CS-349 NETWORKS ASSIGNMENT-2

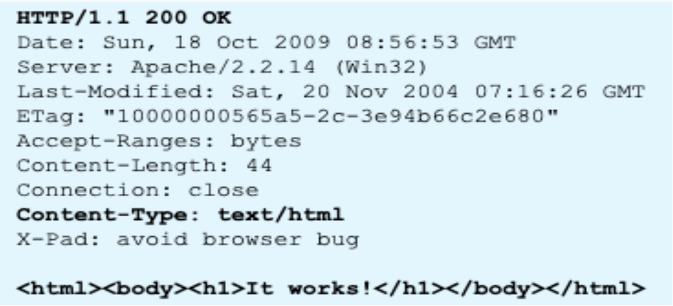
-MAYANK WADHWANI (170101038)

TOPIC ASSIGNED: VIDEO STREAMING ON HOTSTAR.COM
TRACES: bit.ly/397rhsc

QUESTION-1 PROTOCOLS USED IN DIFFERENT LAYERS AND THEIR PACKET FORMATS

A) APPLICATION LAYER:

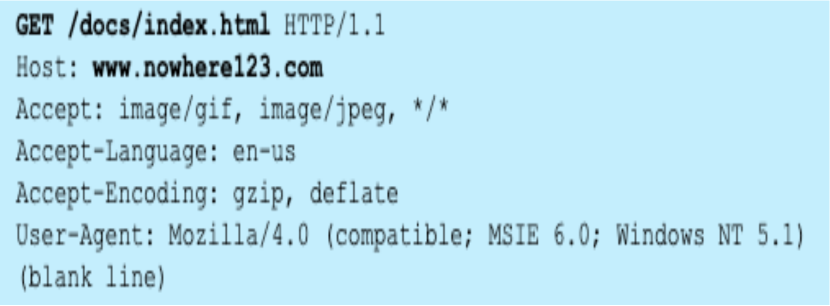
1) **HTTP**: HTTP stands for **HyperText Transfer Protocol** and it is used in the application layer. There are two types of messages that are sent between the web server and the web client: namely **HTTP request** and **HTTP response**. An HTTP request consists of the request line which specifies what type of request is asked for(it can be GET, POST, PUT, DELETE or HEAD) and the HTTP version and the headers which contain additional information like language accepted. It may also contain an HTTP Body to send additional information. The web server responds to the HTTP request by generating an HTTP response. HTTP response comprises of an initial status line(like 404 means not found), header lines and an entity body.



```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<html><body><h1>It works!</h1></body></html>
```

Figure 1 An HTTP response



```
GET /docs/index.html HTTP/1.1
Host: www.nowhere123.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
(blank line)
```

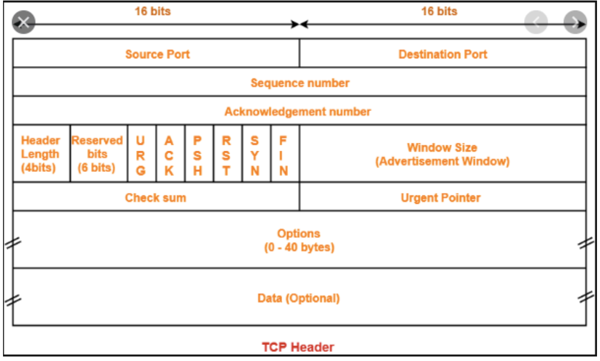
Figure 2 An HTTP request

B) SESSION LAYER

TLS: The TLS layer sits between the Application layer and the transport layer. It provides security in transmission by encrypting the data to be sent. The basic unit of SSL is a **record**. Each record consists of a five-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

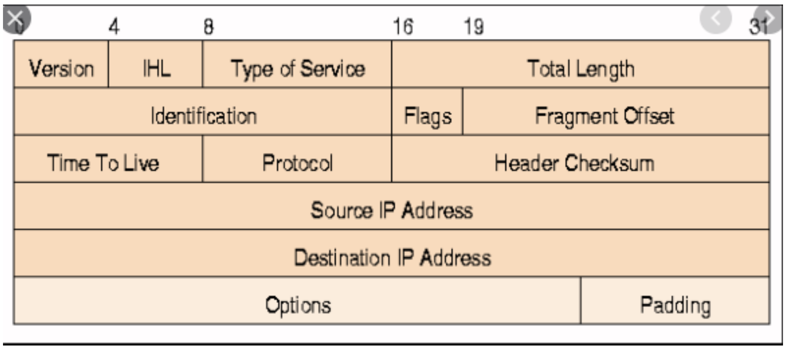
C) TRANSPORT LAYER

TCP: The **Transmission Control Protocol** is a standard that defines how to establish and maintain a network connection between application programs that wish to send data over the network. The TCP packet header consists of the following fields: **Source Port** (16bits) and **Destination Port** (16 bits) to correctly identify the address of the communicating hosts. **Sequence Number** and **Acknowledgement Number** (32 bits each) are used to maintain the ordering of the packets being transferred. **Header Length** (4 Bits) specifies the total size of the TCP header in multiples of 4 bytes. **Reserved bits** (3 bits) serves the purpose of aligning the total header size so that it becomes a multiple of 4. **Window size** (32 bits) is used to regulate the amount of data to be sent to a receiver before requiring an acknowledgement in return. The **Check Sum** (2 bits) is used for error detection. The **Urgent Pointer** field is used to point to data that is urgently required and is often set to zero and ignored. **Data** (variable length) contains upper layer information.



