CS342 Computer Networks Assignment 2 (Skype Application)

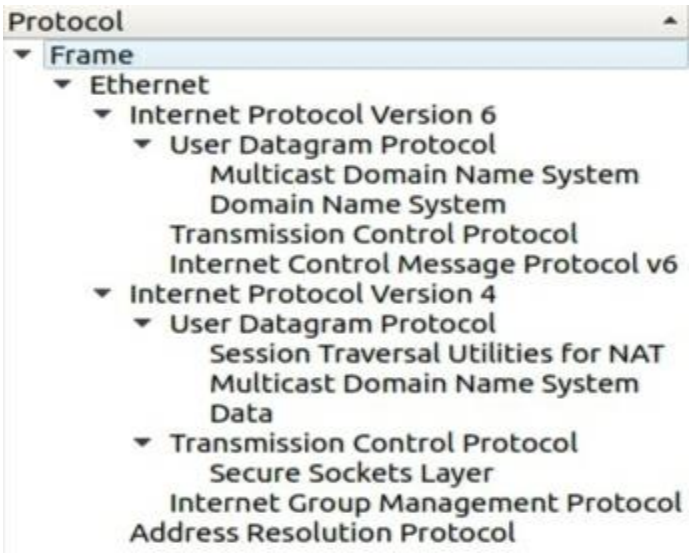
Name: Samay Varshney *Roll No: 180101097*

My main source of internet for all readings was Jio No-Proxy Hotspot.

Traces Link:

https://drive.google.com/drive/folders/10KSrXjJIzByXVwyznALsYkm4ovSipv8M?usp=sharing

Answer 1: All the protocols used by the application at different layers are listed below:



Protocols in Different Layers:

Transport Layer: TCP (Transmission Control Protocol), ICMPv6 (Internet Control Message Protocol v6), UDP (User Datagram Protocol)
Network Layer: ARP (Address Resolution Protocol), IGMP (Internet Group Management Protocol)
Application Layer: DNS (Domain Name System), mDNS (Multicast Domain Name System), STUN (Session Traversal Utilities for NAT)
Secure Socket/Session Layer: TLSv1.2 (Transport Layer Security v1.2), SSLv2 (Secure Sockets Layer)

Flags and options for packet format of all protocols in the application are:
Source port: Port number associated with the sender side.
Destination port: Port number associated with the recipient side.
Sequence number: Unique values that are used to ensure reliable delivery of data.
Acknowledgement number: Response from the receiver side as a part of the confirmation process that the packet was successfully received.
Data offset: Indicates where the data packet begins and also gives the length of the TCP header.
Flags: There are various types of flag bits present. They initiate connection, carry data and tear down connections. They are as follows:

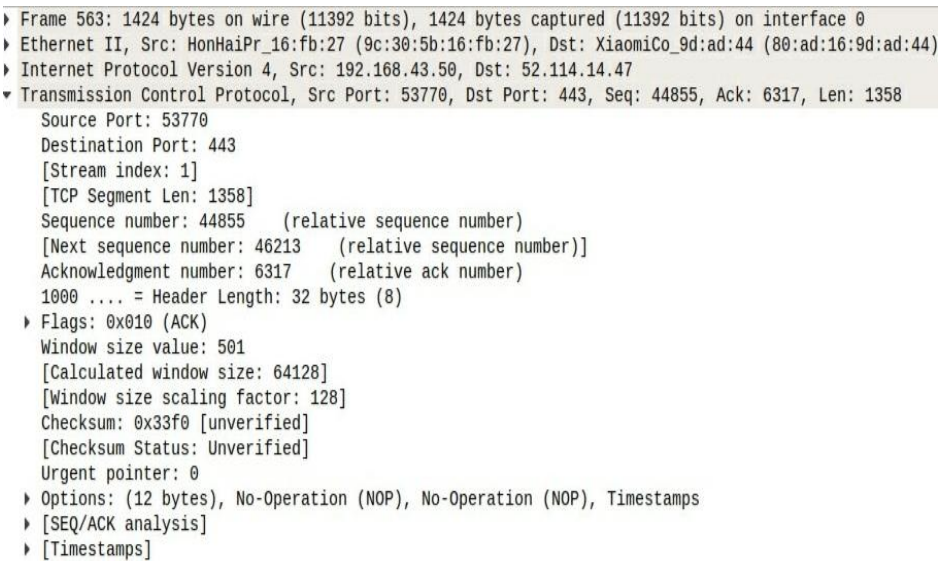
1) **SYN (synchronize):** Packets that are used to initiate a connection that is commonly known as the handshake process.
2) **ACK (acknowledgement):** These packets are used to confirm that the data packets have been received, and this also confirms the initiation and tear down of the connections.
3) **RST (reset):** These packets signify that the connection you were trying to create has been shut down or may be the application we were trying to communicate with is not accepting connections.
4) **FIN (finish):** These packets indicate that the connection is being torn down after the successful delivery of data packets.
5) **PSH (push):** These packets indicate that the incoming data should be passed on directly to the application instead of getting buffered.
6) **URG (urgent)**: Marked packets indicate that the data that the packet is carrying should be processed immediately by the TCP stack and the urgent pointer field should be examined if it is set.
7) **CWR (Congestion Window Reduced):** These packets are used by the sender to inform the receiver that the buffer is getting overfilled and because of congestion, both the parties should slow down the transmission process to avoid any packet loss that might happen.

