

CS-342 Assignment 1

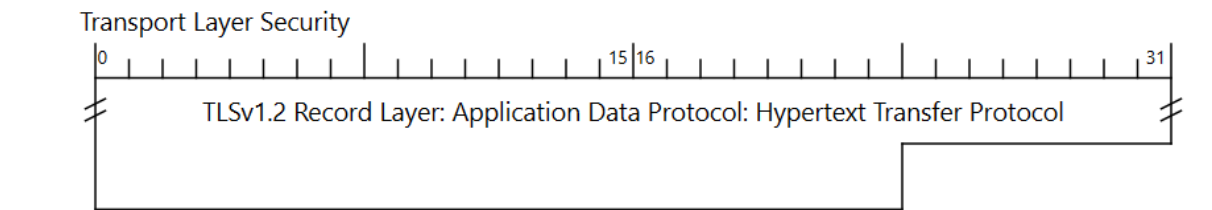
Group 24

1)Udbhav Gupta: 220101106

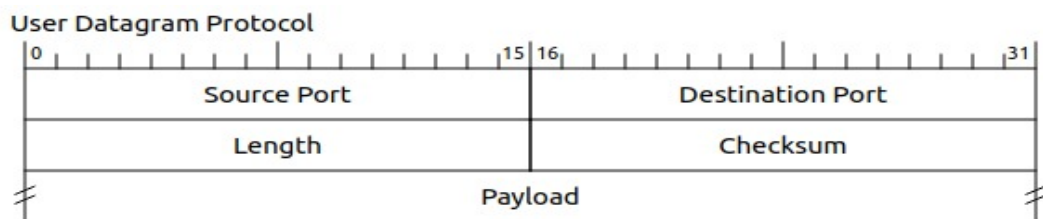
2)Tanay Goenka: 220101098

Packet Formats of Various Protocols used in different applications -

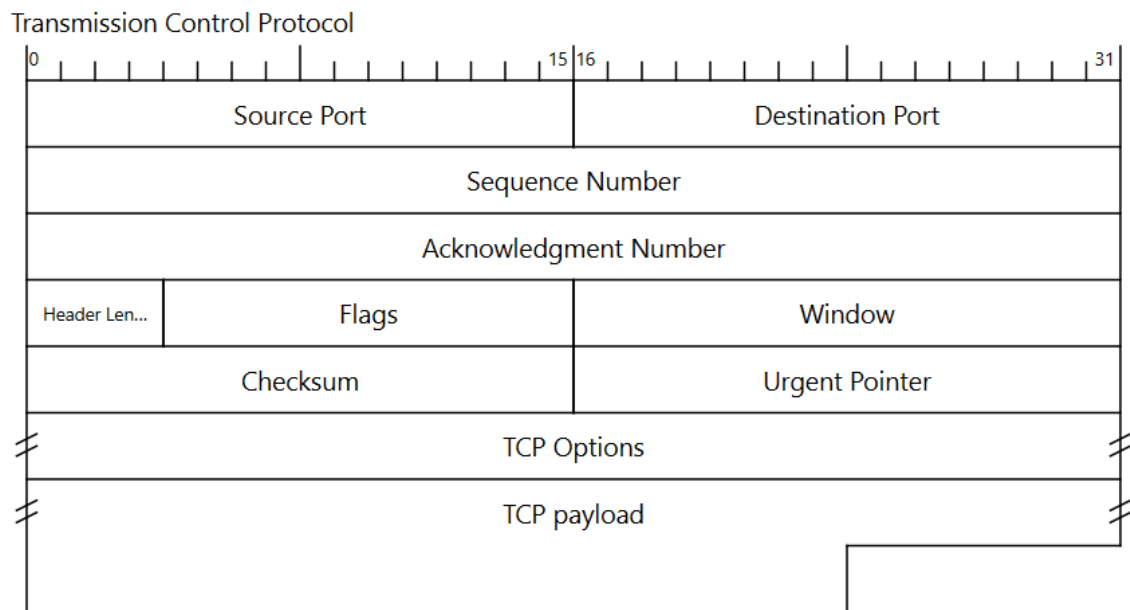
Packet format for TLS(Transport Layer Security) :-



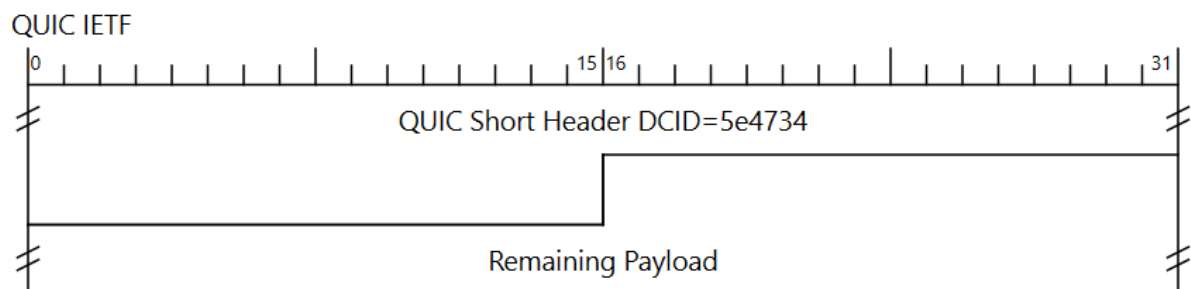
Packet format for UDP(User Datagram Protocol) :-



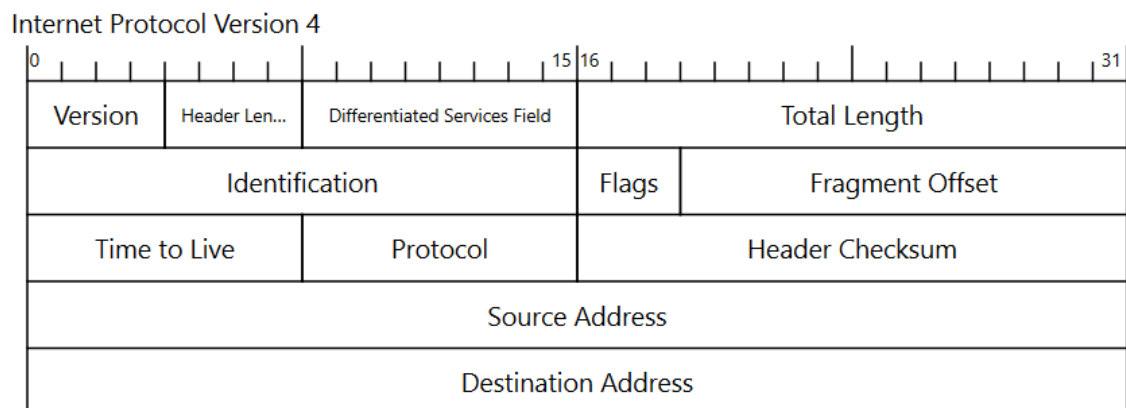
Packet format for TCP(Transmission Control Protocol) :-



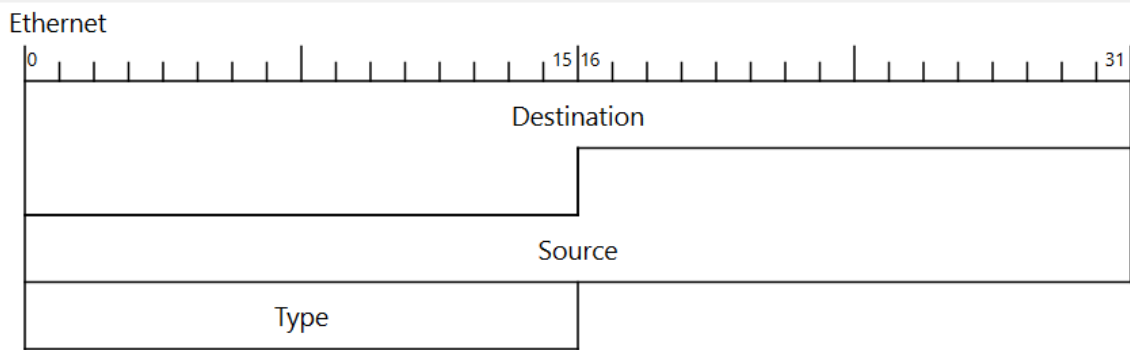
Packet format for QUIC IETF :-



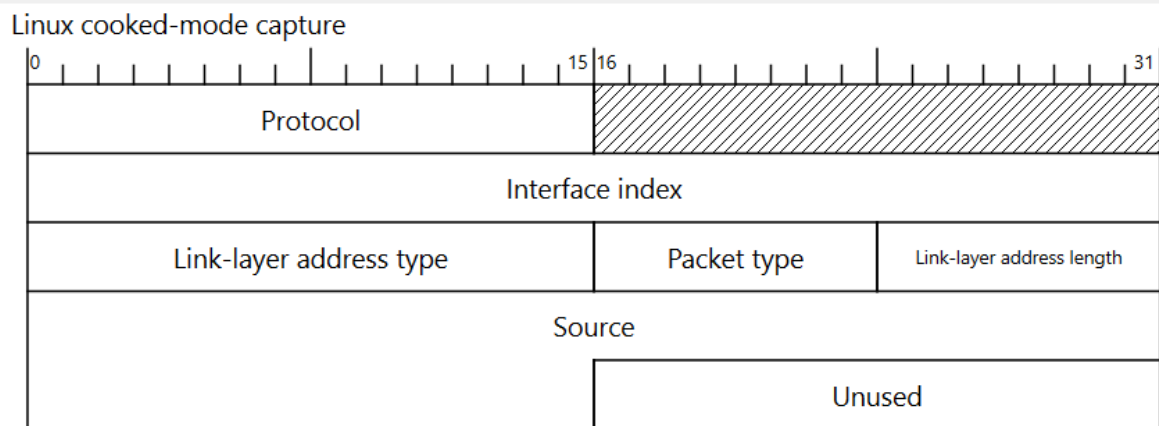
Packet format for IPv4(Internet Protocol Version 4) :-



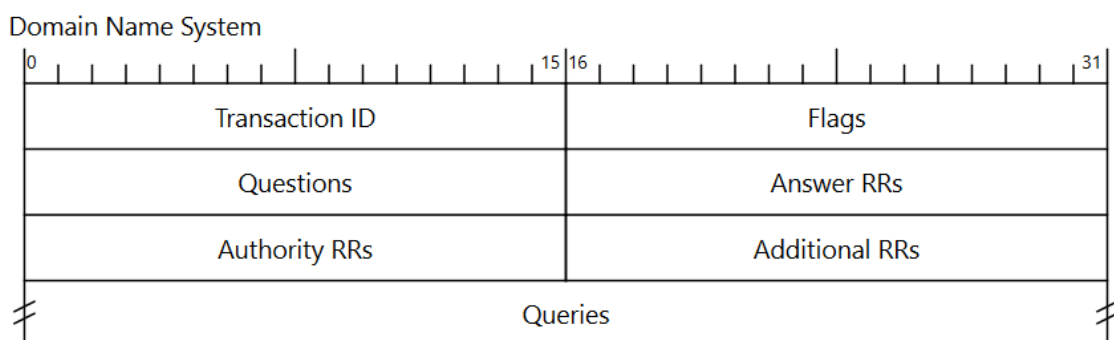
Packet format for Ethernet :-



Packet format for Linux cooked mode-capture :-



Packet format for DNS(Domain Name System) :-



1)Youtube

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:

Wireshark - Protocol Hierarchy Statistics - youtubeDownload.pcap										
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs	
▼ Frame	100.0	484	100.0	321849	105 k	0	0	0	484	
▼ Ethernet	100.0	484	2.1	6776	2219	0	0	0	484	
▼ Internet Protocol Version 4	100.0	484	3.0	9680	3170	0	0	0	484	
▼ User Datagram Protocol	96.9	469	1.2	3752	1228	0	0	0	469	
Data	96.9	469	93.5	300771	98 k	469	300771	98 k	469	
▼ Transmission Control Protocol	3.1	15	0.3	870	284	5	160	52	15	
Transport Layer Security	2.1	10	0.1	390	127	10	390	127	10	

Wireshark - Protocol Hierarchy Statistics - youtubeUpload.pcap										
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs	
▼ Frame	100.0	219	100.0	137488	53 k	0	0	0	219	
▼ Ethernet	100.0	219	2.2	3066	1185	0	0	0	219	
▼ Internet Protocol Version 4	100.0	219	3.2	4380	1694	0	0	0	219	
▼ User Datagram Protocol	79.9	175	1.0	1400	541	0	0	0	175	
QUIC IETF	14.6	32	13.9	19046	7366	32	13775	5327	40	
Data	65.3	143	72.7	99996	38 k	143	99996	38 k	143	
▼ Transmission Control Protocol	20.1	44	8.6	11807	4566	26	5048	1952	44	
Transport Layer Security	8.2	18	8.5	11650	4505	18	11650	4505	18	

Layer 7 (Application): TLS for securing data transmission.

Layer 4 (Transport): UDP and TCP for connectionless and connection-oriented transport, respectively. QUIC operates in Layer 4 (Transport Layer), providing fast, reliable data transfer with built-in encryption and reduced latency.

Layer 3 (Network): IPv4 for routing packets across networks.

Layer 2 (Data Link): Ethernet for data framing and local transmission.

Packet format for Ethernet :-

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

Transport Layer Security (TLS) (Layer 7 - Application Layer)

```

▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 34
    Encrypted Application Data: 8628dbb71806e2387b79752f4e2100f013f09704ad87c18b77baeed15deaca276cb7
    [Application Data Protocol: Hypertext Transfer Protocol]
  
```

- **Version:** `TLS 1.2 (0x0303)`
 - **Explanation:** This indicates that the version of TLS being used is 1.2, which is a protocol for encrypting the communication over the network.
 - **Length:** `34`
 - **Explanation:** The length of the TLS record layer data is 34 bytes.
 - **Encrypted Application Data:**
 - **Explanation:** The actual application data is encrypted, making it secure from eavesdropping. This data likely contains HTTP(S) payloads, as indicated by the protocol (`Hypertext Transfer Protocol`).
-

QUIC IRTF (Layer 4 - Transport Layer)

- ✓ **QUIC IETF**
 - > **QUIC Connection information**
[Packet Length: 26]
 - > **QUIC Short Header DCID=5e4734**
Remaining Payload: `9215e852eb32b71667423c8d841e159c0904599dcf2b`

1. QUIC Connection Information

- **[Packet Length: 26]:** This indicates the total length of the QUIC packet, which is 26 bytes. It includes both the header and the payload data.

