CS342 Assignment 2 Samiksha Sachdeva 180123040

Drive link of traces:

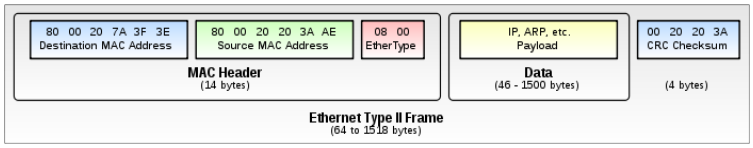
https://drive.google.com/drive/folders/1CoizwFdyy_7x7aVI-0y_o8kSSN0o9vcq?usp=sharing

Question 1:

The following protocols were used by the app(YouTube live) at different layers. Since there were many applications running on the PC at the same time, many protocols correspond to the irrelevant apps and are therefore of no use to us.(Assumption: The values given for each protocol is after applying the filters of IP addresses and we have randomly chosen the packet corresponding to each protocol so the values may differ with other packets.)

a) Ethernet II (Link layer): The Ethernet frame starts with a **Preamble** and a **SFD** (start frame delimiter), both of which work at the physical layer. The header contains both the **source** as well as **destination** MAC addresses, after which the **payload** of the frame is present. The last field corresponds to the Frame Check Sequence which is basically a **Cyclic Redundancy Check (CRC)** for the detection of errors.

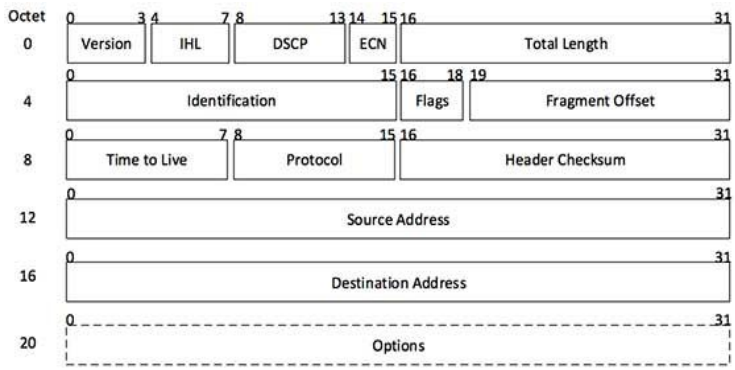
```
Ethernet II, Src: HuaweiTe_42:38:37 (50:1d:93:42:38:37), Dst: IntelCor_d6:6c:e9 (e0:9d:31:d6:6c:e9)
  Destination: IntelCor_d6:6c:e9 (e0:9d:31:d6:6c:e9)
  Source: HuaweiTe_42:38:37 (50:1d:93:42:38:37)
  Type: IPv4 (0x0800)
```



b) Internet Protocol Version 4 (Network layer): It provides the logical connection between network devices by providing identification for each device. IP provides a mechanism to uniquely identify hosts by an IP addressing scheme which consists of 32-bit logical addresses. The IPv4 packet header consists of 14 fields, of which 13 are required. The 14th field is optional and aptly named: options. It has relevant information including version number (in this case 4), total length of the entire packet, TTL, protocol, source and destination address and so on.

```
Internet Protocol Version 4, Src: 150.242.84.77, Dst: 192.168.100.28
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   Total Length: 1378
   Identification: 0x0002 (2)
 > Flags: 0x4000, Don't fragment
   Fragment offset: 0
   Time to live: 124
   Protocol: UDP (17)
   Header checksum: 0xe984 [validation disabled]
   [Header checksum status: Unverified]
   Source: 150.242.84.77
   Destination: 192.168.100.28
```

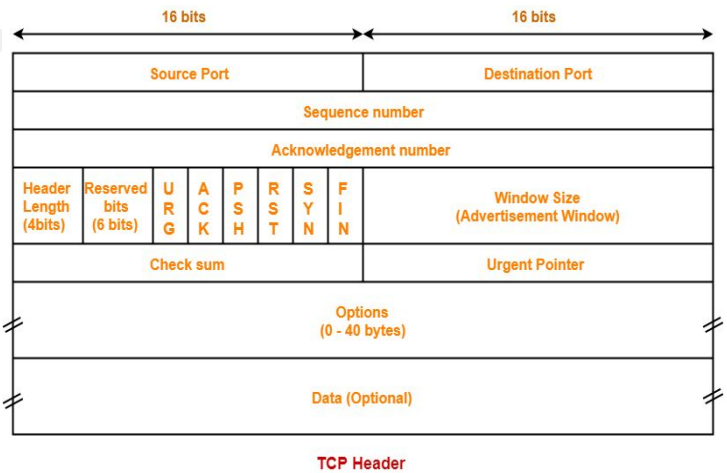
c)



