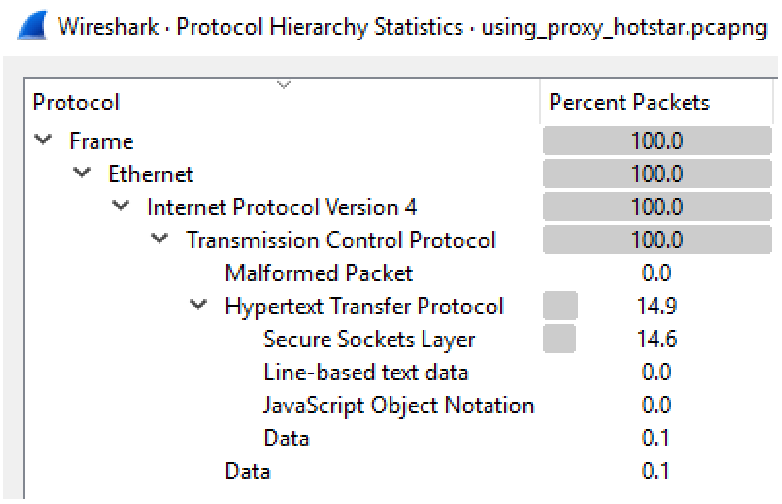
Cs-349 Network lab

Assignment -2 ***phoolchandra (160101051)***

Google Drive link
https://drive.google.com/drive/folders/1ARGBExOPPYSri4ppqo1ssSHscXvyw-7a?usp=sharing

Answer 1.

The protocols hierarchy used by the application (www.hotstar.com) a video live streaming website at different layer are listed in just below image. This protocol hierarchy comes when I collected the data using proxy.



Wireshark · Protocol Hierarchy Statistics · using_proxy_hotstar.pcapng

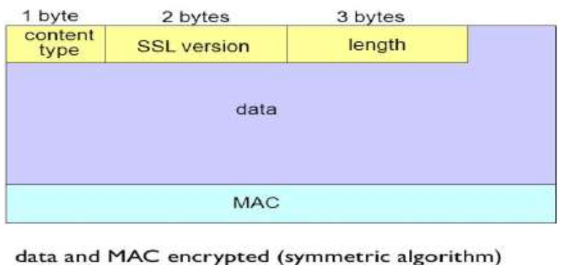
Protocol	Percent Packets
Frame	100.0
Ethernet	100.0
Internet Protocol Version 4	100.0
Transmission Control Protocol	100.0
Malformed Packet	0.0
Hypertext Transfer Protocol	14.9
Secure Sockets Layer	14.6
Line-based text data	0.0
JavaScript Object Notation	0.0
Data	0.1
Data	0.1

From left picture we know that udp and dns packets are zero because udp protocol is used by browser not by hotstar.com.

Application protocols:

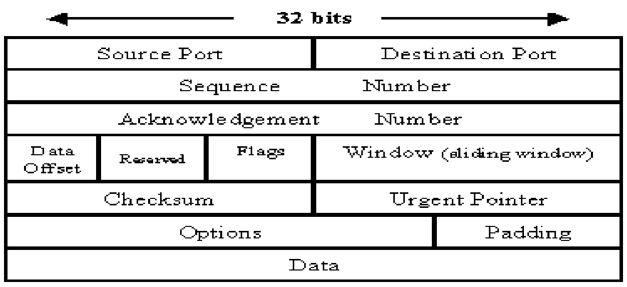
The Hypertext Transfer Protocol (HTTP): is an Application protocol for distributed, collaborative, hypermedia information systems. A typical HTTP request contains many fields like Accept (to specify media types), accept-charset, accept-encoding and accept language.It also contains fields for proxy authorization, range, host and User-agent. An 'expires' field is also present which specifies the time after which the response is considered stale. HTTP functions as a request–response protocol in the client–server computing model.

SSL - hotstar uses the TLSv1.2 which is upgraded version of SSL 3.0 . It provides security in the communication between two hosts. It provides integrity, authentication and confidentiality. The basic unit of data in SSL is a record. Each record consists of a five-byte record header, followed by data. The header contains – Record Type can be of four types(Handshake, Change Cipher Spec, Alert, Application Data), Record Version is 16-byte value formatted in network order, Record Length is 16-bytes Value.



<- SSL packet frame

Data - When Wireshark can't determine how part of a packet should be formatted, it marks that chunk as "Data". The "Data" is just the normal data payload.



Transport Layer protocols:

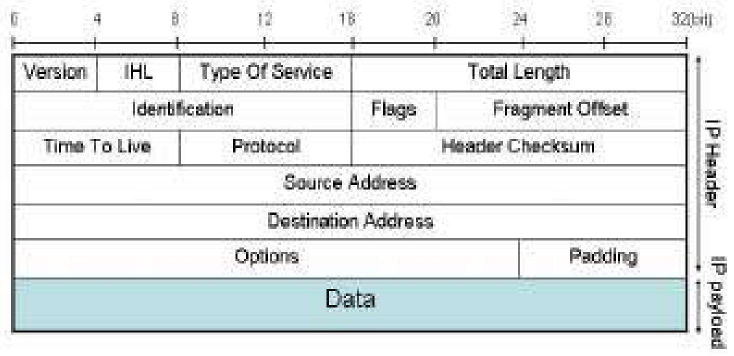
<- TCP packet frame

TCP – Each TCP header has ten required fields totaling 20 bytes in size. They can also optionally include an additional data section up to 40 bytes in size. TCP headers has - Source and destination TCP ports which are the communication endpoints for sending and receiving devices, sequence and acknowledgement numbers to mark the ordering in a group of messages, data offset stores the total size of a TCP header in multiples of four bytes, Reserved data in TCP headers always has a value of zero, a set of six standard and three extended control flags (each an individual bit representing on or off) to manage data flow in specific situations, window size to regulate how much data sender sends to a receiver before requiring an acknowledgment in return, checksum for error detection, urgent pointer can be used as a data offset to mark a subset of a message which require priority processing. Optional TCP data can be used to include support for special acknowledgment and window scaling algorithms.

Network Layer protocol:

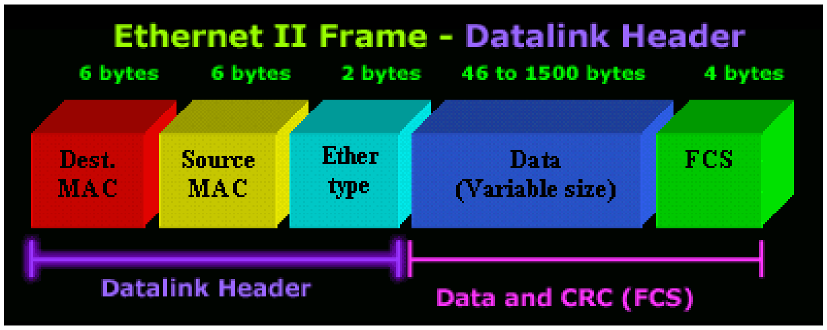
IPv4 is one of the core protocols of standards-based internetworking methods in the Internet. It is a connectionless protocol for use on packet-switched networks. The header consists of 14 fields, of which 13 are required. They are –

Version is always equal to 4, Internet Header Length (IHL) has 4 bits which is the number of 32-bit words in header, Differentiated Services Code Point (DSCP) used in QoS, Explicit Congestion Notification (ECN) allows end-to-end notification of network congestion without dropping packets, Total
Length is 16-bit field which defines the entire packet size in bytes, identification field is primarily used for uniquely identifying the group of fragments of a single IP datagram, flags bit is used to control or identify fragments (0th bit: Reserved and is always 0; 1st bit: Don't Fragment (DF); 2nd bit: More Fragments (MF)), Fragment Offset specifies the offset of a particular fragment, Time To Live (TTL) helps prevent datagrams from persisting on network forever, Protocol defines the protocol used in the data portion of the IP datagram, Header Checksum is used for error-checking of the header, Source address & Destination address is the IPv4 address of the sender and receiver of the packet respectively



<- IP4 packet frame

Link Layer protocols



Ethernet II - An Ethernet frame is preceded by a preamble and start frame delimiter (SFD), to identify the start of the frame. Each Ethernet frame starts with an Ethernet header, which contains destination and source MAC addresses as its first two fields. The middle section of the frame is payload data including any headers for other protocols (for example, Internet Protocol) carried in the frame. The frame ends with a frame check sequence (FCS), which is a 32-bit cyclic redundancy check used to detect any in-transit corruption of data.

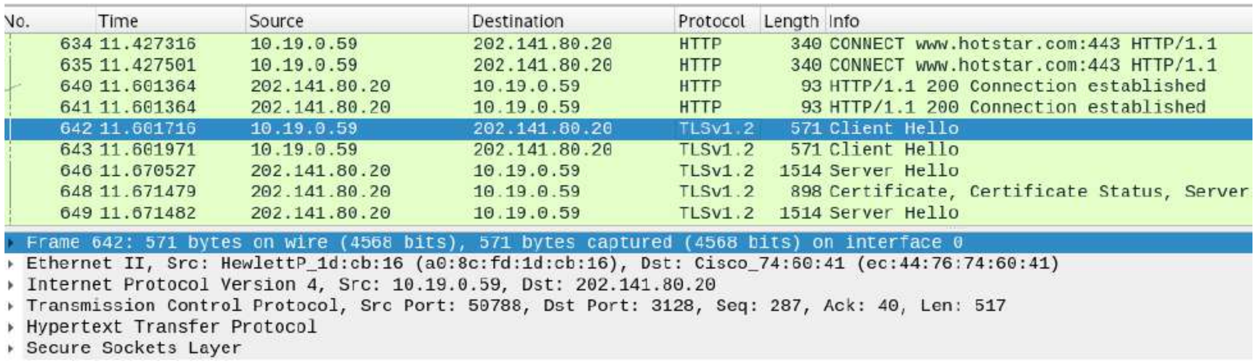
Answer 2.

