CS 349 Assignment 2

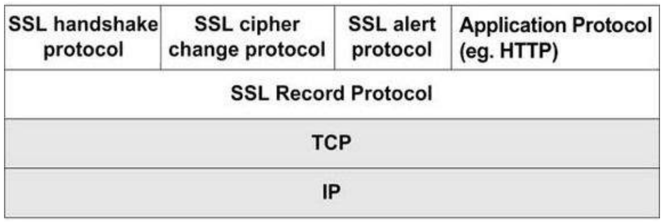
Chirag Gupta (170101019)

Drive Link for Trace Files: https://drive.google.com/open?id=1XYjbvyx5N_p27lxXS4hGfOeC-1Xilo2G

1) Packet formats of Protocols used in different layers

A. Application Layer

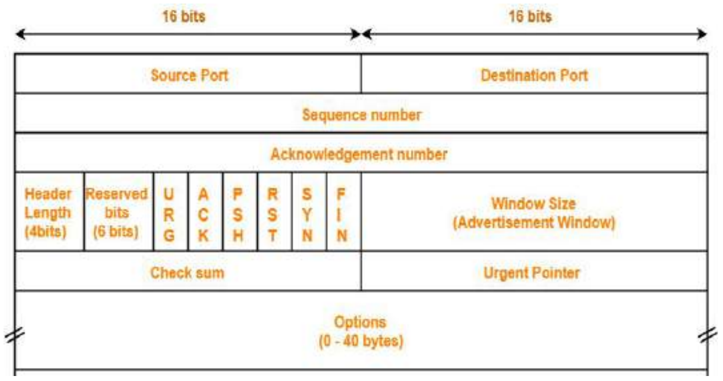
- **SSL/SSLv2**: SSL provides security in the communication between two hosts. It provides integrity, authentication and confidentiality. It can be used with any protocol that uses TCP as the transport layer. The basic unit of data in SSL is a record. Each record consists of a five-byte record header, 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.



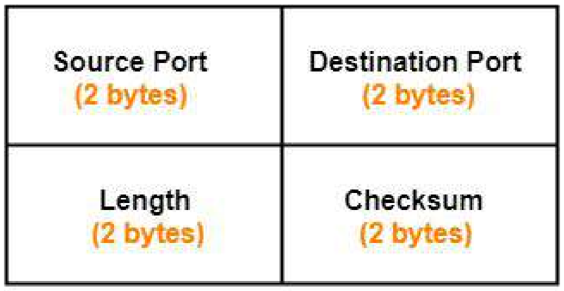
- **Data** - When Wireshark can't determine how part of a packet should be formatted, it marks that chunk as "Data". The "Data" is just a normal data payload.

B. Transport Layer

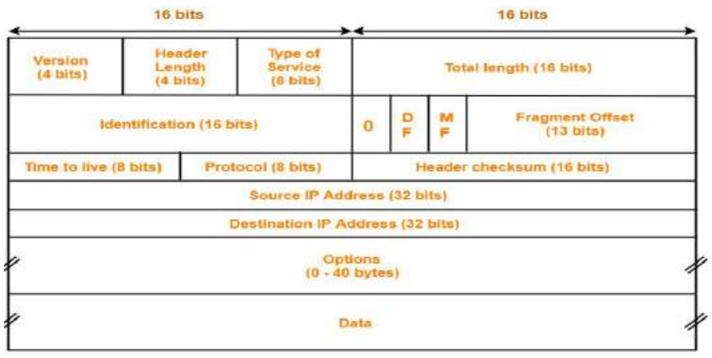
- **TCP(Transmission Control Protocol)**: Each TCP header has ten required fields totalling 20 bytes in size. They can also optionally include an additional data section up to 40 bytes in size. TCP headers have:
  - ❖ *Source Port:* 16-bit field and identifies the port of the sending application.
  - ❖ *Destination Port*: It identifies the port of the receiving application.
  - ❖ *Sequence Number:* It is used to mark an order in a group of messages and is assigned to each byte of data.
  - ❖ *Ack Number*: It contains the sequence number of the data byte that the receiver expects to receive next.
  - ❖ *Header Length*: It is a 4-bit field and contains the length of the TCP header.
  - ❖ *Flags:* There is a total of 6 types of Flags of 1 bit each. Some of them are Ack, Psh and Syn.
  - ❖ *Window Size:* It contains the size of the receiving window of the sender. It advertises how much data (in bytes) the sender can receive without acknowledgement.
  - ❖ *Checksum:* It verifies the integrity of data in the TCP payload.
  - ❖ *Urgent Pointer*: It indicates how much data in the current segment counting from the first data byte is urgent.