2. QUIC Short Header

- **DCID (Destination Connection ID): 5e4734:** The Destination Connection ID (DCID) is a unique identifier used by the server to identify the connection. It helps route the packet to the correct session or stream within the QUIC connection. The value here is `5e4734`.

3. Remaining Payload

- **9215e852eb32b71667423c8d841e159c8904599dcf2b:** This is the encrypted or encoded data that makes up the rest of the QUIC packet after the header. This payload could contain application data, such as a portion of a video being streamed or some other information being transferred. The exact content is not interpretable directly due to encryption and encoding.

Explanation:

- **DCID:** The Destination Connection ID is essential for ensuring that the packet reaches the intended session within the server, particularly in cases where multiple connections are managed simultaneously.
- **Remaining Payload:** This field contains the actual data being transmitted, though in an encrypted form, ensuring security and integrity during transport.

Transmission Control Protocol (TCP) (Layer 4 - Transport Layer)

```

v Transmission Control Protocol, Src Port: 44916, Dst Port: 443, Seq: 1, Ack: 1, Len: 39
  Source Port: 44916
  Destination Port: 443
  [Stream index: 0]
  > [Conversation completeness: Incomplete (12)]
  [TCP Segment Len: 39]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 4180958113
  [Next Sequence Number: 40 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 2487825051
  1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x018 (PSH, ACK)
  Window: 501
  [Calculated window size: 501]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0xf5b2 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [Timestamps]
  > [SEQ/ACK analysis]
  TCP payload (39 bytes)

```

- **Source Port: 44916**
 - **Explanation:** This is the port number used by the source device to send the TCP segment. Port 44916 is a randomly chosen ephemeral port.
- **Destination Port: 443**
 - **Explanation:** Port 443 is the standard port for HTTPS traffic, indicating that this TCP segment is part of an encrypted HTTPS session.
- **Sequence Number: 1 (relative sequence number)**
 - **Explanation:** The sequence number is used to order the data segments. Here, the relative sequence number is 1, meaning this is likely the beginning of a sequence.
- **Acknowledgment Number: 1 (relative ack number)**
 - **Explanation:** The acknowledgment number indicates that the sender has successfully received all bytes up to and including the number 1.
- **Flags: 0x018 (PSH, ACK)**
 - **Explanation:** The flags PSH (Push) and ACK (Acknowledgment) are set. PSH means the data should be sent to the application immediately, and ACK means the acknowledgment field is significant.
- **Window: 501**
 - **Explanation:** The window size indicates how much data (in bytes) the sender is willing to receive before requiring an acknowledgment.
- **Checksum: 0xf5b2 [unverified]**
 - **Explanation:** This is the checksum value for error-checking the TCP header. The status shows that verification is disabled in this capture.
- **Options: No-Operation (NOP), Timestamps**
 - **Explanation:** TCP options are used to enhance protocol operations. Here, NOP is used for padding, and Timestamps are used for Round-Trip Time (RTT) measurement.
- **TCP Payload: 39 bytes**
 - **Explanation:** This is the amount of actual data being carried in this TCP segment.

Internet Protocol Version 4 (IPv4) (Layer 3 - Network Layer)

```

v Internet Protocol Version 4, Src: 172.16.115.44, Dst: 142.250.71.46
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 91
    Identification: 0x602c (24620)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0xe50b [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.16.115.44
    Destination Address: 142.250.71.46

```

- **Version: 4**
 - **Explanation:** This indicates that the IP version used in this packet is IPv4.
- **Header Length: 20 bytes (5)**
 - **Explanation:** The header length is 20 bytes, which is typical for an IPv4 header without any options.
- **Differentiated Services Field (DS Field): 0x00 (DSCP: CS0, ECN: Not-ECT)**
 - **Explanation:** The Differentiated Services Code Point (DSCP) is used for packet priority. **CS0** indicates that there is no special priority. **ECN: Not-ECT** shows that Explicit Congestion Notification is not being used.
- **Total Length: 91**
 - **Explanation:** This is the total length of the IP packet, including both the header and the payload, in bytes.
- **Identification: 0x602c (24620)**
 - **Explanation:** This unique identifier helps in reassembling fragmented packets.
- **Flags: 0x2, Don't fragment**
 - **Explanation:** The flag **0x2** indicates that the packet should not be fragmented.
- **Time to Live (TTL): 64**
 - **Explanation:** TTL determines the lifetime of the packet, i.e., how many hops (routers) it can pass through before being discarded. A value of 64 is typical for many operating systems.
- **Protocol: TCP (6)**
 - **Explanation:** This field indicates that the encapsulated protocol in the IP packet is TCP (Transmission Control Protocol).
- **Header Checksum: 0xe50b [validation disabled]**
 - **Explanation:** This is the checksum value for error-checking the IP header. The status shows that validation is disabled in this capture.
- **Source Address: 172.16.115.44**
 - **Explanation:** This is the IPv4 address of the source device sending the packet.
- **Destination Address: 142.250.71.46**
 - **Explanation:** This is the IPv4 address of the destination device. The IP address **142.250.71.46** belongs to Google's network, indicating that this packet is likely destined for a Google server, possibly related to YouTube.

Ethernet II (Layer 2 - Data Link Layer)

```

v Ethernet II, Src: Dell_42:ef:2d (d8:9e:f3:42:ef:2d), Dst: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)
  > Destination: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)
  > Source: Dell_42:ef:2d (d8:9e:f3:42:ef:2d)
  Type: IPv4 (0x0800)

```

- **Source MAC Address: Dell_42:ef:2d (d8:9e:f3:42:ef:2d)**
 - **Explanation:** This is the Media Access Control (MAC) address of the source device, likely your device (a Dell machine), which is sending the data packet.
- **Destination MAC Address: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)**
 - **Explanation:** This is the MAC address of the destination device, possibly a router or a switch made by Cisco.
- **Type: 0x0800**
 - **Explanation:** This hexadecimal value represents the protocol type of the payload. 0x0800 corresponds to IPv4, indicating that the next encapsulated protocol is Internet Protocol Version 4.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

Types of messages exchanged and handshaking mechanisms

3.1- Launch the application

When we launch the application, at first a DNS query is sent to IITG Server to get the IP address of the main Youtube servers. After that the server responds with the DNS reply and the IP address of the Youtube server is resolved. Followed by this, we observe the conversation, to see that there are 2 hand-shakings that occur.

- 3 way handshaking in the TCP protocol. Initially, the sender sends an [SYN] (Synchronise Sequence Number) message saying that the client is likely to start communication, and sends along with it the initial sequence number that the client is planning to use. Server responds by a [SYN, ACK] reply acknowledging the client and telling the client the initial sequence number the server plans to use. Finally the client acknowledges this also before the actual data transfer starts.

- 2 way TLS handshake in which the client and server exchange the keys to be used for communication, both the client and host computer agree upon an encryption method from the cipher suites to create keys and encrypt information.

166	4.686112	172.16.115.34	142.250.77.142	TCP	80 41184 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=173188751 TSecr=0 WS=128
177	4.750865	142.250.77.142	172.16.115.34	TCP	80 443 → 41184 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM TSval=1542293871 TSecr=173188751 WS=256
178	4.750884	172.16.115.34	142.250.77.142	TCP	72 41184 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=173188815 TSecr=1542293871

TCP 3 way Handshaking

179	4.752062	172.16.115.34	142.250.77.142	TLSv1.3	740 Client Hello (SNI=accounts.youtube.com)
186	4.816931	142.250.77.142	172.16.115.34	TCP	72 443 → 41184 [ACK] Seq=1 Ack=669 Win=65024 Len=0 TSval=1542293937 TSecr=173188816
187	4.849104	142.250.77.142	172.16.115.34	TLSv1.3	1472 Server Hello, Change Cipher Spec
188	4.849113	172.16.115.34	142.250.77.142	TCP	72 41184 → 443 [ACK] Seq=669 Ack=1401 Win=64128 Len=0 TSval=173188914 TSecr=1542293969

TLS 2-way handshake and exchange of keys for encryption

3.2- Play a video

We can observe that packets come in regular intervals of time and the client also sends ACK(acknowledgment) to the server in regular intervals of time. Here we can also notice that the Application data from the server is sent using TLS protocol(adding an extra layer of security) to the client while the ACK(acknowledgment) from the client to server is sent using normal TCP protocol.

209	4.916593	142.250.77.142	172.16.115.34	TLSv1.3	686 Application Data, Application Data
210	4.916594	142.250.77.142	172.16.115.34	TLSv1.3	103 Application Data
211	4.917031	172.16.115.34	142.250.77.142	TLSv1.3	103 Application Data
213	4.986688	142.250.77.142	172.16.115.34	TCP	72 443 → 41184 [ACK] Seq=7231 Ack=934 Win=65536 Len=0 TSval=1542294107 TSecr=173188981

Also a lot of packets are transmitted using QUIC protocol.

167	4.714930	142.250.77.142	172.16.115.34	QUIC	1405 Handshake, DCID=5e4734, SCID=fb2471677f659124
168	4.715072	172.16.115.34	142.250.77.142	QUIC	90 Handshake, DCID=fb2471677f659124, SCID=5e4734
169	4.715181	142.250.77.142	172.16.115.34	QUIC	1405 Handshake, DCID=5e4734, SCID=fb2471677f659124
170	4.715182	142.250.77.142	172.16.115.34	QUIC	1405 Handshake, DCID=5e4734, SCID=fb2471677f659124
171	4.715182	142.250.77.142	172.16.115.34	QUIC	177 Protected Payload (KP0), DCID=5e4734
172	4.716015	172.16.115.34	142.250.77.142	QUIC	90 Handshake, DCID=fb2471677f659124, SCID=5e4734
173	4.717588	172.16.115.34	142.250.77.142	QUIC	179 Protected Payload (KP0), DCID=fb2471677f659124
174	4.717596	172.16.115.34	142.250.77.142	QUIC	119 Protected Payload (KP0), DCID=fb2471677f659124
175	4.718034	172.16.115.34	142.250.77.142	QUIC	1405 Protected Payload (KP0), DCID=fb2471677f659124
176	4.718058	172.16.115.34	142.250.77.142	QUIC	1302 Protected Payload (KP0), DCID=fb2471677f659124

3.3- Pause a video

When we pause a video the packet rate decreases but there occurs some communication between server and client to keep the connection alive. Also at the time of pausing an ACK is sent keeping PSH bit as 1 indicating that it is paused.

188	4.849113	172.16.115.34	142.250.77.142	TCP	72 41184 → 443 [ACK] Seq=669 Ack=1401 Win=64128 Len=0 TSval=173188914 TSecr=1542293969
189	4.849355	142.250.77.142	172.16.115.34	TCP	1472 443 → 41184 [PSH, ACK] Seq=1401 Ack=669 Win=65536 Len=1400 TSval=1542293969 TSecr=173188816 [TCP segment of a reassembled PDU]
190	4.849355	142.250.77.142	172.16.115.34	TCP	1472 443 → 41184 [ACK] Seq=2801 Ack=669 Win=65536 Len=1400 TSval=1542293969 TSecr=173188816 [TCP segment of a reassembled PDU]

3.4-Downloading

We can see that there is a continuous packet exchange at a good rate between server and client. This is because application data of the file being downloaded is exchanged at a good rate.

3.5 Uploading

When uploading a video to YouTube, a TCP handshake occurs to establish a reliable connection between your device and YouTube's servers. Afterwards, data packets are exchanged between the client and server using TLS.

3.5-Skip the video to a particular part

The packet transmission rate has a sudden spike and takes some time, this is because we are loading a particular part not in sequence and it had to load that part. It takes time mostly due to the fact that it was not preloaded in cache as it does not belong to same sequence.

3.6-Closing the application

We can see a 3-way handshake . First client sends to the server keeping [FIN, ACK] on indicating that we can finish the conversation (FIN bit represents it). After that server acknowledges it by again sending a TCP message keeping [FIN, ACK] on. Then again client just acknowledges it keeping [ACK] on and connection is closed

306	53.058532	172.16.115.34	142.250.77.142	TCP	72	41184 → 443 [FIN, ACK] Seq=997 Ack=7231 Win=64128 Len=0 TSval=173237123 TSecr=1542294107
307	53.118426	142.250.77.142	172.16.115.34	TCP	72	443 → 41184 [ACK] Seq=7231 Ack=973 Win=65536 Len=0 TSval=1542342239 TSecr=173237118
308	53.123142	142.250.77.142	172.16.115.34	TCP	72	443 → 41184 [ACK] Seq=7231 Ack=997 Win=65536 Len=0 TSval=1542342244 TSecr=173237123

-> Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

1. Searching for Videos

- **HTTPS:** Encrypts your search queries and the responses from YouTube's servers, ensuring privacy and security during the search process.

2. Opening a Video

- **DNS:** Resolves YouTube's domain name into an IP address, allowing your device to connect to YouTube's servers.
- **TCP:** Establishes a reliable connection, ensuring the video data is correctly received and played without errors.
- **TLS:** Encrypts the connection, securing the video content and user data from interception.

3. Buffering a Video

- **HTTPS:** Securely streams video segments to your device, protecting the content and maintaining data integrity.
- **TCP:** Manages the flow of data, ensuring continuous and orderly delivery of video packets for smooth playback.

4. Downloading a Video

- **HTTPS:** Safeguards the download process, ensuring the video file is transmitted securely and without interference.
- **TCP:** Ensures all parts of the video file are correctly delivered, preventing data loss or corruption during download.