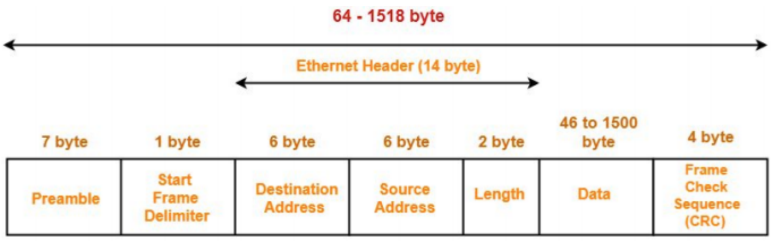
D) NETWORKS LAYER

IPV4 is a connectionless protocol or use on packet switched networks. An IP packet can contain up to 60 bytes of IP header data. The following are the fields one observes in an IP packet header. The **Version** (4 bits) specifies the version no of the internet protocol used (4 in this case). The **Internet Header Length (IHL**) (4 bits) represents the length of the header. The **Type of service** field itself contains 2 fields viz. the **Differentiated Services Code Point(DSCP)** (6 bits) which is used to define differentiated services like VoIP (Voice over IP) and **Explicit Congestion Notification(ECN)** (2 bits) allows end-to-end notification of network congestion without dropping any packets. **Total length** (16 bits) field defines the entire length of the IP packet. The **Identification** (16 bits) field is used to uniquely identify group of fragments for a single IP datagram. The **Fragment Offset** (13 bits) is used to tell the exact position of the fragment in the original IP Packet. The **TTL or Time To Live** (8 bits) tells how many routers at maximum the packet can hop through. So at each router, its value is decremented by 1 and as a result of which if we exceed to max routers allowed the value becomes 0 and the packet is destroyed. The **Protocol** (8 bits) specifies the protocol used in the next layer. The **Header Checksum**(16 bits) is used for error detection just as in the TCP packet. The **Source and Destination IP Address** (16 bits each) specifies the IP address of the client and the server in our case i.e. the source and the destination respectively. The **Options** field is used for additional information.



E) LINK LAYER

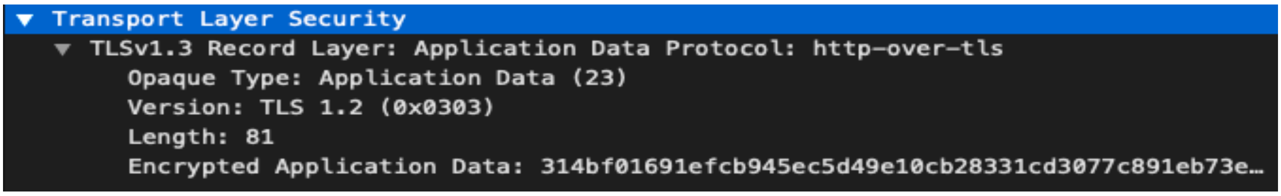
Ethernet II is used at the link layer. A data unit on an Ethernet Link transports an Ethernet frame as its payload. An Ethernet frame consists of the following: A **Preamble** and a **Start Frame Delimiter(SFD**) to mark the starting of the frame. Then **Destination Address** and **Source Address** follows which specified the MAC Addresses of the client and the host in our case which is the source and the destination respectively. The **Data** also called as the **Payload** is inserted next. In the end comes the **Frame Check Sequence (FCS)** which is used to detect errors, if any using the **Cyclic Redundancy Check (CRC).**



QUESTION-2 OBSERVED VALUES

The protocols whose values were observed and listed below are **TLS, TCP** and **Ethernet II.**

