<u>Wireshark Analysis</u>

(HTTP, Web Cache, DNS, and Transport Layer Protocols)

We have analysed the data collected by using the **application 'Netflix'**. We ran the application on our laptop on **IITG Connect** Wifi Network and proceeded to analyse the packets obtained from the same to complete all the tasks.

Report By: Group 13
Achintya Gupta- 210101005
Kannan Rustagi- 210101054
Kshitij Maurya- 210101059

<u>TASK 1</u>:

List out all the protocols used by the application at different layers (only those which you can figure out from traces). Study and briefly describe their packet formats.

APPLICATION LAYER

1. TLS (Transport layer Security Protocol)

Transport Layer Security (TLS) **encrypts data** sent over the Internet to ensure that eavesdroppers and hackers are unable to see what you transmit which is particularly useful for private and sensitive information.

```
v Transport Layer Security
  v TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 16352
      Encrypted Application Data: fc79ad695d61af4fae36c8d6d4b46c05d3fb5a5d2a2cd2df1d6699f65af3e2fce15b14ba…
      [Application Data Protocol: Hypertext Transfer Protocol]
```

Packet format for TLS-

Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			

The basic unit of a TLS is an SSL(Secure Socket Layer). Content Type specifies whether the content is Handshake, Application Data, Alert, Change Cipher Spec etc. Version field specifies the version of TLS being used. Length gives the length of packet including header. Payload contains the data of the packet and MAC stands for the Message authentication code. When data is transmitted between client and server in TLS , it is divided into records and each record contains a MAC for data integrity and authentication. Block ciphers have a specified fixed length and most of them require that the input data is a multiple of their size. It is common that the last block contains data that does not meet this requirement. In this case, padding (usually random data) is used to bring it to the required block length.

2. DNS(Domain Name System)

DNS is a distributed database implemented in a hierarchy of DNS servers, and an application-layer protocol that allows hosts to query the distributed database.

DNS query and response packets as observed from wireshark trace-

1665	5.021879	10.150.46.190	172.17.1.2	DNS	75	Standard query 0x46af A www.netflix.com
1666	5.024004	172.17.1.2	10.150.46.190	DNS	437	Standard query response 0x46af A www.netflix.com CNAME www.dradis.netflix.com CNAME www.us-west-2.internal

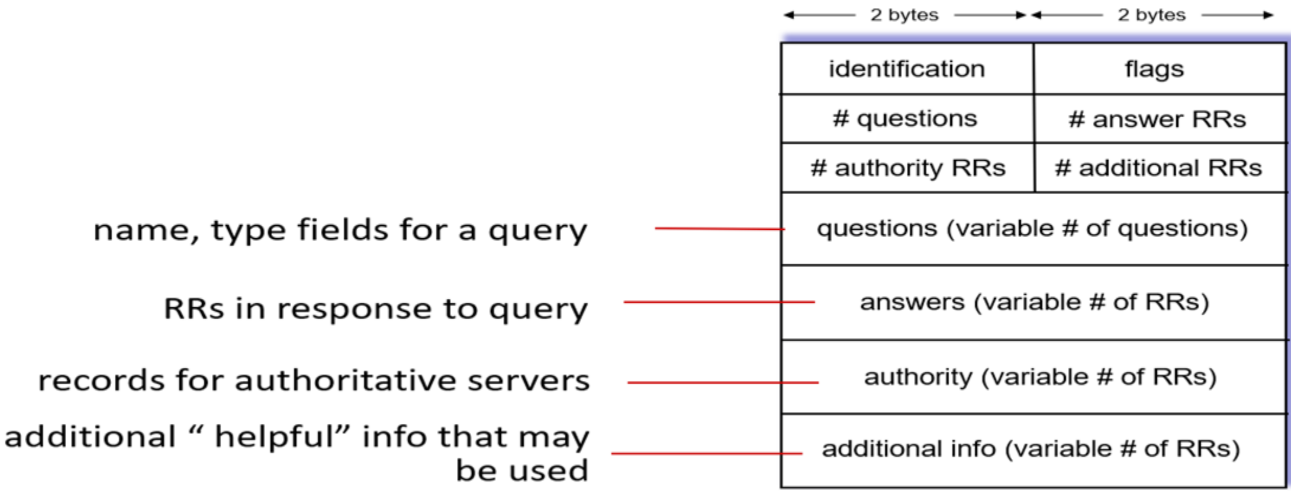
DNS query sent from my device to IITG server-

```
Domain Name System (query)
    Transaction ID: 0x46af
  Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  Queries
    www.netflix.com: type A, class IN
        Name: www.netflix.com
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
    [Response In: 1666]
```