5. Closing the Application

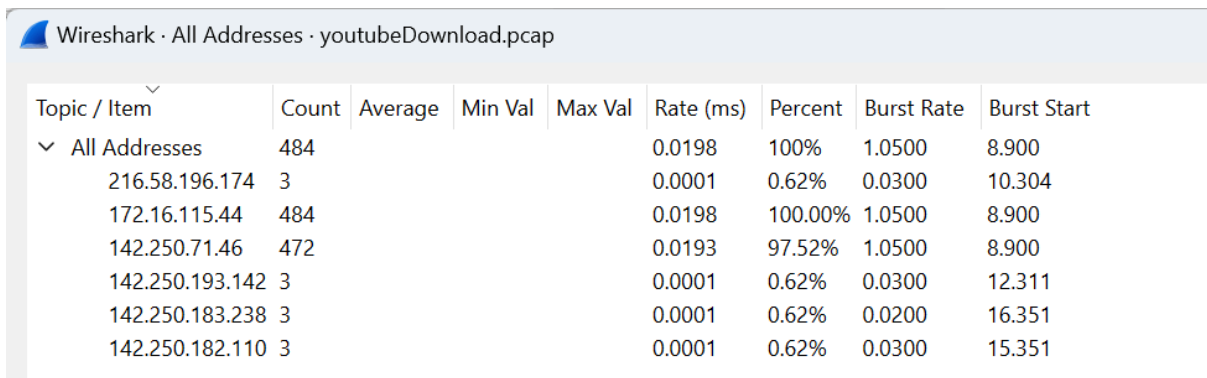
- **TCP:** Gracefully closed the connection with YouTube's servers, ensuring all data exchanges are completed before termination.

In summary, **HTTPS** secures your interactions with YouTube, **TCP** guarantees reliable delivery of video data, **TLS** ensures privacy through encryption, and **DNS** enables your device to connect to YouTube's servers efficiently.

-> Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

Time	Morning	Evening
Throughput (kilobytes/sec)	1165	1351
RTT (s)	0.04	0.07
Average Packet Size (Bytes)	1437	1390
No. of packet lost	0	0
No. of TCP Packets	2178	1982
No. of UDP Packets	593	589
No. of TLS Packets	2084	1843
No. of responses per request	2	2

-> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.



The screenshot shows the Wireshark interface with the 'All Addresses' pane selected. It displays a list of IP addresses and their associated metrics, including count, average, min/max values, rate, percent, burst rate, and burst start.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
✓ All Addresses	484				0.0198	100%	1.0500	8.900
216.58.196.174	3				0.0001	0.62%	0.0300	10.304
172.16.115.44	484				0.0198	100.00%	1.0500	8.900
142.250.71.46	472				0.0193	97.52%	1.0500	8.900
142.250.193.142	3				0.0001	0.62%	0.0300	12.311
142.250.183.238	3				0.0001	0.62%	0.0200	16.351
142.250.182.110	3				0.0001	0.62%	0.0300	15.351

The image displays several IP addresses along with metrics that seem to relate to packet counts, average rates, and percentages.

Multiple Content Sources

IP Addresses:

- **172.16.115.44**: This IP address has the highest packet count, indicating that the majority of the data was sent from this IP. This is likely the main server sending the content.
- **142.250.71.46**: This IP address also has a significant packet count and likely contributes to content delivery.
- The remaining IP addresses have very few packets (only 3 packets each), suggesting they might be used for auxiliary functions like buffering, analytics, or other minor tasks.

Explanation for Multiple Sources:

- **Content Delivery Networks (CDNs)**: YouTube uses CDNs, which are a network of servers distributed geographically to deliver content efficiently. The main reason for multiple IP addresses is that the video data might be coming from different servers in the CDN to optimise download speeds, balance load, and improve reliability.
- **Geo-distribution**: The different IP addresses might be serving different parts of the content or related functionalities like buffering or pre-loading, based on the user's location to minimise latency.
- **Load Balancing**: Traffic might be spread across multiple servers to ensure no single server is overwhelmed, leading to smoother streaming experiences.

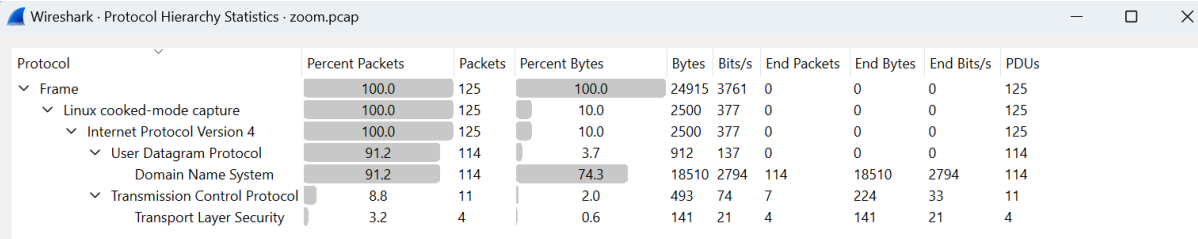
Conclusion:

The content is not being sent from a single source. Instead, multiple IP addresses are involved, which is a common practice for large-scale streaming services like YouTube. The different IP addresses correspond to servers within YouTube's CDN, ensuring efficient content delivery and improved performance.

2)ZOOM

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:



The image shows a screenshot of the Wireshark Protocol Hierarchy Statistics window for a file named 'zoom.pcap'. The window displays a tree view of protocols on the left and a corresponding table of statistics on the right. The protocols listed are Frame, Linux cooked-mode capture, Internet Protocol Version 4, User Datagram Protocol, Domain Name System, Transmission Control Protocol, and Transport Layer Security. The statistics table includes columns for Percent Packets, Packets, Percent Bytes, Bytes, Bits/s, End Packets, End Bytes, End Bits/s, and PDUs.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	125	100.0	24915	3761	0	0	0	125
Linux cooked-mode capture	100.0	125	10.0	2500	377	0	0	0	125
Internet Protocol Version 4	100.0	125	10.0	2500	377	0	0	0	125
User Datagram Protocol	91.2	114	3.7	912	137	0	0	0	114
Domain Name System	91.2	114	74.3	18510	2794	114	18510	2794	114
Transmission Control Protocol	8.8	11	2.0	493	74	7	224	33	11
Transport Layer Security	3.2	4	0.6	141	21	4	141	21	4

Layer 2 (Data Link):

- **Ethernet Frame** for data link encapsulation.
- **Linux cooked-mode capture** for capturing packets in certain environments, like loopback interfaces.

Layer 3 (Network):

- **IPv4** for routing packets across networks.

Layer 4 (Transport):

- **UDP** for fast, connectionless communication.
- **TCP** for reliable, connection-oriented communication.
- **TLS** for securing data transmission.

Layer 7 (Application):

- **DNS** for resolving domain names to IP addresses.
-

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

1. Linux Cooked Capture v2 (Data Link Layer)

```
✓ Linux cooked capture v2
  Protocol: IPv4 (0x0800)
  Interface index: 1
  Link-layer address type: Loopback (772)
  Packet type: Unicast to us (0)
  Link-layer address length: 6
  Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Unused: 0000
```

- **Interface index: 1:** Refers to the network interface from which the packet was captured, often the loopback interface for local traffic.
- **Link-layer address type: Loopback (772):** Indicates that the capture is from a loopback interface, typically used for communication within the host.
- **Packet type: Unicast to us (0):** Specifies that the packet is addressed to the capturing host.
- **Link-layer address length: 6:** The length of the MAC address in bytes.
- **Source: 00:00:00_00:00:00:** The source MAC address is shown as zeroed out, which is typical for loopback addresses.
- **Unused: 0000:** This field is not used in this context.

2. Internet Protocol Version 4 (Network Layer)

```
✓ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 68
    Identification: 0xc825 (51237)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0x744d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.53
```

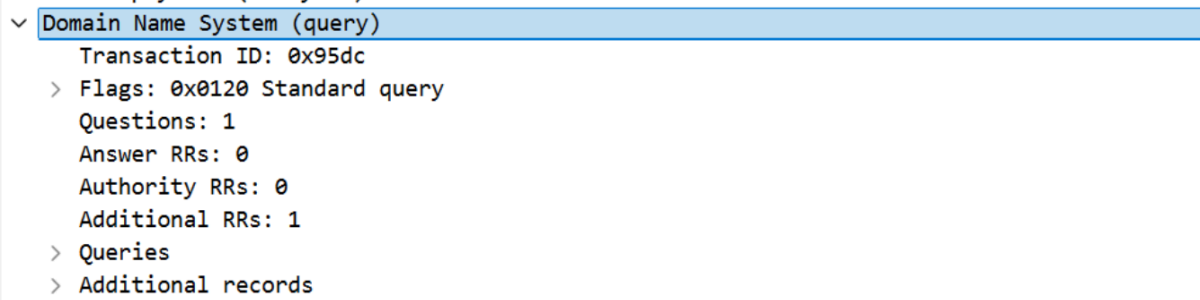
- **Version: 4:** Specifies that this is an IPv4 packet.
- **Header Length: 20 bytes:** The length of the IPv4 header, typically 20 bytes if there are no optional fields.
- **Differentiated Services Field: 0x00:** This field is used for QoS (Quality of Service), with the value indicating that no specific service is requested.
- **Total Length: 68:** Total size of the IP packet including the header and data.
- **Identification: 0xc825 (51237):** Unique ID assigned to the packet, used for reassembling fragmented packets.
- **Flags: 0x2, Don't fragment:** The packet has the "Don't Fragment" flag set, meaning it should not be fragmented during transmission.
- **Time to Live (TTL): 64:** This value decrements by one for each hop the packet takes, used to prevent infinite loops.
- **Protocol: UDP (17):** Indicates that the encapsulated protocol is UDP, with the protocol number 17.
- **Header Checksum: 0x744d:** Used to check the integrity of the header; validation is disabled in this case.
- **Source Address: 127.0.0.1:** The packet's source IP address, which is localhost in this case.
- **Destination Address: 127.0.0.53:** The packet's destination IP address, which is also a localhost address often used for DNS resolution in systemd-resolved setups.

3. User Datagram Protocol (Transport Layer)

```
✓ User Datagram Protocol, Src Port: 47189, Dst Port: 53
  Source Port: 47189
  Destination Port: 53
  Length: 48
  Checksum: 0xfe77 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 1]
  > [Timestamps]
  UDP payload (40 bytes)
```

- **Source Port: 47189:** The port number on the source machine that sent the packet.
- **Destination Port: 53:** The port number on the destination machine, which is typically used for DNS queries.
- **Length: 48:** Total length of the UDP packet, including the header and data.
- **Checksum: 0xfe77:** Used for error-checking the data; in this case, it is unverified.

4. Domain Name System (Application Layer)



- **Transaction ID: 0x95dc:** A unique ID assigned to the DNS query, used to match responses with queries.
- **Flags: 0x0120 Standard query:** This indicates that this is a standard DNS query.
- **Questions: 1:** The number of questions in the DNS query, which is one in this case.
- **Answer RRs: 0:** The number of answers in the DNS response, which is zero in this query (since it's just the query).
- **Authority RRs: 0:** The number of authoritative records, not applicable here.
- **Additional RRs: 1:** The number of additional records included, which is one in this case.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

3.1 Creating a Meeting

The client sends a DNS query to resolve the domain name of the Zoom server.

The DNS server responds with the IP address of the Zoom server. This is followed by a 2 way TLS handshake to establish a secure connection with the Zoom server.

3.2. Joining a Meeting

DNS resolution and TLS handshake takes place similarly like before. Afterwards, a lot of packets are transmitted using UDP and QUIC.

3.3 Chatting in a Meeting

A TLS handshaking sequence is initiated which ensures secure transmission of chat messages.

3.4 Quitting a Meeting

A 3 way TLS handshaking sequence is initiated which marks the end of communication between the server and the client.

-> Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

1. Creating a Meeting

- **Protocols Involved:** TLS, TCP, DNS, HTTP/HTTPS
- **Relevance:**
 - **TLS (Transport Layer Security):** Ensures secure communication when creating a meeting by encrypting data between the Zoom client and the server.
 - **TCP (Transmission Control Protocol):** Provides reliable communication for the exchange of data required to set up the meeting (e.g., sending meeting details to the server).
 - **DNS (Domain Name System):** Resolves the Zoom server's domain name to an IP address, allowing the client to connect to the server.
 - **HTTP/HTTPS (Application Layer):** Used to transmit the meeting creation request to the Zoom server securely over TLS.

2. Joining a Meeting

- **Protocols Involved:** TLS, TCP, UDP, DNS
- **Relevance:**
 - **TLS:** Secures the authentication process when a user joins a meeting.
 - **TCP:** Ensures the reliability of connection establishment and data exchange during the join process.
 - **UDP (User Datagram Protocol):** Used for real-time data transmission like voice and video, which prioritises speed over reliability.
 - **DNS:** Resolves the Zoom server's domain name to an IP address to connect to the meeting.

3. Chatting in a Meeting

- **Protocols Involved:** TLS, TCP, HTTPS
- **Relevance:**
 - **TLS:** Ensures that the chat messages are encrypted and secure between the client and server.
 - **TCP:** Provides reliable transmission of chat messages to ensure they are delivered in the correct order and without errors.
 - **HTTPS:** Securely transmits chat messages to and from the Zoom server.

4. Quitting a Meeting

- **Protocols Involved:** TLS, TCP, HTTPS
- **Relevance:**
 - **TLS:** Secures the communication when the client notifies the server that a user is leaving the meeting.
 - **TCP:** Ensures the reliable transmission of the quit request to the Zoom server.
 - **HTTPS:** Sends the quit request and any relevant data securely to the server, terminating the user's session in the meeting.