[Image: IP Header]

Transmission Control Protocol (Transport layer): TCP is a connection-oriented Layer 4 protocol that provides full-duplex, acknowledged, and flow-controlled service to upper-layer protocols. It moves data in a continuous, unstructured byte stream. Sequence numbers identify bytes within that stream. TCP can also support numerous simultaneous upper-layer conversations. A TCP segment consists of a segment header and a data section. The TCP header contains 10 mandatory fields and an optional extension field. It includes a number of fields including source and destination addresses, sequence, and acknowledgment number and checksum.

```
Transmission Control Protocol, Src Port: 50281, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
   Source Port: 50281
   Destination Port: 443
   [Stream index: 5]
   [TCP Segment Len: 0]
   Sequence number: 1    (relative sequence number)
   Sequence number (raw): 1227140812
   [Next sequence number: 1    (relative sequence number)]
   Acknowledgment number: 1    (relative ack number)
   Acknowledgment number (raw): 2117835762
   0101 .... = Header Length: 20 bytes (5)
 > Flags: 0x010 (ACK)
   Window size value: 259
   [Calculated window size: 66304]
   [Window size scaling factor: 256]
   Checksum: 0x4232 [unverified]
   [Checksum Status: Unverified]
   Urgent pointer: 0
 > [SEQ/ACK analysis]
```



d) User Datagram Protocol (Transport layer): Source Port refers to the port number associated with process sending packet from source. **Destination Port** refers to the port number associated with process receiving packet at the destination. **Length** of the UDP segment. **Checksum** is the field used for error checking of header and data.

```
User Datagram Protocol, Src Port: 443, Dst Port: 62738
    Source Port: 443
    Destination Port: 62738
    Length: 1358
    Checksum: 0x2d47 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 3]
  > [Timestamps]
```

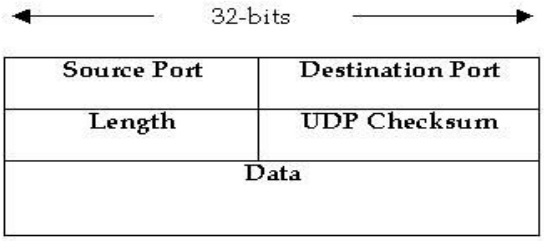
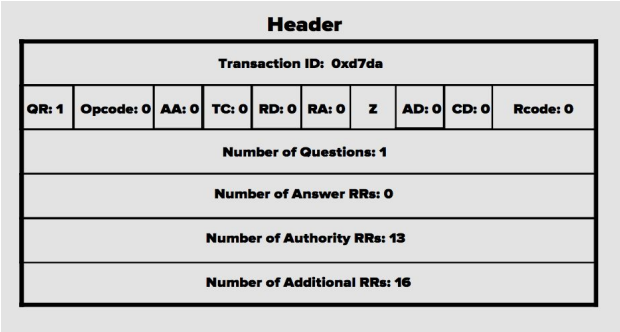


Figure 8: UDP segment structure

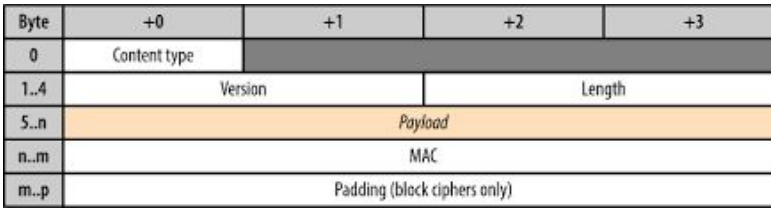
e) Domain Name System (Application layer): The **header** describes the type of packet and which fields are contained in the packet. Following the header are a **number of questions, answers, authority records, and additional records.**

```
Domain Name System (query)
    Transaction ID: 0x04fd
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Response In: 13]
```



f) Transport Layer Security (Application layer): Content Type indicates which type of data is being contained in the packet. **Version** is the TLS version used. **Length** is the length of the data being sent. **Data** is the data sent in encrypted form.

```
Transport Layer Security
  TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 2454
      Encrypted Application Data: 41c306b4dad5c230ffa62e2011ebb262193e4ad90353a06a…
```



