CS349 Networks Lab – Assignment 2

Roopansh Bansal (150101053) - P2P Application(Teamvier)

CAPTURE DATA https://drive.google.com/open?id=1-XrI2hwuTrZM3n2gRi8jiqsH4NXT_sRT

1. PACKET FORMATS OF PROTOCOLS USED IN DIFFERENT LAYERS

A. Application Layer

- **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* have applicability for both request and response messages, *Client Request-header* have applicability only for request messages, *Server Response-header* have applicability only for response messages, *Entity-header* defines meta information about the entity-body. The header fields are - *Connection general-header* allows the sender to specify options that are desired for that particular connection and must not be communicated by proxies over further connections, *Date*, *Authorization request-header* field value consists of credentials containing the authentication information of the user agent, *Cookie request-header* field value contains a name/value pair of information stored for that URL, *From request-header* contains an Internet e-mail address for the human user who controls the requesting user agent, *Host request-header* is used to specify the Internet host and the port number of the resource being requested, *Proxy-Authorization request-header* allows the client to identify itself to a proxy which requires authentication, *User-Agent request-header* contains information about the user agent, *Location response-header* is used to redirect the recipient to a location other than the Request-URI for completion, *Server response-header* contains information about the software used by the origin server to handle the request.

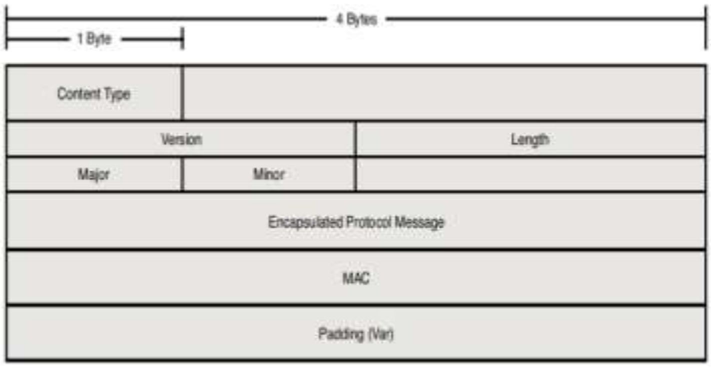


FIGURE 1 SSL PACKET HEADER

- **SSL/ SSLv2** - SSL provides security in the communication between two hosts. It provides integrity, authentication and confidentiality. It can be used with any protocol that uses TCP as the transport layer. 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-byte value.
- **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.

B. Transport layer

- **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.

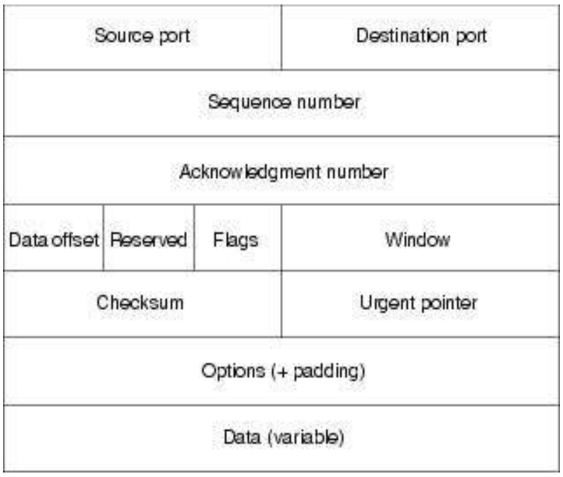
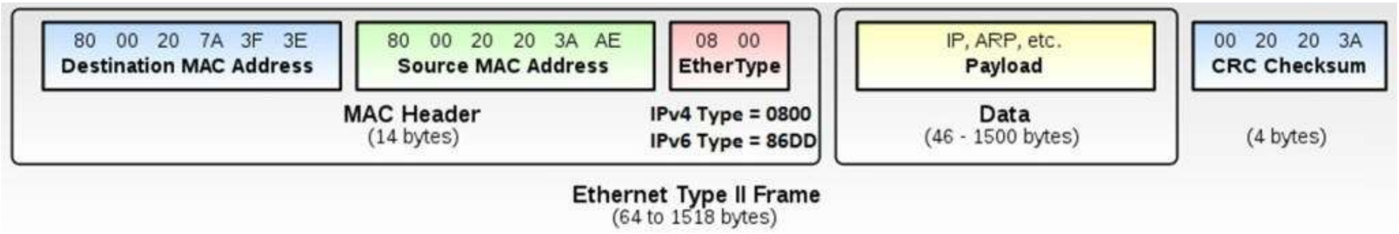