DNS response received from IITG server-

```
Domain Name System (response)
    Transaction ID: 0x46af
  Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 6
    Authority RRs: 4
    Additional RRs: 2
  Queries
    www.netflix.com: type A, class IN
        Name: www.netflix.com
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

A DNS query and response packet have the same format as follows-

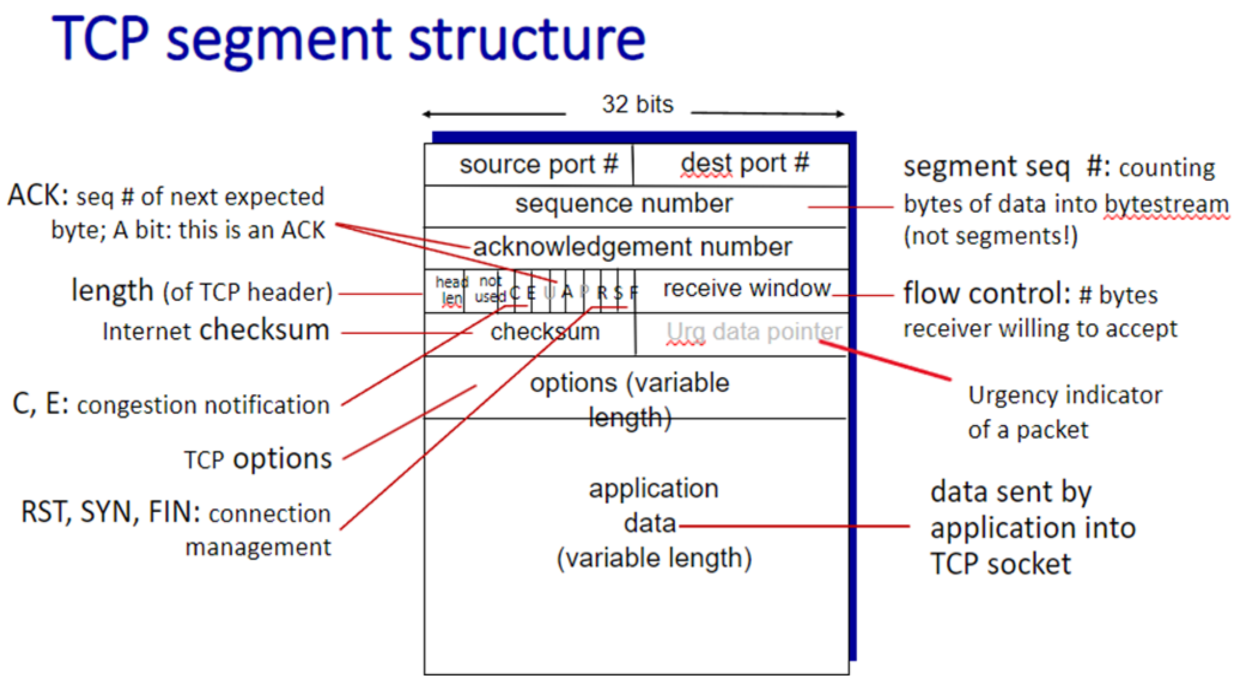


The first 12 bytes is the header section. The identification field is a 16-bit number that identifies the query. Flags in the flag field include query/reply flag, and authoritative flag. The next four fields indicate the number of occurrences of the four types of data sections that follow. The question section contains information about the query that is being made. The answer section contains the resource records for the name that was originally queried. The authority section contains records of other authoritative servers. The additional section contains other helpful records.

TRANSPORT LAYER-

1. TCP(Transmission Control Protocol)

TCP is the Internet's transport-layer, connection-oriented, **reliable transport protocol**. The TCP segment structure is as follows-

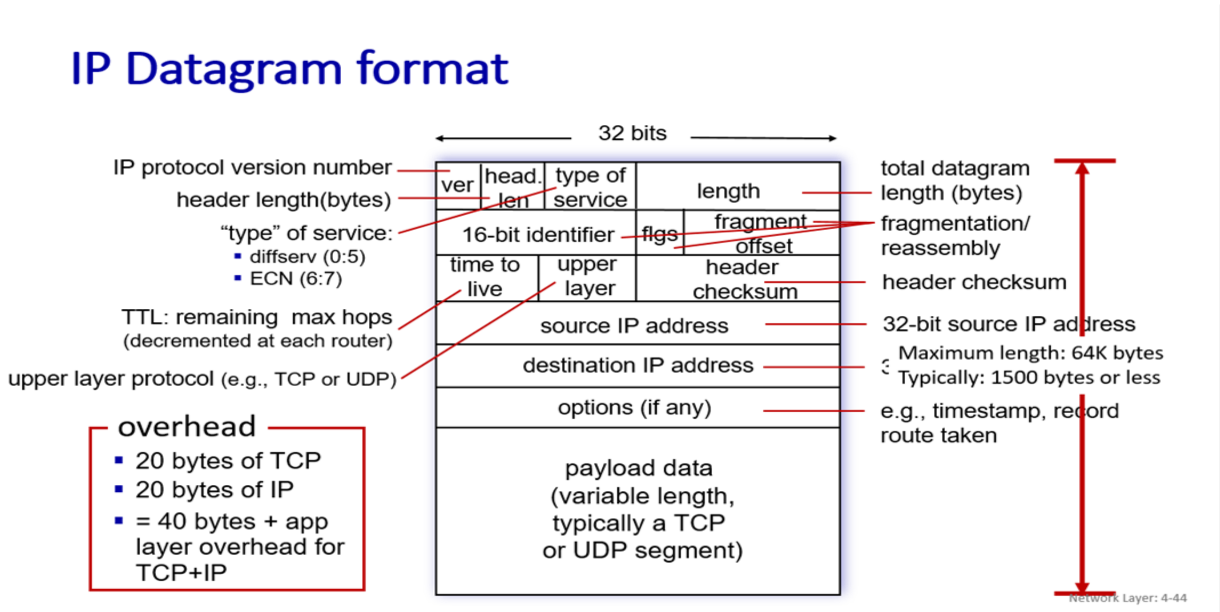


Source Port and Destination Port indicates the port of the sending and receiving application. Sequence Number contains the sequence number of the first data byte. Acknowledgement Number contains the sequence number of databyte that receiver expects to receive next from the sender. Header Length specifies the length of the TCP header. There is a total of 6 types of Flags of 1 bit each. Some of them are ACK, PSH and SYN to specify the kind of TCP message, for eg: ACK stands for acknowledgment. Checksum is used to verify the integrity of data in the TCP payload(method of error correction in transport layer). Window Size contains the size of the receiving window of the sender. It advertises how much data (in bytes) the sender can receive without acknowledgement. It helps in flow control. Urgent Pointer indicates the number of data byte which urgent, counting from the first data byte. Options are used for different purposes like timestamp, window size extension, parameter negotiation, padding.