Observed flags and options in collected data:

- ❏ **Source port**: this is the port number associated with the sender side
- ❏ **Destination number**: port number associated with the recipient side
- ❏ **Sequence number**: these are the unique values that are used to ensure reliable delivery of data
- ❏ **Acknowledgement number**: response from the receiver side as part of the confirmation process that the packet was successfully received.
- ❏ **Data set:** this indicates where the data packet begin and the length of the tcp header.
- ❏ **Flag:** there are various types of flag bits present. They initiate connection, carry data, and tear down connection. Their functionality is as follows:
- ❏ **SYN**: packets that are used to initiate a connection that is commonly known as the handshake process.
- ❏ **ACK(acknowledgement):** these packets are used to confirm that data the data packets have been received, and this also confirms the initiation and tear down of the connections.
- ❏ **rst(reset)**: these packets signify that the connection you were trying to create has been shut down or may be the application we were trying to communicate with is not accepting connections.
- ❏ **fin:** these packets indicate that the connection is being torn down after the successful delivery of data
- ❏ **psh(push):** these packets indicate that the incoming data should be passed on directly to the application instead of getting buffered.
- ❏ **URG(urgent):** market packets indicate that the data that the packet is carrying should be processed immediately by the tcp stack and the urgent pointer field should be examined if it is set.
- ❏ **cwr(congestion window reduced)**: these packets are used by the senders to inform the receiver that the buffer is getting overfilled ,and because of congestion , both the parties should slow down the transmission process to avoid any packet loss that might happen.

- ❑ **Window size:** this field in the header indicates the amount of data that the sender can send
- ❑ **Checksum**: to cross the content of the tcp segments.
- ❑ **Urgent pointer:** this field tells about the value that the urgent pointer contains. It specifically indicates the sequence number of the octet that lies before the data
- ❑ **Options:** this field has three parts: length of the option, option being used, option in use. One of the important option maximum segment size (mss) is also part of this field
- ❑ **data :** the last part in the tcp header is the real data travels around

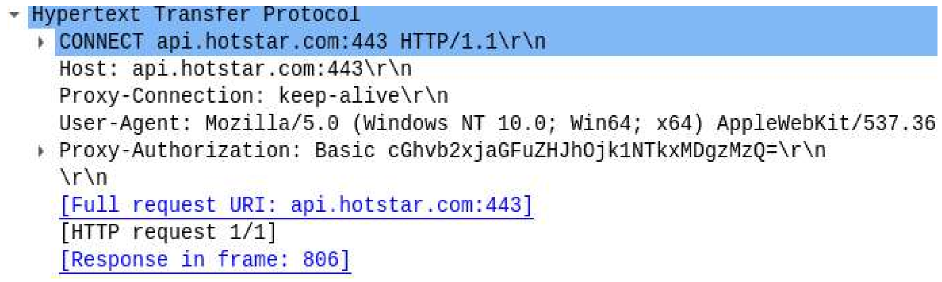
First I have launched browser and then started the wireshark. After It I open only www.hotstar.com in browser and played one video get the following data



```
No.   Time       Source          Destination     Protocol Length Info
  634 11.427316  10.19.0.59      202.141.80.20   HTTP       340 CONNECT www.hotstar.com:443 HTTP/1.1
  635 11.427501  10.19.0.59      202.141.80.20   HTTP       340 CONNECT www.hotstar.com:443 HTTP/1.1
  640 11.601364  202.141.80.20   10.19.0.59      HTTP        93 HTTP/1.1 200 Connection established
  641 11.601364  202.141.80.20   10.19.0.59      HTTP        93 HTTP/1.1 200 Connection established
  642 11.601716  10.19.0.59      202.141.80.20   TLSv1.2    571 Client Hello
  643 11.601971  10.19.0.59      202.141.80.20   TLSv1.2    571 Client Hello
  646 11.670527  202.141.80.20   10.19.0.59      TLSv1.2   1514 Server Hello
  648 11.671479  202.141.80.20   10.19.0.59      TLSv1.2    898 Certificate, Certificate Status, Server
  649 11.671482  202.141.80.20   10.19.0.59      TLSv1.2   1514 Server Hello

▸ Frame 642: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface 0
▸ Ethernet II, Src: HewlettP_1d:cb:16 (a0:8c:fd:1d:cb:16), Dst: Cisco_74:60:41 (ec:44:76:74:60:41)
▸ Internet Protocol Version 4, Src: 10.19.0.59, Dst: 202.141.80.20
▸ Transmission Control Protocol, Src Port: 50788, Dst Port: 3128, Seq: 287, Ack: 40, Len: 517
▸ Hypertext Transfer Protocol
▸ Secure Sockets Layer
```

In the above figure, ***time*** is the time elapsed since the starting of packets capture, ***source & destination*** are the sender & the receiver of the packets respectively, ***protocol*** is the protocol(of the highest layer) that the wireshark could identify, ***length*** is the length of the packet and info is a brief information contained in the packet decoded by wireshark. From above we got that there five protocols HTTP and SSL are application protocol, TCP is transport layer protocol, IPv4 is a network layer protocol and Ethernet II is a link layer protocol