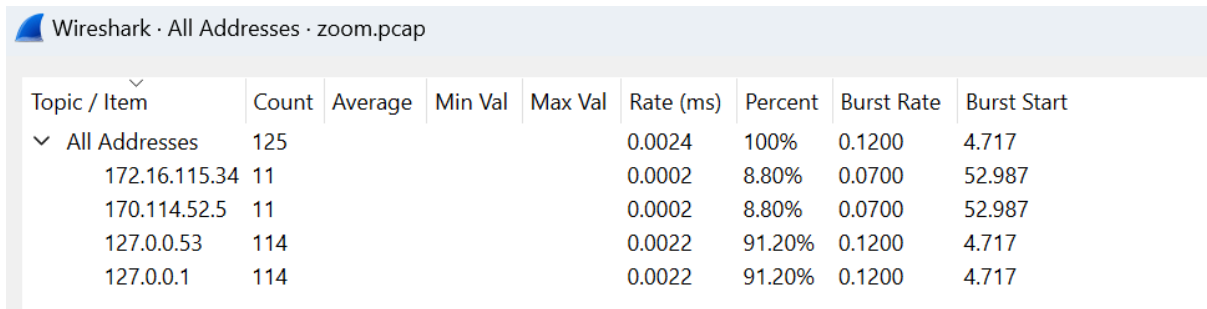
Summary:

Zoom relies on a combination of **TLS** for security, **TCP** for reliable communication, **UDP** and **RTP** for real-time data transmission (such as audio, video, and screen sharing), and **DNS** for resolving server addresses. These protocols ensure that Zoom can provide a secure, reliable, and real-time experience for all its functionalities, from creating and joining meetings to sharing screens and chatting.

-> Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

Time	Morning	Evening
Throughput (kilobytes/sec)	786	892
RTT (s)	0.07	0.09
Average Packet Size (Bytes)	1241	1348
No. of packet lost	0	0
No. of TCP Packets	1521	1862
No. of UDP Packets	5631	6459
No. of TLS Packets	1587	1793
No. of responses per request	2	2

 -> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.



Wireshark · All Addresses · zoom.pcap

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ All Addresses	125				0.0024	100%	0.1200	4.717
172.16.115.34	11				0.0002	8.80%	0.0700	52.987
170.114.52.5	11				0.0002	8.80%	0.0700	52.987
127.0.0.53	114				0.0022	91.20%	0.1200	4.717
127.0.0.1	114				0.0022	91.20%	0.1200	4.717

Analysis:

- **127.0.0.1 and 127.0.0.53:** These IP addresses refer to the localhost. The communication between these two addresses indicates internal communications within the same machine, likely related to DNS resolution or local services. They make up the majority of the packets (91.20% each), suggesting that a significant portion of the traffic is internal.
- **172.16.115.34 and 170.114.52.5:** These addresses represent external communication. They both have the same packet count and percentage (8.80% each), indicating that these could be related to external servers involved in the Zoom session, possibly different servers in Zoom's infrastructure or associated services.

Conclusion:

The content is not being sent from a single source but from multiple locations. The presence of multiple external IP addresses (172.16.115.34 and 170.114.52.5) suggests that Zoom is communicating with different servers, which is typical for cloud-based services like Zoom that rely on distributed Content Delivery Networks (CDNs) and multiple servers to manage load, enhance reliability, and optimise performance. The internal IPs (127.0.0.1 and 127.0.0.53) handle local processes, possibly related to DNS queries or loopback communications.

This setup allows Zoom to efficiently manage and distribute the workload, ensuring a smooth experience for users by utilising multiple sources and internal processing.

3)NPTEL video lectures

The analysis for NPTEL video lectures will be identical to that of youtube video playing in the previous section

4) Hotstar video streaming

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
Frame	100.0	191	100.0	107938	11 k	0	0	0	191
Linux cooked-mode capture	100.0	191		3820	391	0	0	0	191
Internet Protocol Version 4	100.0	191	3.5	3820	391	0	0	0	191
User Datagram Protocol	36.6	70	0.5	560	57	0	0	0	70
Domain Name System	36.6	70	14.7	15902	1629	70	15902	1629	70
Transmission Control Protocol	63.4	121	77.7	83836	8593	97	57900	5934	121
Transport Layer Security	12.6	24	74.1	79964	8196	24	79964	8196	24

ayer 2 (Data Link):

- **Ethernet Frame** for data link encapsulation.
- **Linux cooked-mode capture** for capturing packets in certain environments, like loopback interfaces.

Layer 3 (Network):

- **IPv4** for routing packets across networks.

Layer 4 (Transport):

- **UDP** for fast, connectionless communication.
- **TCP** for reliable, connection-oriented communication.
- **TLS** for securing data transmission.

Layer 7 (Application):

- **DNS** for resolving domain names to IP addresses.

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

1. Linux Cooked Capture v2

```
✓ Linux cooked capture v2
  Protocol: IPv4 (0x0800)
  Interface index: 1
  Link-layer address type: Loopback (772)
  Packet type: Unicast to us (0)
  Link-layer address length: 6
  Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Unused: 0000
```

- **Protocol:** IPv4 (0x0800)
 - Indicates that the captured packet is using the IPv4 protocol.
- **Interface index:** 1
 - This is the index of the network interface used to capture the packet.
- **Link-layer address type:** Loopback (772)
 - Specifies that the packet is from the loopback interface, indicating that it's a local communication on the host machine.
- **Packet type:** Unicast to us (0)
 - The packet is addressed directly to the host.
- **Link-layer address length:** 6
 - The length of the link-layer (MAC) address.
- **Source:** 00:00:00:00:00:00 (loopback address)
- **Unused:** 0000
 - These are usually padding bytes, not used in the protocol.

2. Internet Protocol Version 4 (IPv4)

```
✓ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 80
  Identification: 0x94bf (38079)
  > 010. .... = Flags: 0x2, Don't fragment
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 64
  Protocol: UDP (17)
  Header Checksum: 0xa7a7 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 127.0.0.1
  Destination Address: 127.0.0.53
```

- **Version:** 4
 - This field indicates that the packet is using the IPv4 protocol.
- **Header Length:** 20 bytes
 - The length of the IPv4 header is 20 bytes.

- **Differentiated Services Field:** `0x00` (DSCP: CS0, ECN: Not-ECT)
 - This field is used for Quality of Service (QoS) marking. `0x00` indicates that no special QoS features are applied.
- **Total Length:** 80 bytes
 - The total length of the IP packet, including both header and payload.
- **Identification:** `0x94bf` (38079)
 - A unique identifier for the IP packet, used in fragment reassembly.
- **Flags:** `0x2`, Don't fragment
 - The "Don't Fragment" flag is set, indicating that the packet should not be fragmented during transmission.
- **Time to Live (TTL):** 64
 - This field is decremented by one each time the packet passes through a router. If it reaches 0, the packet is discarded.
- **Protocol:** UDP (17)
 - Indicates that the encapsulated protocol is UDP.
- **Header Checksum:** `0x7a7` [validation disabled]
 - The checksum for the IPv4 header, used to detect errors in the header. Here, validation is disabled.
- **Source Address:** `127.0.0.1`
 - The IP address of the sender, which is the localhost.
- **Destination Address:** `127.0.0.53`
 - The IP address of the receiver, which is another localhost address used typically for DNS queries.

3. User Datagram Protocol (UDP)

```

Destination Host Port: 127.0.0.1:53
User Datagram Protocol, Src Port: 37652, Dst Port: 53
  Source Port: 37652
  Destination Port: 53
  Length: 60
  Checksum: 0xfe83 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 0]
  > [Timestamps]
  UDP payload (52 bytes)

```

- **Source Port:** 37652
 - The source port used by the sender application.
- **Destination Port:** 53
 - Port 53 is used by DNS, indicating that this packet is a DNS query.
- **Length:** 60 bytes
 - The length of the UDP header and payload.
- **Checksum:** `0xfe83` [unverified]
 - The checksum is used to validate the integrity of the UDP header and payload. It is marked as unverified in this capture.

4. Domain Name System (DNS) Query

- ✓ **Domain Name System (query)**
 - Transaction ID: 0xc1a7
 - > Flags: 0x0120 Standard query
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 1
 - > Queries
 - > Additional records
- **Transaction ID: 0xc1a7**
 - A unique identifier for this DNS query, used to match responses with requests.
- **Flags: 0x0120 Standard query**
 - This field includes flags that control the behaviour of the query. 0x0120 indicates a standard DNS query.
- **Questions: 1**
 - Indicates that there is one DNS question in this query.
- **Answer RRs: 0**
 - The number of answers in the DNS response, which is 0 at this point since it's a query.
- **Authority RRs: 0**
 - The number of authority records in the response, which is 0 in this query.
- **Additional RRs: 1**
 - Indicates the number of additional resource records, typically containing information like the address of the DNS server.
- **Response In: 5**
 - This indicates that the DNS response was captured in packet number 5.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

Types of messages exchanged and handshaking mechanisms

3.1- Launch the application

When we launch the application, at first a DNS query is sent to IITG Server to get the IP address of the main Hotstar servers. After that the server responds with the DNS reply and the IP address of the Hotstar server is resolved. Followed by this, we observe the conversation, to see that there are 2 hand-shakings that occur.

- 3 way handshaking in the TCP protocol. Initially, the sender sends an [SYN] (Synchronise Sequence Number) message saying that the client is likely to start communication, and sends along with it the initial sequence number that the client is planning to use. Server responds by a [SYN, ACK] reply acknowledging the client and telling the client the initial sequence number the server plans to use. Finally the client acknowledges this also before the actual data transfer starts.

- 2 way TLS handshake in which the client and server exchange the keys to be used for communication, both the client and host computer agree upon an encryption method from the cipher suites to create keys and encrypt information.

277	5.268200	172.20.10.3	23.58.95.227	TCP	66 63017 → 443 [SYN] Seq=0 Win=65280 Len=0 MSS=1360 WS=256 SACK_PERM
287	5.279462	23.58.95.227	172.20.10.3	TCP	66 443 → 63017 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1400 SACK_PERM WS=128
291	5.283416	172.20.10.3	23.58.95.227	TCP	54 63017 → 443 [ACK] Seq=1 Ack=1 Win=131840 Len=0

TCP 3 way Handshaking

295	5.283668	172.20.10.3	23.58.95.227	TLSv1.3	786 Client Hello (SHA1ing.hotstar.com)
315	5.311349	23.58.95.227	172.20.10.3	TCP	54 443 → 63017 [ACK] Seq=1 Ack=1361 Win=64128 Len=0
324	5.313062	23.58.95.227	172.20.10.3	TCP	54 443 → 63017 [ACK] Seq=1 Ack=2093 Win=63488 Len=0
325	5.313067	23.58.95.227	172.20.10.3	TLSv1.3	302 Server Hello, Change Cipher Spec, Application Data, Application Data

TLS 2-way handshake and exchange of keys for encryption

3.2- Play a video

We can observe that packets come in regular intervals of time and the client also sends ACK(acknowledgment) to the server in regular intervals of time. Here we can also notice that the Application data from the server is sent using TLS protocol(adding an extra layer of security) to the client while the ACK(acknowledgment) from the client to server is sent using normal TCP protocol.

49048	434.138565	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38789600 Win=4221440 Len=0
49049	434.138620	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38789600 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49050	434.138625	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38789600 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49051	434.138631	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38792320 Win=4221440 Len=0
49052	434.138658	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38792320 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49053	434.138724	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38793680 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49054	434.138730	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38795040 Win=4221440 Len=0
49055	434.138823	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38795040 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49056	434.138879	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38796400 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49057	434.138883	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [PSH, ACK] Seq=38797760 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49058	434.138934	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38799120 Win=4221440 Len=0
49059	434.138944	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38799120 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49060	434.138950	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38800480 Win=4221440 Len=0
49061	434.138987	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38800480 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49062	434.139030	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [PSH, ACK] Seq=38801840 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49063	434.139033	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38803200 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49064	434.139038	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38804560 Win=4221440 Len=0
49065	434.139090	23.36.177.155	172.20.10.3	TLSv1.3	1414 Application Data, Application Data
49066	434.139095	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38805920 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49067	434.139102	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38807280 Win=4221440 Len=0
49068	434.139130	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38807280 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49069	434.139137	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38808640 Win=4221440 Len=0
49070	434.139231	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38808640 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49071	434.139283	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38810000 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49072	434.139287	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38811360 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49073	434.139294	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38812720 Win=4221440 Len=0
49074	434.139336	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38812720 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49075	434.139388	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38814080 Win=4221440 Len=0
49076	434.139445	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38814080 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49077	434.139480	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38815440 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49078	434.139517	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38816800 Win=4221440 Len=0
49079	434.139531	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38816800 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49080	434.139536	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38818160 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49081	434.139546	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38819520 Win=4221440 Len=0

However, no QUIC packets were observed. This is in contrast to youtube video streaming where the majority of data was transferred using QUIC.

3.3- Pause a video

When we pause a video the packet rate decreases but there occurs some communication between server and client to keep the connection alive. Also at the time of pausing an ACK is sent keeping PSH bit as 1 indicating that it is paused.

49125	434.140944	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38850960 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49126	434.140950	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38860320 Win=4221440 Len=0
49127	434.140983	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [ACK] Seq=38860320 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49128	434.140988	23.36.177.155	172.20.10.3	TCP	1414 443 → 63126 [PSH, ACK] Seq=38861680 Ack=11957 Win=64128 Len=1360 [TCP segment of a reassembled PDU]
49129	434.140994	172.20.10.3	23.36.177.155	TCP	54 63126 → 443 [ACK] Seq=11957 Ack=38863040 Win=4221440 Len=0

3.5-Skip the video to a particular part

The packet transmission rate has a sudden spike and takes some time, this is because we are loading a particular part not in sequence and it had to load that part. It takes time mostly due to the fact that it was not preloaded in cache as it does not belong to same sequence.

3.6-Closing the application

We can see a 3-way handshake . First client sends to the server keeping [FIN, ACK] on indicating that we can finish the conversation (FIN bit represents it). After that server acknowledges it by again sending a TCP message keeping [FIN, ACK] on. Then again client just acknowledges it keeping [ACK] on and connection is closed

Time	Source	Destination	Protocol	Length	Info
1796.	1638.198013	172.20.10.1	TCP	54	53 → 63664 [FIN, ACK] Seq=106 Ack=41 Win=262144 Len=0
1796.	1638.198195	172.20.10.3	TCP	54	63664 → 53 [FIN, ACK] Seq=41 Ack=107 Win=131584 Len=0
1817.	1638.369895	172.20.10.3	TCP	54	63663 → 443 [FIN, ACK] Seq=4580 Ack=748 Win=131872 Len=0
1817.	1638.412949	23.58.120.72	TCP	54	443 → 63663 [FIN, ACK] Seq=772 Ack=4581 Win=64128 Len=0
1818.	1638.447259	172.20.10.3	TCP	54	63666 → 443 [FIN, ACK] Seq=5144 Ack=748 Win=131872 Len=0
1833.	1638.509538	23.58.120.72	TCP	54	443 → 63666 [FIN, ACK] Seq=772 Ack=5145 Win=64128 Len=0

-> Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

1. Searching for Videos

- **HTTPS:** Encrypts your search queries and the responses from YouTube's servers, ensuring privacy and security during the search process.

