1.a)

Attached below are snapshots of the 5 different network protocols

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.7.45.155	31.13.79.2	TLSv1.2	83	Application Data
2	0.020443	31.13.79.2	10.7.45.155	TCP	54	443 → 63878 [ACK] Seq=1 Ack=30 Win=709 Len=0
3	0.061163	172.217.20.206	10.7.45.155	UDP	68	443 → 65163 Len=26
4	0.100317	142.250.178.138	10.7.45.155	UDP	68	443 → 64616 Len=26
5	0.154673	31.13.79.2	10.7.45.155	TLSv1.2	79	Application Data

52	9.702091	10.7.45.155	42.99.128.153	TLSv1.3	348	Client Hello

130	15.622915	Cisco_bb:7c:c0	Broadcast	ARP	42	Gratuitous ARP for 0.0.0.0 (Request)

1055	39.873946	10.7.45.155	31.13.79.174	QUIC	115	0-RTT, DCID=7e2eee53e51f7d8f
1056	39.888571	31.13.79.174	10.7.45.155	QUIC	1274	Initial, SCID=9a1d006541c1b92e, PKN: 9742921, CRYPTO,…
1057	39.888571	31.13.79.174	10.7.45.155	QUIC	253	Handshake, SCID=9a1d006541c1b92e

1693	56.059496	fe80::6534:1ee9:da8…	ff02::16	ICMPv6	90	Multicast
1694	56.059640	10.7.45.155	224.0.0.22	IGMPv3	54	Membership
1695	56.068444	fe80::6534:1ee9:da8…	ff02::16	ICMPv6	90	Multicast
1696	56.068452	10.7.45.155	224.0.0.22	IGMPv3	54	Membership
1697	56.068711	fe80::6534:1ee9:da8…	ff02::16	ICMPv6	90	Multicast
1698	56.068953	10.7.45.155	224.0.0.22	IGMPv3	54	Membership
1699	56.069356	fe80::6534:1ee9:da8…	ff02::1:2	DHCPv6	156	Solicit XI

1) TLSv1.2 protocol

   TLS 1.2 is a cryptographic protocol that operates at the Transport Layer (layer 4) of the OSI model. It is used to secure communications between two applications by encrypting the data that is being transmitted.

   The RFC number for TLS 1.2 is RFC 5246.

2) TLSv1.3 protocol

   TLS 1.3 is the latest version of the Transport Layer Security (TLS) protocol, which is used to secure communications between two applications over the Internet. TLS 1.3 operates at the Transport Layer (layer 4) of the OSI model. It is used to secure communications between two applications by encrypting the data that is being transmitted.

   The RFC number for TLSv 1.3 is RFC 8446.

3) ARP protocol
   The Address Resolution Protocol (ARP) is a network protocol that maps an IP address to a physical address, such as a Media Access Control (MAC) address. This is necessary because IP addresses are used to identify hosts on a network, while MAC addresses are used to identify devices on a network. ARP operates at the Data Link Layer (layer 2) of the OSI model. It works by broadcasting an ARP request message to all devices on the network.

   The RFC number for ARP is RFC 826.

4) QUIC Protocol

QUIC stands for Quick UDP Internet Connections. It is a new transport layer protocol that is designed to improve the performance and security of web applications. QUIC is based on UDP, but it adds a number of features that make it more reliable and secure. QUIC operates at the Transport Layer (layer 4) of the OSI model. It is designed to replace TCP as the default transport protocol for web applications.

The RFC number for QUIC is RFC 9000.

5) ICMPv6 protocol

The Internet Control Message Protocol version 6 (ICMPv6) is a network layer protocol that is used to send control messages between IPv6 nodes.

ICMPv6 operates at the Network Layer (layer 3) of the OSI model. It is defined in RFC 4443.

b)

1	0.000000	10.7.45.155	31.13.79.2	TLSv1.2	83 Application Data
2	0.020443	31.13.79.2	10.7.45.155	TCP	54 443 → 63878 [ACK] Seq=1 Ack=30 Win=709 Len=0
3	0.061163	172.217.20.206	10.7.45.155	UDP	68 443 → 65163 Len=26
4	0.100317	142.250.178.138	10.7.45.155	UDP	68 443 → 64616 Len=26
5	0.154673	31.13.79.2	10.7.45.155	TLSv1.2	79 Application Data

From data point 1& 5 we can see the RTT for the process using TLSv1.2 is 0.154673 seconds.