NETWORK LAYER

1. Internet Protocol Verion-4(IPv4) :-

IPv4 is a widely used protocol over different kinds of network in the Network layer. IP addressing is a logical means of assigning addresses to devices on a network. Each device connected to the internet requires a unique IP address. An IP address has two parts—-one part identifies the host, such as a computer or other device. The structure of an IPv4 datagram is as follows-

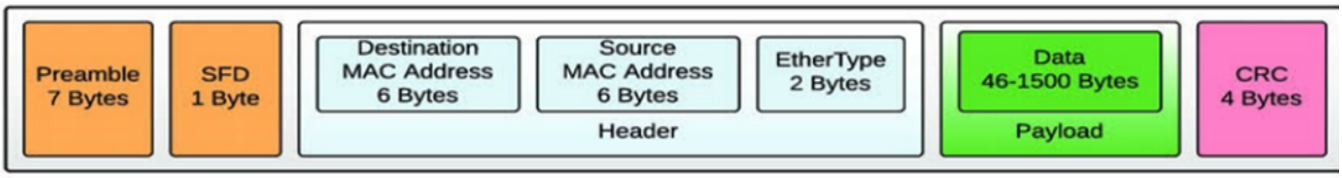


LINK LAYER

1. Ethernet

The Ethernet frame commences with a Preamble and SFD (Start Frame Delimiter), both operating within the realm of the physical layer. The preamble, a seven-byte sequence of alternating 1s and 0s (56 bits), facilitates bit-level synchronization among network devices by aiding in clock alignment. The SFD is an eight-bit (one-byte) value, signaling the conclusion of the preamble.

Following this, the Destination and Source MAC addresses denote the hardware addresses of the receiving and transmitting devices, respectively. The EtherType field stands as the sole distinguishing factor between 802.3 and Ethernet II, representing the Ethernet type. Subsequently, the Data segment encompasses the actual information to be transmitted, while the CRC (Cyclic Redundancy Check) employs a CRC-32 polynomial code to facilitate error detection. The structure of Ethernet layer is shown below:-



We can see the above mentioned protocols on the wireshark interface as well by selecting the **'protocol hierarchy'** option under statistics as shown below-

Wireshark · Protocol Hierarchy Statistics · bestCapture.pcapng

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	41	100.0	41970	637	0	0	0	41
Ethernet	100.0	41	1.4	574	8	0	0	0	41
Internet Protocol Version 4	100.0	41	2.0	820	12	0	0	0	41
Transmission Control Protocol	100.0	41	96.7	40576	616	3	3154	47	41
Transport Layer Security	92.7	38	87.4	36662	557	38	36662	557	38

<u>TASK-2:</u>

Highlight and explain the observed values for various fields of the protocols.

APPLICATION LAYER

1. TLSv1.3 (Transport Layer Security protocol)