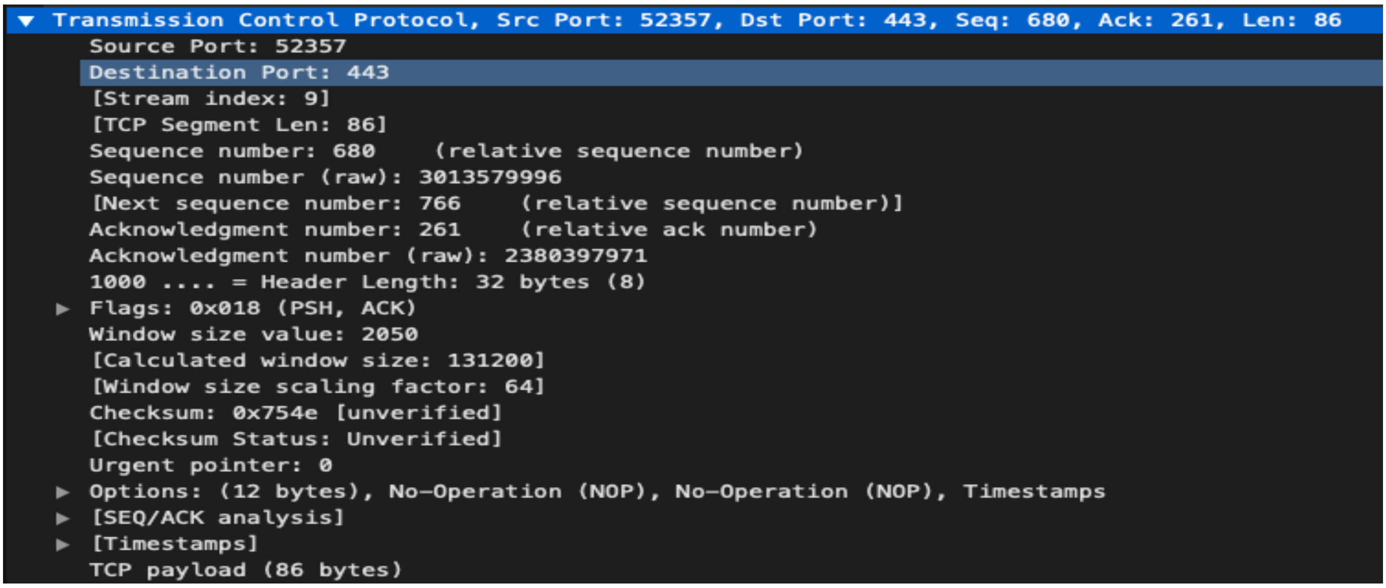
A) TLS:



```
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 81
      Encrypted Application Data: 314bf01691efcb945ec5d49e10cb28331cd3077c891eb73e…
```

Opaque Type	Application Data	Refers to the type of the content being transferred. 23 is an identifier for the same and it means that Application Data will be transferred
Version	TLS 1.2	Refers to the version of the protocol that is used
Length	81	The Length of the application data being transferred, excluding the protocol headers and including the MAC and padding trailers
Encrypted Application Data	314bf0169…	The encrypted form of the data that is sent over the network to provide security.
Application Data Protocol	http-over-tls	The protocol ensures secure communication over the network by encrypting the data and sending it through http requests.

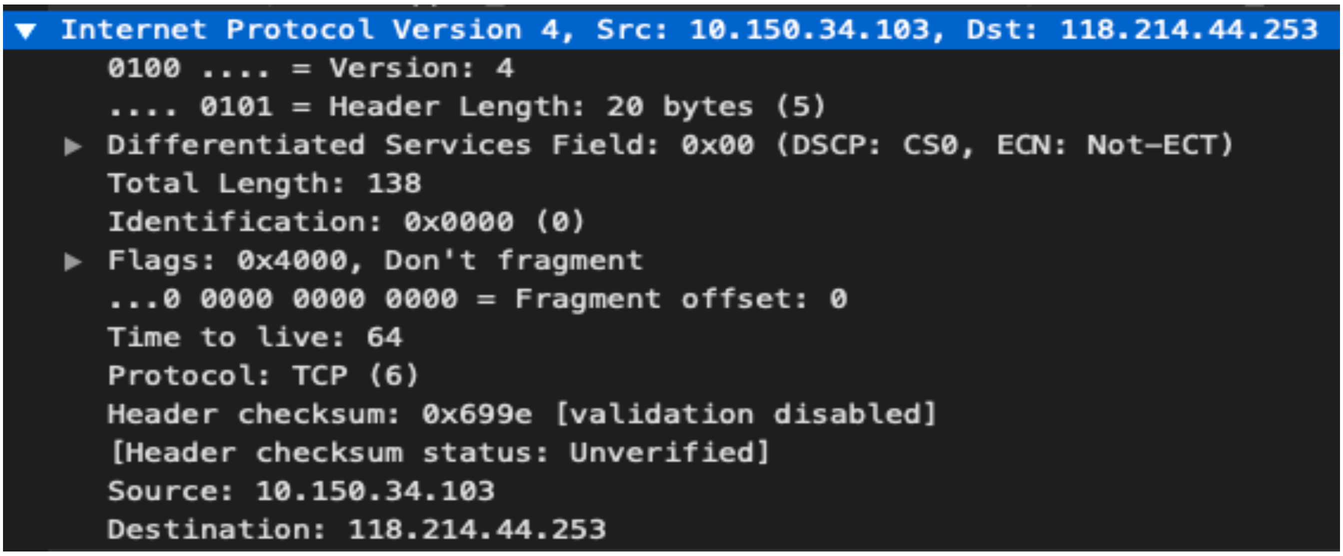
B) TCP



```
▼ Transmission Control Protocol, Src Port: 52357, Dst Port: 443, Seq: 680, Ack: 261, Len: 86
      Source Port: 52357
      Destination Port: 443
      [Stream index: 9]
      [TCP Segment Len: 86]
      Sequence number: 680    (relative sequence number)
      Sequence number (raw): 3013579996
      [Next sequence number: 766    (relative sequence number)]
      Acknowledgment number: 261    (relative ack number)
      Acknowledgment number (raw): 2380397971
      1000 .... = Header Length: 32 bytes (8)
   ▶ Flags: 0x018 (PSH, ACK)
      Window size value: 2050
      [Calculated window size: 131200]
      [Window size scaling factor: 64]
      Checksum: 0x754e [unverified]
      [Checksum Status: Unverified]
      Urgent pointer: 0
   ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
   ▶ [SEQ/ACK analysis]
   ▶ [Timestamps]
      TCP payload (86 bytes)
```