- **UDP(User Datagram Protocol)**: It is the simplest transport layer protocol. It simply takes the datagram from the network layer, attaches its header and sends it to the user. The header contains only 4 fields:
  - ❖ *Source Port*: It is a 16-bit field and identifies the port of the sender application.
  - ❖ *Destination Port*: It identifies the port of receiver application.
  - ❖ *Length*: It identifies the combined length of UDP Header and Encapsulated data.
  - ❖ *Checksum:* It is calculated on UDP Header, encapsulated data and IP pseudo-header and used for error control.



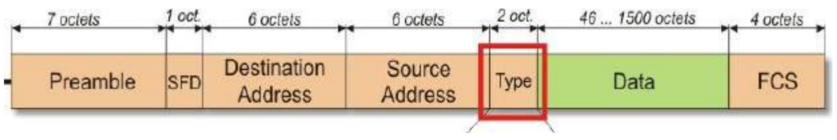
C. Network Layer



- **IPv4** is one of the core protocols of standards-based internetworking methods on the Internet.IP is responsible to deliver data packets from the source host to the destination host.Pv4 is a connectionless protocol for use on packet-switched networks. IP headers contain:
  - ❖ *Version*: Indicates the IP version used.
  - ❖ *Header Length*: Contains the length of the IP header.
  - ❖ *Types of Services:* Used for Quality of Service(QoS).
  - ❖ *Total Length:* It is a 16-bit field that contains the total length of the datagram (in bytes).
  - ❖ *Identification*: It is used for the identification of the fragments of an original IP datagram.
  - ❖ *DF/MF bits*: DF bit stands for Do Not Fragment and MF stands for More Fragment bits.
  - ❖ *Time to Live*: It indicates the maximum number of hops a datagram can take to reach the destination.
  - ❖ *Protocol*: It tells the network layer at the destination host to which protocol the IP datagram belongs.
  - ❖ *Source/Destination IP address*: It contains the logical address of sender and receiver of the datagram.

D. Link Layer

- **Ethernet(II)** is the most common LAN technology. Header contains:



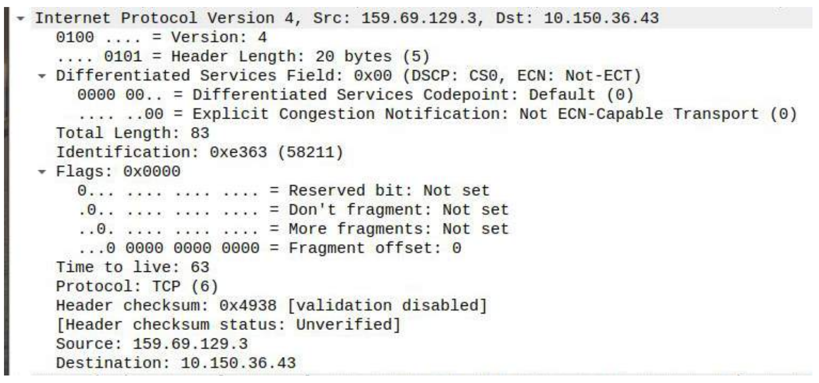
  - ❖ *Preamble/SFD*: This indicates the starting of the frame and allows the sender and receiver to establish bit synchronization.
  - ❖ *Destination/Source address:* Both are 6-byte field and contains the MAC address of the receiver/sender machine.
  - ❖ *Length:* It indicates the length of the entire Ethernet frame.
  - ❖ *Data*: This is the place where actual data is stored and both IP header and data is stored here in general.
  - ❖ *CRC*: This is used to detect any in-transit corruption of data.

2) Observed fields for the Protocols used

- The protocols being used are Ethernet, IPv4, TCP, and SSL.
- PC was connected to the institute LAN at 6 PM.