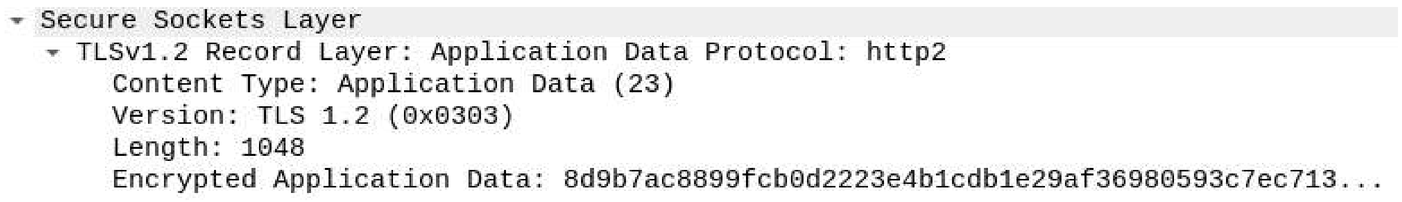


```
▾ Hypertext Transfer Protocol
  ▸ CONNECT api.hotstar.com:443 HTTP/1.1\r\n
    Host: api.hotstar.com:443\r\n
    Proxy-Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  ▸ Proxy-Authorization: Basic cGhvb2xjaGFuZHJhOjk1NTkxMDgzMzQ=\r\n
    \r\n
    [Full request URI: api.hotstar.com:443]
    [HTTP request 1/1]
    [Response in frame: 806]
```

HTTP – As seen in the figure, the Request method is connect to the request **URI**, which is the address of the hotstar server. The version of HTTP used is **1.1**. Since the connection is through IITG Proxy, **Proxy Authorisation** is also added. User agent is **Mozilla**. Wireshark also shows the previous request frame number and the frame number having response to this frame .

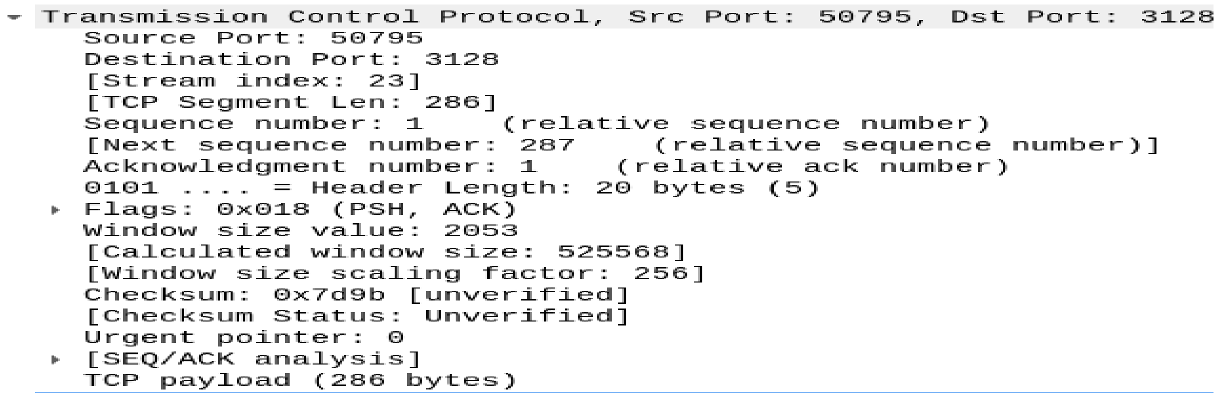
SSL – The HTTP part of the packet contains the hostname and the port, while the SSL part contains the **Encrypted Data.**(Note that both HTTP and TLSv1.2 are in application layer). WireShark is unable to determine the further headers of

the SSL packet. Here TLSv1.2 is



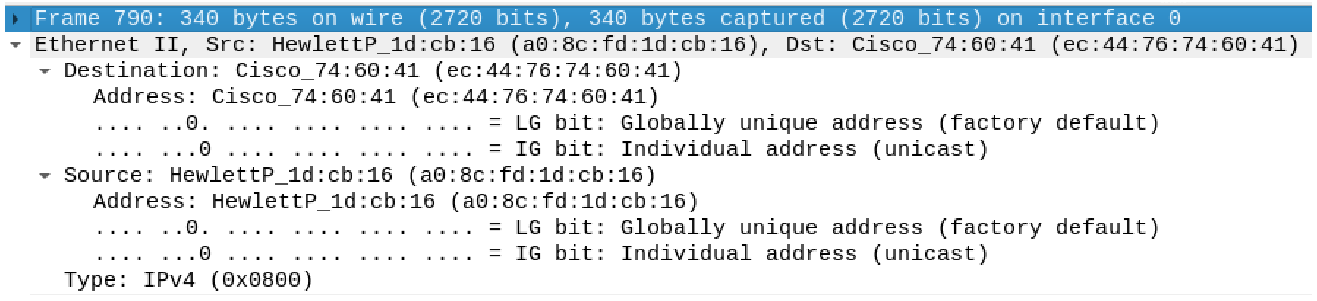
```
▾ Secure Sockets Layer
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http2
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 1048
      Encrypted Application Data: 8d9b7ac8899fcb0d2223e4b1cdb1e29af36980593c7ec713...
```

TCP: The packet contains the Destination and the Source Port, TCP Stream index, sequence number and the acknowledgement number, Header length, Flags. In the following figure, the flags are set as 010 in (HexaDecimal) which corresponds to the Acknowledgement. Window size value is 423. Checksum is used for error detection. Wireshark is remembering the value of Window size scaling factor and presenting it again. Scaling factor shows the number of leftward bit shifts that should be used for an advertised window size.