FIGURE 2 TCP PACKET HEADER



C. Network Layer

- **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

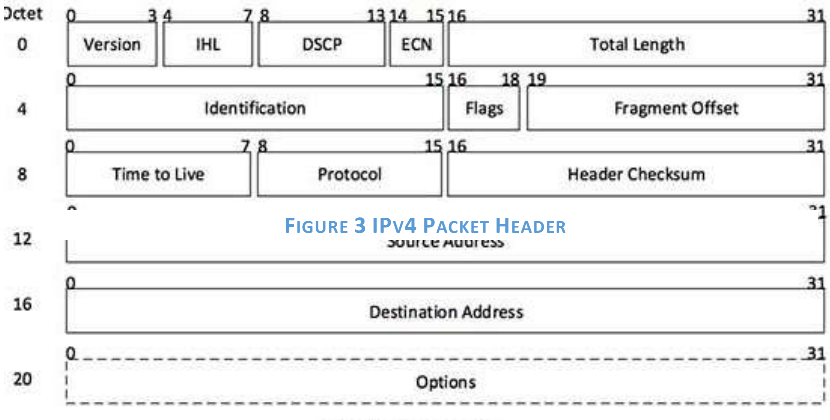


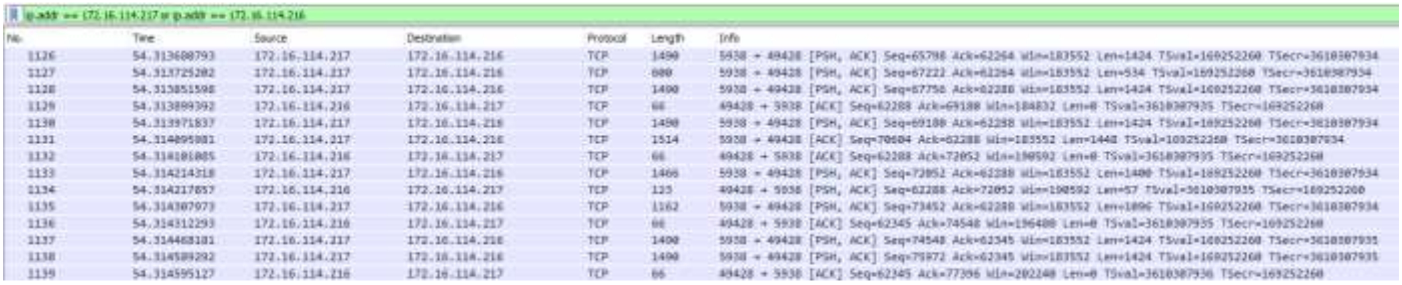
FIGURE 3 IPV4 PACKET HEADER

D. Link Layer

- **Ethernet(II)** is the most common local area networking technology. It has *Preamble* which is 56 bits of alternating 1's and 0's, *Destination MAC Address*, *Source MAC Address*, *Type* that identifies an upper layer protocol encapsulated by the frame data, *Length* of frame and *Frame Checksum* for error detection.