2. Opening a Video

- **DNS:** Resolves YouTube's domain name into an IP address, allowing your device to connect to YouTube's servers.
- **TCP:** Establishes a reliable connection, ensuring the video data is correctly received and played without errors.
- **TLS:** Encrypts the connection, securing the video content and user data from interception.

3. Buffering a Video

- **HTTPS:** Securely streams video segments to your device, protecting the content and maintaining data integrity.
- **TCP:** Manages the flow of data, ensuring continuous and orderly delivery of video packets for smooth playback.

4. Closing the Application

- **TCP:** Gracefully closes the connection with YouTube's servers, ensuring all data exchanges are completed before termination.

In summary, **HTTPS** secures your interactions with YouTube, **TCP** guarantees reliable delivery of video data, **TLS** ensures privacy through encryption, and **DNS** enables your device to connect to YouTube's servers efficiently.

-> Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

Time	Morning	Evening
Throughput (kilobytes/sec)	21	34
RTT (s)	32.4	28.1
Average Packet Size (Bytes)	154	205
No. of packet lost	0	0
No. of TCP Packets	1956	2084
No. of UDP Packets	342	367
No. of TLS Packets	673	651
No. of responses per request	2.14	2.78

-> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.

Wireshark · All Addresses · hotstar.pcap								
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ All Addresses	191				0.0024	100%	0.2600	64.843
184.86.248.170	121				0.0016	63.35%	0.2000	45.976
172.16.115.34	121				0.0016	63.35%	0.2000	45.976
127.0.0.53	70				0.0009	36.65%	0.0900	47.019
127.0.0.1	70				0.0009	36.65%	0.0900	47.019

Observed IP Addresses and Counts:

Explanation and Analysis:

- **184.86.248.170 and 172.16.115.34:**
 - These IP addresses have the highest packet counts (121 packets each), making up 63.35% of the total traffic.
 - **184.86.248.170** appears to be a public IP address likely associated with a content delivery network (CDN) or an external server.
 - **172.16.115.34** is a private IP address typically used in internal networks (RFC 1918). It might be associated with an internal device or a proxy server.

- **127.0.0.53 and 127.0.0.1:**
 - These addresses are loopback addresses, typically used for internal communications on the local machine.
 - **127.0.0.53** is often associated with systemd-resolved or DNS caching/resolving services on Linux systems.
 - **127.0.0.1** is the standard loopback address used to refer to the local machine itself.

Conclusion:

The traffic observed is not being sent entirely from the same location. Multiple sources are involved:

- **184.86.248.170** represents an external content provider, likely delivering parts of the requested content.
- **172.16.115.34** might be an internal proxy or server handling some requests internally.
- The loopback addresses (127.0.0.53 and 127.0.0.1) represent local communications, likely involving DNS queries or local services.

The existence of multiple sources can be due to:

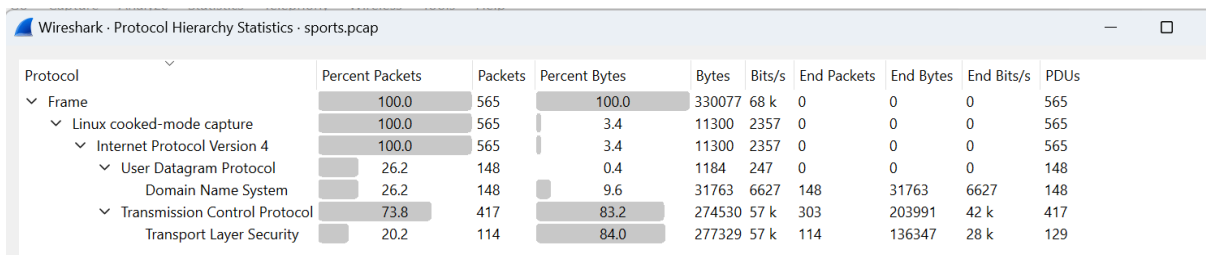
- **Content Delivery Networks (CDNs):** These networks use multiple servers to deliver content faster based on the user's geographical location.
- **Internal Routing:** Internal IPs like 172.16.115.34 may indicate the use of proxies, load balancers, or internal caching mechanisms to optimise content delivery.
- **Local DNS Caching:** The loopback addresses are involved in DNS resolution, helping to speed up repeated requests.

This setup helps optimise performance and reduce latency, ensuring content is delivered efficiently.

5)Live Sport Streaming

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
▼ Frame	100.0	565	100.0	330077	68 k	0	0	0	565
▼ Linux cooked-mode capture	100.0	565	3.4	11300	2357	0	0	0	565
▼ Internet Protocol Version 4	100.0	565	3.4	11300	2357	0	0	0	565
▼ User Datagram Protocol	26.2	148	0.4	1184	247	0	0	0	148
Domain Name System	26.2	148	9.6	31763	6627	148	31763	6627	148
▼ Transmission Control Protocol	73.8	417	83.2	274530	57 k	303	203991	42 k	417
Transport Layer Security	20.2	114	84.0	277329	57 k	114	136347	28 k	129

Layer 2 (Data Link):

- **Ethernet Frame** for data link encapsulation.
- **Linux cooked-mode capture** for capturing packets in certain environments, like loopback interfaces.

Layer 3 (Network):

- **IPv4** for routing packets across networks.

Layer 4 (Transport):

- **UDP** for fast, connectionless communication.
- **TCP** for reliable, connection-oriented communication.
- **TLS** for securing data transmission.

Layer 7 (Application):

- **DNS** for resolving domain names to IP addresses.

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

1. Linux Cooked Capture v2

```
✓ Linux cooked capture v2
  Protocol: IPv4 (0x0800)
  Interface index: 1
  Link-layer address type: Loopback (772)
  Packet type: Unicast to us (0)
  Link-layer address length: 6
  Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Unused: 0000
```

- **Protocol:** IPv4 (0x0800)
 - This field indicates that the packet uses the IPv4 protocol.
- **Interface index:** 1
 - The network interface index where this packet was captured.
- **Link-layer address type:** Loopback (772)
 - This shows that the packet originates from the loopback interface, used for internal communication within the same host.
- **Packet type:** Unicast to us (0)
 - The packet is specifically addressed to the host machine.
- **Link-layer address length:** 6
 - This indicates the length of the link-layer (MAC) address.
- **Source:** 00:00:00:00:00:00 (loopback address)
 - The source MAC address, which is a loopback address in this case.
- **Unused:** 0000
 - Padding bytes or fields not used by the protocol.

2. Internet Protocol Version 4 (IPv4)

```

v Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 70
    Identification: 0xa96b (43371)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0x9305 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.53

```

- **Version:** 4
 - Indicates that the packet is using the IPv4 protocol.
- **Header Length:** 20 bytes
 - The length of the IPv4 header.
- **Differentiated Services Field:** 0x00 (DSCP: CS0, ECN: Not-ECT)
 - This field is used for QoS (Quality of Service). 0x00 indicates default treatment without specific QoS features.
- **Total Length:** 70 bytes
 - The total size of the IP packet, including both header and payload.
- **Identification:** 0xa96b (43371)
 - A unique identifier for this packet, useful for reassembling fragmented packets.
- **Flags:** 0x2, Don't fragment
 - The "Don't Fragment" flag is set, indicating that the packet should not be fragmented.
- **Time to Live (TTL):** 64
 - This value decrements by one each time the packet traverses a router. It determines how many hops the packet can take before being discarded.
- **Protocol:** UDP (17)
 - The protocol field indicates that the encapsulated protocol is UDP.
- **Header Checksum:** 0x9305 [validation disabled]
 - This is the checksum for the IPv4 header, used to check for errors. Validation is disabled in this capture.
- **Source Address:** 127.0.0.1
 - The IP address of the source, which is the localhost.
- **Destination Address:** 127.0.0.53
 - The destination IP address, another localhost address often used for DNS queries.

3. User Datagram Protocol (UDP)

```

v User Datagram Protocol, Src Port: 57961, Dst Port: 53
  Source Port: 57961
  Destination Port: 53
  Length: 50
  Checksum: 0xfe79 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 7]
  > [Timestamps]
  UDP payload (42 bytes)

```

- **Source Port:** 57961
 - The port number from which the UDP packet is sent by the source application.

- **Destination Port:** 53
 - The destination port, which is 53, indicating that this packet is likely a DNS query.
- **Length:** 50 bytes
 - The length of the UDP header and payload.
- **Checksum:** 0xfe79 [unverified]
 - The checksum is used to validate the integrity of the UDP header and payload. It's marked as unverified in this capture.

4. Domain Name System (DNS) Query

```

v Domain Name System (query)
  Transaction ID: 0xfa9f
  > Flags: 0x0120 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 1
  > Queries
  > Additional records

```

- **Transaction ID:** 0xfa9f
 - A unique identifier for this DNS query, used to match responses to requests.
- **Flags:** 0x0120 Standard query
 - This field contains flags that control the behaviour of the DNS query. 0x0120 indicates a standard query.
- **Questions:** 1
 - There is one question in this DNS query.
- **Answer RRs:** 0
 - The number of answers in the DNS response is zero, as this is a query.
- **Authority RRs:** 0
 - The number of authority resource records is zero in this query.
- **Additional RRs:** 1
 - Indicates the number of additional resource records, typically including extra information such as the address of the DNS server.

Summary:

The captured packet is a DNS query sent from the localhost (127.0.0.1) to the local DNS resolver (127.0.0.53). It uses the UDP protocol, as indicated by the protocol field (17) and destination port (53). The packet is small, with a total length of 70 bytes, and has the "Don't Fragment" flag set. The loopback address and interface index confirm that this is an internal communication within the host machine.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

Types of messages exchanged and handshaking mechanisms

3.1- Starting Live Streaming

After initial DNS resolution, TCP handshaking is initiated to establish a connection. TLS handshaking also takes place to establish a secure connection

95	32.946456	172.16.115.34	23.206.173.74	TCP	80 42794 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2016595595 TSecr=0 WS=128
96	32.947898	23.206.173.74	172.16.115.34	TCP	80 443 → 42794 [SYN, ACK] Seq=0 Ack=1 Win=18328 Len=0 MSS=9176 SACK_PERM TSval=1323465679 TSecr=2016595595 WS=128
97	32.947915	172.16.115.34	23.206.173.74	TCP	72 42794 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2016595596 TSecr=1323465679

TCP 3 way Handshaking

98	32.948528	172.16.115.34	23.206.173.74	TLSv1.3	734 Client Hello (SNI=v3img.voot.com)
99	32.950602	23.206.173.74	172.16.115.34	TCP	72 443 → 42794 [ACK] Seq=1 Ack=663 Win=19712 Len=0 TSval=1323465679 TSecr=2016595597
100	33.188447	23.206.173.74	172.16.115.34	TLSv1.3	1520 Server Hello, Change Cipher Spec, Application Data
101	33.188460	172.16.115.34	23.206.173.74	TCP	72 42794 → 443 [ACK] Seq=663 Ack=1449 Win=64128 Len=0 TSval=2016595837 TSecr=1323465703

TLS 2-way handshake and exchange of keys for encryption

Afterwards we observe that there is a continuous packet exchange at a good rate between server and client

105	33.190588	172.16.115.34	23.206.173.74	TLSv1.3	242 Application Data
106	33.190600	172.16.115.34	23.206.173.74	TLSv1.3	427 Application Data
107	33.190908	23.206.173.74	172.16.115.34	TCP	72 443 → 42794 [ACK] Seq=3625 Ack=743 Win=19712 Len=0 TSval=1323465703 TSecr=2016595839
108	33.190908	23.206.173.74	172.16.115.34	TCP	72 443 → 42794 [ACK] Seq=3625 Ack=913 Win=20992 Len=0 TSval=1323465703 TSecr=2016595839
109	33.190909	23.206.173.74	172.16.115.34	TCP	72 443 → 42794 [ACK] Seq=3625 Ack=1268 Win=22400 Len=0 TSval=1323465703 TSecr=2016595839
110	33.252867	23.206.173.74	172.16.115.34	TLSv1.3	359 Application Data
111	33.253150	23.206.173.74	172.16.115.34	TLSv1.3	359 Application Data
112	33.253165	172.16.115.34	23.206.173.74	TCP	72 42794 → 443 [ACK] Seq=1268 Ack=4199 Win=64128 Len=0 TSval=2016595902 TSecr=1323465710
113	33.253677	23.206.173.74	172.16.115.34	TLSv1.3	133 Application Data
114	33.253677	23.206.173.74	172.16.115.34	TLSv1.3	103 Application Data

-> Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

1. Streaming the Live Content:

- **Protocols Used:** HTTP/HTTPS, HLS (HTTP Live Streaming) , UDP/TCP.
- **Relevance:**
 - **HLS/HTTP/HTTPS:** These protocols are essential for adaptive bitrate streaming. The streaming service typically uses HLS or DASH (Dynamic Adaptive Streaming over HTTP) to deliver video content in small segments. HTTP/HTTPS ensures that these segments are securely and reliably transmitted over the internet.
 - **UDP/TCP:** UDP is often used in real-time streaming because it provides lower latency than TCP, making it suitable for live content. TCP may still be used for control messages or ensuring reliability in certain parts of the stream delivery.

-> Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

Time	Morning	Evening
Throughput (kilobytes/sec)	54	57

RTT (s)	35.1	29.8
Average Packet Size (Bytes)	180	199
No. of packet lost	0	0
No. of TCP Packets	673	1124
No. of UDP Packets	97	138
No. of TLS Packets	184	210
No. of responses per request	4.12	4.4

-> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.