Source Port	52357	The source port number is used by the sending host to help keep track of new incoming connections and existing data streams.
Destination Port	443	Similar to the source port, the destination port is used by the receiver to keep track of new incoming connections.
Sequence Number	680	The number assigned to the packet relative to the advent of the TCP connection. So we can say that 679 packets have been sent since the TCP connection has been set up.
Acknowledgement Number	261	The acknowledgement number is the sequence number of the next byte the receiver expects to receive
Checksum	0x754e	The checksum field is used for error detection.
Flags	PSH,ACK	PSH or the PUSH flag is an option provided by TCP that allows the sending application to start sending the data even when the buffer is not full. The ACK which stands for "Acknowledgment", is used to acknowledge the successful receipt of a packet
Urgent Pointer	0	The Urgent Pointer field is used to point to data that is urgently required. Here there is no such requirement and so its value is set to 0.
Header Length	8	Header length is 32 bytes so the value 8 is stored as the counting happens in multiples of 4 bytes.

C) IPv4

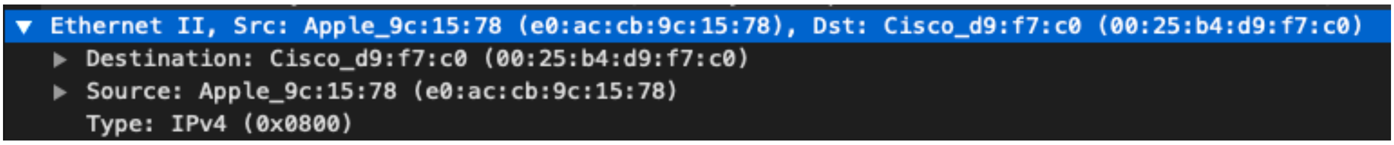


```
▼ Internet Protocol Version 4, Src: 10.150.34.103, Dst: 118.214.44.253
     0100 .... = Version: 4
     .... 0101 = Header Length: 20 bytes (5)
   ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
     Total Length: 138
     Identification: 0x0000 (0)
   ▶ Flags: 0x4000, Don't fragment
     ...0 0000 0000 0000 = Fragment offset: 0
     Time to live: 64
     Protocol: TCP (6)
     Header checksum: 0x699e [validation disabled]
     [Header checksum status: Unverified]
     Source: 10.150.34.103
     Destination: 118.214.44.253
```

Version	4	The version of internet protocol used
Header Length	5	Header length is 20 bytes so the value is 5 as counting is done in multiples of 4 bytes
Src	10.150.34.103	The IP address of the sender, in this case my laptop
Dst	118.214.44.253	The IP address of the receiver, in this case the hotstar server
Flags	Don't Fragment	If this flag is set and fragmentation is required to route the packet, then the packet is simply dropped.

TTL	64	This represents the maximum hops the packet can make through the routers. So after 64 hops the packet will be dropped.
DSCP	CS0	This means standard service is applied in the network and this will have undifferentiated applications.
ECN	Not-ECT	This means not-ECN capable transport i.e. ECN is not used in the connection.

D) Ethernet II



```
▼ Ethernet II, Src: Apple_9c:15:78 (e0:ac:cb:9c:15:78), Dst: Cisco_d9:f7:c0 (00:25:b4:d9:f7:c0)
  ▶ Destination: Cisco_d9:f7:c0 (00:25:b4:d9:f7:c0)
  ▶ Source: Apple_9c:15:78 (e0:ac:cb:9c:15:78)
    Type: IPv4 (0x0800)
```

Destination	00:25:b4:d9:f7:c0	Refers to the MAC address of the destination server, in this case the server of hotstar.com
Source	E0:ac:cb:9c:15:78	Refers to the MAC address of the source, in this case my laptop
Type	IPv4	Means that the upper layer protocol used is IPv4

Question-3 : Protocols helping in various functionalities

TLS:

a) TLS is used for prevention of unwanted **eavesdropping** and modification on internet traffic.

b) The protocol helps to provide security to the website from **external hackers** as now the data is encrypted so even if someone breaks into the website and gets access to the data, he/she will not be able to decipher it.

c) So whenever we are entering, let's say login credentials on a website, in this case I entered my credentials on hotstar.com, the password and username are safely transmitted from my machine to the server by using TLS.