A. IPv4

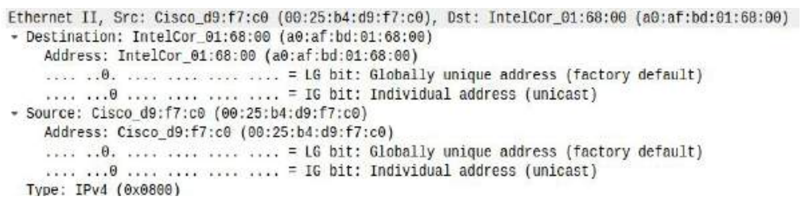
The *version* of IP is 4(IPv4). The *header length* is 20 bytes and as each word is 4 bytes, hence total 5 words length of the header. CS0 implies that the *service type* is the best effort and Not-ECT implies that the packet is not using ECN Transport. Total *Length* of the packet is 83 bytes. *The fragment offset* is 0 and both DF and MF are set to 0. The value of *Time to live(TTL)* is 63 hops. *Header checksum* is used to detect any error. *Source and destination* contain the IP address of sender and receiver machine.



```
▾ Internet Protocol Version 4, Src: 159.69.129.3, Dst: 10.150.36.43
    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: 83
    Identification: 0xe363 (58211)
  ▾ Flags: 0x0000
      0... .... .... .... = Reserved bit: Not set
      .0.. .... .... .... = Don't fragment: Not set
      ..0. .... .... .... = More fragments: Not set
      ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 63
    Protocol: TCP (6)
    Header checksum: 0x4938 [validation disabled]
    [Header checksum status: Unverified]
    Source: 159.69.129.3
    Destination: 10.150.36.43
```

B. Ethernet

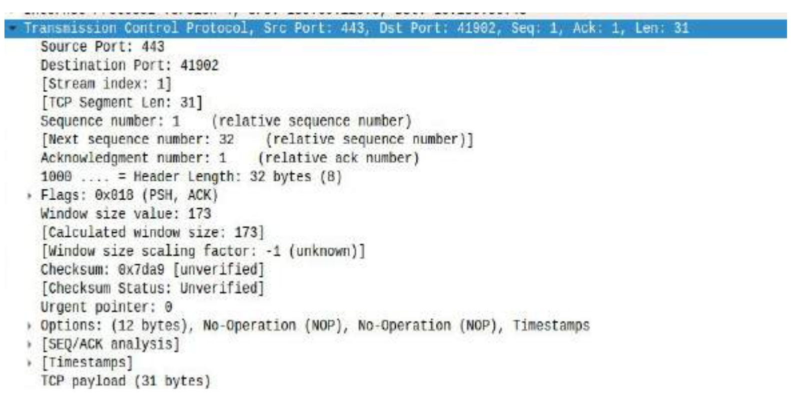
Destination and source contain the MAC address of destination endpoint and sender's endpoint.
Both addresses are unicast and globally unique.
Type indicates which protocol is encapsulated in the payload of the frame.



```
Ethernet II, Src: Cisco_d9:f7:c0 (00:25:b4:d9:f7:c0), Dst: IntelCor_01:68:00 (a0:af:bd:01:68:00)
 ▾ Destination: IntelCor_01:68:00 (a0:af:bd:01:68:00)
     Address: IntelCor_01:68:00 (a0:af:bd:01:68:00)
     .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
     .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
 ▾ Source: Cisco_d9:f7:c0 (00:25:b4:d9:f7:c0)
     Address: Cisco_d9:f7:c0 (00:25:b4:d9:f7:c0)
     .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
     .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
   Type: IPv4 (0x0800)
```

C. TCP

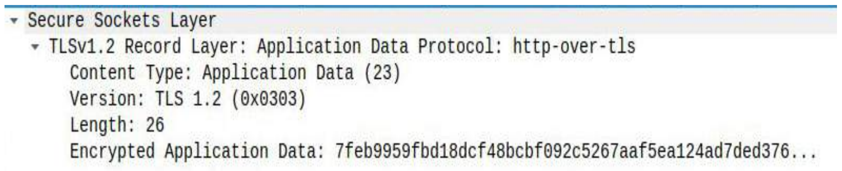
The *source and destination ports* are the numbers of ports of the server and my PC between which communication is happening. The *header length* is 32 bytes and hence a total of 8 words are present. Here both the *sequence* and *acknowledgement number* is 1. Total of two *flags* is set (Push and Acknowledgment). Here ACK means that the machine sending the packet is acknowledging data that is received from another end. PSH is an indication to push the entire buffer immediately to the receiving application. The *Window Size* Value on packets from A to B indicates how much buffer space is available on A for receiving packets and its current value is 173. The *checksum* is again, used for error detection in the entire packet. *Scaling Factor* is the number by which the window size displayed is to be scaled. Also, the value of the *urgent pointer* is 8 meaning that a total of 8 data bytes is urgent in the current segment.



```
▾ Transmission Control Protocol, Src Port: 443, Dst Port: 41902, Seq: 1, Ack: 1, Len: 31
    Source Port: 443
    Destination Port: 41902
    [Stream index: 1]
    [TCP Segment Len: 31]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 32    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 173
    [Calculated window size: 173]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x7da9 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (31 bytes)
```