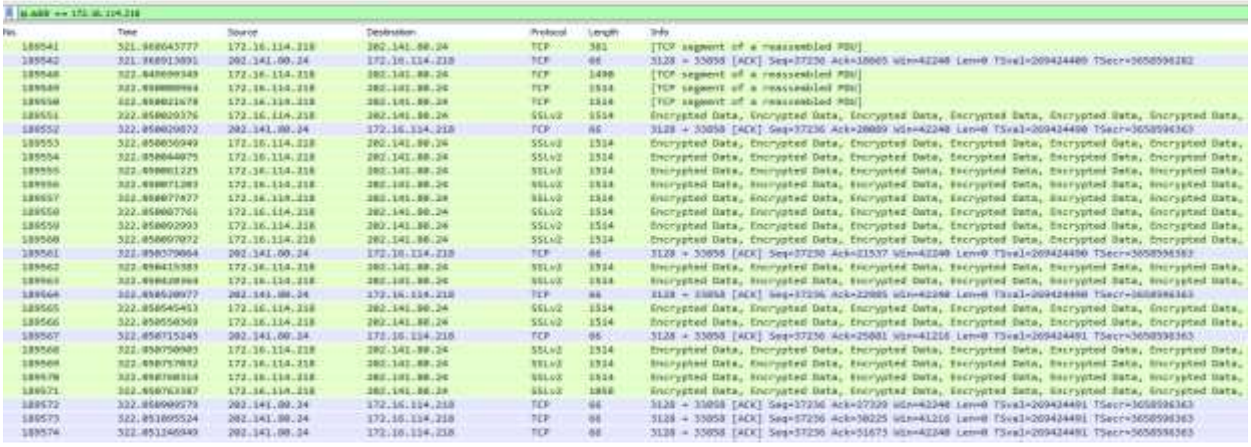
2. OBSERVED VALUES

When using Remote desktop(Teamviewer) on LAN, the following stream of packets were observed :-



```
ip.addr == 172.16.114.217 or ip.addr == 172.16.114.216
No.    Time           Source           Destination      Protocol  Length  Info
1126   54.313608793   172.16.114.217   172.16.114.216   TCP       1490    5938 → 49428 [PSH, ACK] Seq=65798 Ack=62264 Win=183552 Len=1424 TSval=169252260 TSecr=3610307934
1127   54.313725282   172.16.114.217   172.16.114.216   TCP       600     5938 → 49428 [PSH, ACK] Seq=67222 Ack=62264 Win=183552 Len=534 TSval=169252260 TSecr=3610307934
1128   54.313851598   172.16.114.217   172.16.114.216   TCP       1490    5938 → 49428 [PSH, ACK] Seq=67756 Ack=62288 Win=183552 Len=1424 TSval=169252260 TSecr=3610307934
1129   54.313899392   172.16.114.216   172.16.114.217   TCP       66      49428 → 5938 [ACK] Seq=62288 Ack=69180 Win=184832 Len=0 TSval=3610307935 TSecr=169252260
1130   54.313971837   172.16.114.217   172.16.114.216   TCP       1490    5938 → 49428 [PSH, ACK] Seq=69180 Ack=62288 Win=183552 Len=1424 TSval=169252260 TSecr=3610307934
1131   54.314095981   172.16.114.217   172.16.114.216   TCP       1514    5938 → 49428 [ACK] Seq=70604 Ack=62288 Win=183552 Len=1448 TSval=169252260 TSecr=3610307934
1132   54.314101085   172.16.114.216   172.16.114.217   TCP       66      49428 → 5938 [ACK] Seq=62288 Ack=72052 Win=190592 Len=0 TSval=3610307935 TSecr=169252260
1133   54.314214318   172.16.114.217   172.16.114.216   TCP       1466    5938 → 49428 [PSH, ACK] Seq=72052 Ack=62288 Win=183552 Len=1400 TSval=169252260 TSecr=3610307934
1134   54.314217857   172.16.114.216   172.16.114.217   TCP       123     49428 → 5938 [PSH, ACK] Seq=62288 Ack=72052 Win=190592 Len=57 TSval=3610307935 TSecr=169252260
1135   54.314307973   172.16.114.217   172.16.114.216   TCP       1162    5938 → 49428 [PSH, ACK] Seq=73452 Ack=62288 Win=183552 Len=1096 TSval=169252260 TSecr=3610307934
1136   54.314312293   172.16.114.216   172.16.114.217   TCP       66      49428 → 5938 [ACK] Seq=62345 Ack=74548 Win=196480 Len=0 TSval=3610307935 TSecr=169252260
1137   54.314468181   172.16.114.217   172.16.114.216   TCP       1490    5938 → 49428 [PSH, ACK] Seq=74548 Ack=62345 Win=183552 Len=1424 TSval=169252260 TSecr=3610307935
1138   54.314589292   172.16.114.217   172.16.114.216   TCP       1490    5938 → 49428 [PSH, ACK] Seq=75972 Ack=62345 Win=183552 Len=1424 TSval=169252260 TSecr=3610307935
1139   54.314595127   172.16.114.216   172.16.114.217   TCP       66      49428 → 5938 [ACK] Seq=62345 Ack=77396 Win=202240 Len=0 TSval=3610307936 TSecr=169252260
```