TCP:

a) It was the only used protocol in this case i.e. hotstar.com only uses TCP for transmission of data and not UDP. There are several reasons for the same.

b) It provides **reliability** of data i.e. if we send a packet using TCP, then we can be absolutely sure that the packet will reach the destination safely without any loss. UDP does not guarantee this, it is not as reliable as TCP.

c) It works by creating a TCP connection. Also, the TLS protocol discussed above works only for the applications that send their data through TCP. So if we are using UDP for transmission then we will not get **security** which is a major disadvantage. So TCP is both secure and reliable.

d) Since the user should be given the facility to go back and watch that part of the clip, this means that the application can't afford to lose any information. This can be ensured by TCP as there is a process of **hand shaking** done at the start. As a result of which, the error rates are minimized.

IPv4:

a) **IPv4** is a connection less protocol used for packet-switched networks like the Internet.

b) It delivers packets using IP headers from source to the destination.

c) This protocol is used in the **networks** layer.

Ethernet:

a) Ethernet is used in the **data link** layer.

b) It stores the actual physical MAC Addresses of the source and the destination hosts.

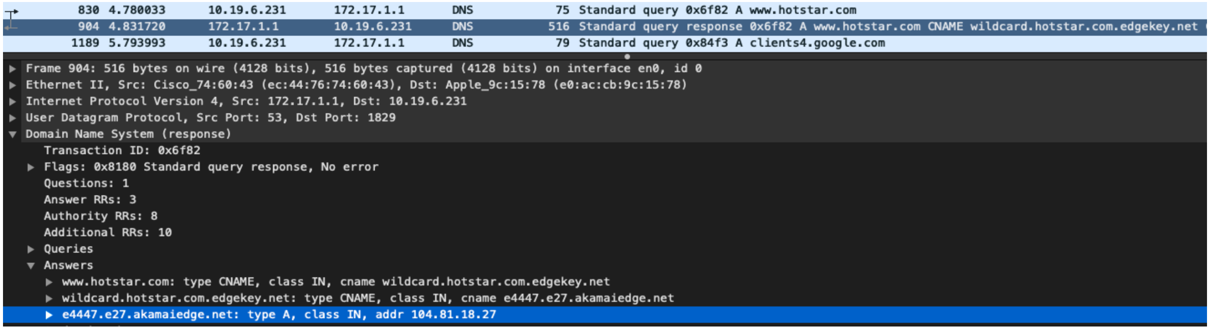
c) Lying on the data link layer, it is responsible for **error detection** and correctness. Various algorithms and techniques exist for the same including **CRC** (Cyclic Redundancy Check).

QUESTION-4 SEQUENCE OF MESSAGES

The sequence of messages were observed for various functionalities like playing a video, pausing it, closing the web browser, etc. All these have been listed below:

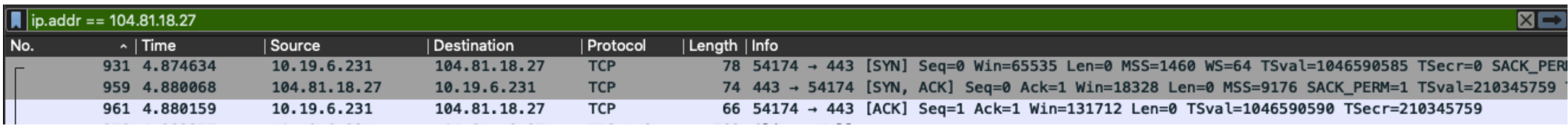
A) DNS Querying

When we open the site, DNS querying is done by the browser, here Google Chrome. A series of messages are exchanged so that the browser can know the IP Address of the server ie hotstar.com. In the attached screenshot, it can be seen that a query was made to hotstar.com and the response contains the IP address of the server. **The answer part is selected(blue colour) in the screenshot**. This IP was then used as a filter to get all the messages.



```
  830 4.780033    10.19.6.231     172.17.1.1      DNS        75 Standard query 0x6f82 A www.hotstar.com
  904 4.831720    172.17.1.1      10.19.6.231     DNS       516 Standard query response 0x6f82 A www.hotstar.com CNAME wildcard.hotstar.com.edgekey.net
 1189 5.793993    10.19.6.231     172.17.1.1      DNS        79 Standard query 0x84f3 A clients4.google.com
▶ Frame 904: 516 bytes on wire (4128 bits), 516 bytes captured (4128 bits) on interface en0, id 0
▶ Ethernet II, Src: Cisco_74:60:43 (ec:44:76:74:60:43), Dst: Apple_9c:15:78 (e0:ac:cb:9c:15:78)
▶ Internet Protocol Version 4, Src: 172.17.1.1, Dst: 10.19.6.231
▶ User Datagram Protocol, Src Port: 53, Dst Port: 1829
▼ Domain Name System (response)
    Transaction ID: 0x6f82
  ▶ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 8
    Additional RRs: 10
  ▶ Queries
  ▼ Answers
    ▶ www.hotstar.com: type CNAME, class IN, cname wildcard.hotstar.com.edgekey.net
    ▶ wildcard.hotstar.com.edgekey.net: type CNAME, class IN, cname e4447.e27.akamaiedge.net
    ▶ e4447.e27.akamaiedge.net: type A, class IN, addr 104.81.18.27
```

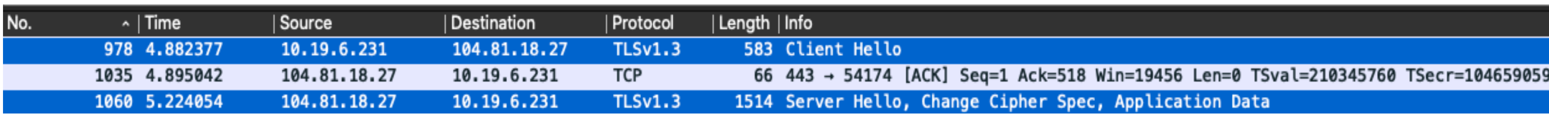
B) TCP Handshaking



```
ip.addr == 104.81.18.27
No.      Time       Source          Destination     Protocol  Length Info
  931 4.874634    10.19.6.231     104.81.18.27    TCP         78 54174 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1046590585 TSecr=0 SACK_PERI
  959 4.880068    104.81.18.27    10.19.6.231     TCP         74 443 → 54174 [SYN, ACK] Seq=0 Ack=1 Win=18328 Len=0 MSS=9176 SACK_PERM=1 TSval=210345759
  961 4.880159    10.19.6.231     104.81.18.27    TCP         66 54174 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1046590590 TSecr=210345759
```

After DNS querying, we get the IP address of the server. Following this, there is a **three-way-handshake** between the source, the destination. We have already seen In the previous question that port **541474** is being used in my laptop and port **443** is being used by hotstar.com. The handshaking process advents when the host sends a **SYN** packet to the destination. This SYN packet helps in **SYNchronizing** the sequence number. The destination then responds to this packet by sending an **ACK** (acknowledgement) and another SYN packet which asks the source to SYNchronize the packet number with its sequence number. In the end, the source sends a final ACK packet and with this, we say that the handshaking process is successful.

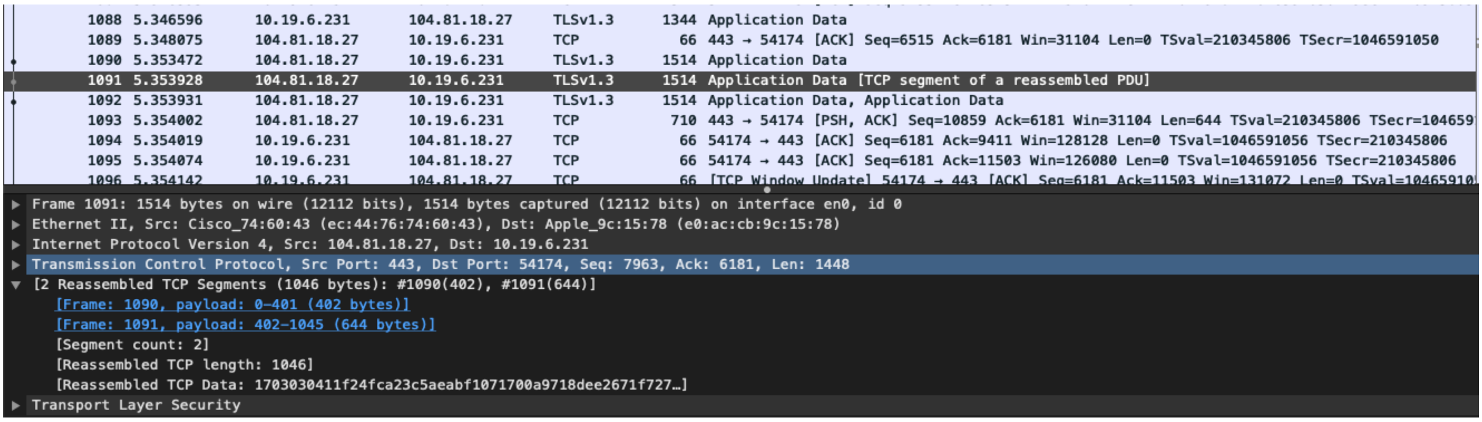
C) TLS Handshaking



```
No.      Time       Source          Destination     Protocol  Length Info
  978 4.882377    10.19.6.231     104.81.18.27    TLSv1.3    583 Client Hello
 1035 4.895042    104.81.18.27    10.19.6.231     TCP         66 443 → 54174 [ACK] Seq=1 Ack=518 Win=19456 Len=0 TSval=210345760 TSecr=104659059
 1060 5.224054    104.81.18.27    10.19.6.231     TLSv1.3   1514 Server Hello, Change Cipher Spec, Application Data
```

