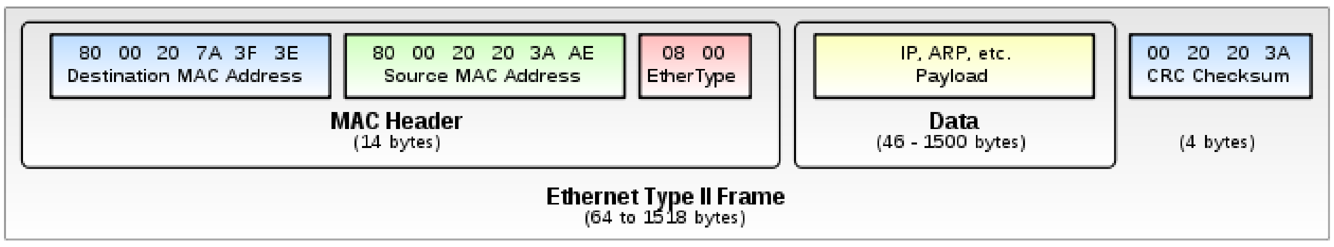
NAME :- ANURAG RAMTEKE
ROLL NUMBER:-150101010
NETWORKS LAB ASSIGNMENT 2

1) Following are the protocols used by "hotstar" at different layers as found using wireshark.

Ethernet------link protocol used in data link layer of the OSI model
Internet protocol version 4-------internet layer protocols
User datagram protocol------transport layer protocols
Domain Name System------Application layer protocol
Transmission control protocol------transport layer protocols
Secure sockets layer-------application layer protocols
Hypertext transfer protocol-------------application layer protocol

Description:-



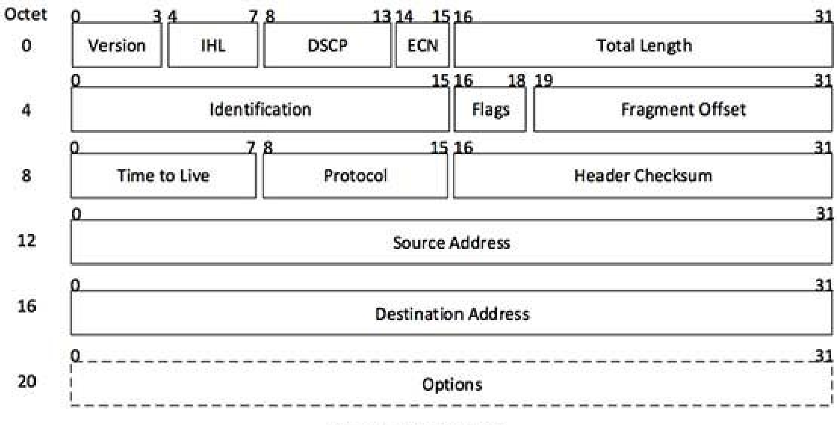
Ethernet:- This framing defines the two-octet EtherType field in an Ethernet frame, preceded by destination and source MAC addresses, that identifies an upper layer protocol encapsulated by the frame data.

ipv4:-

Internet Protocol being a layer-3 protocol (OSI) takes data Segments from layer-4 (Transport) and divides it into packets. IP packet encapsulates data unit received from above layer and add to its own header. The IPv4 packet header consists of 14 fields, of which 13 are required. The 14th field is optional and aptly named: options. It is a connectionless protocol for use on packet switched networks information. The encapsulated data is referred to as IP Payload. IP header contains all the necessary information to deliver the packet at the other end..**Version** no. of Internet Protocol used (e.g. IPv4).**IHL** means Internet Header Length; Length of entire IP header. **DSCP:** Differentiated Services Code Point; this is Type of Service.**ECN:** Explicit Congestion Notification; It carries information about the congestion seen in the route.**Total Length:** Length of entire IP Packet (including IP header and IP Payload).**Identification:** If IP packet is fragmented during the transmission, all the fragments contain same identification number. to identify original IP packet they belong to.**Flags:** As required by the network resources, if IP Packet is too large to handle, these 'flags' tells if they can be fragmented or not. In this 3-bit flag, the MSB is always set to '0'.**Fragment Offset:** This offset tells the exact position of the fragment in the original IP Packet.**Time to Live:** To avoid looping in the network, every packet is sent with some TTL value set, which tells the network how many routers (hops) this packet can cross. At each hop, its value is decremented by one and when the value reaches zero, the packet is discarded.**Protocol:** Tells the Network layer at the destination host, to which Protocol this packet belongs to, i.e. the next level Protocol. For example protocol number of ICMP is 1, TCP is 6 and UDP is 17.**Header Checksum:** This field is used to keep checksum value of entire header which is then used to check if the packet is received error-free.**Source Address:** 32-bit address of the Sender (or source) of the packet.**Destination Address:** 32-bit address of the Receiver (or destination) of the packet.**Options:** This is optional field, which is used if the value of IHL is greater than 5. These options may contain values for options such as Security, Record Route, Time Stamp, etc.