```
Transport Layer Security
  TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 16352
    Encrypted Application Data: fc79ad695d61af4fae36c8d6d4b46c05d3fb5a5d2a2cd2df1d6699f65af3e2fce15b14ba…
    [Application Data Protocol: Hypertext Transfer Protocol]
```

Firstly, in this packet, the <u>content type</u> as mentioned is Application Data which is encrypted to ensure security. We can observe that the <u>version</u> on TLS protocol being used here is 1.2. Moreover, the <u>length</u> of the entire packet, i.e, including the header is 16352.

2. DNS(Domain Name System)

DNS uses **UDP protocol** in the transport layer to query for the IP address corresponding to the required host. The observed DNS query that was sent from our local device to the IITG server-

```
˅ Domain Name System (query)
    Transaction ID: 0x46af
  ˅ Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ˅ Queries
    ˅ www.netflix.com: type A, class IN
        Name: www.netflix.com
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
    [Response In: 1666]
```

The query consists of flags which defines the type of DNS query, whether it is a question, recursive or iterative search should to be followed etc. It also defines the number of questions and number of different types of answers(RR's) followed by them in the bottom.

The response received to the above DNS query is as follows-

```
˅ Domain Name System (response)
    Transaction ID: 0x46af
  ˅ Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 6
    Authority RRs: 4
    Additional RRs: 2
  ˅ Queries
    ˅ www.netflix.com: type A, class IN
        Name: www.netflix.com
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

```
v Answers
   > www.netflix.com: type CNAME, class IN, cname www.dradis.netflix.com
   > www.dradis.netflix.com: type CNAME, class IN, cname www.us-west-2.internal.dradis.netflix.com
   > www.us-west-2.internal.dradis.netflix.com: type CNAME, class IN, cname apiproxy-website-nlb-prod-1-bcf28d21f4bbcf2c.elb.us-west-
   > apiproxy-website-nlb-prod-1-bcf28d21f4bbcf2c.elb.us-west-2.amazonaws.com: type A, class IN, addr 44.240.158.19
   > apiproxy-website-nlb-prod-1-bcf28d21f4bbcf2c.elb.us-west-2.amazonaws.com: type A, class IN, addr 52.38.7.83
   > apiproxy-website-nlb-prod-1-bcf28d21f4bbcf2c.elb.us-west-2.amazonaws.com: type A, class IN, addr 44.242.13.161
v Authoritative nameservers
   > elb.us-west-2.amazonaws.com: type NS, class IN, ns ns-748.awsdns-29.net
   > elb.us-west-2.amazonaws.com: type NS, class IN, ns ns-1283.awsdns-32.org
   > elb.us-west-2.amazonaws.com: type NS, class IN, ns ns-1870.awsdns-41.co.uk
   > elb.us-west-2.amazonaws.com: type NS, class IN, ns ns-424.awsdns-53.com
v Additional records
   > ns-424.awsdns-53.com: type A, class IN, addr 205.251.193.168
   > ns-748.awsdns-29.net: type A, class IN, addr 205.251.194.236
  [Request In: 1665]
  [Time: 0.002125000 seconds]
```

This screenshot shows how the records are returned in the request. It provides the different types of records like A, CNAME etc. in which there are IP addresses and aliases.

TRANSPORT LAYER-

1. TCP(Transmission Control Protocol)-

```
v Transmission Control Protocol, Src Port: 443, Dst Port: 54001, Seq: 1, Ack: 303, Len: 1396
     Source Port: 443
     Destination Port: 54001
     [Stream index: 30]
     [Conversation completeness: Complete, WITH_DATA (63)]
     [TCP Segment Len: 1396]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 529659735
     [Next Sequence Number: 1397    (relative sequence number)]
     Acknowledgment Number: 303    (relative ack number)
     Acknowledgment number (raw): 744388453
     0101 .... = Header Length: 20 bytes (5)
   > Flags: 0x010 (ACK)
     Window: 152
     [Calculated window size: 19456]
     [Window size scaling factor: 128]
     Checksum: 0x437e [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   v [Timestamps]
        [Time since first frame in this TCP stream: 2.256335000 seconds]
        [Time since previous frame in this TCP stream: 0.101357000 seconds]
   v [SEQ/ACK analysis]
        [iRTT: 0.002073000 seconds]
        [Bytes in flight: 1396]
        [Bytes sent since last PSH flag: 1396]
      > [TCP Analysis Flags]
     TCP payload (1396 bytes)
     Retransmitted TCP segment data (1396 bytes)
```

An example of TCP packet is shown above.

We can see that it contains source port, destination port, sequence number, acknowledgment number, header length etc. It also contains flags which contains various metadata like urgent pointer, acknowledgment (whether it is an acknowledgement or not), SYN, FIN, reserved etc. It also contains Window size and checksum (for error detection).

NETWORK LAYER

1. Internet Protocol Verion-4 (IPv4) :-

```
v Internet Protocol Version 4, Src: 52.25.127.28, Dst: 10.150.46.190
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1436
    Identification: 0xbc75 (48245)
  v 000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 63
    Protocol: TCP (6)
    Header Checksum: 0xcd5d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 52.25.127.28
    Destination Address: 10.150.46.190
```

The structure of IPv4 packet is shown above. It has fields containing source IP address, destination IP address, version, header length, total length etc. It also has flags specifying wheather it is a reserved bit, fragment offset, wheather there are more fragments to come etc (It is used for flow control). It also contains Time to Live (TTL) representing max number of hops of that packet in the network before it is dropped, followed by checksum for error detection at Network level and data payload from above layers.