Once TCP connection is successfully established, we move on to TLS establishment. The TLS protocol send a ‘Client Hello’ message to the server. This can be seen from the first line of the screenshot where the source is my laptop and destination is server’s IP address ie. 104.81.18.27. The server responds to this message by a ‘Server Hello’ message and a Server Certificate(used primarily for authentication). The Server Hello signifies that the server is now ready to take requests from the client. We say that the TLS session is established and communication can take place successfully.

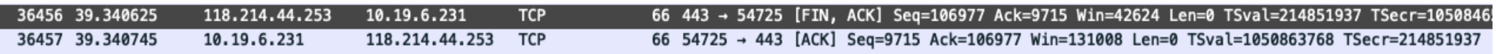
D) Streaming Videos



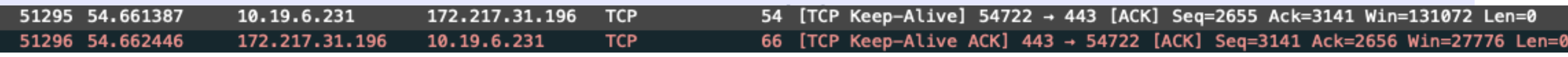
```
 1088 5.346596    10.19.6.231     104.81.18.27    TLSv1.3   1344 Application Data
 1089 5.348075    104.81.18.27    10.19.6.231     TCP         66 443 → 54174 [ACK] Seq=6515 Ack=6181 Win=31104 Len=0 TSval=210345806 TSecr=1046591050
 1090 5.353472    104.81.18.27    10.19.6.231     TLSv1.3   1514 Application Data
 1091 5.353928    104.81.18.27    10.19.6.231     TLSv1.3   1514 Application Data [TCP segment of a reassembled PDU]
 1092 5.353931    104.81.18.27    10.19.6.231     TLSv1.3   1514 Application Data, Application Data
 1093 5.354002    104.81.18.27    10.19.6.231     TCP        710 443 → 54174 [PSH, ACK] Seq=10859 Ack=6181 Win=31104 Len=644 TSval=210345806 TSecr=104659
 1094 5.354019    10.19.6.231     104.81.18.27    TCP         66 54174 → 443 [ACK] Seq=6181 Ack=9411 Win=128128 Len=0 TSval=1046591056 TSecr=210345806
 1095 5.354074    10.19.6.231     104.81.18.27    TCP         66 54174 → 443 [ACK] Seq=6181 Ack=11503 Win=126080 Len=0 TSval=1046591056 TSecr=210345806
 1096 5.354142    10.19.6.231     104.81.18.27    TCP         66 [TCP Window Update] 54174 → 443 [ACK] Seq=6181 Ack=11503 Win=131072 Len=0 TSval=10465910
▶ Frame 1091: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface en0, id 0
▶ Ethernet II, Src: Cisco_74:60:43 (ec:44:76:74:60:43), Dst: Apple_9c:15:78 (e0:ac:cb:9c:15:78)
▶ Internet Protocol Version 4, Src: 104.81.18.27, Dst: 10.19.6.231
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 54174, Seq: 7963, Ack: 6181, Len: 1448
▼ [2 Reassembled TCP Segments (1046 bytes): #1090(402), #1091(644)]
    [Frame: 1090, payload: 0-401 (402 bytes)]
    [Frame: 1091, payload: 402-1045 (644 bytes)]
    [Segment count: 2]
    [Reassembled TCP length: 1046]
    [Reassembled TCP Data: 1703030411f24fca23c5aeabf1071700a9718dee2671f727…]
▶ Transport Layer Security
```

When we play the video on hotstar.com, packets are sent from the server to the client. Now these packets may take different paths to reach the client. These paths may depend on a lot of factors like load balancing. So it is possible that these packets arrive out of order. But when the packets arrive out of order, they are not directly passed to the application layer. We wait for the rest packets to arrive. So in this situation, we can see in the screenshot that the packet number 1091 and 1090 were reassembled so that the data becomes in order. Once reassembling is done, we send the data to the Application layer.