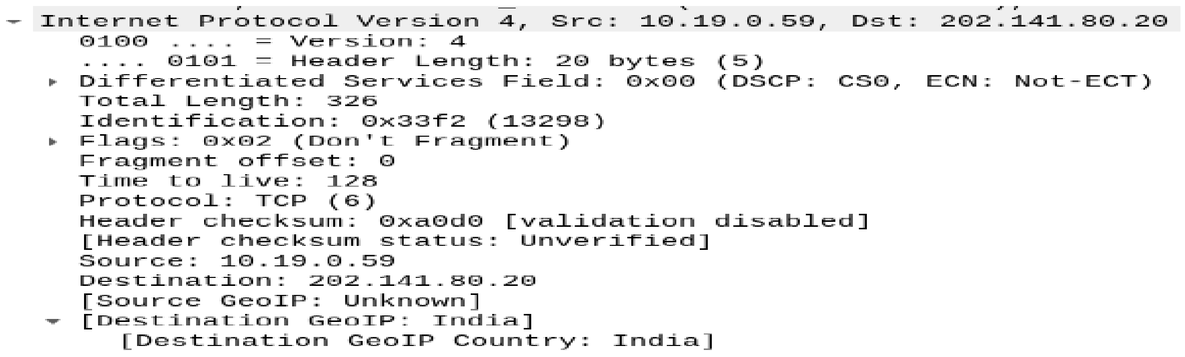
```
▾ Transmission Control Protocol, Src Port: 50795, Dst Port: 3128
    Source Port: 50795
    Destination Port: 3128
    [Stream index: 23]
    [TCP Segment Len: 286]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 287    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 2053
    [Calculated window size: 525568]
    [Window size scaling factor: 256]
    Checksum: 0x7d9b [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
    TCP payload (286 bytes)
```

IPv4 – Version header field is always 4 as we are using IPv4. Header length is 5 (which means 20 bytes because it counts in 4 Bytes word). Total length of the packet is 52 bytes. The flag set is don't fragment which instructs all the nodes through which the packet passes to not fragment the packet. TTL is 63. The protocol contained in it is TCP. The packet is sent by the proxy (202.141.80.20) to my Device (10.19.0.59).



```
▸ Frame 790: 340 bytes on wire (2720 bits), 340 bytes captured (2720 bits) on interface 0
▾ Ethernet II, Src: HewlettP_1d:cb:16 (a0:8c:fd:1d:cb:16), Dst: Cisco_74:60:41 (ec:44:76:74:60:41)
  ▾ Destination: Cisco_74:60:41 (ec:44:76:74:60:41)
      Address: Cisco_74:60:41 (ec:44:76:74:60:41)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: HewlettP_1d:cb:16 (a0:8c:fd:1d:cb:16)
      Address: HewlettP_1d:cb:16 (a0:8c:fd:1d:cb:16)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

Ethernet II – It contains the physical MAC address of the devices communicating. Destination is my HP Device and the source is the Switch to which my device is connected. Source is always Unicast. Destination is Unicast in this case. In both of them, it is Globally Unique Address and not a Local Address.



```
▾ Internet Protocol Version 4, Src: 10.19.0.59, Dst: 202.141.80.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 326
    Identification: 0x33f2 (13298)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0xa0d0 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.19.0.59
    Destination: 202.141.80.20
    [Source GeoIP: Unknown]
  ▾ [Destination GeoIP: India]
      [Destination GeoIP Country: India]
```

Answer 3:

for this question I have used the file named = data_at_2_am_using_proxy_overlan

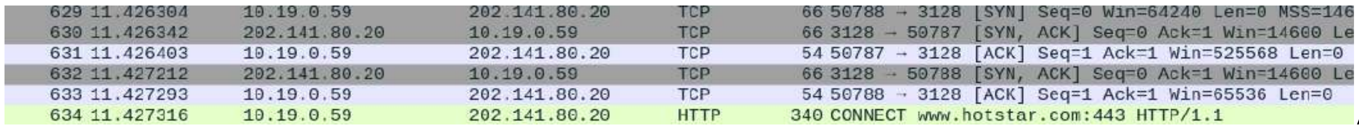
Messages exchanged by the application:

TCP connection Establishment ON OPENING WEBSITE

To establish a connection, TCP uses a three-way handshake. 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.

The three-way handshake: **SYN**: The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a some value A. **SYN-ACK**: In response, the server replies with a SYN-ACK

(acknowledging our SYN request). The ack number is set to one more than the received sequence number (A + 1), and the sequence number that the server chooses for the packet is another some number, B.
ACK: Finally, the client sends an ACK back to the server to acknowledge the SYN-ACK packet from server. The sequence number is set to the received ack value i.e. A + 1, and the ack number is set to one more than the received sequence number i.e. B + 1.
soon as TCP connection is established HTTP request is sent.

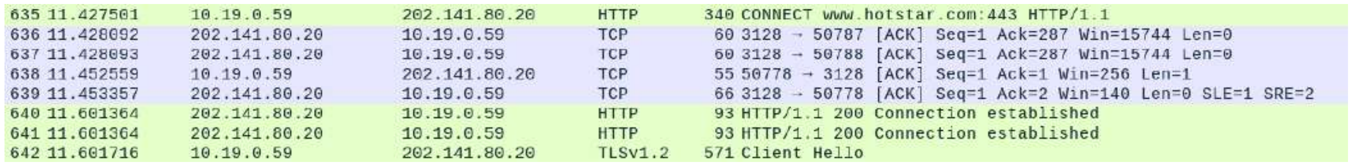