Question 2:

All the packets follow Ethernet (II) protocol at the data link layer. It is one of the most widely used and reliable link layer protocols. It has a well defined preamble for synchronization and CRC field for error detection. It ensures reliable data transfer between 2 network devices on the path of the packet. It gives collision free interconnection of multiple devices via a common bus.
All the packets except some DNS packets follow Internet Protocol Version 4 protocol at the network layer. This protocol is used because it lays the foundation of rules on which the internet is based upon.
Now, we will focus on the **Application Layer** and **Transport Layer** protocols.

TCP is a connection-oriented reliable data transfer protocol and involves handshaking between the client and the server. It also ensures proper error handling and flow control mechanisms to minimize the error loss rate which is needed in my application.
UDP is an alternative communications protocol to TCP used primarily for establishing low-latency and loss-tolerating connections between applications on the internet.
DNS is used to translate domain names into IP Addresses, which computers can understand. It responds with the IP address of the server. It allows users to have human-readable domains while ensuring a mapping to the IP addresses corresponding to the domain names.
TLSv1.3 protocol helps your system (clients and servers) communicate over the secured layer where data travels over the wire in encrypted format which could be understood only by the involved parties not by intrusions or outside audience.

Important functionalities that I discovered are:

a) Play/Starting the video:

Protocols used: TCP, UDP, TLSv1.2

Reason: While starting the video, reliability and accuracy is more important than speed. That is why TCP is used while we click on the play button. After that when the video is playing, UDP is used since at that time speed is more important. Also whenever we are entering, let's say login credentials on a website, in this case I entered my credentials on youtube.com, the password and username are safely transmitted from my machine to the server by using TLS.

b) Pause:

Protocols used: TCP, UDP

Reason: While pausing the video, reliability and accuracy is more important than speed. That is why TCP is used while we click on the pause button. After that when the video is not playing, UDP is used since at that time speed is more important.

c) Live stream:

Protocols used: TCP, UDP

Reason: When we are lagging behind and want to get back live again, then accuracy is more important. We can't tolerate any errors in that case. This is why TCP is used. After that when the video is playing normally, UDP is used since at that time speed is more important.

d) Increase/decrease volume:

Protocols used: TCP, UDP

Reason: While increasing/decreasing the volume, reliability and accuracy is more important than speed. That is why TCP is used while we click on the pause button. After we are done changing the volume and video plays at the new volume, UDP is used since at that time speed is more important.

e) Change quality of the live stream:

Protocols used: TCP, UDP

Reason: While changing the quality of the video, reliability and accuracy is more important than speed. That is why TCP is used. After that when the video is playing at that quality, UDP is used since at that time speed is more important.

f) Decrease playback speed:

Protocols used: TCP, UDP

Reason: While decreasing the playback speed of the video, reliability and accuracy is more important than speed. That is why TCP is used. After that when the video is playing at that speed, UDP is used since at that time speed is more important.

Question 3

3- way TCP Handshaking to establish the connection:

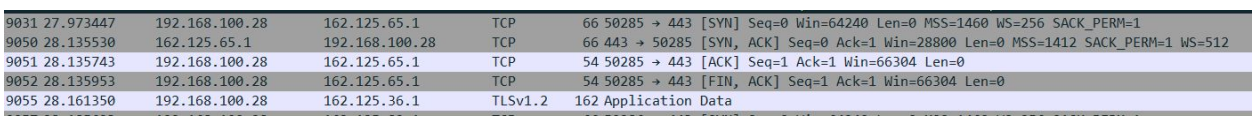
This 3-way handshake process is designed so that both ends can initiate and negotiate separate TCP socket connections at the same time.

Step1(SYN): In the first step, the client wants to establish a connection with the server, so it sends a segment with SYN(Synchronize Sequence Number) which informs the server that the client is likely to start communication and with what sequence number it starts segments with.

Step2([SYN,ACK]): Server responds to the client request with SYN-ACK signal bits set. The ACK signifies the response of the segment it received and SYN signifies the sequence number it is likely to start the segments with.