2)

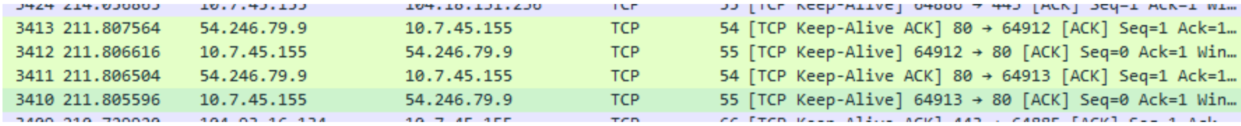
The public IP address of google server is 142.250.192.142, which is found by ping google.com in command terminal. Attached below is the snapshot of the packet information captured by wireshark on host machine.

337	17.813673	10.7.45.155	142.250.192.142	QUIC	1292 Initial, DCID=852e07f4c8921173, PKN: 1, PADDING, …
339	17.813869	10.7.45.155	142.250.192.142	QUIC	122 0-RTT, DCID=852e07f4c8921173
347	17.885351	10.7.45.155	142.250.192.142	QUIC	121 Handshake, DCID=e52e07f4c8921173
348	17.885566	10.7.45.155	142.250.192.142	QUIC	73 Protected Payload (KP0), DCID=e52e07f4c8921173
351	17.885836	10.7.45.155	142.250.192.142	QUIC	868 Protected Payload (KP0), DCID=e52e07f4c8921173
362	17.895503	10.7.45.155	142.250.192.142	QUIC	74 Protected Payload (KP0), DCID=e52e07f4c8921173
371	17.922042	10.7.45.155	142.250.192.142	QUIC	74 Protected Payload (KP0), DCID=e52e07f4c8921173
374	17.959289	10.7.45.155	142.250.192.142	QUIC	77 Protected Payload (KP0), DCID=e52e07f4c8921173
375	17.959962	10.7.45.155	142.250.192.142	QUIC	72 Protected Payload (KP0), DCID=e52e07f4c8921173
377	17.960244	10.7.45.155	142.250.192.142	QUIC	78 Protected Payload (KP0), DCID=e52e07f4c8921173
394	17.985817	10.7.45.155	142.250.192.142	QUIC	74 Protected Payload (KP0), DCID=e52e07f4c8921173

The public IP address of GitHub server is 20.207.73.82 , which is found by ping google.com in command terminal. Attached below is the snapshot of the packet information captured by wireshark on host machine.

156	8.764876	10.7.45.155	20.207.73.82	TCP	55 64731 → 443 [ACK] Seq=1 Ack=1 Win=515 Len=1 [TCP …
2158	29.259820	10.7.45.155	20.207.73.82	TCP	54 64675 → 443 [ACK] Seq=1 Ack=65 Win=515 Len=0
2159	29.259927	10.7.45.155	20.207.73.82	TCP	54 64675 → 443 [FIN, ACK] Seq=1 Ack=65 Win=515 Len=0
3032	33.465067	10.7.45.155	20.207.73.82	TLSv1.2	141 Ignored Unknown Record
3033	33.465127	10.7.45.155	20.207.73.82	TLSv1.2	93 Application Data
3050	33.478121	10.7.45.155	20.207.73.82	TCP	54 64731 → 443 [ACK] Seq=128 Ack=40 Win=515 Len=0
3069	33.742514	10.7.45.155	20.207.73.82	TCP	54 64731 → 443 [ACK] Seq=128 Ack=5123 Win=515 Len=0
3078	33.743166	10.7.45.155	20.207.73.82	TCP	54 64731 → 443 [ACK] Seq=128 Ack=15063 Win=515 Len=0
3079	33.744222	10.7.45.155	20.207.73.82	TLSv1.2	89 Application Data
3103	33.744588	10.7.45.155	20.207.73.82	TCP	54 64731 → 443 [ACK] Seq=163 Ack=39032 Win=515 Len=0
3113	33.744958	10.7.45.155	20.207.73.82	TCP	54 64731 → 443 [ACK] Seq=163 Ack=46660 Win=515 Len=0

The public IP address of Netflix.com server is 10.7.45.155 , which is found by ping google.com in command terminal. Attached below is the snapshot of the packet information captured by wireshark on host machine.