IP Header
Layer – 4 Data

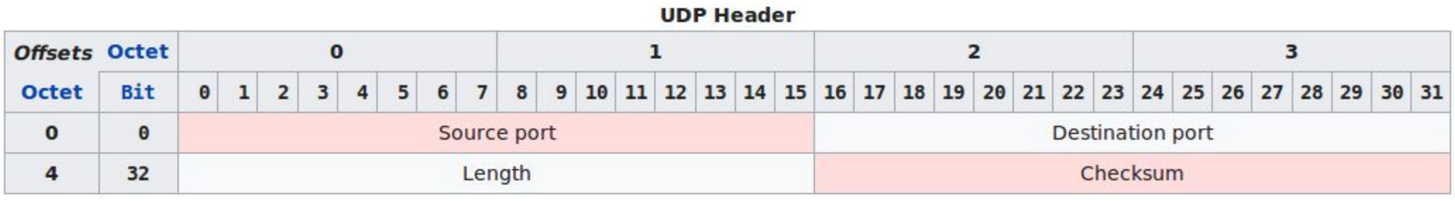
(IP Encapsulation)



[Image: IP Header]

UDP:-

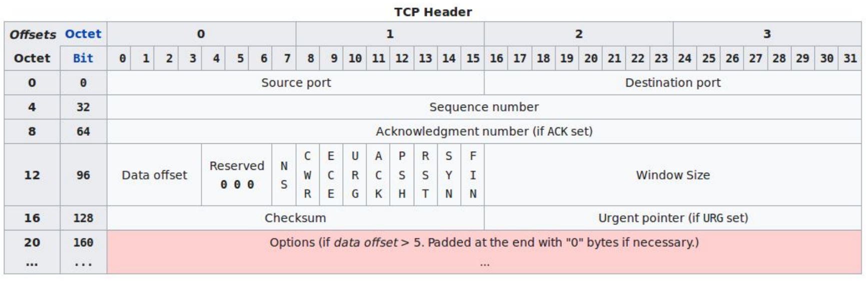
UDP uses a simple connectionless communication model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. The UDP header consists of 4 fields, each of which is 2 bytes (16 bits). The use of the fields "Checksum" and "Source port" is optional in IPv4 (pink background in table). **Source port** number identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number. **Destination port number** identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number. **Length** specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes because that is the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. However the actual limit for the data length, which is imposed by the underlying IPV4 protocol, is 65,507 bytes (65,535 − 8 byte UDP header − 20 byte IP header). **Checksum** may be used for error-checking of the header and data. This field is optional in IPv4, and mandatory in IPv6. The field carries all-zeros if unused.



UDP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

TCP:-TCP header contains 10 mandatory fields, and an optional extension field. It identifies source and destination ports and control data. The data section follows the header. Its contents are the payload data carried for the applicaiton. The length of the data section is not specified in the TCP segment header.



TCP Header

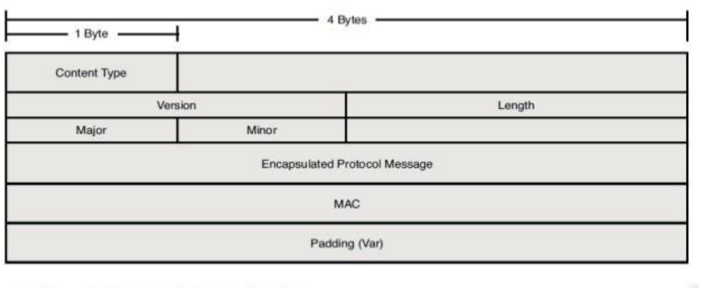
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