```
629 11.426304   10.19.0.59      202.141.80.20   TCP     66 50788 → 3128 [SYN] Seq=0 Win=64240 Len=0 MSS=146
630 11.426342   202.141.80.20   10.19.0.59      TCP     66 3128 → 50787 [SYN, ACK] Seq=0 Ack=1 Win=14600 Le
631 11.426403   10.19.0.59      202.141.80.20   TCP     54 50787 → 3128 [ACK] Seq=1 Ack=1 Win=525568 Len=0
632 11.427212   202.141.80.20   10.19.0.59      TCP     66 3128 → 50788 [SYN, ACK] Seq=0 Ack=1 Win=14600 Le
633 11.427293   10.19.0.59      202.141.80.20   TCP     54 50788 → 3128 [ACK] Seq=1 Ack=1 Win=65536 Len=0
634 11.427316   10.19.0.59      202.141.80.20   HTTP   340 CONNECT www.hotstar.com:443 HTTP/1.1
```

As
In above image there is two [ACK] and [SYN, ACK] it may be due to packet loss. When client sends its first ACK then it may be lost so after time out server again sent [SYN, ACK]

HTTP connection establishment once tcp connection has been built.

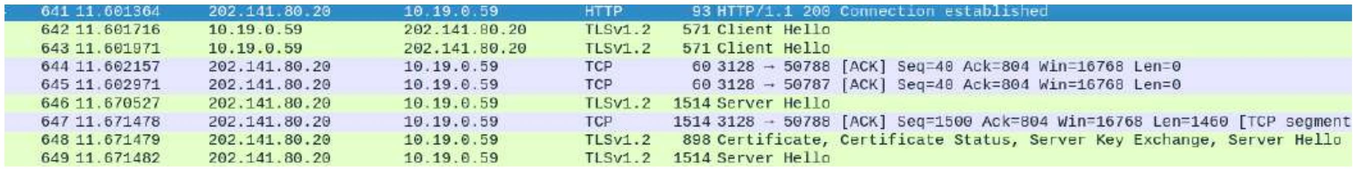
In client-server protocols, like HTTP, sessions consist of three phases:The client establishes a TCP connection (HTTP request is sent once TCP connection is established). The client sends its request, and waits for the acknowledgement from server. The server processes the request, sending back its acknowledgement, providing a status code and appropriate



```
635 11.427501   10.19.0.59      202.141.80.20   HTTP     340 CONNECT www.hotstar.com:443 HTTP/1.1
636 11.428092   202.141.80.20   10.19.0.59      TCP       60 3128 → 50787 [ACK] Seq=1 Ack=287 Win=15744 Len=0
637 11.428093   202.141.80.20   10.19.0.59      TCP       60 3128 → 50788 [ACK] Seq=1 Ack=287 Win=15744 Len=0
638 11.452559   10.19.0.59      202.141.80.20   TCP       55 50778 → 3128 [ACK] Seq=1 Ack=1 Win=256 Len=1
639 11.453357   202.141.80.20   10.19.0.59      TCP       66 3128 → 50778 [ACK] Seq=1 Ack=2 Win=140 Len=0 SLE=1 SRE=2
640 11.601364   202.141.80.20   10.19.0.59      HTTP      93 HTTP/1.1 200 Connection established
641 11.601364   202.141.80.20   10.19.0.59      HTTP      93 HTTP/1.1 200 Connection established
642 11.601716   10.19.0.59      202.141.80.20   TLSv1.2  571 Client Hello
```

Handshaking Sequences between source and destination: While opening website after HTTP connection

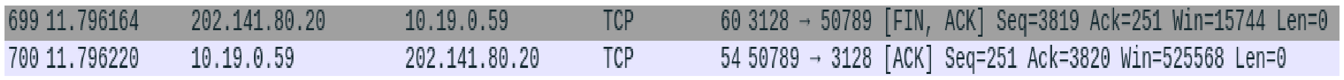
The TLS Handshake Protocol is responsible for the authentication and key exchange necessary to establish or resume secure sessions. When establishing a secure session, the Handshake Protocol manages the following:.
The client sends a "Client hello" message to the server, along with the client's random value and supported cipher suites. The server responds by sending a "Server hello" message to the client, along with the server's random value.The server sends its certificate to the client for authentication and may request a certificate from the client. The server sends the "Server hello done" message.If the server has requested a certificate from the client, the client sends it.The client creates a random Pre-Master Secret and encrypts it with the public key from the server's certificate, sending the encrypted Pre-Master Secret to the server.The server receives the Pre-Master Secret. The server and client each generate the Master Secret and session keys based
on the Pre-Master Secret.The client sends "Change cipher spec" notification to server to indicate that the client will start using the new session keys for hashing and encrypting messages. Client also sends "Client finished" message.Server receives "Change cipher spec" and switches its record layer security state to symmetric encryption using the session keys. Server sends "Server finished" message to the client.
Client and server can now exchange application data over the secured channel they have established. All messages sent from client to server and from server to client are encrypted using session key.