LINK LAYER

1. Ethernet

```
v Ethernet II, Src: Dell_f0:ee:42 (c8:f7:50:f0:ee:42), Dst: Chongqin_ab:3f:ff (b4:b5:b6:ab:3f:ff)
  v Destination: Chongqin_ab:3f:ff (b4:b5:b6:ab:3f:ff)
      Address: Chongqin_ab:3f:ff (b4:b5:b6:ab:3f:ff)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: Dell_f0:ee:42 (c8:f7:50:f0:ee:42)
      Address: Dell_f0:ee:42 (c8:f7:50:f0:ee:42)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

The structure of ethernet packet is shown above. The Src and Dst field contain the source and destination device's MAC addresses. Type indicates the upper layer protocol used which is IPv4 in this case.The LG bit in both cases is 0,so the addresses are vendor assigned,not administratively assigned.(Here unicast refers to sending data to single destination). This is followed by Data from upper layers.

TASK-3:

Explain the sequence of messages exchanged by the application for using the available functionalities in the application.

3.1> Launching the application:

Standard query is sent to DNS for "www.netflix.com" and its IP is returned. Multiple DNS requests to different servers of Netflix are also seen to be sent. These DNS Requests are sent to 172.17.1.1 that is the DNS server of IITG.

1665	5.021879	10.150.46.190	172.17.1.2	DNS	75	Standard query 0x46af A www.netflix.com
1666	5.024004	172.17.1.2	10.150.46.190	DNS	437	Standard query response 0x46af A www.netflix.com CNAME www.dradis.netflix.com CNAME www.us-west-2.internal…
1667	5.024730	10.150.46.190	44.240.158.19	TCP	66	54000 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1668	5.027198	44.240.158.19	10.150.46.190	TCP	66	443 → 54000 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=1396 SACK_PERM WS=128
1669	5.027308	10.150.46.190	44.240.158.19	TCP	54	54000 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
1670	5.030120	10.150.46.190	44.240.158.19	TLSv1.3	343	Client Hello
1671	5.032104	44.240.158.19	10.150.46.190	TCP	54	443 → 54000 [ACK] Seq=1 Ack=290 Win=19456 Len=0

2140	6.361382	10.150.46.190	172.17.1.2	DNS	103	Standard query 0xcc10 A oca-api.us-east-2.origin.prodaa.netflix.com
2141	6.361382	10.150.46.190	172.17.1.2	DNS	103	Standard query 0xb50b A oca-api.us-east-1.origin.prodaa.netflix.com
2142	6.361413	10.150.46.190	172.17.1.2	DNS	103	Standard query 0xd05a A oca-api.us-west-2.origin.prodaa.netflix.com
2143	6.361461	172.17.1.1	10.150.46.190	DNS	320	Standard query response 0xb50b A oca-api.us-east-1.origin.prodaa.netflix.com A 18.
2144	6.362092	172.17.1.1	10.150.46.190	DNS	320	Standard query response 0xd05a A oca-api.us-west-2.origin.prodaa.netflix.com A 34.

3.2> Establishing the connection:

Request for connection is then sent to the server at the received IP address and then we can see a handshake mechanism between the server and client. This can be seen as a **[SYN]** (Synchronization) request. The server then sends an acknowledgement for the SYN request with SYN and ACK flags. The client then sends a final ACK message for connection and the connection is successfully established through this **3-way handshake mechanism**.

No.	Time	Source	Destination	Protocol	Length	Info
1729	5.155449	10.150.46.190	52.25.127.28	TCP	66	54001 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1730	5.157399	52.25.127.28	10.150.46.190	TCP	66	443 → 54001 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=1396 SACK_PERM WS=128
1731	5.157522	10.150.46.190	52.25.127.28	TCP	54	54001 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
1732	5.159786	10.150.46.190	52.25.127.28	TLSv1.2	356	Client Hello

After this, for a secure connection, there is a **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.

177165	479.320881	10.150.46.190	52.25.127.28	TLSv1.2	356	Client Hello
177413	480.128745	52.25.127.28	10.150.46.190	TLSv1.2	1450	Server Hello
177418	480.129247	52.25.127.28	10.150.46.190	TLSv1.2	661	Certificate, Server Key Exchange, Server Hello Done
177420	480.131842	10.150.46.190	52.25.127.28	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
177423	480.138315	10.150.46.190	52.25.127.28	TLSv1.2	1280	Application Data