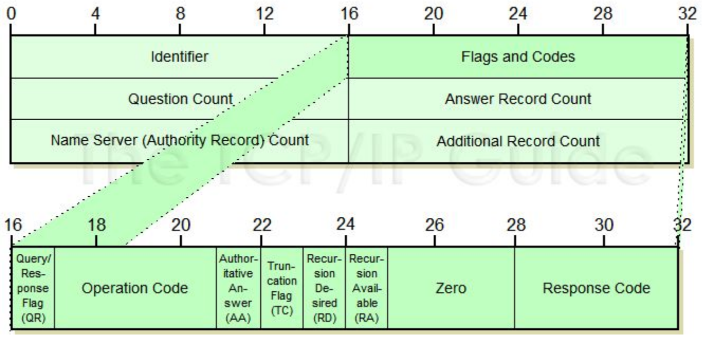
Source port (16 bits) identifies the sending port. **Destination port** (16 bits) identifies the receiving port. **Sequence number** (32 bits) has a dual role which is if the SYN flag is set

(1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1 else this is the accumulated sequence number of the first data byte of this segment for the current session. **Acknowledgment number** (32 bits):- the value of this field is the next sequence number that the sender of the ACK is expecting. **Data offset**(4 bits) specifies the size of the TCP header in 32-bit words. **Reserved** (3 bits) is for future use and should be set to zero. **Flags** (9 bits) (aka Control bits) Contains 9 1-bit flags which are NS, CWR,ECE, URG, ACK, PSH, RST, SYN, FIN. **Window size** (16 bits) is the size of the *receive window*, which specifies the number of window size units. **Checksum** (16 bits) is used for error-checking of the header, the Payload and a Pseudo-Header. **Urgent pointer** (16 bits) if the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte. Options (Variable 0–320 bits, divisible by 32). The length of this field is determined by the data offset field. Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable).

SSL:-SSL (TLSv1.2) or SSL/TLS is used in every browser worldwide to provide https (http secure) functionality. TLS is a transport layer cryptographic protocol that adds to the- security of the layer. A general SSL record format is given on the right with the understanding of stacking.

This is the general format of all TLS records.





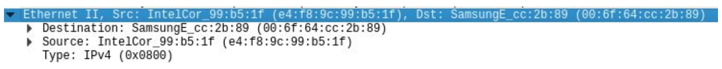
.**HTTP:**HTTP headers allow the client and the server to pass additional informatoin with the request or the response. There are varoius types of HTTP requests. A request header consists of its case-insensitive name folowed by a colon ':'. Then by its value (without line breaks). Leading white space before the value is ignored.

HTTP-message = <Request> | <Response> ; HTTP/1.1 messages

HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers which are **General-header** which have general applicability for both request and response messages. **Request-header** which has applicability only for request messages. **Response-header** which has applicability only for response messages. **Entity-header** defines meta information about the entity-body or, if no body is present, about the resource identified by the request.

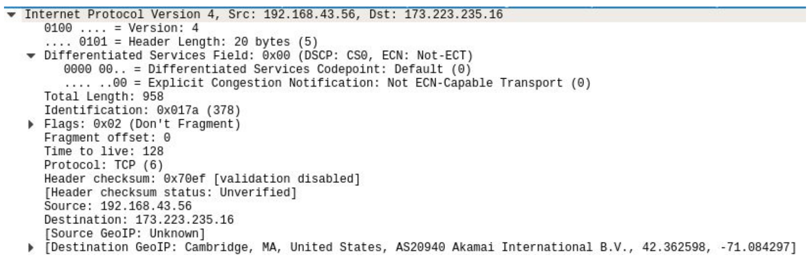
DNS:-The client/server information exchange in DNS is facilitated using query/response messaging. Both queries and responses have the same general format, containing up to five individual sections carrying information. Of these, two are usually found in both queries and responses: the *Header* section and the *Question* section.

2)



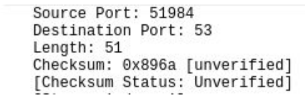
```
▾ Ethernet II, Src: IntelCor_99:b5:1f (e4:f8:9c:99:b5:1f), Dst: SamsungE_cc:2b:89 (00:6f:64:cc:2b:89)
   ▸ Destination: SamsungE_cc:2b:89 (00:6f:64:cc:2b:89)
   ▸ Source: IntelCor_99:b5:1f (e4:f8:9c:99:b5:1f)
     Type: IPv4 (0x0800)
```

Ethernet II:- From total of 14 bytes from the MAC Header, first 6 bytes in the header are **destination address** which specifies the MAC address of the destination/target system. The next 6 bytes specifies the MAC address of the **source/transmitting system** and the rest 2 bytes which tells about the **ethertype** is used to specify the higher level protocol frame encapsulated in the packet.