```
641 11.601364   202.141.80.20   10.19.0.59      HTTP       93 HTTP/1.1 200 Connection established
642 11.601716   10.19.0.59      202.141.80.20   TLSv1.2   571 Client Hello
643 11.601971   10.19.0.59      202.141.80.20   TLSv1.2   571 Client Hello
644 11.602157   202.141.80.20   10.19.0.59      TCP        60 3128 → 50788 [ACK] Seq=48 Ack=804 Win=16768 Len=0
645 11.602971   202.141.80.20   10.19.0.59      TCP        60 3128 → 50787 [ACK] Seq=48 Ack=804 Win=16768 Len=0
646 11.670527   202.141.80.20   10.19.0.59      TLSv1.2  1514 Server Hello
647 11.671478   202.141.80.20   10.19.0.59      TCP      1514 3128 → 50788 [ACK] Seq=1500 Ack=804 Win=16768 Len=1460 [TCP segment
648 11.671479   202.141.80.20   10.19.0.59      TLSv1.2   898 Certificate, Certificate Status, Server Key Exchange, Server Hello
649 11.671482   202.141.80.20   10.19.0.59      TLSv1.2  1514 Server Hello
```

When I paused the video

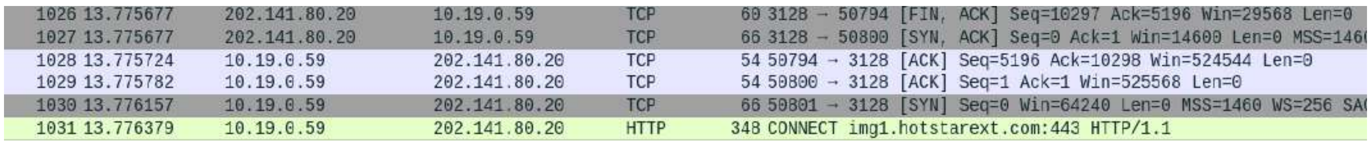
 When a playing live video stream was paused, the current ongoing TCP connection was closed and we received a FIN packet from server.



```
699 11.796164   202.141.80.20   10.19.0.59      TCP   60 3128 → 50789 [FIN, ACK] Seq=3819 Ack=251 Win=15744 Len=0
700 11.796220   10.19.0.59      202.141.80.20   TCP   54 50789 → 3128 [ACK] Seq=251 Ack=3820 Win=525568 Len=0
```

Resume a paused video

On playing the video again after pause, all the old tcp connection were terminated which can be seen from FIN packages received from the server. A new TCP connection (3-way connection) is made which is evident from sequence of SYN, SYN-ACK and ACK(in similar way as explained above).



