CS 349: Networks Lab
Assignment 1

Akul Agrawal
160101085

Find captured wireshark packets at: http://tiny.cc/sm482y

Q1.

1. Application Layer

a. HTTP

HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers:

•**General-header:** They have general applicability for both request and response messages.

•**Client Request-header:** These header fields have applicability only for request messages.

•**Server Response-header:** These header fields have applicability only for response messages.

•**Entity-header:** These header fields define meta information about the entity-body or, if no body is present, about the resource identified by the request.

HTTP/1.1 200 OK	**Status Line**	**HTTP Response**
Date: Thu, 20 May 2004 21:12:58 GMT Connection: close	**General Headers**	
Server: Apache/1.3.27 Accept-Ranges: bytes	**Response Headers**	
Content-Type: text/html Content-Length: 170 Last-Modified: Tue, 18 May 2004 10:14:49 GMT	**Entity Headers**	
<html> <head> <title>Welcome to the Amazing Site!</title> </head> <body> This site is under construction. Please come back later. Sorry! </body> </html>	**Message Body**	

HTTP messages consist of requests from client to server and responses from server to client.

HTTP-message = Request | Response ; HTTP/1.1 messages

Request and Response messages use the generic message format of RFC 822 for transferring entities (the payload of the message). Both types of message consist of a start-line, zero or more header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and possibly a message-body.

generic-message = start-line
*(message-header CRLF)
CRLF
[message-body]
start-line = Request-Line | Status-Line

2. SSL/TSL Layer

SSL Alert : This protocol is used to convey SSL-related alerts to the peer entity. It consists of two bytes the first of which takes the values 1 (warning) or 2 (fatal). The second byte contains a code that indicates the specific alert.

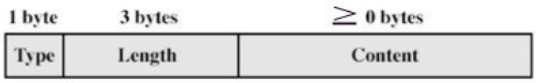
SSL Record: Content type (8 bits) - The higher layer protocol used to process the en-closed fragment.

Major Version (8 bits) - Indicates major version of SSL in use. For SSLv3, the value is 3.

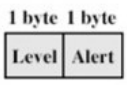
Minor Version (8 bits) - Indicates minor version in use. For SSLv3, the value is 0.

Compressed Length (16 bits) - The length in bytes of the compressed (or plaintext) fragment.

(a) Change Cipher Spec Protocol



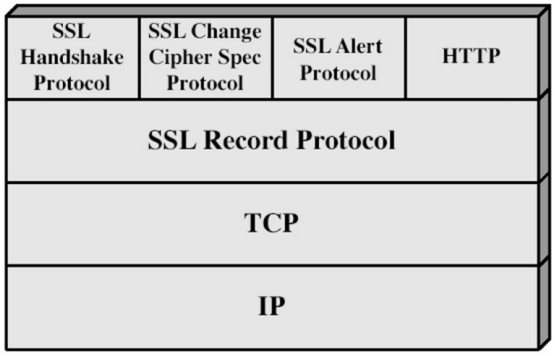
(c) Handshake Protocol

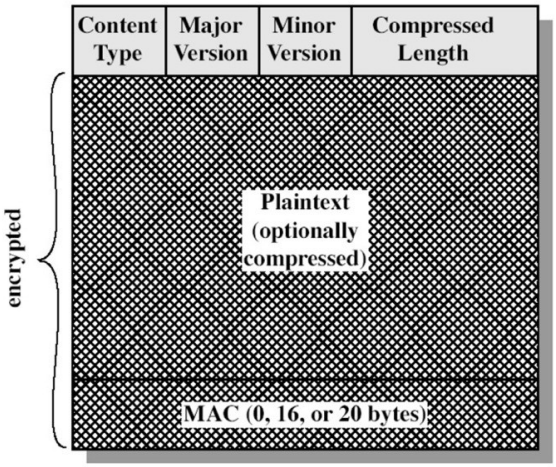


(b) Alert Protocol

(d) Other Upper-Layer Protocol (e.g., HTTP)