E) Pausing A Video



```
36456  39.340625    118.214.44.253   10.19.6.231      TCP     66  443 → 54725 [FIN, ACK] Seq=106977 Ack=9715 Win=42624 Len=0 TSval=214851937 TSecr=1050846
36457  39.340745    10.19.6.231      118.214.44.253   TCP     66  54725 → 443 [ACK] Seq=9715 Ack=106977 Win=131008 Len=0 TSval=1050863768 TSecr=214851937
```

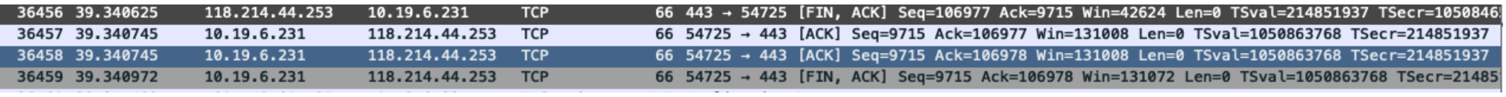


```
51295  54.661387    10.19.6.231      172.217.31.196   TCP     54  [TCP Keep-Alive] 54722 → 443 [ACK] Seq=2655 Ack=3141 Win=131072 Len=0
51296  54.662446    172.217.31.196   10.19.6.231      TCP     66  [TCP Keep-Alive ACK] 443 → 54722 [ACK] Seq=3141 Ack=2656 Win=27776 Len=0
```

When we pause the video, some data still comes at the client end till the receiving buffer is full. Once it is full, the client sends a FIN piggybacked by an ACK to the server to stop the packet flow and the server acknowledges. But since the user has only paused the video and can play it anytime in the future, the client keeps sending Keep-Alive messages to the server notifying it to not close the connection. Once the user presses the play button again, the client sends a SYN packet to notify the server to start sending the packets again. In this manner the pause functionality works and the TCP connection stays established.

F) Closing the website

When the user closes the website, a 4-way termination handshake takes place. First the client sends a FIN packey piggybacked by an ACK packet. The server responds to this by sending an ACK packet from its end. It also sends a FIN packet to the client. The client acknowledges it by sending the final ACK packet to the server after which the connection successfully closes.



```
36456  39.340625    118.214.44.253   10.19.6.231      TCP     66  443 → 54725 [FIN, ACK] Seq=106977 Ack=9715 Win=42624 Len=0 TSval=214851937 TSecr=1050846
36457  39.340745    10.19.6.231      118.214.44.253   TCP     66  54725 → 443 [ACK] Seq=9715 Ack=106977 Win=131008 Len=0 TSval=1050863768 TSecr=214851937
36458  39.340745    10.19.6.231      118.214.44.253   TCP     66  54725 → 443 [ACK] Seq=9715 Ack=106978 Win=131008 Len=0 TSval=1050863768 TSecr=214851937
36459  39.340972    10.19.6.231      118.214.44.253   TCP     66  54725 → 443 [FIN, ACK] Seq=9715 Ack=106978 Win=131072 Len=0 TSval=1050863768 TSecr=21485
```

QUESTION-5 STATISTICS FROM TRACES

Following are the required statistics:

Property	Time 1 : 5:00 AM (CSE LAB)	Time 2: 2:30 PM (Lohit Hostel)	Time 3: 7:00 PM (Library)
Avg. Throughput (in Kilo Bytes per second)	254	212	593
RTT	0.00059272	0.0010439	0.0003713
Avg. Packet Size (in Bytes)	757	771	592
Number of packets lost	0	0	0
Number of UDP packets	142	1488	5301
Number of TCP packets	34343	17666	76188
Number of responses per request	23644/10699	9439/8227	47647/28541

The UDP packets seen in the table can be attributed mainly to DNS and other activities performed by the browser. However, hotstar.com does not use UDP for transmission of packets.

QUESTION-6 CONTENT SOURCES

For the Source:

It was observed that there was a change in source IP while sending the data. When the messages were sent in the morning from the lab, the observed IP address was 10.150.33.220. Whereas in the afternoon from my hostel (Lohit), observed IP was 10.19.6.231 and lastly observed IP address from the library in the evening was 10.150.34.103. For all these cases, I used IIT_CONNECT for internet connection. The main reason for difference in these IP's can be because at different locations, different routers have been set up. These routers have different IP's. So when I connect my laptop to these different routers, I observed different IP addresses for the sources. We can conclude that source IP depends on the **location** of the laptop.

For the Destination:

For this part, it was observed that the destination IP address changed at different times of the day. In the morning, the destination address was 104.81.18.27, whereas in the evening the observed IP was 180.149.60.168. Finally, in the afternoon, observed IP was 118.214.44.253. We observe different values because of:

a) **Load balancing**: Most of the websites have several servers set up across the world. This helps in **load balancing** i.e. if one particular server receives a lot of requests at one time, the next request is sent to some other server by the router. This helps to keep the network traffic stable.

b) **Reliability**: If due to any unforeseen reasons, the server fails, then there should be other servers which respond to the clients. If however we are using only one server, then the website crashes if that server goes down. So using multiple servers helps to provide **reliability**.