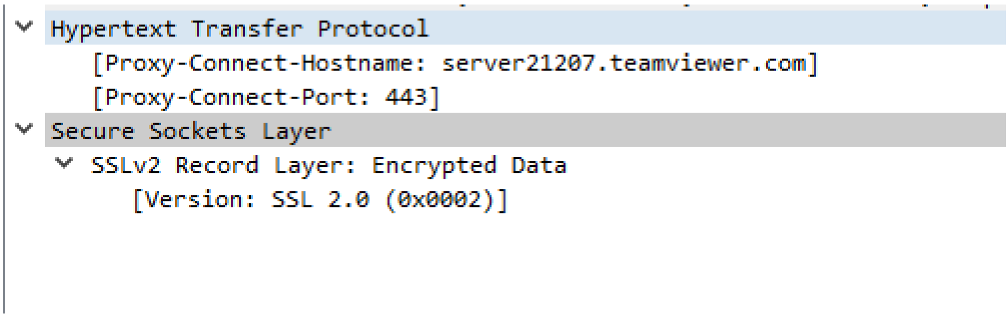
When using Remote desktop(Teamviewer) through internet, the following stream of packets were observed :-



```
ip.addr == 172.16.114.218
No.      Time            Source           Destination      Protocol  Length  Info
189541   321.988643777   172.16.114.218   202.141.80.24    TCP       381     [TCP segment of a reassembled PDU]
189542   321.988913091   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37230 Ack=18005 Win=42240 Len=0 TSval=209424489 TSecr=3658596282
189548   322.049699349   172.16.114.218   202.141.80.24    TCP       1490    [TCP segment of a reassembled PDU]
189549   322.050000964   172.16.114.218   202.141.80.24    TCP       1514    [TCP segment of a reassembled PDU]
189550   322.050021678   172.16.114.218   202.141.80.24    TCP       1514    [TCP segment of a reassembled PDU]
189551   322.050029376   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189552   322.050029872   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37236 Ack=20089 Win=42240 Len=0 TSval=209424490 TSecr=3658596363
189553   322.050036949   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189554   322.050044075   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189555   322.050061225   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189556   322.050071283   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189557   322.050077477   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189558   322.050087761   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189559   322.050092993   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189560   322.050097072   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189561   322.050379064   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37230 Ack=21537 Win=42240 Len=0 TSval=209424490 TSecr=3658596363
189562   322.050415383   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189563   322.050420964   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189564   322.050520977   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37236 Ack=22985 Win=42240 Len=0 TSval=209424490 TSecr=3658596363
189565   322.050545453   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189566   322.050550369   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189567   322.050715245   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37236 Ack=25081 Win=41216 Len=0 TSval=209424491 TSecr=3658596363
189568   322.050750905   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189569   322.050757032   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189570   322.050760314   172.16.114.218   202.141.80.24    SSLv2     1514    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189571   322.050763387   172.16.114.218   202.141.80.24    SSLv2     1058    Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data,
189572   322.050909579   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37236 Ack=27329 Win=42240 Len=0 TSval=209424491 TSecr=3658596363
189573   322.051095524   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37236 Ack=30225 Win=41216 Len=0 TSval=209424491 TSecr=3658596363
189574   322.051246949   202.141.80.24    172.16.114.218   TCP       66      3128 → 33058 [ACK] Seq=37236 Ack=31673 Win=42240 Len=0 TSval=209424491 TSecr=3658596363
```

