166.238	TCP	66 56282 → 443 [ACK]	Seq=2281 Ack=1045 Win=273 Len=0 TSval=1461687057 TSecr=186453345			
8122	203.875468975	10.19.4.77	172.217.166.238	TLSv1.2	105 Application Data				
8123	203.876873938	172.217.166.238	10.19.4.77	TCP	66 443 → 56282 [ACK]	Seq=1045 Ack=2320 Win=227 Len=0 TSval=186453345 TSecr=1461687058			

Figure 20: Segments Transferred on Play

## F) TCP Connection Termination

The TCP connection termination phase uses a four-way handshake, with each side of the connection terminating independently. The client sends a FIN (piggybacked by an ACK) to the server which is ACKed by it. The server in turn sends a FIN which is ACKed by the client.

6643	39.287422852	10.19.4.77	103.254.155.35	TCP	66 47246 → 443 [FIN, ACK]	Seq=1318 Ack=6137 Win=46080 Len=0 TSval=3859463561 TSecr=161584542			
6644	39.289306541	103.254.155.35	10.19.4.77	TCP	66 443 → 47248 [ACK]	Seq=1288 Ack=2118 Win=22400 Len=0 TSval=161585084 TSecr=3059463560			
6645	39.289341458	103.254.155.35	10.19.4.77	TCP	66 443 → 47248 [FIN, ACK]	Seq=1288 Ack=2119 Win=22400 Len=0 TSval=161585084 TSecr=3059463560			
6646	39.289362892	10.19.4.77	103.254.155.35	TCP	66 47248 → 443 [ACK]	Seq=2119 Ack=1289 Win=32640 Len=0 TSval=3059463563 TSecr=161585084			

Figure 21: TCP Connection Termination

## Question 5: Statistical Analysis

The data has been collected at 3 different times of the day :-

1. **10:30 AM** in my room in Lohit using Mozilla Firefox web browser.
2. **5:00 PM** in my room in Lohit using Mozilla Firefox web browser.
3. **22:00 PM** in **Anchorenza and RadioG room** in New Sac using Mozilla Firefox web browser.

Statistic Name	10:30 (Room)	17:00 (Room)	22:00 (New Sac)
Throughput	38795.027 Bytes/sec	217567.01 Bytes/sec	994580.82 Bytes/sec
RTT	40.56 ms	9.52 ms	0.2 ms
Packet Size	1171.67 Bytes	1221.18 Bytes	1433.02 Bytes
Number of Packets Lost	0	0	0
Number of UDP Packets	950	794	2180
Number of TCP Packets	9476	29635	41782
Number of Responses Received per Request	6138/3511 (1.74)	19530/10446 (1.87)	26376/15475 (1.70)

Table 1: Statistical Ananlysis at 3 Different Times

Although Hotsar does not use UDP at the transport layer, UDP packets are still observed. The reason is that DNS uses UDP packets and DNS resolution is an integral part of running any web application to find its servers IP address from the given hostname.

## Question 6: Multiple Sources

The data arrived from many different IP addresses. A few of which have been listed below.

- ★ **180.149.60.171**
- ★ **180.149.60.168**
- ★ **180.149.60.139**
- ★ **124.124.201.232**
- ★ **172.217.166.238**
- ★ **103.254.155.35**

Most of the data arrived from the first 2 addresses. It can be seen from the attached screenshot that the canonical name for hotstar is **cname.hotstar.edgesuite.net** and the two servers are actually **Akamai** servers. Akamai is the leading **Content Delivery Network** (CDN) services provider for media and software delivery. So we can conclude that hotstar uses Akamai's services for content delivery.

Data is sent from multiple servers because it leads to **Load Balancing**. Data comes

through different paths so there is less network congestion and lesser load on each individual server. Multiple servers also leads to increased reliability as there is **no single point of failure**. Even if one server crashes, the service does not stop.

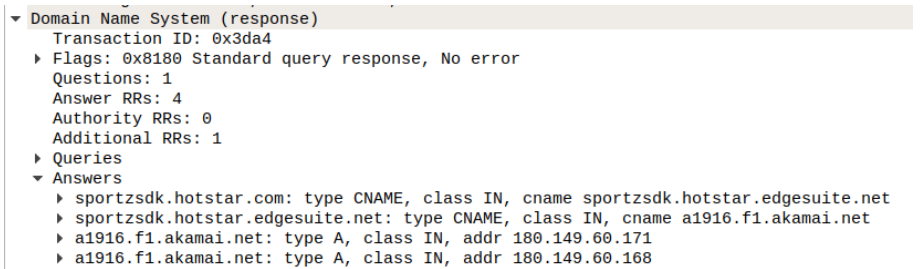


Figure 22: DNS Response