D. Secure Sockets Layer(SSL)

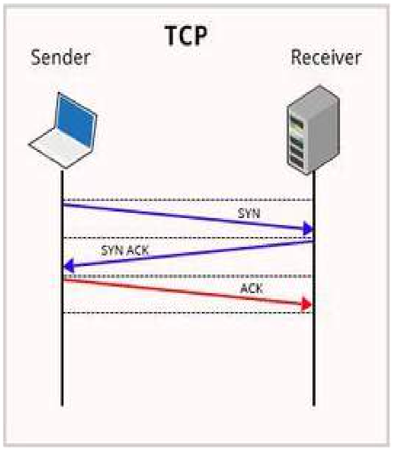
The *Application-Data-Protocol* indicates the protocol used at the application layer, which is HTTP (secured with TLS) in this case. *Content-Type* indicates the type of payload, which is simply the Application Data in this case. In total there is a total of 4 types of Content-type. *Version* tells the protocol version used for securing the communication, which is TLSv1.2 here. *Length* is simply the length of the message. *Encrypted Application Data* is unreadable by anyone who does not have the required key.



```
▾ Secure Sockets Layer
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 26
      Encrypted Application Data: 7feb9959fbd18dcf48bcbf092c5267aaf5ea124ad7ded376...
```

3) The Relevance of the protocols for the application.

As Vimeo is a video streaming site, *reliability is of utmost importance.* Hence, the use of TCP is inevitable. TCP/IP protocol sends the packet only if the link is connected at all times during the connection.

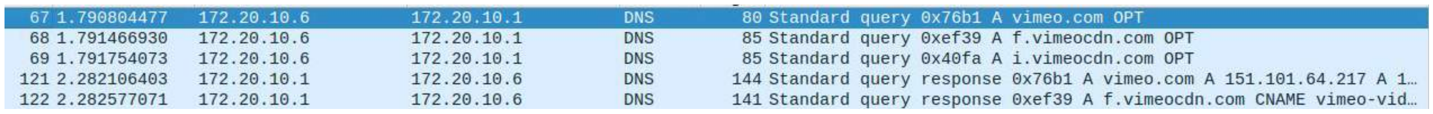


- ***TCP(Transmission Control Protocol)*** is a **reliable** protocol, meaning that the data that is sent is reached by the receiving party. Data packets that are lost are resent again, if the connection fails then the data is re-requested, thus making sure that data is received at the other end.
- Also as TCP is a **stream-oriented** protocol, it means that it is considered to be along the stream of data that is transmitted from one end of the connection to the other end. The TCP/IP stack is responsible for breaking the stream of data into packets and sending those packets while the stack at the other end is responsible for **reassembling the packets into a data stream** using the information in the packet headers.
- Also, it is necessary that data from the server reaches us in an ordered manner, as the video is to be streamed from beginning to end. TCP enables data to be received in an **ordered way,** meaning if 5 data packets are sent, then data packet 1 should be received before data packet 2.
- TCP is **compatible only with IP at Network Layer**, so Vimeo uses IP at the network layer. ***Internet Protocol(IP)* is a connectionless** protocol, hence neither reliable delivery nor correctly ordered data is guaranteed and therefore TCP is needed to provide connection-based protocol which ensures the problem caused by IP protocol.
- ***Ethernet II*** is used at the link layer as it ensures **reliable data transfer** between two nodes and involves proper error handling and flow control mechanisms to minimize errors. Other features include CRC for **error detection** and preamble for **synchronization**.
- ***SSL*** is used on top of HTTP to prevent hackers to snoop on the packets being sent between the two endpoints. SSL **encrypts** the message data, which is not readable without the required key. Its basic core function is to protect server-client information. Apart from encryption and authentication, SSL certificates are vital from a **customer trust point** of view. The easy to identify signs inform the users that the data they send will be secured.