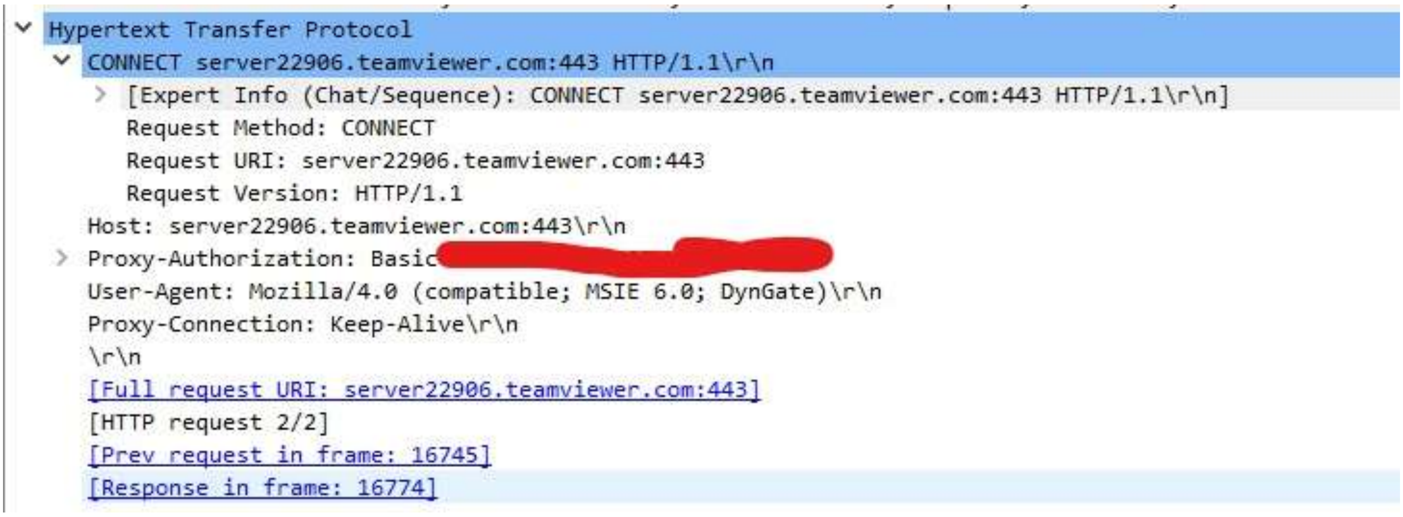
In the both the above, *time* is the time elapsed since the starting of packets capture, *source & destination* are the sender & the reciever of the packets respectively, *protocol* is the protocol(of the highest layer) that the wireshark could identify, *length* is the length of the packet and *info* is a brief information contained in the packet decoded by wireshark.

- **SSLv2 –** 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 SSLv2 are in application layer). WireShark is unable to determine the further headers of the SSL packet. It just identifies the version – 2 as seen in the following figure.



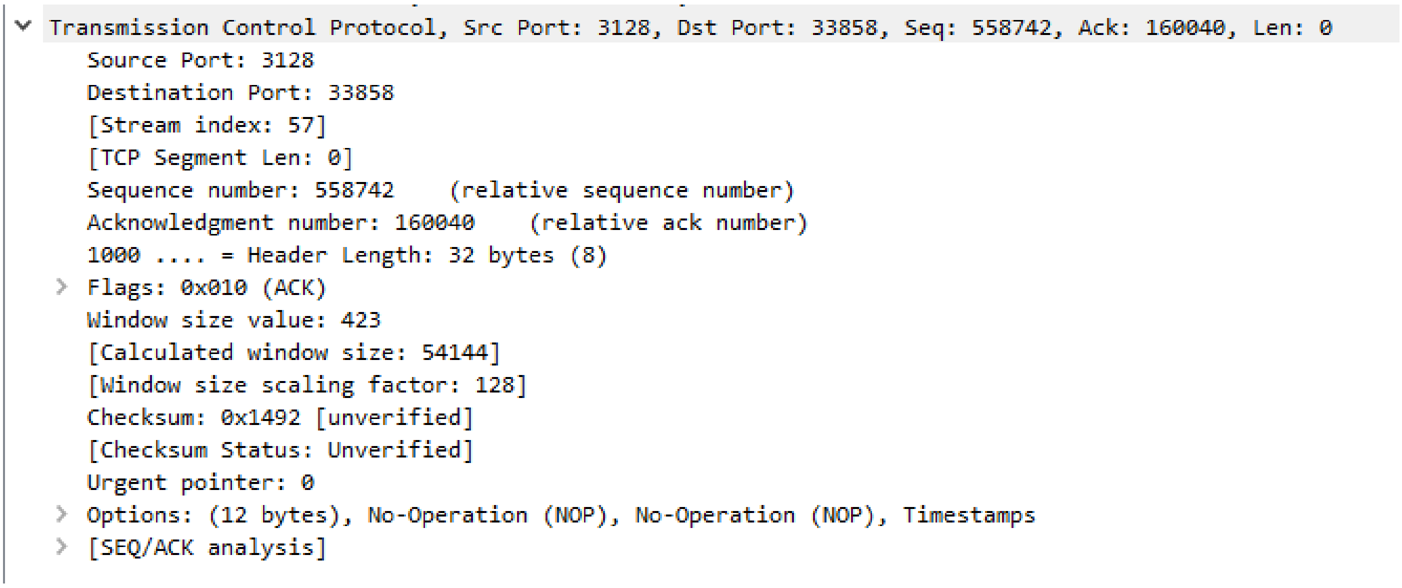
```
v Hypertext Transfer Protocol
    [Proxy-Connect-Hostname: server21207.teamviewer.com]
    [Proxy-Connect-Port: 443]
v Secure Sockets Layer
  v SSLv2 Record Layer: Encrypted Data
      [Version: SSL 2.0 (0x0002)]
```

- **HTTP –** As seen in the fighure, the Request method is connect to the request URI, which is the address of the teamviewer 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 by default in Wireshark. Wireshark also shows the previous request frame number and the frame number having response to this frame .