SSL/TSL Layer

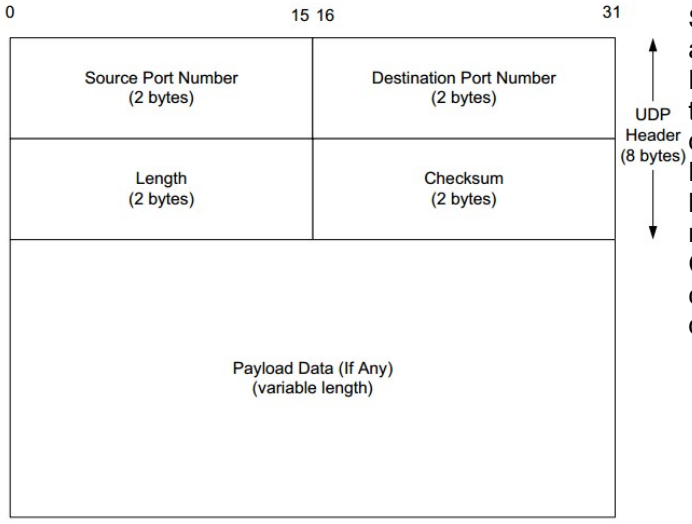




SSL Record Protocol

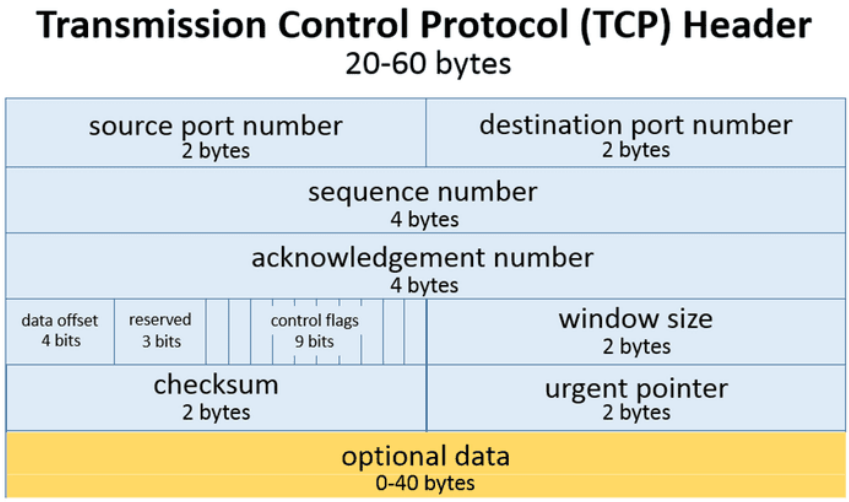
3. Transport Layer

a. UDP



Source Port Number : the port address of the application that is sending the data segment.
Destination Port Number : the port address of the application in the host that is receiving the data segment
Length : The length in bytes of the UDP header and the encapsulated data. The minimum value for this field is 8.
Checksum : checksum for error control. If the checksum is set to zero, then checksuming is disabled.

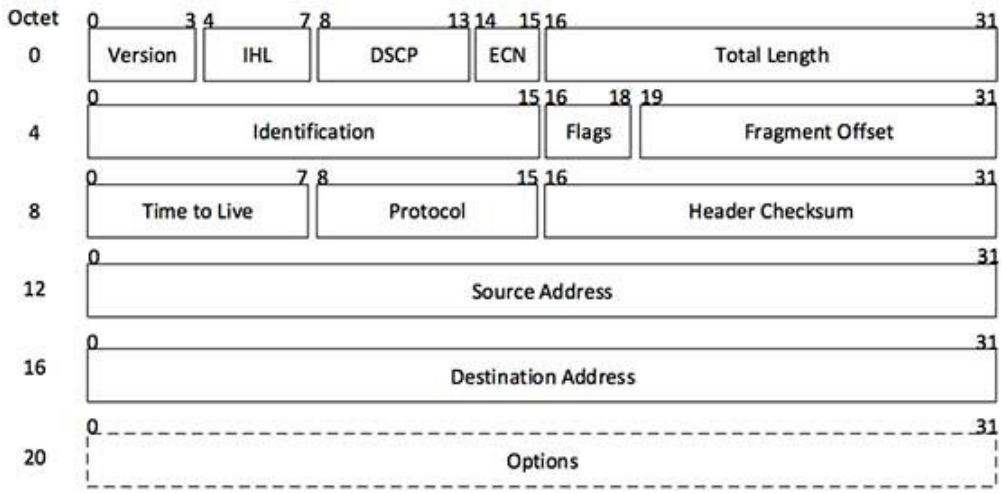
b. TCP



Source Port : port address of the application that is sending the data segment.
Destination Port : the port address of the application in the host that is receiving the data segment.
Sequence Number : the byte no. of rst byte that is sent in that particular segment. It is used to reassemble the message at the receiving end if the segments are received out of order.
Acknowledgement Number : byte number that receiver expects to receive next. It is an acknowledgment for the previous bytes being received successfully.
Data offset : indicates the length of the TCP header by number of 4-byte words in header
Control flags : 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc. Eg:
URG: Urgent pointer is valid
ACK: Acknowledgement number is valid(used in case of cumulative acknowledgement)
PSH: Request for push

RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
Window size : the window size of the sending TCP in bytes.
Checksum : checksum for error control. It is mandatory in TCP as opposed to UDP.
Urgent pointer : (valid only if the URG control ag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. Its value is added to the sequence number to get the byte number of the last urgent byte.

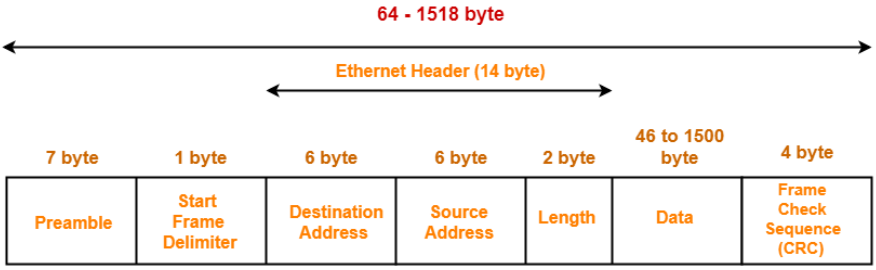
4. Network Layer
   a. IPv4



[Image: IP Header]

Internet Protocol being a layer-3 protocol (OSI) takes data Segments from layer-4 (Transport) and divides it into packets. IP packet encapsulates data unit received and add to its own header information.
Version: Version no. of Internet Protocol used (e.g. IPv4).
IHL: Internet Header Length; Length of entire IP header.
DSCP: Differentiated Services Code Point; this is Type of Service.
ECN: Explicit Congestion Notification; It carries information about the congestion seen in the route.
Total Length: Length of entire IP Packet (including IP header and IP Payload).
Identification: If IP packet is fragmented during the transmission, all the fragments contain same identification number. to identify original IP packet they belong to.
Flags: If IP Packet is too large to handle, these 'flags' tell if they can be fragmented or not.
Fragment Offset: This offset tells the exact position of the fragment in the original IP Packet.
Time to Live: To avoid looping in the network, every packet is sent with some TTL value set, which tells the network how many routers (hops) this packet can cross.
Protocol: Tells the Network layer at the destination host, to which Protocol this packet belongs to, i.e. the next level Protocol. Eg. protocol number of ICMP is 1, TCP is 6 and UDP is 17.
Header Checksum: checksum value of header which is used to check if the packet is received error-free.
Source Address: 32-bit address of the Sender (or source) of the packet.
Destination Address: 32-bit address of the Receiver (or destination) of the packet.
Options: This is optional field, which is used if the value of IHL is greater than 5. These options may contain values for options such as Security, Record Route, Time Stamp, etc.