Window size: This field in the header indicates the amount of data that the sender can send.

Checksum: To cross check the contents of the TCP segments.
Urgent pointer: Tells us about the value that the urgent pointer contains. It specifically indicates the sequence number of the octet that lies before the data.
Options: This field has three parts: length of the option, options being used, options in use. One of the important options **Maximum Segment Size (MSS)** is also part of this field.
Data: The last part in the TCP header is the real data that travels around.

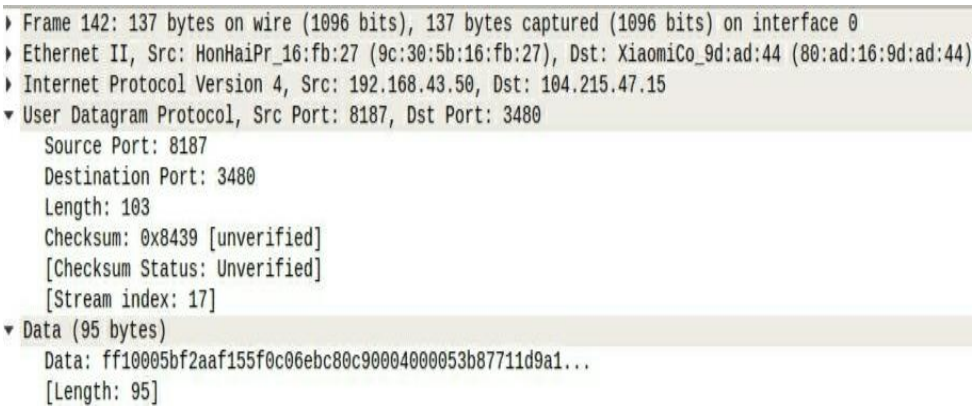
Source Ethernet Address: **9c:30:5b:16:fb:27**, Destination Ethernet Address: **80:ad:16:9d:ad:44** in each case below.

TCP:



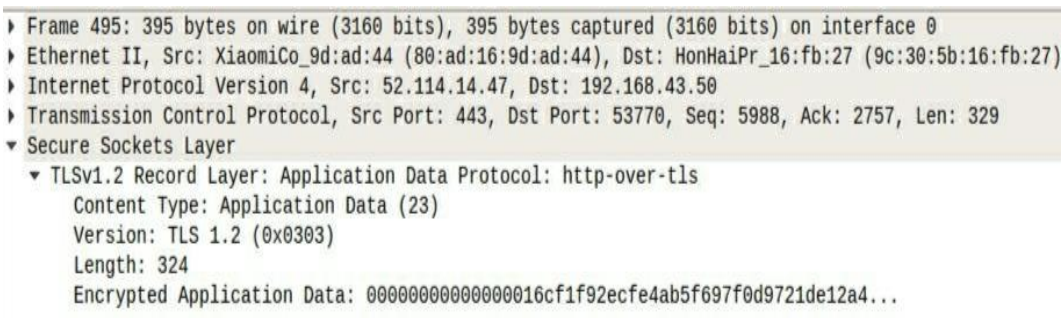
The packet contains the Destination Port (**443**), Source Port (**53770**), TCP Stream index (**1**), sequence number (**44855**) and the acknowledgement number (**6317**), Header length, Flags etc. In the following figure, the flags are set as **0x010**. Window size value is **493.** Checksum is used for error detection. Wireshark is remembering the value of Window size scaling factor and presenting it again. Scaling factor shows the number of leftward bits shift that should be used for an advertised window size.