Wireshark · All Addresses · sports.pcap

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ All Addresses	565				0.0147	100%	1.2300	36.910
23.206.173.74	417				0.0109	73.81%	1.2300	36.910
172.16.115.34	417				0.0109	73.81%	1.2300	36.910
127.0.0.53	148				0.0039	26.19%	0.1800	30.618
127.0.0.1	148				0.0039	26.19%	0.1800	30.618

Explanation and Analysis:

- **23.206.173.74 and 172.16.115.34:**
 - **23.206.173.74** appears to be a public IP address, likely associated with a content delivery network (CDN) or an external server delivering the streaming content.
 - **172.16.115.34** is a private IP address, typically used within an internal network. This could represent an internal proxy, server, or device handling content requests within a local or corporate network.
 - Both addresses have identical packet counts (417), suggesting that the content is being served or processed equally by both the external CDN and an internal network entity.
- **127.0.0.53 and 127.0.0.1:**
 - These addresses are loopback addresses, used for internal communications within the same machine.
 - **127.0.0.53** is commonly used by DNS resolver services on modern Linux systems (like `systemd-resolved`) for DNS resolution.
 - **127.0.0.1** is the standard loopback address, often used for general internal communications, indicating that some of the network activities, such as DNS queries, are happening locally.

Conclusion:

The streaming content is not being sent from a single source. Instead, multiple sources are involved:

- **23.206.173.74** is likely an external CDN or streaming server delivering the content directly to the client.
- **172.16.115.34** suggests internal network involvement, potentially a proxy or internal server that is either caching or handling requests before they reach the external CDN.
- **127.0.0.53** and **127.0.0.1** are loopback addresses managing internal processes such as DNS queries and other local machine interactions.

Reason for Multiple Sources:

1. **Content Delivery Network (CDN):**
 - CDNs like the one represented by **23.206.173.74** are designed to distribute content efficiently across the globe. They reduce latency by serving content from a location geographically closer to the user.
2. **Internal Network Routing:**
 - The private IP address **172.16.115.34** indicates that part of the content or request handling is occurring within a local network, possibly for security (e.g., through a firewall or proxy) or performance reasons (caching).
3. **DNS Resolution and Local Communication:**
 - **127.0.0.53** and **127.0.0.1** handle DNS and other local services, ensuring that the application can efficiently resolve domain names and manage local resources without needing to communicate externally for every request.

This multi-source setup helps ensure that the content is delivered efficiently, securely, and with minimal latency, which is crucial for a smooth streaming experience, especially in live sports streaming scenarios.



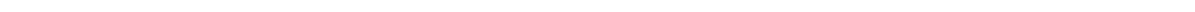
6)Microsoft Teams

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:

Wireshark · Protocol Hierarchy Statistics · msTeamsFinal.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
▼ Frame	100.0	347	100.0	172303	26 k	0	0	0	347
▼ Ethernet	100.0	347	2.8	4858	741	0	0	0	347
▼ Internet Protocol Version 4	100.0	347	4.0	6940	1058	0	0	0	347
▼ Transmission Control Protocol	100.0	347	93.2	160505	24 k	226	105696	16 k	347
Transport Layer Security	34.9	121	86.7	149407	22 k	121	146141	22 k	122



Layer 2 (Data Link):

- **Ethernet Frame** for data link encapsulation.

Layer 3 (Network):

- **IPv4** for routing packets across networks.

Layer 4 (Transport):

- **TCP** for reliable, connection-oriented communication.
- **TLS** for securing data transmission.

Layer 7 (Application):

- **DNS** for resolving domain names to IP addresses.

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

1. Ethernet II Layer

```

v Ethernet II, Src: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4), Dst: Dell_42:ef:2d (d8:9e:f3:42:ef:2d)
  > Destination: Dell_42:ef:2d (d8:9e:f3:42:ef:2d)
  > Source: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)
  Type: IPv4 (0x0800)

```

- **Source MAC Address:** **Cisco_cb:49:e4** (f8:0b:cb:cb:49:e4)
 - This is the MAC address of the source device, which is identified by the Cisco prefix, indicating the manufacturer of the network interface.
- **Destination MAC Address:** **Dell_42:ef:2d** (d8:9e:f3:42:ef:2d)
 - This is the MAC address of the destination device, which is identified by the Dell prefix, indicating that the destination device has a network interface made by Dell.
- **Type:** IPv4 (0x0800)
 - This field indicates that the payload of this Ethernet frame is an IPv4 packet.

2. Internet Protocol Version 4 (IPv4)

```

v Internet Protocol Version 4, Src: 52.123.129.14, Dst: 172.16.115.44
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x4948 (18760)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 62
    Protocol: TCP (6)
    Header Checksum: 0x590e [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 52.123.129.14
    Destination Address: 172.16.115.44

```

- **Version:** 4
 - This indicates that the packet is using the IPv4 protocol.
- **Header Length:** 20 bytes
 - The length of the IPv4 header, which in this case is 20 bytes.
- **Differentiated Services Field:** **0x00** (DSCP: CS0, ECN: Not-ECT)
 - This field is used for Quality of Service (QoS) settings, though in this packet, the DSCP and ECN fields are not actively used.
- **Total Length:** 1500 bytes
 - The total size of the IPv4 packet, including both the header and the payload.
- **Identification:** **0x4948** (18760)

- This is a unique identifier for the packet, used for fragment reassembly in case the packet is fragmented during transmission.
- **Flags:** 0x0
 - In this packet, the flags are set to 0, indicating no fragmentation is needed.
- **Time to Live (TTL):** 62
 - The TTL value, which decrements by one each time the packet traverses a router, indicating how many hops this packet can take before being discarded.
- **Protocol:** TCP (6)
 - The protocol field indicates that the encapsulated protocol is TCP.
- **Header Checksum:** 0x590e [validation disabled]
 - This is the checksum for the IPv4 header, used to detect errors in the header. Validation is disabled in this capture.
- **Source Address:** 52.123.129.14
 - The IP address of the source, which is an external IP address potentially associated with a web server or other internet service.
- **Destination Address:** 172.16.115.44
 - The IP address of the destination, which is a private IP address likely within an internal network.

3. Transmission Control Protocol (TCP)

```

✓ Transmission Control Protocol, Src Port: 443, Dst Port: 41062, Seq: 13233, Ack: 70038, Len: 1448
  Source Port: 443
  Destination Port: 41062
  [Stream index: 0]
  > [Conversation completeness: Incomplete (12)]
  [TCP Segment Len: 1448]
  Sequence Number: 13233 (relative sequence number)
  Sequence Number (raw): 2328399790
  [Next Sequence Number: 14681 (relative sequence number)]
  Acknowledgment Number: 70038 (relative ack number)
  Acknowledgment number (raw): 1227790885
  1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x010 (ACK)
  Window: 1215
  [Calculated window size: 1215]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0xde2a [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [Timestamps]
  > [SEQ/ACK analysis]
  TCP payload (1448 bytes)
  [Reassembled PDU in frame: 186]
  TCP segment data (1448 bytes)

```

- **Source Port:** 443
 - The source port used by the sender, which is port 443, typically associated with HTTPS, indicating secure web traffic.
- **Destination Port:** 41062
 - The destination port on the receiving side, which is dynamically assigned by the client's operating system.
- **Sequence Number:** 13233 (relative sequence number)
 - The sequence number indicates the position of the first data byte in this segment within the overall TCP data stream.
- **Acknowledgment Number:** 70038 (relative acknowledgment number)
 - This is the acknowledgment number, which tells the sender that the receiver has successfully received up to this byte in the data stream.
- **Flags:** 0x010 (ACK)
 - The ACK flag is set, indicating that this packet is acknowledging the receipt of data.

- **Window Size:** 1215
 - The window size field, which specifies the size of the receiver's buffer space available. It controls the flow of data and helps prevent congestion.
- **Checksum:** 0xde2a [unverified]
 - The checksum for the TCP segment, used to ensure data integrity. It's marked as unverified in this capture.
- **TCP Payload Length:** 1448 bytes
 - The length of the data payload carried by this TCP segment.
- **Reassembled PDU in frame:** 186
 - This field indicates that the data from this TCP segment has been reassembled in a later frame (frame 186).

Summary:

- **Ethernet Layer:** The source and destination MAC addresses indicate communication between devices made by Cisco and Dell, respectively, over an IPv4 network.
- **IP Layer:** The packet originates from an external IP address (possibly a web server) and is destined for an internal network device (private IP). The packet is carrying TCP data.
- **TCP Layer:** The source port is 443, indicating HTTPS traffic, while the destination port is dynamically assigned. The packet is part of an ongoing TCP connection, as evidenced by the sequence and acknowledgment numbers, and it carries 1448 bytes of data.

These fields collectively show a secure web communication (likely HTTPS) between an external server and an internal client within a private network.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

3.1 Video/Voice Calling

The DNS server responds with the IP address of the Teams server. This is followed by a 3 way TCP handshake and a 2 way TLS handshake to establish a secure connection with the Teams server. Afterwards, a lot of packets are transmitted using UDP. The client sends a DNS query to resolve the domain name of the MS teams server.

3.2 Chatting

A TLS handshaking sequence is initiated which ensures secure transmission of chat messages. Afterwards the packets are shared using TLS to ensure reliability.

3.3 Quitting a Meeting

A 3 way TLS handshaking sequence is initiated which marks the end of communication between the server and the client

-> Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

1. Chatting (Instant Messaging)

- **Protocols Used:** HTTPS, TLS, WebSockets.
- **Relevance:**
 - **HTTPS and TLS:** All communication, including chat messages, is encrypted using HTTPS (Hypertext Transfer Protocol Secure) and TLS (Transport Layer Security). This ensures that messages are securely transmitted and cannot be intercepted or tampered with.
 - **WebSockets:** WebSockets enable real-time communication between clients and servers, allowing for instant delivery of messages without the need for repeated polling.

2. Video/Voice Calling

- **Protocols Used:** RTP (Real-Time Transport Protocol), SRTP (Secure Real-Time Transport Protocol),
- **Relevance:**
 - **RTP and SRTP:** Video streams are transmitted using RTP, while SRTP provides encryption, ensuring that video content is securely transmitted.
 - **HTTPS and TLS:** All signalling traffic, including call setup and participant management, is encrypted to protect against eavesdropping and unauthorized access.

Summary:

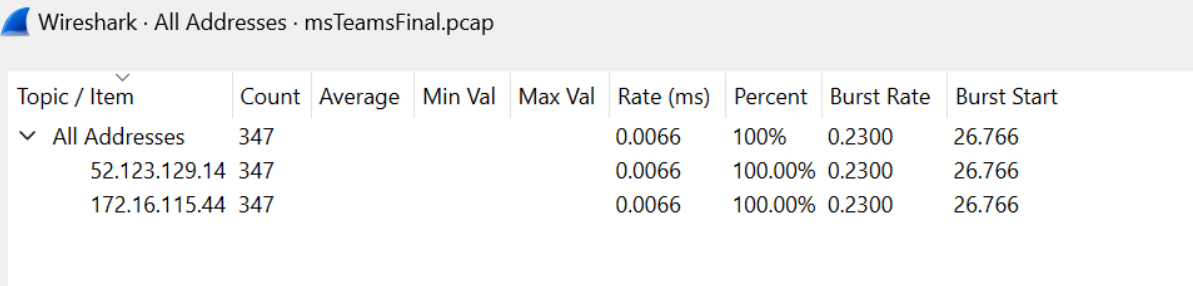
Each protocol used by Microsoft Teams is critical for enabling its wide range of features in a secure, reliable, and efficient manner. From secure communication (HTTPS, TLS) to real-time interaction (WebSockets, RTP/SRTP), these protocols ensure that users can collaborate, communicate, and share information seamlessly across various devices and environments. By understanding how these protocols work together, we gain insight into how Teams manages to deliver a robust and integrated user experience.

-> Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

Time	Morning	Evening
Throughput (kilobytes/sec)	545	673
RTT (s)	0.04	0.06
Average Packet Size (Bytes)	1121	1233
No. of packet lost	0	0
No. of TCP Packets	1276	1349
No. of UDP Packets	6591	6372
No. of TLS Packets	1416	1673

No. of responses per request	2	2
------------------------------	---	---

-> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.



Wireshark · All Addresses · msTeamsFinal.pcap

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ All Addresses	347				0.0066	100%	0.2300	26.766
52.123.129.14	347				0.0066	100.00%	0.2300	26.766
172.16.115.44	347				0.0066	100.00%	0.2300	26.766

Explanation:

1. Single Source vs. Multiple Sources:

- The presence of two distinct IP addresses suggests that the content or communication is being exchanged between these two endpoints. This is typical in a network capture where data is transferred between a client and a server or between two peers.

2. IP Address Details:

- **52.123.129.14:** This appears to be a public IP address, which might be associated with an external server or service (potentially part of a content delivery network or a cloud service provider).
- **172.16.115.44:** This is a private IP address, which is likely associated with a local device on the internal network. Private IP addresses are typically used within local networks and are not routable on the public internet.

Conclusion:

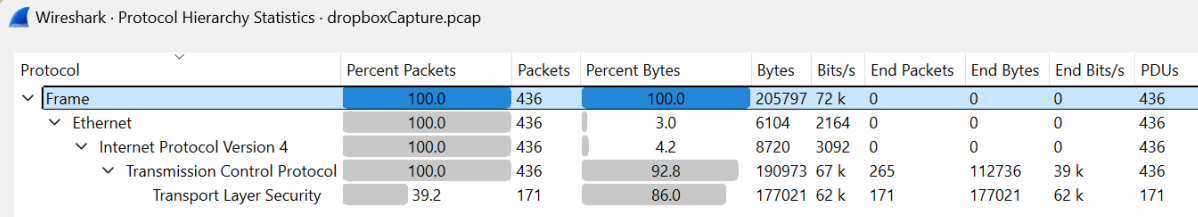
The content appears to be exchanged between a local device (172.16.115.44) and an external service or server (52.123.129.14). This setup is common in scenarios where a client on an internal network is connecting to an external service, such as Microsoft Teams. The communication seems to originate from these two sources, indicating that data is being sent and received between them.