Observation: Load Balancing
Multiple servers are connected across the globe. So there is constant service even if some connection fails. This also prevents too much load on a single server.

3.3> Playing video on application

We observed that packets come in regular time intervals and the client also sends ACK(acknowledgment) to the server in regular time intervals. Here, we also noticed that the Application data from the server is sent using TLS protocol(for adding an extra layer of security) to the client while the ACK(acknowledgment) from the client to server is sent using normal TCP protocol.
Also just after starting the video, huge number of packets are sent from the netflix server to the client.

ip.dst == 44.242.60.85 || ip.src == 44.242.60.85

No.	Time	Source	Destination	Protocol	Length	Info
149576	260.588621	10.150.33.40	44.242.60.85	TLSv1.3	166	Application Data
149577	260.588790	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=263582 Ack=158599 Win=261632 Len=1396 [TCP segment of a reassembled PDU]
149578	260.588790	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=264978 Ack=158599 Win=261632 Len=1396 [TCP segment of a reassembled PDU]
149579	260.588790	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=266374 Ack=158599 Win=261632 Len=1396 [TCP segment of a reassembled PDU]
149580	260.588790	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=267770 Ack=158599 Win=261632 Len=1396 [TCP segment of a reassembled PDU]
149581	260.588790	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=269166 Ack=158599 Win=261632 Len=1396 [TCP segment of a reassembled PDU]
149582	260.588790	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=270562 Ack=158599 Win=261632 Len=1396 [TCP segment of a reassembled PDU]
149583	260.588790	10.150.33.40	44.242.60.85	TLSv1.3	861	Application Data
149584	260.588900	10.150.33.40	44.242.60.85	TLSv1.3	85	Application Data
149585	260.590223	44.242.60.85	10.150.33.40	TCP	54	443 → 60974 [ACK] Seq=158599 Ack=263582 Win=100736 Len=0
149586	260.590223	44.242.60.85	10.150.33.40	TCP	54	443 → 60974 [ACK] Seq=158599 Ack=264978 Win=99456 Len=0
149587	260.590223	44.242.60.85	10.150.33.40	TCP	54	443 → 60974 [ACK] Seq=158599 Ack=266374 Win=99328 Len=0
149588	260.590223	44.242.60.85	10.150.33.40	TCP	54	443 → 60974 [ACK] Seq=158599 Ack=270562 Win=107776 Len=0
149589	260.590675	44.242.60.85	10.150.33.40	TCP	54	443 → 60974 [ACK] Seq=158599 Ack=272796 Win=113280 Len=0
149640	260.909241	44.242.60.85	10.150.33.40	TCP	1450	443 → 60974 [ACK] Seq=158599 Ack=272796 Win=113280 Len=1396 [TCP segment of a reassembled PDU]
149641	260.909499	10.150.33.40	44.242.60.85	TCP	54	60974 → 443 [ACK] Seq=272796 Ack=159995 Win=262144 Len=0
149642	260.910210	44.242.60.85	10.150.33.40	TCP	1450	443 → 60974 [ACK] Seq=159995 Ack=272796 Win=113280 Len=1396 [TCP segment of a reassembled PDU]
149643	260.910210	44.242.60.85	10.150.33.40	TCP	158	443 → 60974 [PSH, ACK] Seq=161391 Ack=272796 Win=113280 Len=104 [TCP segment of a reassembled PDU]
149644	260.910296	10.150.33.40	44.242.60.85	TCP	54	60974 → 443 [ACK] Seq=272796 Ack=161495 Win=262144 Len=0
149684	260.920191	44.242.60.85	10.150.33.40	TLSv1.3	152	Application Data
149685	260.920242	10.150.33.40	44.242.60.85	TCP	54	60974 → 443 [ACK] Seq=272796 Ack=161593 Win=261888 Len=0
155599	276.091214	10.150.33.40	44.242.60.85	TLSv1.3	101	Application Data
155600	276.091419	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=272843 Ack=161593 Win=261888 Len=1396 [TCP segment of a reassembled PDU]
155601	276.091419	10.150.33.40	44.242.60.85	TCP	1450	60974 → 443 [ACK] Seq=274239 Ack=161593 Win=261888 Len=1396 [TCP segment of a reassembled PDU]

3.4> Pausing video

When we pause the video, packets are still received from the server, however, the rate at which packets are received from the server decreases. There is also some communication between the server and client to keep the connection alive.

340888	918.539229	10.150.33.40	52.123.170.29	TCP	54	61650 → 443 [ACK] Seq=516 Ack=5689 Win=131072 Len=0