UDP:



Packets contains Source Port (**8187**), Destination Port (**3480**), checksum status (**0x8439**), length (**103**) of the packet. Data is given (**95 bytes**) in string format. **UDP** is a communication **protocol** that is primarily **used for** establishing low-latency and loss-tolerating connections between applications on the internet. It speeds up transmissions by enabling the transfer of data before an agreement is provided by the receiving party.

TLSv1.2:



Transport Layer Security (TLS) are cryptographic protocols that provide communications security over a computer network. Packet contains Source Port (**443**), Destination Port (**53770**), length (**324**), encrypted data, version (**1.2**). TLS is

implemented on two levels- The TLS Record protocol and the TLS Handshake protocol to ensure **security, efficiency and extensibility**.

DNS: The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. Transaction Id (**0x3a56**), Flags(**0x0100**), queries, additional records are there in the packet.

```
▸ Frame 435: 101 bytes on wire (808 bits), 101 bytes captured (808 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_16:fb:27 (9c:30:5b:16:fb:27), Dst: XiaomiCo_9d:ad:44 (80:ad:16:9d:ad:44)
▸ Internet Protocol Version 4, Src: 192.168.43.50, Dst: 8.8.8.8
▸ User Datagram Protocol, Src Port: 49784, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0x3a56
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▸ ea-prod-msgsearch.cloudapp.net: type AAAA, class IN
  ▾ Additional records
    ▸ <Root>: type OPT
```

STUN:

```
▸ Frame 144: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
▸ Ethernet II, Src: XiaomiCo_9d:ad:44 (80:ad:16:9d:ad:44), Dst: HonHaiPr_16:fb:27 (9c:30:5b:16:fb:27)
▸ Internet Protocol Version 4, Src: 104.215.47.15, Dst: 192.168.43.50
▸ User Datagram Protocol, Src Port: 3480, Dst Port: 8187
▾ Session Traversal Utilities for NAT
  ▸ Message Type: 0x0101 (Binding Success Response)
    Message Length: 60
    Message Cookie: 2112a442
    Message Transaction ID: 53e91f227139704884a14587
  ▾ Attributes
    ▸ MS-IMPLEMENTATION-VERSION: Unknown (0x7)
    ▸ XOR-MAPPED-ADDRESS: 47.31.141.207:8187
    ▸ Unknown
    ▸ MESSAGE-INTEGRITY
    ▸ FINGERPRINT
```

It contains information about messages type (**0x0101**), length (**60**), transaction Id, cookie for IP and XOR-MAPPED-ADDRESS and binding success response. In order to get the public IP address, we use the **STUN** Protocol. It uses the **XOR Mapped Address** attribute to indicate its reflexive transport address which further identifies the public address of that client as seen by a protocol server. The address is communicated to the protocol client through the XOR MAPPED ADDRESS attribute in a success response message.

ARP:

```
▸ Frame 642: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: XiaomiCo_9d:ad:44 (80:ad:16:9d:ad:44), Dst: HonHaiPr_16:fb:27 (9c:30:5b:16:fb:27)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: XiaomiCo_9d:ad:44 (80:ad:16:9d:ad:44)
    Sender IP address: 192.168.43.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.43.50
```

Address Resolution Protocol (ARP) is a protocol for mapping an Internet Protocol address (IP address) to a physical machine address that is recognized in the local network. Sender's and receiver's MAC and IP address are mentioned as shown in figure. Protocol type is **IPv4**.