If you're investigating the content delivery or suspecting issues related to multiple sources, this setup does not indicate multiple independent content providers but rather a typical client-server interaction.

7)Dropbox

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:



Wireshark · Protocol Hierarchy Statistics · dropboxCapture.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
Frame	100.0	436	100.0	205797	72 k	0	0	0	436
Ethernet	100.0	436	3.0	6104	2164	0	0	0	436
Internet Protocol Version 4	100.0	436	4.2	8720	3092	0	0	0	436
Transmission Control Protocol	100.0	436	92.8	190973	67 k	265	112736	39 k	436
Transport Layer Security	39.2	171	86.0	177021	62 k	171	177021	62 k	171

Layer 2 (Data Link):

- **Ethernet Frame** for data link encapsulation.

Layer 3 (Network):

- **IPv4** for routing packets across networks.

Layer 4 (Transport):

- **TCP** for reliable, connection-oriented communication.
- **TLS** for securing data transmission.

Layer 7 (Application):

- **DNS** for resolving domain names to IP addresses.

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

Ethernet II:

```
▼ Ethernet II, Src: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4), Dst: Dell_42:ef:2d (d8:9e:f3:42:ef:2d)
  > Destination: Dell_42:ef:2d (d8:9e:f3:42:ef:2d)
  > Source: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)
  Type: IPv4 (0x0800)
```

- **Source MAC Address:** `Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)`
 - This is the MAC address of the device sending the packet. The prefix "f8:0b" indicates the manufacturer (Cisco in this case).
- **Destination MAC Address:** `Dell_42:ef:2d (d8:9e:f3:42:ef:2d)`
 - This is the MAC address of the device intended to receive the packet. The prefix "d8:9e" suggests it is a Dell device.
- **Type:** `IPv4 (0x0800)`
 - This indicates that the Ethernet frame is carrying an IPv4 packet.

Internet Protocol Version 4 (IP Layer):

```
Internet Protocol Version 4, Src: 162.125.69.18, Dst: 172.16.115.44
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 52
  Identification: 0xd177 (53623)
  > 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 62
  Protocol: TCP (6)
  Header Checksum: 0xa480 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 162.125.69.18
  Destination Address: 172.16.115.44
```

- **Source IP Address: 162.125.69.18**
 - This is the IP address of the device that sent the packet, likely a server given the address range.
- **Destination IP Address: 172.16.115.44**
 - This is the IP address of the device receiving the packet. The 172.16.x.x address is a private IP address, indicating it belongs to an internal network.
- **Time to Live (TTL): 62**
 - TTL is a field that indicates the remaining lifespan of the packet. It starts at a certain number (commonly 64, 128, or 255) and is decremented by one each time the packet passes through a router. A TTL of 62 suggests this packet has traversed 2 routers so far.
- **Protocol: TCP (6)**
 - This indicates that the packet is encapsulating a TCP segment.
- **Header Checksum: 0xa480 [validation disabled]**
 - This is the checksum used to verify the integrity of the IP header. Validation is disabled in this capture, so the checksum status is unverified.

TCP Layer:

```
Transmission Control Protocol, Src Port: 443, Dst Port: 37124, Seq: 1, Ack: 3991, Len: 0
  Source Port: 443
  Destination Port: 37124
  [Stream index: 0]
  > [Conversation completeness: Incomplete (12)]
  [TCP Segment Len: 0]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 3086481830
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 3991 (relative ack number)
  Acknowledgment number (raw): 337080376
  1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x010 (ACK)
  Window: 1213
  [Calculated window size: 1213]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0xcf32 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [Timestamps]
  > [SEQ/ACK analysis]
```

- **Source Port: 443**
 - This is the source port of the TCP segment. Port 443 is commonly used for HTTPS traffic, indicating encrypted web communication.
- **Destination Port: 37124**
 - This is the destination port on the client side, which is dynamically assigned by the client's operating system when establishing the connection.
- **Sequence Number: 1**

- This is the initial sequence number of the TCP segment, indicating the first byte in the TCP segment.
- **Acknowledgment Number: 3991**
 - This number acknowledges the receipt of a specific number of bytes from the sender. It indicates that the receiver has successfully received all data up to byte 3990 and is expecting the next byte.
- **Flags:**
 - **ACK (0x010)** - The Acknowledgment flag indicates that the acknowledgment number field is significant, meaning this packet is acknowledging receipt of data.
- **Window Size: 1213**
 - The window size indicates how many bytes the sender is willing to receive beyond the acknowledged bytes. It's part of TCP's flow control mechanism.
- **Checksum: 0xcf32 [unverified]**
 - This is the checksum used to verify the integrity of the TCP segment. Again, it's marked as unverified in this capture.

Summary:

The packet captures show communication between two devices, where the source IP **162.125.69.18** (likely an external server) is sending data to a local device with the IP **172.16.115.44**. The Ethernet frame shows the communication between the MAC addresses of the devices. The use of port 443 indicates secure communication (HTTPS), and the presence of TCP flags, sequence numbers, and acknowledgments suggests that this is part of an established TCP connection.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

3.1 File Sharing

The client sends a DNS query to resolve the domain name of the Dropbox server.

The DNS server responds with the IP address of the Dropbox server. This is followed by a 3 way TCP handshake and a 2 way TLS handshake to establish a secure connection with the Dropbox server.

Afterwards, for file uploading as well as downloading ,we have observed that majority of the data transmission takes through TCP (encrypted using TLS).

419	21.309910	172.16.115.44	162.125.69.18	TLSv1.2	439 Application Data
420	21.310491	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42003 Ack=126123 Win=1230 Len=0 TSval=1323566325 TSecr=168808344
421	21.310745	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42003 Ack=129019 Win=1230 Len=0 TSval=1323566325 TSecr=168808344
422	21.310745	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42003 Ack=130840 Win=1230 Len=0 TSval=1323566325 TSecr=168808344
423	21.681032	162.125.69.18	172.16.115.44	TLSv1.2	194 Application Data
424	21.681075	172.16.115.44	162.125.69.18	TCP	66 37124 → 443 [ACK] Seq=130840 Ack=42131 Win=753 Len=0 TSval=168808715 TSecr=1323566362
425	22.218359	172.16.115.44	162.125.69.18	TLSv1.2	938 Application Data
426	22.218392	172.16.115.44	162.125.69.18	TLSv1.2	251 Application Data
427	22.218958	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42131 Ack=131897 Win=1230 Len=0 TSval=1323566416 TSecr=168809252
428	22.220328	172.16.115.44	162.125.69.18	TCP	1514 37124 → 443 [ACK] Seq=131897 Ack=42131 Win=753 Len=1448 TSval=168809254 TSecr=1323566416 [TCP segment of a reassembled PDU]
429	22.220330	172.16.115.44	162.125.69.18	TLSv1.2	292 Application Data
430	22.220378	172.16.115.44	162.125.69.18	TLSv1.2	1156 Application Data

-> Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

File Sharing

- **Relevant Protocols:** HTTPS, TLS/TCP
- **Explanation:**
 - Dropbox allows users to share files and collaborate with others. When sharing a file or folder, HTTPS ensures that the shared links and access permissions are securely transmitted and stored.
 - TCP (Transmission Control Protocol) is used in applications like Dropbox primarily because of its reliability and the need for accurate data transmission. TCP is essential for Dropbox and similar file-sharing applications because of Reliable Data Transmission, Data Integrity and Orderly Data Transfer.

-> Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

Time	Morning	Evening
Throughput (kilobytes/sec)	1220	1121
RTT (s)	434	562
Average Packet Size (Bytes)	780	813
No. of packet lost	3	6
No. of TCP Packets	87	98
No. of UDP Packets	0	0
No. of TLS Packets	43	56
No. of responses per request	3.1	2.6

-> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.

Wireshark · All Addresses · dropboxCapture.pcap								
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ All Addresses	436				0.0193	100%	0.5100	16.172
172.16.115.44	436				0.0193	100.00%	0.5100	16.172
162.125.69.18	436				0.0193	100.00%	0.5100	16.172

Analysis:

Single Source vs. Multiple Sources:

- The presence of two distinct IP addresses suggests that the content or communication is being exchanged between these two endpoints. This is typical in a network capture where data is transferred between a client and a server or between two peers.

IP Address Details:

- **172.16.115.44**: This is a private IP address, which is likely associated with a local device on the internal network. Private IP addresses are typically used within local networks and are not routable on the public internet.
- **162.125.69.18**: This is a public IP address, which might be associated with Dropbox's servers or a content delivery network (CDN) used by Dropbox to store or manage files.

Conclusion:

The content in this capture appears to be exchanged between a local device (172.16.115.44) and an external server (162.125.69.18). This setup is common in scenarios where a client on an internal network (e.g., a user's computer) is connecting to an external service (e.g., Dropbox's cloud servers).

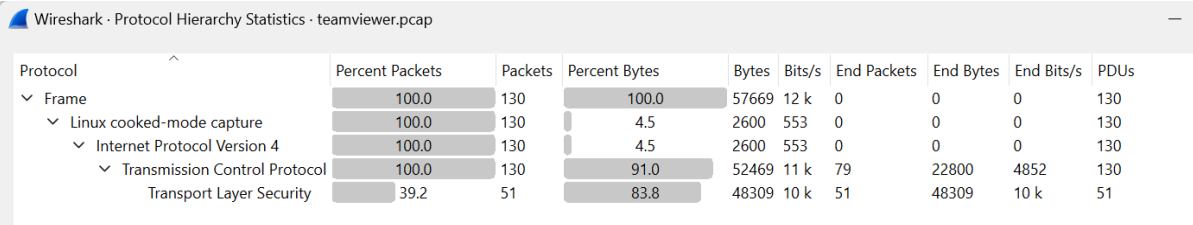
The fact that both IP addresses have identical statistics (packet count, rate, etc.) suggests that all content is being sent and received between these two IPs, implying that the content originates from and is directed to these two endpoints only. There are no additional content sources or destinations involved in this specific communication, indicating that Dropbox is handling the data exchange directly between the user's device and the Dropbox server.

This direct connection is typical for Dropbox operations, where files are uploaded, downloaded, or synchronised between a user's device and Dropbox's cloud storage.

8) P2P connectivity using Remote desktop and softwares like Teamviewer

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:



The image shows a screenshot of the Wireshark Protocol Hierarchy Statistics window for a file named 'teamviewer.pcap'. The window displays a tree view of protocols and their corresponding statistics. The protocols listed are Frame, Linux cooked-mode capture, Internet Protocol Version 4, Transmission Control Protocol, and Transport Layer Security. The statistics for each protocol are shown in a table format with columns for Percent Packets, Packets, Percent Bytes, Bytes, Bits/s, End Packets, End Bytes, End Bits/s, and PDUs.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	130	100.0	57669	12 k	0	0	0	130
Linux cooked-mode capture	100.0	130	4.5	2600	553	0	0	0	130
Internet Protocol Version 4	100.0	130	4.5	2600	553	0	0	0	130
Transmission Control Protocol	100.0	130	91.0	52469	11 k	79	22800	4852	130
Transport Layer Security	39.2	51	83.8	48309	10 k	51	48309	10 k	51

ayer 2 (Data Link):

- **Ethernet Frame** for data link encapsulation.
- **Linux cooked-mode capture** for capturing packets in certain environments, like loopback interfaces.

Layer 3 (Network):

- **IPv4** for routing packets across networks.

Layer 4 (Transport):

- **UDP** for fast, connectionless communication.
- **TCP** for reliable, connection-oriented communication.
- **TLS** for securing data transmission.

Layer 7 (Application):

- **DNS** for resolving domain names to IP addresses.

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

Linux Cooked Capture (LCC)

```
✓ Linux cooked capture v2
  Protocol: IPv4 (0x0800)
  Interface index: 2
  Link-layer address type: Ethernet (1)
  Packet type: Unicast to us (0)
  Link-layer address length: 6
  Source: Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)
  Unused: 0000
```

- **Protocol:** IPv4 (0x0800)
 - This indicates that the packet being captured is an IPv4 packet.
- **Link-layer address type:** Ethernet (1)
 - This specifies the type of link-layer protocol used, which is Ethernet in this case.
- **Packet Type:** Unicast to us (0)
 - The packet is a unicast, meaning it is intended for a single specific destination.
- **Link-layer address length:** 6
 - Indicates the length of the MAC address in bytes.
- **Source Link-layer Address:** Cisco_cb:49:e4 (f8:0b:cb:cb:49:e4)
 - The MAC address of the device sending the packet, likely a Cisco device given the manufacturer prefix "f8:0b".

Internet Protocol Version 4 (IP Layer)

Internet Protocol Version 4, Src: 52.233.254.206, Dst: 172.16.115.34

0100 = Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 52
Identification: 0x90c9 (37065)
> 000. = Flags: 0x0
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 62
Protocol: TCP (6)
Header Checksum: 0x9910 [validation disabled]
[Header checksum status: Unverified]
Source Address: 52.233.254.206
Destination Address: 172.16.115.34

- **Source IP Address: 52.233.254.206**
 - This is the IP address of the device that sent the packet, likely an external server, possibly related to a cloud service provider.
- **Destination IP Address: 172.16.115.34**
 - This is the IP address of the device receiving the packet. The 172.16.x.x address is a private IP address, indicating it belongs to an internal network.
- **Time to Live (TTL): 62**
 - TTL is a field that indicates the remaining lifespan of the packet. It starts at a certain number (commonly 64, 128, or 255) and is decremented by one each time the packet passes through a router. A TTL of 62 suggests this packet has traversed 2 routers so far.
- **Protocol: TCP (6)**
 - This indicates that the packet is encapsulating a TCP segment.
- **Header Checksum: 0x9910 [validation disabled]**
 - This is the checksum used to verify the integrity of the IP header. Validation is disabled in this capture, so the checksum status is unverified.