340889	918.539357	52.123.170.29	10.150.33.40	TLSv1.2	613	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
340890	918.543299	10.150.33.40	52.123.170.29	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
340891	918.543632	10.150.33.40	52.123.170.29	TLSv1.2	153	Application Data
340892	918.544052	10.150.33.40	52.123.170.29	TCP	1450	61650 → 443 [ACK] Seq=775 Ack=6248 Win=130560 Len=1396 [TCP segment of a reassembled PDU]
340893	918.544052	10.150.33.40	52.123.170.29	TLSv1.2	829	Application Data
340894	918.544156	10.150.33.40	52.123.170.29	TLSv1.2	161	Application Data
340895	918.546639	52.123.170.29	10.150.33.40	TCP	54	443 → 61650 [ACK] Seq=6248 Ack=676 Win=20608 Len=0
340896	918.547409	52.123.170.29	10.150.33.40	TCP	54	443 → 61650 [ACK] Seq=6248 Ack=775 Win=20608 Len=0
340897	918.548008	52.123.170.29	10.150.33.40	TCP	54	443 → 61650 [ACK] Seq=6248 Ack=3053 Win=26112 Len=0
340898	918.730452	52.123.170.29	10.150.33.40	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
340899	918.730452	52.123.170.29	10.150.33.40	TLSv1.2	123	Application Data
340900	918.730579	10.150.33.40	52.123.170.29	TCP	54	61650 → 443 [ACK] Seq=3053 Ack=6368 Win=130304 Len=0
340901	918.730951	10.150.33.40	52.123.170.29	TLSv1.2	92	Application Data
340902	918.732100	52.123.170.29	10.150.33.40	TCP	54	443 → 61650 [ACK] Seq=6368 Ack=3091 Win=26112 Len=0
340931	918.912213	52.123.170.29	10.150.33.40	TLSv1.2	92	Application Data
340982	918.963543	10.150.33.40	52.123.170.29	TCP	54	61650 → 443 [ACK] Seq=3091 Ack=6406 Win=130304 Len=0
340983	919.121861	52.123.170.29	10.150.33.40	TLSv1.2	304	Application Data
341000	919.170542	10.150.33.40	52.123.170.29	TCP	54	61650 → 443 [ACK] Seq=3091 Ack=6656 Win=130048 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 pre-loaded in buffer as it does not belong to same sequence.

3.6> 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 higher rate than normal.
Other effects like increase/decrease the volume, brightness of application does not have a contribution in packet exchange rate.

3.7> Closing the application

The above shown example is a case of abrupt exit, where one end exits abruptly and TCP exit handshaking does not occur as a result. There would be retransmissions from the other end for a particular time limit after that it also exits.

Now coming to graceful exit, we can see a **3-way handshaking** . First client sends to server keeping **[FIN, ACK]** on indicating that we can finish 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.

3412	10.400264	10.150.46.190	52.41.213.68	TCP	54	54015 → 443 [FIN, ACK] Seq=294 Ack=3025 Win=262144 Len=0
3413	10.403752	52.41.213.68	10.150.46.190	TCP	54	443 → 54015 [FIN, ACK] Seq=3025 Ack=295 Win=19456 Len=0

RST/ACK is used to end a TCP session. The packet is ACKnowledging receipt of the previous packet in the stream, and then closing that same session with a RST (Reset) packet being sent to the far end to let it know the connection is being closed.

	Time	Source	Destination	Protocol	Length	Info
113977	305.097758	10.150.46.190	52.25.127.28	TCP	54	54045 → 443 [RST, ACK] Seq=16443 Ack=11881 Win=0 Len=0
113979	305.098116	10.150.46.190	52.25.127.28	TCP	54	54001 → 443 [RST, ACK] Seq=23776 Ack=38562 Win=0 Len=0

TASK 4:

Explain how the particular protocol(s) used by the application is relevant for functioning of the Application.

At the **application layer-**

1. **TLS protocol** is used for encrypting and securing content, user data, and decryption keys while packets are exchanged between the server and client. This helps to prevent eavesdropping from other malicious parties. TLS certificates are used to verify the identity of a website or server, hence, it essentially verifies Netflix servers, that we are indeed connecting to a legitimate server and not a malicious one.
2. **DNS protocol** is used to transalate domain names to IP addresses. When we start Netflix application, a DNS query is sent from our local device for resolving the corresponding hostname. We also observed that it optimizes content delivery through load balancing and CDN selection.