SSLv2:

```
▸ Frame 264: 1424 bytes on wire (11392 bits), 1424 bytes captured (11392 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_16:fb:27 (9c:30:5b:16:fb:27), Dst: XiaomiCo_9d:ad:44 (80:ad:16:9d:ad:44)
▸ Internet Protocol Version 4, Src: 192.168.43.50, Dst: 52.114.14.47
▸ Transmission Control Protocol, Src Port: 53770, Dst Port: 443, Seq: 495671, Ack: 1, Len: 1358
▸ [6 Reassembled TCP Segments (6664 bytes): #257(758), #258(1358), #259(1358), #261(1358), #263(1358),
▾ Secure Sockets Layer
  ▾ SSLv2 Record Layer: Encrypted Data
      [Version: SSL 2.0 (0x0002)]
```

The SSL version is the language the client and server will use to talk with each other. It controls the encryption process by **encrypting data**. In the application its version used is **2.0**.

Answer 2:

The important functionalities of the skype application are initiating a video/phone call, terminating a video/phone call and sending videos/chat messages over a video conferencing call. **Main Protocols used:**

Initiating a phone/video call: UDP, STUN, TLSv1.2, DNS, ARP

Terminating a phone/video call: TLSv1.2, ICMP, DNS, UDP, TCP

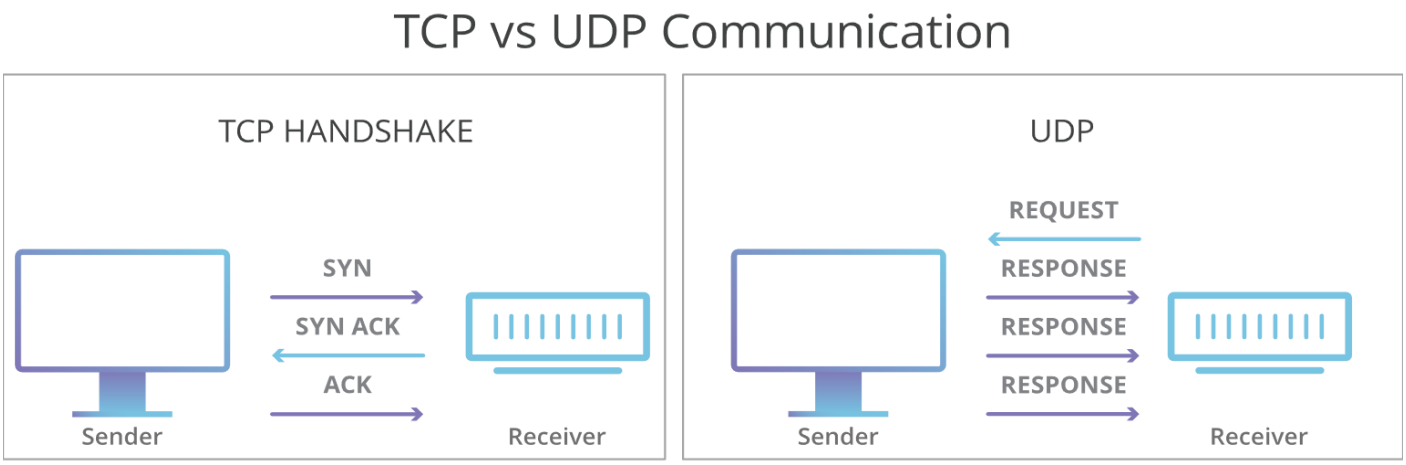
Sending video/chat messages over conference call: SSLv2, TCP, TLSv1.2

Firstly, whenever we initiate a phone or video call, host to IP lookup is triggered using DNS protocol (UDP) to connect to the server. We may also have some ARP packets due to our ethernet/wireless connection's broadcast messages.

Then UDP is used in initiating and resuming video or phone calls as these are time-sensitive transmissions and designed to handle some sort of loss and UDP accomplishes this process in a simple fashion: it sends packets (units of data transmission) directly to a target computer, without establishing a connection first, indicating the order of said packets or checking whether they arrived as intended. This reduces the time of not checking for connections but also results in some packet loss. In video and phone calls we wanted to get the response from different users as soon as possible and if some data loss happens in between, it doesn't matters much, so UDP is the best protocol used for this purpose and that's why in initiating and terminating phone/video calls UDP is used.