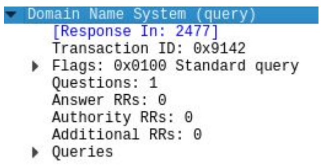
```
▾ Internet Protocol Version 4, Src: 192.168.43.56, Dst: 173.223.235.16
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 958
    Identification: 0x017a (378)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x70ef [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.43.56
    Destination: 173.223.235.16
    [Source GeoIP: Unknown]
  ▸ [Destination GeoIP: Cambridge, MA, United States, AS20940 Akamai International B.V., 42.362598, -71.084297]
```

IPv4:Version of the IP protocol used here is 4 for this frame, **IHL** is the length of entire IP header, 20 bytes in this case, **DSCP=CS0** implies best effort service, **ECN** is which requires specific support at both the Internet.Layer and the transport layer so ECN=not ECT implies Non ECN-capable Transport.**Total Length=**958 is the length of entire IP packet .**Identification:-** if packet is fragmented during the transmission, all the fragments contain same identification number to identify original IP packet they belong to. **Flags** as required by the network resources, if IP Packet is too large to handle, these 'flags' tells if they can be fragmented. Here the Don't Fragment bit is set, hence this frame will not be fragmented even if the IP packet is large. **Fragment Offset** tells the exact position of the fragment in the origianl IP packet. In this packe tit is set to 0. **Time to Live P:-**to avoid looping in the network, every packet is sent with some TTL value set, which tells the network ho wmany routers (hops) this packet can cross. At each hop, its value is decremented by one and when the value reaches zero, the packet is discarded. Protocol tells the Network layer at the destination host, to which Protocol this packet belongs to. For example protocol number of ICMP is 1, TCP is 6 and UDP is 17. **Header Checksum:** This field is used to keep checksum value of entire header which is then used to check if the packet is received error-free. This also contains a pseudo header which again contains the source and destination IP addresses. **Source Address: Options:** This is optional field, which is used if the value of IHL is greater than 5. These options may contain values for options such as Security, Record Route, Time Stamp, etc.



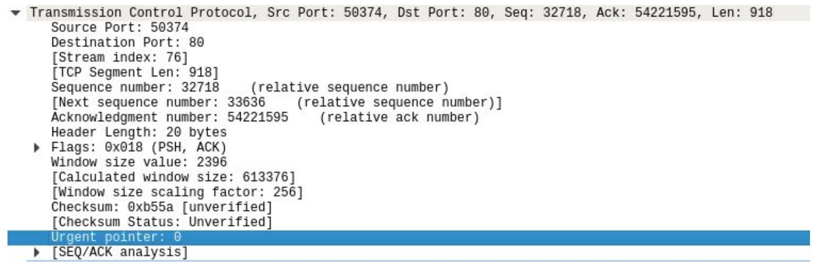
```
    Source Port: 51984
    Destination Port: 53
    Length: 51
    Checksum: 0x896a [unverified]
    [Checksum Status: Unverified]
```

UDP:-Source port =51984 number identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed **Destination port number** =53 identifies the receiver's port and is required. **Length=51** specifies the length in bytes of the UDP header and UDP data.**Checksum** may be used for error-checking of the header and data. This field is optional in IPv4, and mandatory in IPv6.



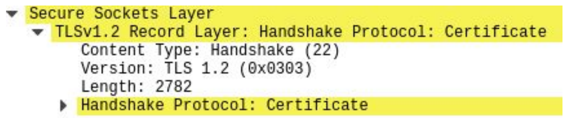
```
▾ Domain Name System (query)
    [Response In: 2477]
    Transaction ID: 0x9142
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▸ Queries
```

DNS:-The client/server information exchange in DNS is facilitated using query/response messaging. Both queries and responses have the same general format, containing up to five individual sections carrying information. Of these, two are usually found in both queries and responses: the *Header* section and the *Question* section. **Questions[].** Variable length=1 is a list of zero or more *Query* structures. **Answer RRs[].** Variable length is a list of zero or more Answer *Resource Record* structures. **Authority RRs[].** Variable length is a list of zero or more Authority *Resource Record* structures. **Additional RRs[].** Variable length is a list of zero or more Additional *Resource Record* structures.