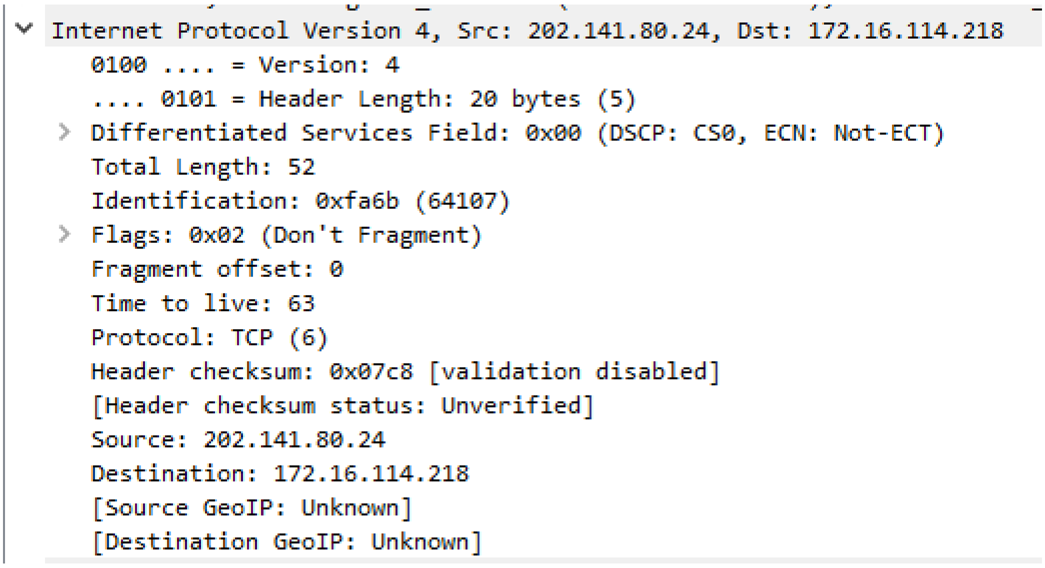
```
v Hypertext Transfer Protocol
  v CONNECT server22906.teamviewer.com:443 HTTP/1.1\r\n
    > [Expert Info (Chat/Sequence): CONNECT server22906.teamviewer.com:443 HTTP/1.1\r\n]
      Request Method: CONNECT
      Request URI: server22906.teamviewer.com:443
      Request Version: HTTP/1.1
    Host: server22906.teamviewer.com:443\r\n
  > Proxy-Authorization: Basic[redacted]
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; DynGate)\r\n
    Proxy-Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: server22906.teamviewer.com:443]
    [HTTP request 2/2]
    [Prev request in frame: 16745]
    [Response in frame: 16774]
```

- **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.



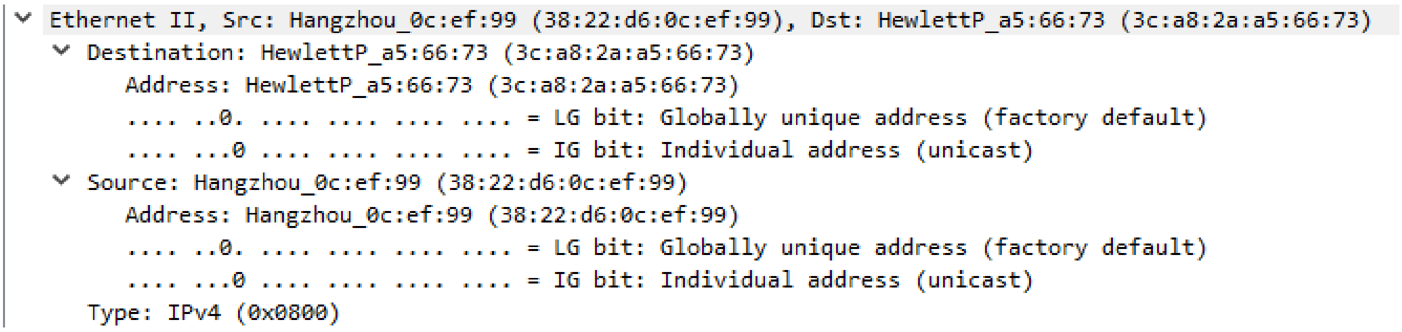
```
v Transmission Control Protocol, Src Port: 3128, Dst Port: 33858, Seq: 558742, Ack: 160040, Len: 0
    Source Port: 3128
    Destination Port: 33858
    [Stream index: 57]
    [TCP Segment Len: 0]
    Sequence number: 558742    (relative sequence number)
    Acknowledgment number: 160040    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x010 (ACK)
    Window size value: 423
    [Calculated window size: 54144]
    [Window size scaling factor: 128]
    Checksum: 0x1492 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [SEQ/ACK analysis]
```