At the **Transport layer,**

1. **TCP protocol** is connection oriented and ensures reliable content delivery that delivers in order packets. This service is very essential to Netflix, that is a video streaming platform, so in order and reliable delivery of packets is very important from their service point of view.

At the **Network layer,**

1. **IPv4** is crucial for Netflix as it assigns unique IP addresses to user devices, facilitating data transmission over the Internet. This enables Netflix to deliver streaming content to viewers' devices, ensuring proper routing and communication between Netflix servers and user devices, essential for seamless video streaming.

At the **Link layer**,

1. **Ethernet** at the link layer is crucial for the Netflix application by providing reliable, high-speed data transmission within local networks. It ensures seamless streaming of content from Netflix servers to user devices, maintaining a stable connection and preventing data loss, ultimately delivering a smooth and uninterrupted viewing experience.

TASK-5

Did you observe any caching mechanisms in the captured packets?Explain.

Indeed, we noticed that video packets that have been played are subject to caching. These incoming video packets are initially stored in a buffer and then subsequently moved to a cache once they have been viewed. This caching mechanism comes into play when using Netflix's rewind feature, allowing you to go back 10 seconds in the video.

We also observed that the cache for video packets is noticeably smaller in size compared to the buffer. If we attempt a significant rewind, such as going back 20 minutes, the video playback is paused until the browser requests the subsequent packets starting from that point. If the packets preceding the 20-minute mark had already been cached, the video would have loaded much more quickly, and there would have been no buffering delays.

In DNS query, there exists a field called TTL (Time to live) that indicates the presence of web cache for holding DNS requests, as TTL represents the maximum time a packet can stay in cache before getting replaced. We made 3 consecutive requests. First DNS request has RTT of around 0.24s, but subsequent requests take considerably less time, around 0.02s and 0.01s.

Task 6: Statistics

Calculate the following statistics from your traces while performing experiments at different time of the day.

Statistics

Measurement	Captured
Packets	212836
Time span, s	557.099
Average pps	382.0
Average packet size, B	715
Bytes	152159641
Average bytes/s	273 k
Average bits/s	2185 k

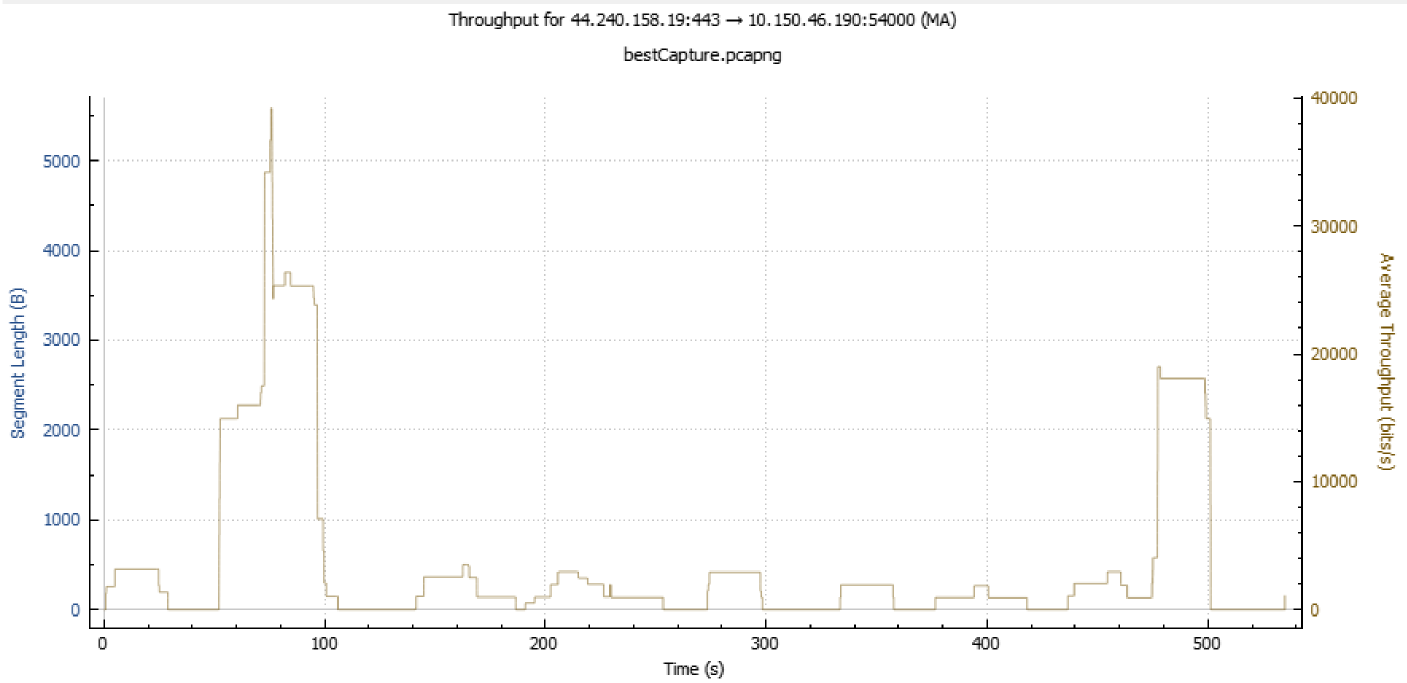
```
[SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 24327]
    [The RTT to ACK the segment was: 0.001776000 seconds]
    [iRTT: 0.002578000 seconds]
```

RTT-0.001776

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Total Packets	366				0.0007	100%	0.3400	79.574
rcode	366				0.0007	100.00%	0.3400	79.574
opcodes	366				0.0007	100.00%	0.3400	79.574
Query/Response	366				0.0007	100.00%	0.3400	79.574
Response	183				0.0003	50.00%	0.1700	79.634
Query	183				0.0003	50.00%	0.1800	79.574
Query Type	366				0.0007	100.00%	0.3400	79.574
HTTPS (HTTPS Specific Service Endpoints)	132				0.0002	36.07%	0.1700	79.574
A (Host Address)	234				0.0004	63.93%	0.1900	6.324
Class	366				0.0007	100.00%	0.3400	79.574
Service Stats	0				0.0000	100%	-	-
request-response time (msec)	183	46.07	0.910000	907.179016	0.0003		0.1700	79.634
no. of unsolicited responses	0				0.0000		-	-
no. of retransmissions	0				0.0000		-	-

DNS Request stats: 183 queries were sent. No packets were dropped.

Graph of average throughput wrt time



Throughput increased when videos were played and remained low at other times.

Analysis Table

	Lab (1:30 AM)	Central Library (2 PM)	Lab (9:30 PM)
Throughput (Kilobytes/s)	273	326	305
Round Trip Time (ms)	1.8	4.6	2.5
Avg. Packet Size (Bytes)	715	1073	795
No. of Packets Lost	0	0	0
No. of TCP Packets	9526	7863	8968
No. of UDP Packets	28	18	22
No. of Responses per Request Sent	5	3	4.3