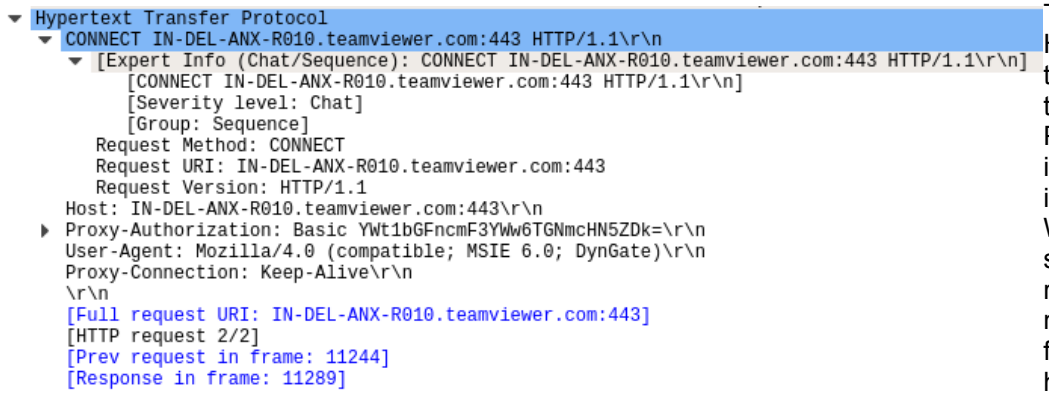
5. Link Layer
   a. Ethernet



IEEE 802.3 Ethernet Frame Format

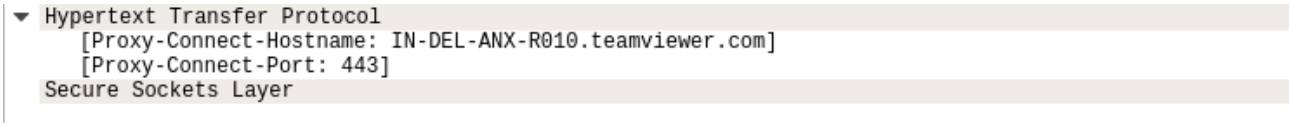
Q2.
The observed values of the protocols are mentioned in the same format as mentioned in Q1.

1. HTTP



The version of HTTP is 1.1. Since the connection is through IITG Proxy, Proxy Authorization is used. User agent is Mozilla by default. Wireshark also shows the previous request frame number and the frame number having response to this frame .

2. SSL

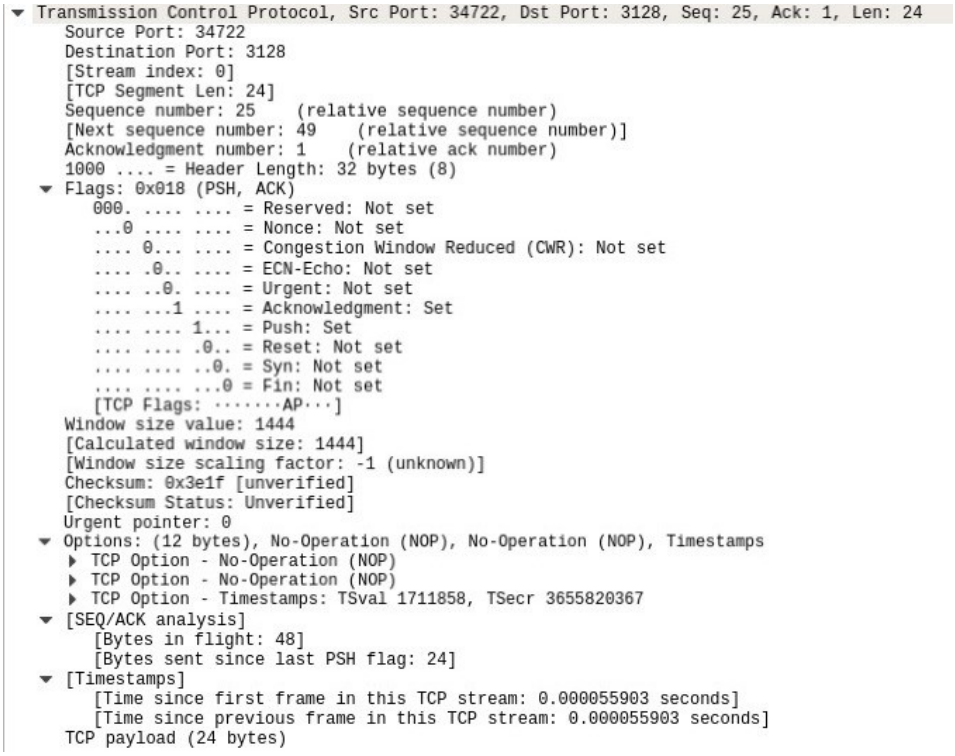


Because these layers (HTTP and SSL) are above the transport layer, wireshark could only identify them and could not capture the entire message.

3. UDP

source port: 52012 destination port: 34660
length: 1032 checksum: 0xb15a (45402 in decimal)
Payload data: (mentioned in second last line in figure) 3a6902003e290000...