4) The Sequence of Messages Exchange

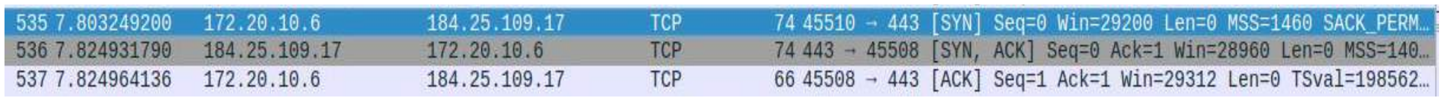
- My Laptop was connected to Airtel's mobile hotspot for screenshots and to carry out this experiment so that we can clearly view the results. The sequence of the message follows broadly the following steps:

- **DNS querying:** First, when the site is loaded, DNS querying is done by the browser. DNS querying is a demand for information sent from the user's computer(DNS client) to a DNS server to ask for the IP address associated with the domain name vimeo.com. As we can see, the querying is for 'A' record type, which is used to relate IP addresses with domain names.



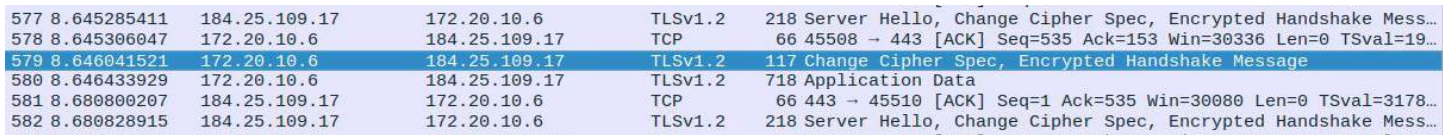
```
 67 1.790804477   172.20.10.6     172.20.10.1     DNS    80 Standard query 0x76b1 A vimeo.com OPT
 68 1.791466930   172.20.10.6     172.20.10.1     DNS    85 Standard query 0xef39 A f.vimeocdn.com OPT
 69 1.791754073   172.20.10.6     172.20.10.1     DNS    85 Standard query 0x40fa A i.vimeocdn.com OPT
121 2.282106403   172.20.10.1     172.20.10.6     DNS   144 Standard query response 0x76b1 A vimeo.com A 151.101.64.217 A 1…
122 2.282577071   172.20.10.1     172.20.10.6     DNS   141 Standard query response 0xef39 A f.vimeocdn.com CNAME vimeo-vid…
```

- **TCP 3-way handshake**: It is a process which is used in a TCP/IP network to make a connection between the server(Vimeo) and client(my PC). It is a three-step process where 53386 is client port and port 443 is Vimeo's server:
  - **Step 1:** The client establishes a connection with a server. It sends a segment with the SYN and informs the server about the client should start communication, and with what should be its sequence number.
  - **Step 2:** The server responds to the client request with an SYN-ACK signal set. ACK helps you to signify the response of segment that is received and SYN signifies what sequence number it should able to start with the segments.
  - **Step 3:** In this final step, the client acknowledges the response of the Server, and they both create a stable connection will begin the actual data transfer process.



```
535 7.803249200   172.20.10.6     184.25.109.17   TCP    74 45510 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM…
536 7.824931790   184.25.109.17   172.20.10.6     TCP    74 443 → 45508 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=140…
537 7.824964136   172.20.10.6     184.25.109.17   TCP    66 45508 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=198562…
```

- **TLS Handshaking:** To make the connection secure, TLSv1.2 is being used. The first message in the TLS Handshake is the *Client Hello* which is sent by the client to initiate a session with the server. The server responds with *Server Hello* and the *Server Certificate (for authentication)*. The server also sends a *Server Key*, which is used by the client to encrypt Client Key Exchange later in the process. *Server Hello Done* indicates that the server is done and is now awaiting the client's response. The client responds with the *Client Key* and is issued a New Session Ticket. The TLS session is now established, and application data can be exchanged between both the server and the client.