3413	211.807564	54.246.79.9	10.7.45.155	TCP	54	[TCP Keep-Alive ACK] 80 → 64912 [ACK] Seq=1 Ack=1…
3412	211.806616	10.7.45.155	54.246.79.9	TCP	55	[TCP Keep-Alive] 64912 → 80 [ACK] Seq=0 Ack=1 Win…
3411	211.806504	54.246.79.9	10.7.45.155	TCP	54	[TCP Keep-Alive ACK] 80 → 64913 [ACK] Seq=1 Ack=1…
3410	211.805596	10.7.45.155	54.246.79.9	TCP	55	[TCP Keep-Alive] 64913 → 80 [ACK] Seq=0 Ack=1 Win…

Difference between QUIC and TCP are summarized under: -

QUIC	TCP
features a faster, one-round trip connection establishment, reducing initial connection latency. It combines the handshake and encryption setup in a single step, making it more efficient for quickly establishing connections.	TCP requires a three-way handshake to establish a connection. This involves a series of messages exchanged between the client and server to ensure reliability and order of data transmission. This process can introduce latency.
QUIC inherently supports multiplexing. It allows multiple streams to be sent concurrently within a single connection, eliminating head-of-line blocking issues.	Multiplexing multiple streams over a single connection in TCP is accomplished using techniques like HTTP/2 or HTTP/3 (over TLS).
QUIC incorporates encryption from the start. It uses Datagram Transport Layer Security (DTLS) for encryption, and this encryption is applied to the entire communication, including the initial handshake.	Encryption in TCP typically relies on TLS (Transport Layer Security) for secure communication. While TLS provides strong security, it can introduce some overhead due to the initial handshake.
QUIC was initially developed by Google and is commonly used for web services, particularly Google services (HTTP/3 is based on QUIC). It excels in situations where low-latency and high-performance are critical, such as real-time communication and video streaming.	TCP is widely used for various applications, including web browsing, email, file transfers, and more. It's a versatile protocol suitable for most scenarios.

Similarity between QUIC and TCP

QUIC	TCP
QUIC also offers reliability. Like TCP, QUIC includes mechanisms for retransmitting lost packets and ensuring the correct order of data delivery. It aims to maintain data integrity, similar to TCP.	TCP is known for its reliability. It provides guaranteed, in-order delivery of data packets. When data is sent over TCP, the protocol ensures that the data arrives at its destination without errors and in the correct order.
QUIC incorporates flow control as well. It uses a similar flow control mechanism to prevent congestion and optimize the transfer of data between sender and receiver.	TCP includes flow control mechanisms to prevent congestion and ensure that the sender doesn't overwhelm the receiver with data. It uses techniques like TCP sliding window to manage the flow of data.
QUIC also includes error detection and correction mechanisms, such as checksums and	TCP employs error detection and correction mechanisms, including checksums,

retransmissions, to maintain data integrity and reliability.	acknowledgments, and retransmissions, to ensure that data is transmitted accurately and reliably.
QUIC integrates encryption from the start. It uses Datagram Transport Layer Security (DTLS) for encryption, ensuring data privacy and security.	TCP itself does not provide encryption. If encryption is desired, it is typically implemented at a higher layer using protocols like TLS (Transport Layer Security) when using HTTPS.

3)

Attached below are the observed attributes of the cookies of *eoffice.iitgn.ac.in*

Name	Value	Domain	Path	Expires / Max...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
AIT	a022bc4674047e438427078785c942a1	eoffice.iitgn.a...	/	2023-10-10T...	35	✓	✓			Medium
PHPSESSID	jpfh895e8fpbiapqf7h6jt1e7k	eoffice.iitgn.a...	/	Session	35					Medium
_ga	GA1.3.1352712354.1663782364	.iitgn.ac.in	/	2024-05-16T...	30					Medium

References:-

1) YouTube tutorial on socket programming
2) https://www.opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/
3) https://www.geeksforgeeks.org/socket-programming-cc/