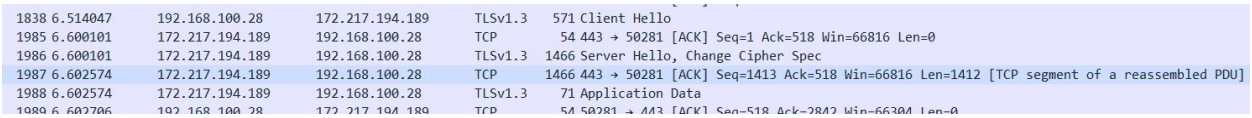
Step3(ACK): The client acknowledges the response of the server and thus establishes a reliable connection with which they start the actual data transfer.

The steps 1, 2 establish the sequence number for one direction and it is acknowledged. With these, a full-duplex communication is thus established.



9031	27.973447	192.168.100.28	162.125.65.1	TCP	66 50285 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
9050	28.135530	162.125.65.1	192.168.100.28	TCP	66 443 → 50285 [SYN, ACK] Seq=0 Ack=1 Win=28800 Len=0 MSS=1412 SACK_PERM=1 WS=512
9051	28.135743	192.168.100.28	162.125.65.1	TCP	54 50285 → 443 [ACK] Seq=1 Ack=1 Win=66304 Len=0
9052	28.135953	192.168.100.28	162.125.65.1	TCP	54 50285 → 443 [FIN, ACK] Seq=1 Ack=1 Win=66304 Len=0
9055	28.161350	192.168.100.28	162.125.36.1	TLSv1.2	162 Application Data

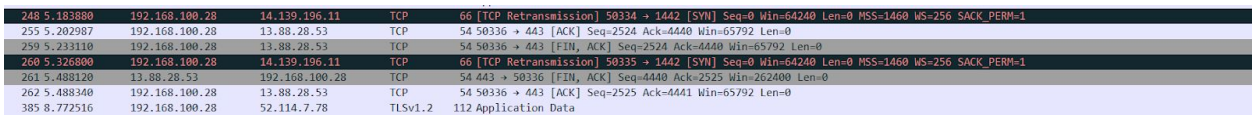
TLSv1.3 Handshaking(Application layer handshaking) starts with a client sending a "Client Hello" to the server to which the server responds with a "Server Hello" and "Server Certificate" to complete the handshake. The server also sends a server key which is used by the client for encryption purposes later in the process. Server is now waiting for the client's response which responds with a client key. This occurs during the initial loading and the application data can now be transferred between the client and the server.



DNS queries and answers: Whenever an application is opened, multiple DNS queries are performed to multiple DNS servers which give answers to resolve the address for app gateway and login.

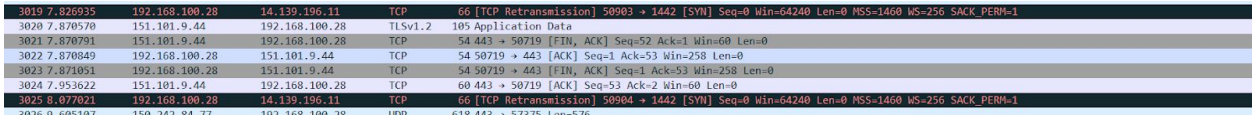
Functionality 1: Pausing the video

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



Functionality 2: Live streaming

On starting a live stream, our PC makes a HTTP GET request for a live stream, each of which is ACKnowledged from my PC. Since the video data is big in size it is broken into multiple TCP segments, indicated by the TCP and TLS segments (of a reassembled PDU), which are ACKs but with non-zero lengths, indicating that they are carrying application data. Once all the packets arrive the segments are reassembled and fed to the application layer, indicated by the HTTP message sent to our PC with the live stream as payload.



Question 4:

	11 PM	3PM	10 AM
Throughput(kilo bytes/s)	258	250	171
RTT (ms)	131.8	94.96	120.3
Avg. Packet size (bytes)	1129	1186	1179
Number of packets lost	0	0	0
Number of TCP, UDP packets	TCP=407, UDP=15286	TCP=101, UDP=8404	TCP=97, UDP=6089
Number of responses per request	12947/2689=4.81	7420/1083=6.85	5387/785=6.86

Question 5:

I repeated the experiment again and again at different times of the day but I was always getting the same IPs, indicating that the request was being processed by a unique server. In case of heavy applications though, we might get multiple servers. The reason for that is:

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

In case there is a change in the IP, one possible reason could be:

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