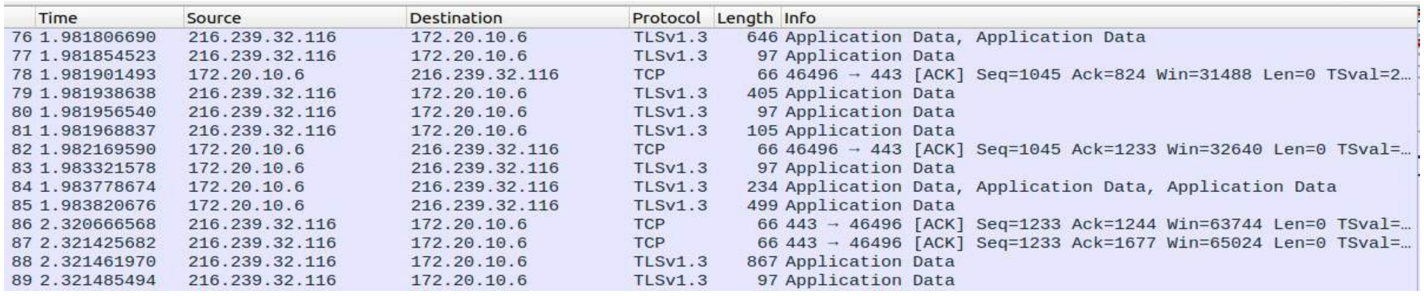


```
577 8.645285411   184.25.109.17   172.20.10.6     TLSv1.2  218 Server Hello, Change Cipher Spec, Encrypted Handshake Mess…
578 8.645306047   172.20.10.6     184.25.109.17   TCP       66 45508 → 443 [ACK] Seq=535 Ack=153 Win=30336 Len=0 TSval=19…
579 8.646041521   172.20.10.6     184.25.109.17   TLSv1.2  117 Change Cipher Spec, Encrypted Handshake Message
580 8.646433929   172.20.10.6     184.25.109.17   TLSv1.2  718 Application Data
581 8.680800207   184.25.109.17   172.20.10.6     TCP       66 443 → 45510 [ACK] Seq=1 Ack=535 Win=30080 Len=0 TSval=3178…
582 8.680828915   184.25.109.17   172.20.10.6     TLSv1.2  218 Server Hello, Change Cipher Spec, Encrypted Handshake Mess…
```

- **Request for Resources**: After handshaking is over and communication is established, the browser sends a CONNECT request to the server and asks for the web page. The server responds by sending the desired resource.

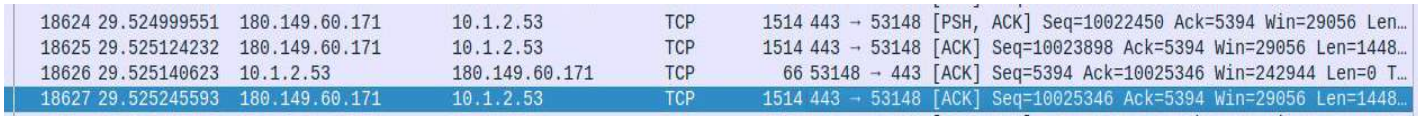
Data flow for different functionalities.

- **Streaming Videos**
  Once the video starts streaming, the Vimeo's server sends application data to the client, each of which is Acknowledged by a message from the client by the ACK flag. For multiple TCP segments of a reassembled PDU(Protocol Data Unit) sent from the server, the client sends a cumulative ACK back indicating the first-byte number that it expects from the server. Once all these segments arrive, they are re-assembled and then data is sent to the application layer (HTTP and SSL).



```
   Time          Source           Destination      Protocol Length Info
76 1.981806690   216.239.32.116   172.20.10.6      TLSv1.3    646 Application Data, Application Data
77 1.981854523   216.239.32.116   172.20.10.6      TLSv1.3     97 Application Data
78 1.981901493   172.20.10.6      216.239.32.116   TCP         66 46496 → 443 [ACK] Seq=1045 Ack=824 Win=31488 Len=0 TSval=2…
79 1.981938638   216.239.32.116   172.20.10.6      TLSv1.3    405 Application Data
80 1.981956540   216.239.32.116   172.20.10.6      TLSv1.3     97 Application Data
81 1.981968837   216.239.32.116   172.20.10.6      TLSv1.3    105 Application Data
82 1.982169590   172.20.10.6      216.239.32.116   TCP         66 46496 → 443 [ACK] Seq=1045 Ack=1233 Win=32640 Len=0 TSval=…
83 1.983321578   172.20.10.6      216.239.32.116   TLSv1.3     97 Application Data
84 1.983778674   172.20.10.6      216.239.32.116   TLSv1.3    234 Application Data, Application Data, Application Data
85 1.983820676   172.20.10.6      216.239.32.116   TLSv1.3    499 Application Data
86 2.320666568   216.239.32.116   172.20.10.6      TCP         66 443 → 46496 [ACK] Seq=1233 Ack=1244 Win=63744 Len=0 TSval=…
87 2.321425682   216.239.32.116   172.20.10.6      TCP         66 443 → 46496 [ACK] Seq=1233 Ack=1677 Win=65024 Len=0 TSval=…
88 2.321461970   216.239.32.116   172.20.10.6      TLSv1.3    867 Application Data
89 2.321485494   216.239.32.116   172.20.10.6      TLSv1.3     97 Application Data
```