After signaling, in skype, for having a successful phone/video call we need to connect all the different clients peer to peer. And for connecting, we must have the public IP address of the clients. So, in order to get the public IP address, we use the **STUN** Server Protocol. We get the public IP addresses of the clients and then, we connect the clients through something similar to WebRTC **(which uses UDP)** to start the phone/video call. It handles all the media streaming.

In sending video/chat messages what is usually more important is the sequence of sent messages and their accuracy compromising the time delay. This is actually achieved through **TCP protocol**. In a TCP communication, the two computers begin by establishing a connection via an automated process called a **handshake**. Only once this handshake has been completed will one computer actually transfer data packets to the other. TCP responsibility includes end-to-end message transfer independent of the underlying network and structure of user data, along with error control, segmentation, **flow control**, and helps to minimize **traffic congestion** control. It addresses numerous **reliability** issues in providing a reliable byte stream: data arrives in-order, data has minimal error (i.e., correctness), duplicate data is discarded and lost or discarded packets are resent.



Apart from this, **TLSv1.2** is used by skype to ensure user data much safer. TLS and UDP were designed to operate on top of a reliable transport protocol such as TCP. Web servers use cookies to identify the web user. They are small piece of data stored into the web user's disk. TLS is used to protect **session cookies** on the rest of the sites from being intercepted to protect user accounts. By not using TLS 1.2, the transactions that users do on the skype (like upgrading to premium etc.) will not remain safer and will become vulnerable to threats like exposing of your details to hackers. TLS allows the peers to negotiate **a shared secret key** without having to establish any prior knowledge of each other, and to do so over an unencrypted channel. TLS provides verification of **identity** of server, which is as important as encryption. The goals of the TLS protocol are cryptographic **security**, **extensibility**, and relative **efficiency**. These goals are achieved through implementation of the TLS protocol on **two** levels: the TLS Record protocol and the TLS Handshake protocol.

Also, while sending video/chat messages, since we are connecting through internet here, we want to avoid hackers snooping packets on wire, so skype goes for something **like SSLv2**. Actually, application's requirement determines the

selection of protocol. Here SSLv2 ensures that even the people at the skype server are unable to view the data. Hence, SSLv2 is used here so that encrypted data is transferred.

In Skype, **ICMP protocol** is used for error reporting or to perform network diagnostics. When a user sends some data to the other user using video/audio conferencing, the ICMP generates errors to share with the sending device in the event that any of the data did not get to its intended destination. For example, if a user sends a much big file (not supported in skype) to other user then the router will drop the packet and send an ICMP message back to the original source for the data.

Answer 3: The skype application was tested under different conditions and the behavior was observed in each. It checks network connections to see what protocol it can efficiently deploy to make the communication between the peer computers efficient. If the computers are in the same LAN, there is no need to route data through ports. But if the data is being sent through Skype server over the internet, then the data is encrypted and sent through SSLv2 ports so that not even the people at Skype can decipher and look at data. First the **DNS** server is used to connect to the server and it used our IPv6 address having value 2409:4050:2e90:aa62:dcf0:d46b:1372:1ef4 as source and 2409:4050:2e90:aa62::75 as destination for server's IPv6 address as shown in below figure.

3-Way TCP Handshake Message sequence observed:

Time	Source	Destination	Protocol	Length	Info
14.629805618	40.82.159.28	192.168.43.50	TCP	66	443 → 47033 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1370 WS=2
14.629880094	192.168.43.50	40.82.159.28	TCP	54	47033 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
14.629912675	40.82.159.28	192.168.43.50	TCP	66	443 → 26994 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1370 WS=2
14.629946014	192.168.43.50	40.82.159.28	TCP	54	26994 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
14.629963603	40.82.159.28	192.168.43.50	STUN	195	Allocate Success Response lifetime: 60 MAPPED-ADDRESS: 20.194.12
14.630496383	192.168.43.50	40.82.159.28	TLSv1	104	Client Hello
14.630631601	192.168.43.50	40.82.159.28	TLSv1	104	Client Hello
14.798910310	2409:4050:2e82:c2f:…	2409:4050:2e82:c2f:…	DNS	97	Standard query 0x0e23 AAAA api3.cc.skype.com