```
1026 13.775677   202.141.80.20   10.19.0.59      TCP    60 3128 → 50794 [FIN, ACK] Seq=10297 Ack=5196 Win=29568 Len=0
1027 13.775677   202.141.80.20   10.19.0.59      TCP    66 3128 → 50800 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=146
1028 13.775724   10.19.0.59      202.141.80.20   TCP    54 50794 → 3128 [ACK] Seq=5196 Ack=10298 Win=524544 Len=0
1029 13.775782   10.19.0.59      202.141.80.20   TCP    54 50800 → 3128 [ACK] Seq=1 Ack=1 Win=525568 Len=0
1030 13.776157   10.19.0.59      202.141.80.20   TCP    66 50801 → 3128 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SA
1031 13.776379   10.19.0.59      202.141.80.20   HTTP  348 CONNECT img1.hotstarext.com:443 HTTP/1.1
```

Answer 4.

Different protocols used by the application work at different layers of the OSI model.They perform a specific task which is necessary for the proper functioning of the application. They are discussed below:

Hypertext Transfer Protocol (HTTP) - Using HTTP and HTML, clients can request different kinds of content (such as text, images, video, and application data) from web and application servers that host the content. It follows a request-response paradigm in which the client makes a request and the server issues a response that includes not only the requested content, but also relevant status information about the request. This self-contained design allows for the distributed nature of the Internet, where a request or response might pass through many intermediate routers and proxy servers. It also allows intermediary servers to perform value-added functions such as load balancing, caching, encryption, and compression.

SSL: hotstar uses new version TLSv1.2 of ssl. It is used to secure (Encrypt) the data to and from the site to clients.The security protocol protects the integrity of the website by helping to prevent intruders tampering with communications between the site and the visitors browsing (a common tactic here is injecting malware) as well as safeguarding privacy and security.Every unprotected HTTP request can potentially reveal information about the behaviors and identities. Thus Login and credentials details are compromised

TCP: this is the majorly used protocol in communication with hotstar.com. It is evident from the traces that TCP layer handles all the handshaking which is done for the establishment and graceful termination of the connection. The hotstar uses 3 way handshaking for establishment while 2 packets are used for termination. It is a connection-oriented protocol, which means a connection is established and maintained until the application-programs at each end have finished exchanging message. It determines how to break application data into packets that networks can deliver, sends packets to and accepts packets from the network layer, manages flow control, and—because it is meant to provide error-free data transmission—handles retransmission of dropped or garbled packets as well as acknowledgement of all packets that arrive.

Internet Protocol (IPv4) – The Internet protocol or the IP handles tasks in the network layer which incorporates all the routing procedures. It has the job of delivering the packets from the source to the destination solely on the basis of the IP addresses in the packet headers. IP also provides security to networks by encapsulating the data into an IP datagram. This makes sure it is received and interpreted by the intended recipient. IP provides the fundamental mechanism using which data is delivered between devices which may or may not be in the same network which is the most fundamental concept in today's internet.

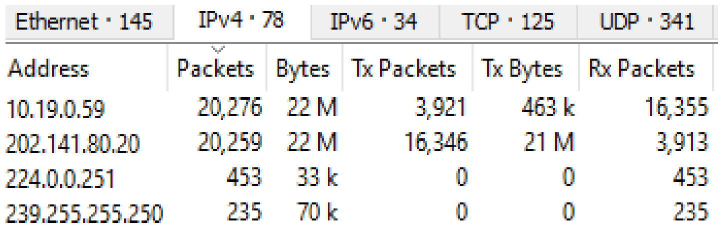
Ethernet – This protocol is the most common local area network technology and works at the Data Link layer. Details regarding the MAC address of the source and destination are found in Ethernet header. System communicating over Ethernet divide a stream of data into frames as was visible through Wireshark. The frame also contains error checking data so that damaged frames can be detected and discarded.

Answer 5

This data is collected using lan at different times of the day. I have taken this data from lohit hostel.
UDP packets are not used after connection established between source and hotstar server.
UDP packets are used by browser when browser search the dns of the given url. Once dns is found then there are not udp packets are used in my application. Below k = 1024.

time	Throughput (bytes/sec)	RTT (ms)	Avg. Packet Size (Bytes)	Number of packets Lost	UDP packets	Tcp Packets	Avg. Response W.r.t. 1 Request
7:30pm	228 k	4.59	1108	0	159	20259	4.17
5:00pm	32 k	5.197	539	0	100	6149	1.65
2:am	389 k	6.418	1289	0	1215	164059	6.5343

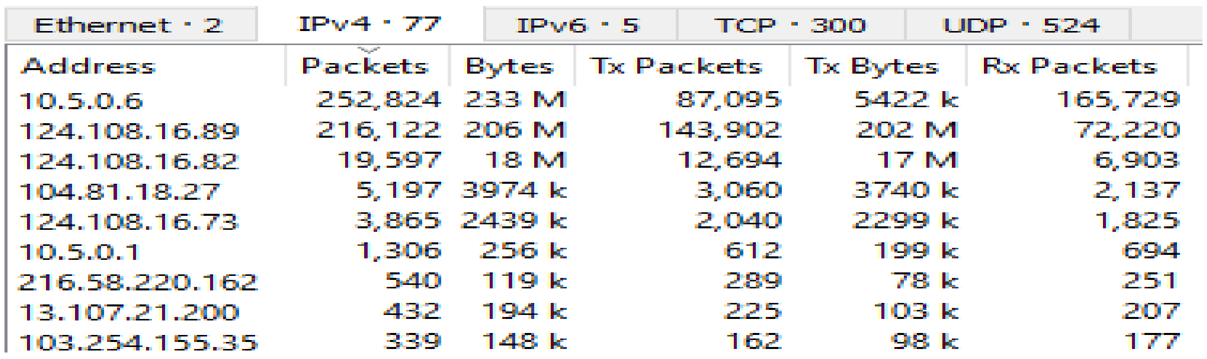
Ans 6. When I take data using proxy then most of the data came from one server which is 202.141.80.20. Above is the ip table using proxy



Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets
10.19.0.59	20,276	22 M	3,921	463 k	16,355
202.141.80.20	20,259	22 M	16,346	21 M	3,913
224.0.0.251	453	33 k	0	0	453
239.255.255.250	235	70 k	0	0	235

In left table source is 10.19.0.59 and destination is 202.141.80.20

Below is ip4 table, when i took the data using vpn. From below we know that a big percentage of data is coming from different different data



Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets
10.5.0.6	252,824	233 M	87,095	5422 k	165,729
124.108.16.89	216,122	206 M	143,902	202 M	72,220
124.108.16.82	19,597	18 M	12,694	17 M	6,903
104.81.18.27	5,197	3974 k	3,060	3740 k	2,137
124.108.16.73	3,865	2439 k	2,040	2299 k	1,825
10.5.0.1	1,306	256 k	612	199 k	694
216.58.220.162	540	119 k	289	78 k	251
13.107.21.200	432	194 k	225	103 k	207
103.254.155.35	339	148 k	162	98 k	177

Reason for data coming from multiple ip are followings:

1. Since modern day websites deploy load balancing on servers, data is sent to clients from various IP in most efficient manner.
2. Geographic location - Ideal scenario is for a server to be as close as possible to the customer or end user
3. It may be possible that video is being streamed from one IP and other site data is being loaded from other server with different IP. Since video and html data might be present on different server and hence different IP.
4. Geographic location - Ideal scenario is for a server to be as close as possible to the customer or end user
5. Maintenance backup
6. As a caution for unwanted fault in the network lines of few servers.