4. TCP



source port number: 34722 destination port number: 3128
sequence number: 25
acknowledgement number: 1
data offset/header: 8 reserved: 000_2 control flags: 000011000_2 window size: 1444
checksum: 0x3e1f urgent pointer: 0
optional data: Options(12 bytes), TCP payload (24 bytes)

5. IPv4

```
▼ Internet Protocol Version 4, Src: 10.12.22.46, Dst: 202.141.80.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▼ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 76
    Identification: 0x366f (13935)
  ▼ Flags: 0x4000, Don't fragment
      0... .... .... .... = Reserved bit: Not set
      .1.. .... .... .... = Don't fragment: Set
      ..0. .... .... .... = More fragments: Not set
      ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xc961 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.12.22.46
    Destination: 202.141.80.20
```

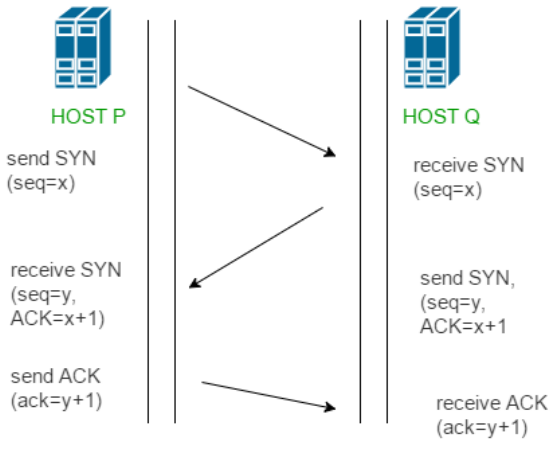
Version: 4 IHL: 5 DSCP: 0 ECN: 0 Total Length: 76
Identification: 0x366f (13935 in decimal) Flags: 010_2 (2 in decimal) Fragment Offset: 0
Time to Live: 64 Protocol: 6 Header Checksum: 0xc961 (51553 in decimal)
Source Address: 10.12.22.46
Destination Address: 202.141.80.20
Protocol 6 stands for TCP. In case of UDP, it's value is 17.

Q3

Teamviewer uses two different sets of protocols depending on the connection between two computers:
1. Over LAN
No data is sent through the teamviewer servers for remote access. Using additional features like chat, login etc is sent through HTTPS to the teamviewer servers as can be observed in SSL topic in Q2. UDP is used by teamviewer in this method to improve performance.
2. Over Internet(other sources than LAN)
Since the probability of losing a packet is high (as the data first goes from source to server, and then to the destination), TCP is used to ensure reliability. HTTPS is used for additional features as in case of LAN.
For additional features like chat, all the packets are routed through the internet, hence, for both the devices the packets are being sent to and received from the IITG proxy server(202.141.80.20). TCP connection setup handshake and the termination handshake are observed. After the TCP connection, an HTTP CONNECT packet is sent to the teamviewer server and the HTTP connection is established. The data is sent using SSL (application layer) protocol. Hence, it is encrypted. All the acknowledgements are not encrypted and hence wireshark displays their protocol as TCP. Finally on connection termination the TCP termination handshake occurs.
Handshakes :-

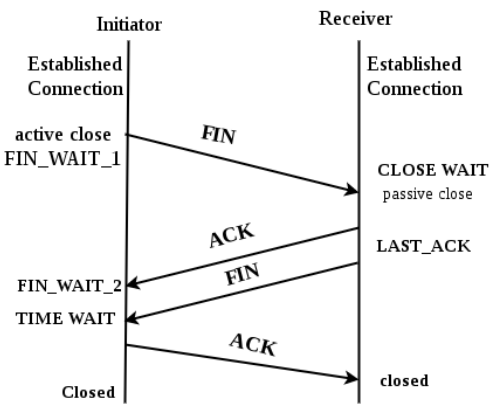
```
57 1.025651329   2402:3a80:9c1:ca3:5… 2a00:11c0:27:351:21… TCP      1454 41970 → 5938 [ACK] Seq=53353 Ack=74 Win=1432 Len=1368 TSval=8240896 TSecr=2081355129
58 1.025655599   2402:3a80:9c1:ca3:5… 2a00:11c0:27:351:21… TCP      1454 41970 → 5938 [ACK] Seq=54721 Ack=74 Win=1432 Len=1368 TSval=8240896 TSecr=2081355129
59 1.026189124   2402:3a80:9c1:ca3:5… 2a00:11c0:27:351:21… TCP      1454 41970 → 5938 [ACK] Seq=56089 Ack=74 Win=1432 Len=1368 TSval=8240896 TSecr=2081355129
60 1.229169378   2a00:11c0:27:351:21… 2402:3a80:9c1:ca3:5… TCP        86 5938 → 41970 [ACK] Seq=74 Ack=50617 Win=1026 Len=0 TSval=2081355310 TSecr=8240896
61 1.229232560   2402:3a80:9c1:ca3:5… 2a00:11c0:27:351:21… TCP       723 41970 → 5938 [PSH, ACK] Seq=57457 Ack=74 Win=1432 Len=637 TSval=8240947 TSecr=2081355310
```