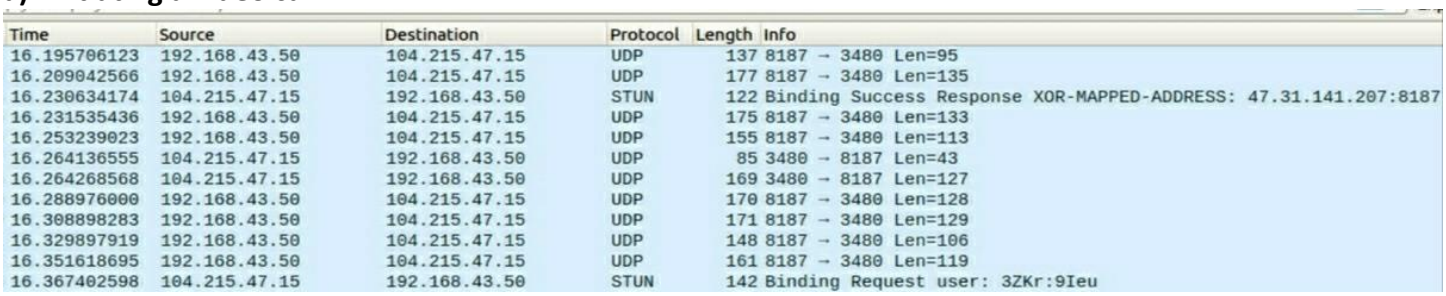
Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. Establishing a normal TCP connection requires three separate steps: **1) SYN:** The client sending a SYN to the server performs the active open. The client sets the segment's sequence number to a random value A. **2) SYN-ACK:** In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B. **3) ACK:** Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgment value i.e. A+1, and the acknowledgment number is set to one more than the received sequence number i.e. B+1. So, first a 3 Way Handshake message sequence is observed in Skype.

TLS Handshaking Message Sequence observed:

Time	Source	Destination	Protocol	Length	Info
14.986719445	192.168.43.50	52.114.15.58	TCP	54	37355 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
14.988156366	192.168.43.50	52.114.15.58	TLSv1.2	353	Client Hello
15.024696915	40.82.159.28	192.168.43.50	TCP	54	443 → 26994 [ACK] Seq=84 Ack=52 Win=524544 Len=0
15.029698529	40.82.159.28	192.168.43.50	TCP	54	443 → 47033 [ACK] Seq=84 Ack=52 Win=524544 Len=0
15.114872334	52.114.15.58	192.168.43.50	TCP	1424	443 → 37355 [ACK] Seq=1 Ack=300 Win=524288 Len=1370 [TCP segment
15.114930397	192.168.43.50	52.114.15.58	TCP	54	37355 → 443 [ACK] Seq=300 Ack=1371 Win=64128 Len=0
15.116092201	52.114.15.58	192.168.43.50	TCP	1424	443 → 37355 [ACK] Seq=1371 Ack=300 Win=524288 Len=1370 [TCP segm
15.116123417	192.168.43.50	52.114.15.58	TCP	54	37355 → 443 [ACK] Seq=300 Ack=2741 Win=64128 Len=0
15.116504926	52.114.15.58	192.168.43.50	TLSv1.2	1478	Server Hello, Certificate, Server Key Exchange, Server Hello Don
15.116539089	192.168.43.50	52.114.15.58	TCP	54	37355 → 443 [ACK] Seq=300 Ack=4165 Win=63104 Len=0
15.127992360	192.168.43.50	52.114.15.58	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Mes
15.341531632	52.114.15.58	192.168.43.50	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message

After the handshake message sequences, TLSv1.2 connection starts. This protocol is used to exchange all the information required by both sides. TLS Handshake protocol allows authenticated communication to commence between the server and client. This protocol allows the client and server to speak the same language, allowing them to agree upon an encryption algorithm and encryption keys before the selected application protocol begins to send data. Different messages like Client Hello, Server Hello, Certificate, Server Key Exchange, Server Hello Done, Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message b/n client (192.168.43.50) and server (52.114.15.58) are seen in above figure which shows this handshaking sequence.

a) Initiating a video call:

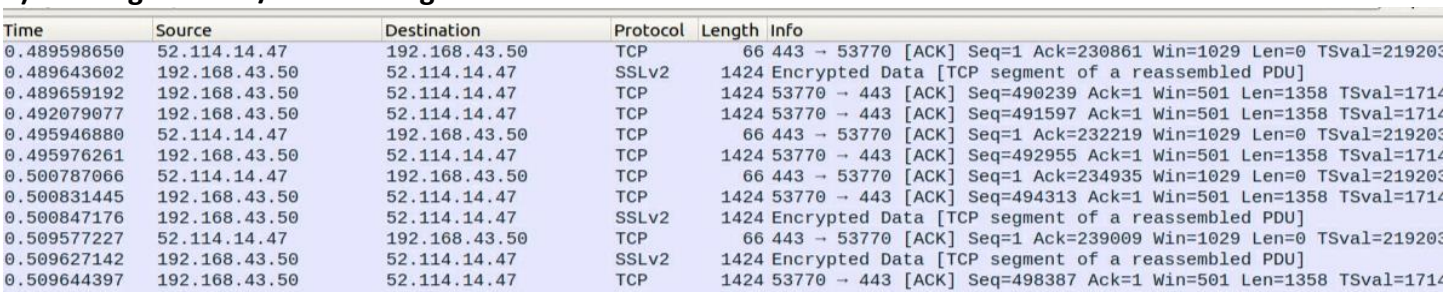


Time	Source	Destination	Protocol	Length	Info
16.195706123	192.168.43.50	104.215.47.15	UDP	137	8187 → 3480 Len=95
16.209042566	192.168.43.50	104.215.47.15	UDP	177	8187 → 3480 Len=135
16.230634174	104.215.47.15	192.168.43.50	STUN	122	Binding Success Response XOR-MAPPED-ADDRESS: 47.31.141.207:8187
16.231535436	192.168.43.50	104.215.47.15	UDP	175	8187 → 3480 Len=133
16.253239023	192.168.43.50	104.215.47.15	UDP	155	8187 → 3480 Len=113
16.264136555	104.215.47.15	192.168.43.50	UDP	85	3480 → 8187 Len=43
16.264268568	104.215.47.15	192.168.43.50	UDP	169	3480 → 8187 Len=127
16.288976000	192.168.43.50	104.215.47.15	UDP	170	8187 → 3480 Len=128
16.308898283	192.168.43.50	104.215.47.15	UDP	171	8187 → 3480 Len=129
16.329897919	192.168.43.50	104.215.47.15	UDP	148	8187 → 3480 Len=106
16.351618695	192.168.43.50	104.215.47.15	UDP	161	8187 → 3480 Len=119
16.367402598	104.215.47.15	192.168.43.50	STUN	142	Binding Request user: 3ZKr:9Ieu

For having a video/phone call successfully, **UDP** protocol is used and packets of different length as mentioned in the figure are transferred from our IPv4 address to server and from server to our IP. Here 192.168.43.50 is our IP and 104.215.47.15 is corresponding server's IP from which we are getting packets for video of other users. The 8187 → 3480 value in the Info in the first row indicates the port number from source → destination.

STUN protocol uses the **XOR Mapped Address** attribute to indicate to the protocol client its reflexive transport address which further identifies the public address of that client as seen by a protocol server. The address is communicated to the protocol client through the XOR MAPPED ADDRESS attribute in a success response message as shown in the figure.

b) Sending of video/chat messages in a call:



Time	Source	Destination	Protocol	Length	Info
0.489598650	52.114.14.47	192.168.43.50	TCP	66	443 → 53770 [ACK] Seq=1 Ack=230861 Win=1029 Len=0 TSval=219203
0.489643602	192.168.43.50	52.114.14.47	SSLv2	1424	Encrypted Data [TCP segment of a reassembled PDU]
0.489659192	192.168.43.50	52.114.14.47	TCP	1424	53770 → 443 [ACK] Seq=490239 Ack=1 Win=501 Len=1358 TSval=1714
0.492079077	192.168.43.50	52.114.14.47	TCP	1424	53770 → 443 [ACK] Seq=491597 Ack=1 Win=501 Len=1358 TSval=1714
0.495946880	52.114.14.47	192.168.43.50	TCP	66	443 → 53770 [ACK] Seq=1 Ack=232219 Win=1029 Len=0 TSval=219203
0.495976261	192.168.43.50	52.114.14.47	TCP	1424	53770 → 443 [ACK] Seq=492955 Ack=1 Win=501 Len=1358 TSval=1714
0.500787066	52.114.14.47	192.168.43.50	TCP	66	443 → 53770 [ACK] Seq=1 Ack=234935 Win=1029 Len=0 TSval=219203
0.500831445	192.168.43.50	52.114.14.47	TCP	1424	53770 → 443 [ACK] Seq=494313 Ack=1 Win=501 Len=1358 TSval=1714
0.500847176	192.168.43.50	52.114.14.47	SSLv2	1424	Encrypted Data [TCP segment of a reassembled PDU]
0.509577227	52.114.14.47	192.168.43.50	TCP	66	443 → 53770 [ACK] Seq=1 Ack=239009 Win=1029 Len=0 TSval=219203
0.509627142	192.168.43.50	52.114.14.47	SSLv2	1424	Encrypted Data [TCP segment of a reassembled PDU]
0.509644397	192.168.43.50	52.114.14.47	TCP	1424	53770 → 443 [ACK] Seq=498387 Ack=1 Win=501 Len=1358 TSval=1714

After clicking on the sending button, initially connection is established using handshakes and then some sequences of messages were observed as shown in the above figure. The TCP segments were transferred from our computer → server and from server → computer continuously. Here the data is sent using SSLv2 (application layer) protocol. Hence, it is encrypted. All the acknowledgements are not encrypted and hence Wireshark displays their protocol as TCP only. TCP packets are sent from port number 443 to 53770 and vice versa.

Answer 4: Data is collected according to the 3 functionalities used above. Data is collected after putting filter in each file with IP's of Skype. **RTT**: Got from No of packets/Total time taken. **Number of UDP/TCP**: Got from protocol hierarchy.

Initiating a video call

	11:00 am	**3:00 pm**	**8:00 pm**
Throughput (bytes/sec)	8969	4794	7216
RTT (milliseconds)	18.832	11.809	26.45
Packet Size (bytes)	169	185	191
No of Packets Lost	0	0	0
No of UDP Packets	5827	2008	2310
No of TCP Packets	152	161	180
No of Responses per one request sent	0.0984	0.1403	0.1269

Sending a video message

	11:00 am	**3:00 pm**	**8:00 pm**
Throughput (bytes/sec)	495000	511000	739000
RTT (milliseconds)	1.947	1.892	1.274
Packet Size (bytes)	964	969	943
No of Packets Lost	0	0	0
No of UDP Packets	1083	914	995

No of TCP Packets	22255	19477	32687
No of Responses per one request sent	0.437	0.4297	0.4999

Terminating a video call

	11:00 am	**3:00 pm**	**8:00 pm**
Throughput (bytes/sec)	5235	17000	7885
RTT (milliseconds)	49.26	33.44	38.30
Packet Size (bytes)	258	594	406
No of Packets Lost	0	0	0
No of UDP Packets	480	165	212
No of TCP Packets	150	435	254
No of Responses per one request sent	0.2652	0.9742	0.5751

Answer 5: While performing the experiments, packets from **Skype** came from multiple destinations across the world. It can be due to **fast switching/load switching**, after a packet has been sent to the next hop, the **routing information** about how to get to the destination is stored in a fast cache. When the router receives another packet that is directed to the same destination, it uses the cache, which is faster than the traditional way. So, now suppose some router IP address is selected from the routing table which was used earlier and now is found to be inactive then another router IP address will be selected, so the routes to same host during the day **can be different** and content can be provided from **various locations**. It can also be due to **Geographic location** - Ideal scenario is for a server to be as close as possible to the customer or end user. It can be also due to **Maintenance Backup** of some servers at the time experiments were performed due to which different servers were chosen at different times of the day.

Listing few of IPs found at different times of the day:
11 am: 104.215.47.15, 52.114.14.47, 40.74.219.49, 52.114.15.58
3 pm: 104.215.6.69, 52.114.14.1, 40.74.219.49, 52.114.6.180
8 pm: 168.63.246.67, 52.114.6.105, 52.114.14.47, 40.83.97.152