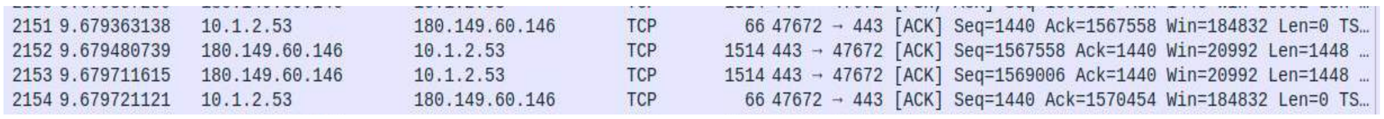
  Operations like playing, pausing etc while streaming can lead to communication between a different port on the client and the same 443 Vimeo server port. In this communication, a message will be sent from the client which is replied by the server along with ACK(piggy-backing protocol). When the video is paused, Keep-Alive messages are exchanged to keep the connection alive. Also, we can see that ACK is sent from 443 servers but the packet length is non-zero, implying that piggybacking protocol is being enforced.



```
18624 29.524999551  180.149.60.171   10.1.2.53        TCP   1514 443 → 53148 [PSH, ACK] Seq=10022450 Ack=5394 Win=29056 Len…
18625 29.525124232  180.149.60.171   10.1.2.53        TCP   1514 443 → 53148 [ACK] Seq=10023898 Ack=5394 Win=29056 Len=1448…
18626 29.525140623  10.1.2.53        180.149.60.171   TCP     66 53148 → 443 [ACK] Seq=5394 Ack=10025346 Win=242944 Len=0 T…
18627 29.525245593  180.149.60.171   10.1.2.53        TCP   1514 443 → 53148 [ACK] Seq=10025346 Ack=5394 Win=29056 Len=1448…
```

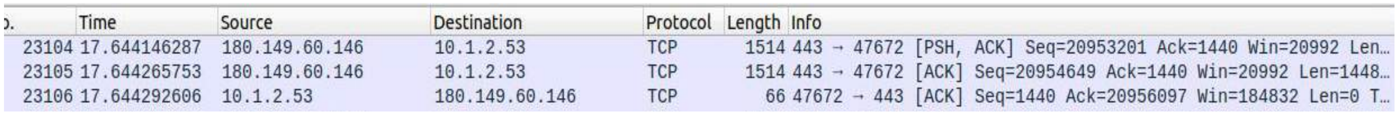
- **Downloading videos**

While downloading the videos, the same sequence of messages as streaming was observed. However, the client(my PC) does not send any message to the server on a different port.



```
2151 9.679363138   10.1.2.53        180.149.60.146   TCP     66 47672 → 443 [ACK] Seq=1440 Ack=1567558 Win=184832 Len=0 TS…
2152 9.679480739   180.149.60.146   10.1.2.53        TCP   1514 443 → 47672 [ACK] Seq=1567558 Ack=1440 Win=20992 Len=1448 …
2153 9.679711615   180.149.60.146   10.1.2.53        TCP   1514 443 → 47672 [ACK] Seq=1569006 Ack=1440 Win=20992 Len=1448 …
2154 9.679721121   10.1.2.53        180.149.60.146   TCP     66 47672 → 443 [ACK] Seq=1440 Ack=1570454 Win=184832 Len=0 TS…
```

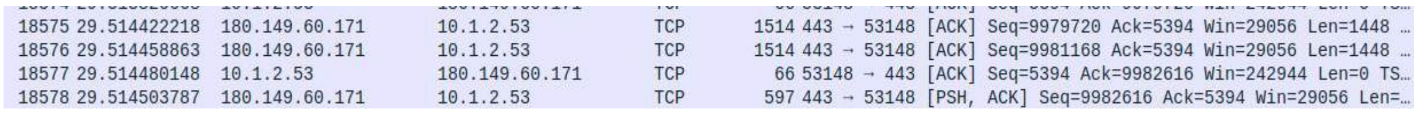
Also when the video is about to be completely downloaded, the server sends a PSH packet (Push Flag) and hence, conveying the client to receive the concerned packet and empty out the buffer by giving data to the relevant application



```
).      Time          Source           Destination      Protocol Length Info
 23104 17.644146287  180.149.60.146   10.1.2.53        TCP      1514 443 → 47672 [PSH, ACK] Seq=20953201 Ack=1440 Win=20992 Len…
 23105 17.644265753  180.149.60.146   10.1.2.53        TCP      1514 443 → 47672 [ACK] Seq=20954649 Ack=1440 Win=20992 Len=1448…
 23106 17.644292606  10.1.2.53        180.149.60.146   TCP        66 47672 → 443 [ACK] Seq=1440 Ack=20956097 Win=184832 Len=0 T…
```