TCP Connection Handshake (3-Way Handshake) :



Step 1 (SYN) : The client sends a SYN(Synchronize Sequence Number) segment which informs server that client is likely to start communication and with what sequence number it starts segments with.
Step 2 (SYN + ACK) : Server responds to the client request with SYN-ACK signal bits set. Acknowledgement(ACK) signifies the response of segment it received and SYN signifies the sequence number it is likely to start the segments with.
Step 3 (ACK) : The client acknowledges the response of server and they both establish a reliable connection.
The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter for the other direction and it is acknowledged. Thus, a full-duplex communication is established.

TCP Termination Handshake :



Step 1 (FIN From Client) : Suppose that the client application decides it wants to close the connection. (Note that the server could also choose to close the connection). This causes the client send a TCP segment with the FIN bit set to 1 to server and to enter the FIN_WAIT_1 state. While in the FIN_WAIT_1 state, the client waits for a TCP segment from the server with an acknowledgment (ACK).
Step 2 (ACK From Server) : When Server received FIN bit segment from Sender (Client), it sends acknowledgement (ACK) segment to the Sender (Client).
Step 3 (Client waiting) : While in the FIN_WAIT_1 state, the client waits for a TCP segment from the server with an acknowledgment. When it receives this segment, the client enters the FIN_WAIT_2 state. While in the FIN_WAIT_2 state, the client waits for another segment from the server with the FIN bit set to 1.
Step 4 (FIN from Server) : Server sends FIN bit segment to the Sender(Client) after some time when Server send the ACK segment (because of some closing process in the Server).
Step 5 (ACK from Client) : When Client receive FIN bit segment from the Server, the client acknowledges the server's segment and enters the TIME_WAIT state. The TIME_WAIT state lets the client resend the final acknowledgment in case the ACK is lost. After the wait, the connection formally closes.

Q4

The teamviewer application can be used through two connections:
1. LAN

If the two computers are connected by LAN, teamviewer provides a direct peer to peer connection between the two Pcs using UDP protocol. For live video sharing, UDP (User Data Protocol) is always recommended over TCP (Transport Control Protocol) as:

1. UDP offers reduced latency over the TCP reliability
2. In case of time sensitive applications, UDP is faster protocol as it doesn't wait for acknowledgement from the client side and retransmission of lost packet.

2. Internet

In this case, TCP protocol is used in this case to maintain reliability and due to the following reasons:

1.Adapt the best picture quality by transmitting every frame.
2.TCP streams can do encryption to prevent theft of videos due to guaranteed receipt of segments in correct order.
3.Due to self clocking mechanism, TCP is better for variable bandwidths that occur on the Internet.
4.TCP provides error recovery by retransmission of missing data.

While streaming the content, packets may be lost due to some reason. Suppose for a minute the teamviewer did not receive the packet. Now in case of TCP, the video streaming will pause till it receives the packet. On top of it, client has to send the right acknowledgment for each segment received. While in case of UDP, the client is not bothered for any acknowledgement. Hence, the transmission is fast which leads to less buffering and reduced video playout delays. UDP does not care for frame loss, what matters is the on-time delivery of the content. So, it results in the complete sync with live streaming.

Q5

Time	Throughput (packets per sec)	RTT (ms)	Packet size (Bytes)	No. of packets lost	No. of UDP and TCP packets	No. of responses recieved per request
10:45pm	145.7	0.02	120	0	2843	0.92
07:00pm	636.3	0.09	306	0	8554	0.18
12:00pm	440.8	0.02	586	0	12876	0.13

Q6

Teamviewer is a Peer-To-Peer application to access a desktop remotely. Hence, all the data is exchanged between two peers. There's no involvement of Teamviewer website/servers in this process. If the connection is established through LAN, the teamviewer application connects the devices directly through the local LAN path. Hence, the data is coming from only one IP and not multiple IP.
If the peers try to connect through Internet (without being directly connected by LAN), then the packets are sent through teamviewer servers. Since in IIT Guwahati, all the traffic goes through the proxy server 202.140.80.20, only the proxy IP address is captured in the packets.