- **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(202.141.80.24) to my device(172.16.114.218).



```
v Internet Protocol Version 4, Src: 202.141.80.24, Dst: 172.16.114.218
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xfa6b (64107)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 63
    Protocol: TCP (6)
    Header checksum: 0x07c8 [validation disabled]
    [Header checksum status: Unverified]
    Source: 202.141.80.24
    Destination: 172.16.114.218
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
```

- **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 Adress and not a Local Address.



```
v Ethernet II, Src: Hangzhou_0c:ef:99 (38:22:d6:0c:ef:99), Dst: HewlettP_a5:66:73 (3c:a8:2a:a5:66:73)
  v Destination: HewlettP_a5:66:73 (3c:a8:2a:a5:66:73)
      Address: HewlettP_a5:66:73 (3c:a8:2a:a5:66:73)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: Hangzhou_0c:ef:99 (38:22:d6:0c:ef:99)
      Address: Hangzhou_0c:ef:99 (38:22:d6:0c:ef:99)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

3. SEQUENCE OF MESSAGES

The application was tested under different conditions and the behaviour was observed in each. TeamViewer checks network connections to see what protocol it can efficiently deploy to make the communication between two peer computers efficient. If both the computers sit in the same LAN, there is no need to route data thorough HTTPS/SSL ports. But if the data is being sent through Teamviewer server over the internet, then the data is encrypted and sent through HTTPS/SSL ports so that not even the people at teamviewer can decipher and look at data.

The behaviour observed was similar when the two devices were connected directly through LAN and when connected through a switch. In both these scenarios, only TCP/IPv4 protocol packets were observed. The TCP three-way handshake (SYN SYN-ACK ACK) is the method used by TCP to set up a TCP/IP connection between the devices. After the connection setup, the devices exchange sequences of data and acknowledgements. The wireshark couldn't determine the application layer protocol and hence it states it as 'Data' protocol. During termination of the connection, another handshake is used(FIN, ACK, FIN, ACK). In both these cases, the source and destination of the packets either of the source or destination addresses and not any intermediate address.

The behaviour was a little different when the two devices were connected through internet using teamviewer. Primarily, all the packets were being routed through the internet, so for both the devices the packets were being sent to and received from the proxy server(202.141.80.24). Here also, the TCP connection setup handshake and the termination handshake were observed. After the TCP connection, an HTTP CONNECT packet is sent to the teamviewer server and the HTTP connection is established. Unlike the previous scenario, here 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 only. Finally on connection termination the TCP termination handshake occurs. Note that in this case, teamviewer.com does handshaking with regular TCP packets. But transmits data with SSL(encrypted) which is an application protocol which sits over TCP. SSL transmission is interlaced with plain TCP transmission with same source/destination IP pairs connected to the same ports. This indicates that the data is being transferred with SSL and handshake is happening with TCP.

Handshakes :-

- TCP CONNECTION HANDSHAKE : To establish a connection, each device must send a SYN and receive an ACK for it from the other device. Thus, conceptually, we need to have four control messages pass between the devices. However, it's inefficient to send a SYN and an ACK in separate messages when one could communicate both simultaneously. Thus, in the normal sequence of events in connection establishment, one of the SYNs and one of the ACKs is sent together by setting both of the relevant bits (a message sometimes called a SYN+ACK). This makes a total of three messages, and for this reason the connection procedure is called a three-way handshake.