- **Uploading Videos**

When we start we start uploading a video on the server, data is being sent from the client which is acknowledged back to the client by the server using piggybacking protocol. Generally, the server sends a cumulative ACK.
<u>*Connection Tear-Down*</u>: When the communication is over, there is an exchange of messaged to tear-down the TCP connection. The client first sends a FIN message to the server, and once the server receives it, it acknowledges it by sending its own FIN-ACK message. Once the client received this message, it again sends an ACK back to the server and the connection formally closes and all resources on the client side are discharged.



```
18575 29.514422218  180.149.60.171   10.1.2.53        TCP   1514 443 → 53148 [ACK] Seq=9979720 Ack=5394 Win=29056 Len=1448 …
18576 29.514458863  180.149.60.171   10.1.2.53        TCP   1514 443 → 53148 [ACK] Seq=9981168 Ack=5394 Win=29056 Len=1448 …
18577 29.514480148  10.1.2.53        180.149.60.171   TCP     66 53148 → 443 [ACK] Seq=5394 Ack=9982616 Win=242944 Len=0 TS…
18578 29.514503787  180.149.60.171   10.1.2.53        TCP    597 443 → 53148 [PSH, ACK] Seq=9982616 Ack=5394 Win=29056 Len=…
```

5) Trace Statistics

- **Streaming Videos**

Time of the Day	Avg. Throughput (Bytes/sec)	Avg. RTT	Avg. Packet Size(Bytes)	Packets Lost	UDP Packets	TCP Packets	Responses Per Packet
11 AM	780102	0.148	1034	0	126	15298	1
5 PM	232601	0.227	880	8	52	4377	1.08
10 PM	371291	0.851	1016	5	405	18398	1

- **Downloading Videos**

Time of the Day	Avg. Throughput (Bytes/sec)	Avg. RTT	Avg. Packet Size(Bytes)	Packets Lost	UDP Packets	TCP Packets	Responses Per Packet
11 AM	1490201	0.403	969	1	132	22177	1.102
5 PM	2517192	0.253	1026	0	55	21867	1
10 PM	241384	0.194	1021	5	91	23597	1.04

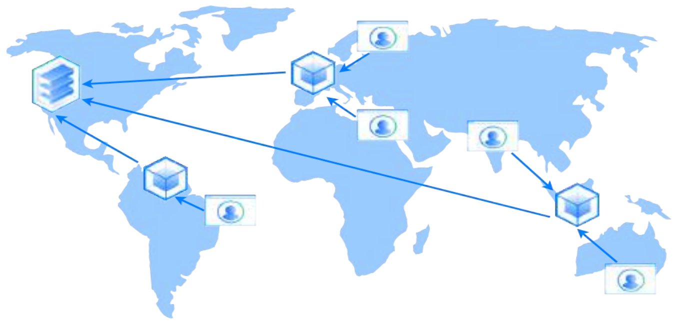
- **Uploading Videos**

Time of the Day	Avg. Throughput (Bytes/sec)	Avg. RTT (ms)	Avg. Packet Size(Bytes)	Packets Lost	UDP Packets	TCP Packets	Responses Per Packet
11 AM	1219707	3.30	1175	26	169	20054	1
5 PM	1065589	5.67	1172	5	156	19066	0.997
10 PM	1120771	3.80	1165	3	273	18414	0.996

6) Content Sources

Vimeo is a globally used video streaming service and supports high-definition video and has a lot of users across the world. To accommodate such heavy data from various clients, Vimeo uses multiple servers with service requests.

When we first open the Vimeo site, we can see that the browser tries to connect to “**a1880.dscc.akamai.net**” and hence we can infer that Vimeo uses the Akamai content distribution network. The browser also tries to connect to “Vimeo-cdn” implying that Vimeo has a CDN of its own as well. Also, certain requests were made to Google APIs related to ad services and safe browsing. From the resolved address section of Wireshark, one can notice that the client (my PC) communicates with Vimeo through different server IP’s. Some of the IP’s commonly used were **151.101.36.217 and 151.101.128.217.** In general, we can state that the IP address of Vimeo servers starts from 151.101.



Content Delivery Network (CDN)

This *Content Distribution Network* has various benefits which include a decrease in server load time, faster delivery of content along with lower network latency and packet loss. Multiple servers help in increasing security and reliability, as there is no single point of error as if one server is damaged, data could be retrieved from another server.