TCP Layer:

Transmission Control Protocol, Src Port: 443, Dst Port: 36038, Seq: 389, Ack: 4240, Len: 0

Source Port: 443
Destination Port: 36038
[Stream index: 0]
> [Conversation completeness: Incomplete (12)]
[TCP Segment Len: 0]
Sequence Number: 389 (relative sequence number)
Sequence Number (raw): 2924327079
[Next Sequence Number: 389 (relative sequence number)]
Acknowledgment Number: 4240 (relative ack number)
Acknowledgment number (raw): 1752882939
1000 = Header Length: 32 bytes (8)
> Flags: 0x010 (ACK)
Window: 1213
[Calculated window size: 1213]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x6cb5 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [Timestamps]
> [SEQ/ACK analysis]

- **Source Port: 443**
 - This is the source port of the TCP segment. Port 443 is commonly used for HTTPS traffic, indicating encrypted web communication.
- **Destination Port: 36038**

- This is the destination port on the client side, which is dynamically assigned by the client's operating system when establishing the connection.
- **Sequence Number: 389**
 - This is the sequence number of the TCP segment, indicating the relative position of the first byte of data in this segment within the overall sequence of bytes sent.
- **Acknowledgment Number: 4240**
 - This number acknowledges the receipt of a specific number of bytes from the sender. It indicates that the receiver has successfully received all data up to byte 4240 and is expecting the next byte.
- **Flags:**
 - **ACK (0x010)** - The Acknowledgment flag indicates that the acknowledgment number field is significant, meaning this packet is acknowledging receipt of data.
- **Window Size: 1213**
 - The window size indicates how many bytes the sender is willing to receive beyond the acknowledged bytes. It's part of TCP's flow control mechanism.
- **Checksum: 0x6cb5 [unverified]**
 - This is the checksum used to verify the integrity of the TCP segment. Again, it's marked as unverified in this capture.

Summary:

The packet captures show communication between two devices, where the source IP **52.233.254.206** (likely an external server) is sending data to a local device with the IP **172.16.115.34**. The Ethernet frame shows the communication between the MAC addresses of the devices. The use of port 443 indicates secure communication (HTTPS), and the presence of TCP flags, sequence numbers, and acknowledgments suggests that this is part of an established TCP connection.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

3.1 Remote Desktop Access/ File transfer

The client sends a DNS query to resolve the domain name of the Teamviewer server. The DNS server responds with the IP address of the Teamviewer server. This is followed by a 3 way TCP handshake and a 2 way TLS handshake to establish a secure connection with the Teamviewer server. Afterwards, for file uploading as well as downloading, we have observed that the majority of the data transmission takes place through TCP (encrypted using TLS).

419	21.309910	172.16.115.44	162.125.69.18	TLsv1.2	439 Application Data
420	21.310491	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42003 Ack=126123 Win=1230 Len=0 TSval=1323566325 TSecr=168808344
421	21.310745	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42003 Ack=129019 Win=1230 Len=0 TSval=1323566325 TSecr=168808344
422	21.310745	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42003 Ack=130840 Win=1230 Len=0 TSval=1323566325 TSecr=168808344
423	21.681032	162.125.69.18	172.16.115.44	TLsv1.2	194 Application Data
424	21.681075	172.16.115.44	162.125.69.18	TCP	66 37124 → 443 [ACK] Seq=130840 Ack=42131 Win=753 Len=0 TSval=168808715 TSecr=1323566362
425	22.218359	172.16.115.44	162.125.69.18	TLsv1.2	938 Application Data
426	22.218392	172.16.115.44	162.125.69.18	TLsv1.2	251 Application Data
427	22.218958	162.125.69.18	172.16.115.44	TCP	66 443 → 37124 [ACK] Seq=42131 Ack=131897 Win=1230 Len=0 TSval=1323566416 TSecr=168809252
428	22.220328	172.16.115.44	162.125.69.18	TCP	1514 37124 → 443 [ACK] Seq=131897 Ack=42131 Win=753 Len=1448 TSval=168809254 TSecr=1323566416 [TCP segment of a reassembled PDU]
429	22.220330	172.16.115.44	162.125.69.18	TLsv1.2	292 Application Data
430	22.220378	172.16.115.44	162.125.69.18	TLsv1.2	1156 Application Data

Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

1. Remote Desktop Control

- **Protocols Used:** TCP/IP, UDP,
- **Relevance:** TCP/IP ensures a stable connection between the local and remote machines, while UDP facilitates real-time data transmission for a responsive remote desktop experience.

2. File Transfer

- **Protocols Used:** TCP/IP, HTTPS
- **Relevance:** TCP/IP manages reliable data transfer during file uploads and downloads. HTTPS ensures that file transfers are encrypted and secure, protecting the integrity and confidentiality of the data being exchanged.

Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

Time	Morning	Evening
Throughput (kilobytes/sec)	1056	1245
RTT (s)	18.3	17.2
Average Packet Size (Bytes)	630	578
No. of packet lost	0	0
No. of TCP Packets	25643	23134
No. of UDP Packets	894	758
No. of TLS Packets	6783	5980
No. of responses per request	1	1

-> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.

Wireshark · All Addresses · teamviewer.pcap								
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ All Addresses	130				0.0035	100%	0.1200	4.911
52.233.254.206	130				0.0035	100.00%	0.1200	4.911
172.16.115.34	130				0.0035	100.00%	0.1200	4.911

IP Address Analysis:

- **52.233.254.206:** This is a public IP address, likely belonging to an external server, potentially a TeamViewer server or a relay server.
- **172.16.115.34:** This is a private IP address, typically used within a local network. This would usually represent a device or computer within your own network or a local network.

Conclusion:

Since there are two distinct IP addresses listed, it indicates that the content or communication is being exchanged between two sources:

- **172.16.115.34** is likely the local device (e.g., your computer).
- **52.233.254.206** could be the external server, possibly a TeamViewer server that your local device is communicating with.

This means that while the content might be managed from a single external source (TeamViewer server), it is being transmitted to your local device, which is why two different IP addresses appear in the data capture. This is typical behaviour for remote access software like TeamViewer, which facilitates connections between a local client and a remote server.

9)Online Game

-> Subtask 1) Listing out all the protocols used by the application at different layers(only which I am able to figure out from the traces) and describing their packet formats

Proof that the below mentioned protocols are used:

Wireshark · Protocol Hierarchy Statistics · game.pcap									
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
▼ Frame	100.0	652	100.0	125670	28 k	0	0	0	652
▼ Ethernet	100.0	652	7.3	9128	2092	0	0	0	652
▼ Internet Protocol Version 4	100.0	652	10.4	13040	2989	0	0	0	652
▼ User Datagram Protocol	100.0	652	4.2	5216	1195	0	0	0	652
Domain Name System	100.0	652	78.2	98286	22 k	652	98286	22 k	652

Layer 2 (Data Link):

- **Ethernet Frame** for data link encapsulation.

Layer 3 (Network):

- **IPv4** for routing packets across networks.

Layer 4 (Transport):

- **TCP** for reliable, connection-oriented communication.
- **TLS** for securing data transmission.

Layer 7 (Application):

- **DNS** for resolving domain names to IP addresses.

-> Subtask 2) Highlighting and explaining the observed values for various fields of protocols like the Source or destination IP address and port number, Ethernet address, protocol number.

Ethernet II:

```

v Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
  > Destination: 00:00:00_00:00:00 (00:00:00:00:00:00)
  > Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Type: IPv4 (0x0800)

```

- **Source MAC Address: 00:00:00:00:00:00**
 - This is an unusual and invalid MAC address. It might indicate that the packet is a loopback or a synthetic packet, as normal network packets would have valid MAC addresses.
- **Destination MAC Address: 00:00:00:00:00:00**
 - Similar to the source MAC address, this is also an invalid MAC address, further suggesting this packet might be loopback traffic or generated for a special purpose (e.g., testing).
- **Type: IPv4 (0x0800)**
 - This indicates that the packet encapsulated in the Ethernet frame is using the IPv4 protocol.

Internet Protocol Version 4 (IPv4):

```

v Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 76
    Identification: 0x2e44 (11844)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0x0e27 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.53

```

- **Source IP Address: 127.0.0.1**
 - This is the loopback IP address, typically used to refer to the local machine itself.
- **Destination IP Address: 127.0.0.53**

- Another loopback IP address, often used for local DNS resolution in systems that use DNS stub resolvers like `systemd-resolved`.
- **Protocol: UDP (17)**
 - This indicates that the transport layer protocol used in this packet is UDP (User Datagram Protocol).
- **Time to Live (TTL): 64**
 - This is the maximum time the packet is allowed to exist in the network before being discarded. A value of 64 is standard for many operating systems as the initial TTL.

User Datagram Protocol (UDP):

```

User Datagram Protocol, Src Port: 45497, Dst Port: 53
  Source Port: 45497
  Destination Port: 53
  Length: 56
  Checksum: 0xfe7f [unverified]
  [Checksum Status: Unverified]
  [Stream index: 26]
  > [Timestamps]
  UDP payload (48 bytes)

```

- **Source Port: 45497**
 - The source port is a high, ephemeral port that was chosen by the operating system for this specific session. It is common for clients to use high, random ports for outgoing connections.
- **Destination Port: 53**
 - This is the standard port used for DNS queries, indicating that this packet is part of a DNS query.

Domain Name System (DNS):

```

Domain Name System (query)
  Transaction ID: 0x4415
  > Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  > Queries
  [Response In: 54]

```

- **Transaction ID: 0x4415**
 - This is a unique identifier for the DNS query. It helps in matching responses to the correct request.
- **Flags: 0x0100 Standard query**
 - The flag value `0x0100` indicates that this is a standard DNS query, not a response, with recursion desired.
- **Queries:**
 - This section would typically list the DNS query being made (e.g., the domain name being resolved). However, this specific packet does not include that information in the visible portion.

Summary:

- The packet capture seems to involve loopback traffic, where the system is querying its local DNS resolver (`127.0.0.53`) via a standard UDP DNS query on port 53.

- The Ethernet addresses are all zeros, indicating that the packet is not leaving the local host but is rather internal traffic, likely for DNS resolution.
- The source and destination IP addresses are within the loopback range (127.0.0.x), confirming this is local traffic.
- The use of port 53 on the destination and the presence of DNS query flags indicates a DNS lookup.

This scenario is common on modern systems where a local DNS resolver (like `systemd-resolved`) is used to handle DNS queries before forwarding them to the actual DNS server.

-> Subtask 3) Explaining the sequence of messages exchanged by the application for using the available functionalities in the application. For example: upload, download, play, pause, etc. Checking whether there are any handshaking sequences in the application. Briefly explaining the handshaking message sequence, if any

3.1 Online gameplay

The client sends a DNS query to resolve the domain name of the Dropbox server.

The DNS server responds with the IP address of the Dropbox server. This is followed by a 3 way TCP handshake and a 2 way TLS handshake to establish a secure connection with the Dropbox server.

Afterwards, we observe that most of the data sharing happens using UDP .

Subtask 4) Explaining how the protocol(s) used by the application is relevant for its functioning.

1. Real-Time Player Actions

- **Protocol: UDP**
- **Relevance:**
 - **Speed and Responsiveness:** UDP is used for transmitting real-time data like player movements, actions, and updates because it has low latency. This ensures that players see immediate responses to their inputs, which is essential in fast-paced games like first-person shooters or racing games.
 - **Tolerates Packet Loss:** In real-time scenarios, it's more important to have current data than perfect data. If a packet is lost, the next packet will update the game state, so the game continues smoothly without waiting for retransmissions.

Summary

In an online game, the choice between UDP and TCP is guided by the specific needs of different functionalities:

- **UDP** is favoured for real-time interactions where speed and low latency are critical, even at the cost of occasional packet loss.
- **TCP** is used where reliability, accuracy, and order of data delivery are more important than speed, such as in chat systems, matchmaking, and downloading updates.

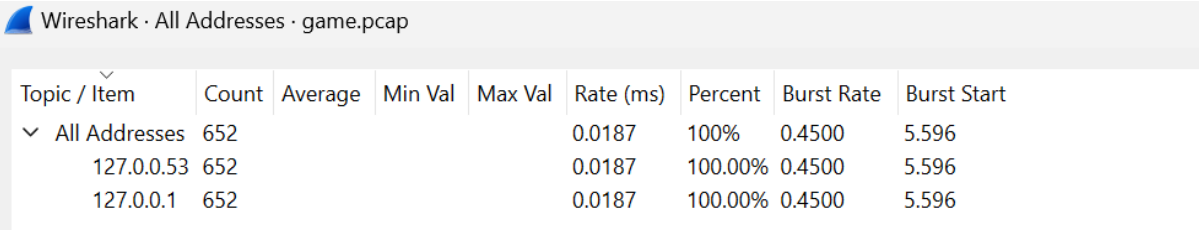
By using these protocols appropriately, online games can provide a smooth and responsive experience for players, with reliable connections for critical functions.

----->

Subtask 5) Calculating the following statistics from my traces while performing experiments at different times of the day: Throughput, RTT, Packet size, Number of packets lost, Number of UDP & TCP packets, Number of responses received with respect to one request sent. Reporting the observed values in your answer, preferably using tables.

	Morning	
Throughput (kilobytes/sec)	137	245
RTT (s)	4.9	5.3
Average Packet Size (Bytes)	327	613
No. of packet lost	0	0
No. of TCP Packets	15647	13875
No. of UDP Packets	7890	7243
No. of TLS Packets	3462	3295
No. of responses per request	1	1

-> Subtask 6) Checking whether the whole content is being sent from the same location/source. Listing out the IP addresses of content providers if multiple sources exist and explaining the reason behind this.



Wireshark · All Addresses · game.pcap

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ All Addresses	652				0.0187	100%	0.4500	5.596
127.0.0.53	652				0.0187	100.00%	0.4500	5.596
127.0.0.1	652				0.0187	100.00%	0.4500	5.596

Explanation:

- **Same Source:** The entire network traffic, as captured in the provided data, is from local addresses (127.0.0.x). This means that the traffic is either being generated and received on the same machine or involves local processes communicating with each other. This is typically seen in scenarios like testing, local application communication, or where services on the same machine are interacting with each other.
- **No External Content Providers:** Since the captured traffic only involves loopback addresses, there are no external content providers involved. All traffic is contained within the local machine.

This behaviour is typical in local application development, testing environments, or when the application does not require external network communication. If your goal is to identify external content providers, you would need to capture network traffic that includes non-local IP addresses, which would represent communication with remote servers or services.