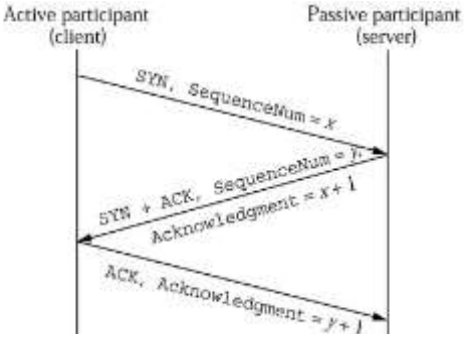

Active participant (client)
Passive participant (server)
SYN, SequenceNum = x
SYN + ACK, SequenceNum = y,
Acknowledgment = x + 1
ACK, Acknowledgment = y + 1

FIGURE 4 TCP CONNECTION HANDSHAKE

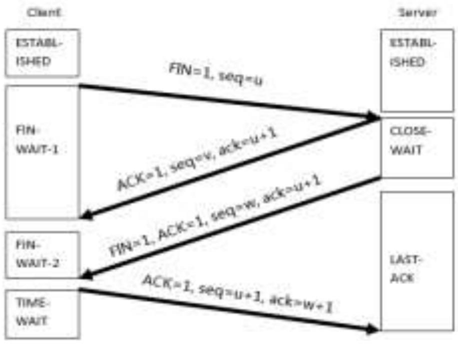


FIGURE 5 TCP TERMINATION HANDSHAKE

- TCP TERMINATION HANDSHAKE : In the normal case, each side terminates its end of the connection by sending a special message with the FIN (finish) bit set. This message serves as a connection termination request to the other device. The device receiving the FIN responds with an acknowledgment to the FIN to indicate that it was received. The connection as a whole is not considered terminated until both sides have finished the shut down procedure by sending a FIN and receiving an ACK. Thus, termination isn't a three-way handshake like establishment: it is a pair of two-way handshakes. The states that the two devices in the connection move through during a normal connection shutdown are different because the device initiating the shutdown must behave differently than the one that receives the termination request. In particular, the TCP on the device receiving the initial termination request must inform its application process and wait for a signal that the process is ready to proceed. The initiating device doesn't need to do this, since the application is what started the ball rolling in the first place.

4. FUNCTIONING OF APPLICATION

The teamviewer application has two modes of connection – Direct LAN and Through Internet as stated earlier. When establishing a session, TeamViewer determines the optimal type of connection. After the handshake through their master servers or directly through LAN, a direct connection via TCP is established (even behind Proxy). All application protocols observed and explained in Question 1 sits above TCP layer. For handshaking TCP would be the best choice because TCP guarantees packet ordering and packet delivery. There are three phases- connection, data transfer and termination between the pairs. During the connection and termination phase, TCP handshake occurs as explained previously. Once connection is established for data transmission TCP is used. According to the teamviewer's site, once connection is established, the data exchange may use TCP or UDP depending on the requirements and conditions. But, I observed only TCP packets. I believe that this is so because IITG-Proxy server blocks UDP packets.

Since when connecting through internet, we want to avoid hackers snooping packets on wire, teamviewer goes for something like SSL. Application's requirement determines the selection of protocol. It is used so that even the people at the teamviewer's server are unable to view the data. Hence, SSL is used so that encrypted data is transferred.

5. STATISTICAL ANALYSIS

Time	Throughput (Packets/Sec)	RTT (ms)	Avg. Packet Size (Bytes)	No. of Packets Lost	UDP Packets	TCP Packets	Avg Response w.r.t. 1 Request
9:00am	45	0.267	493.5	0	0	1941	0.69
12:30am	4.1	0.257	487.5	0	0	1811	0.75
3:00pm	12.8	7.67	485.5	0	0	1615	0.83

6. CONTENT PROVIDERS

Teamviewer is a Peer-To-Peer Remote Desktop application. Hence, all the data is exchanged between two peers. TeamViewer site doesn't provide any content. As stated in earlier parts, if trying to connect through LAN, teamviewer application connects the devices directly through local LAN path. Hence, data is coming from only one IP and not multiple IP.

If the peers try to connect through Internet, then the packets are sent through teamviewer servers, it routes the peer-peer data using standard HTTPS and SSL ports. In our case, all the traffic goes through IIT-Guwahati proxy server 202.140.80.24. Proxy server does NAT (Network Address Translation) because of which we only see proxy IP address in the packets we capture.