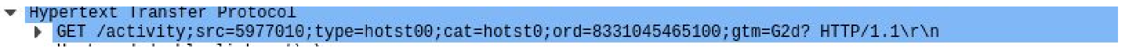
```
▾ Transmission Control Protocol, Src Port: 50374, Dst Port: 80, Seq: 32718, Ack: 54221595, Len: 918
    Source Port: 50374
    Destination Port: 80
    [Stream index: 76]
    [TCP Segment Len: 918]
    Sequence number: 32718    (relative sequence number)
    [Next sequence number: 33636    (relative sequence number)]
    Acknowledgment number: 54221595    (relative ack number)
    Header Length: 20 bytes
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 2396
    [Calculated window size: 613376]
    [Window size scaling factor: 256]
    Checksum: 0xb55a [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
```

TCP:-Source port=50374 (16 bits) identifies the sending port. **Destination port=80** (16 bits) identifies the receiving port. **Sequence number=32718** (32 bits) has a dual role which is if the SYN flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1 else this is the accumulated sequence number of the first data byte of this segment for the current session. **Acknowledgment number=54221595** (32 bits):-the value of this field is the next sequence number that the sender of the ACK is expecting. **Data offset**(4 bits) specifies the size of the TCP header in 32-bit words. **Reserved** (3 bits) is for future use and should be set to zero. **Flags** (9 bits) (aka Control bits) Contains 9 1-bit flags which are NS, CWR,ECE, URG, ACK, PSH, RST, SYN, FIN. **Window size** =256(16 bits) is the size of the *receive window*, which specifies the number of window size units. **Checksum** (16 bits) is used for error-checking of the header, the Payload and a Pseudo-Header. **Urgent pointer** (16 bits) if the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte. Options (Variable 0–320 bits, divisible by 32). The length of this field is determined by the data offset field. Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable).



```
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 2782
    ▶ Handshake Protocol: Certificate
```

SSL:-content type - handshake (22), version TLSv1.2 the version of TLS protocol, length =2782), Encrypted Application Data holds the application data encrypted based on the cipher suite decided between the client and server.



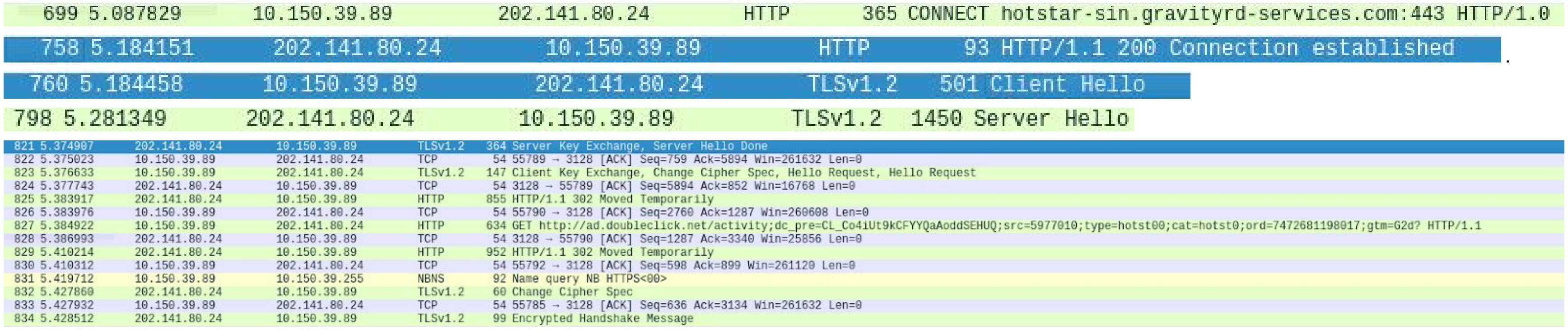
```
▼ Hypertext Transfer Protocol
  ▶ GET /activity;src=5977010;type=hotst00;cat=hotst0;ord=8331045465100;gtm=G2d? HTTP/1.1\r\n
```

HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers which are **General-header** which have general applicability for both request and response messages. **Request-header** which has applicability only for request messages. **Response-header** which has applicable only for response.

3)

On opening or refreshing the hotstar page, the following are first transaction of the packets between client and server. These packets were captured are as follows:-

When the site is opened, a connection request, (No. 699 in the image below), is sent from the client to the server to which the server responds, with an "HTTP/1.1 200 Connection Established" packet (No. 758 in the image below), status 200, implying it has successfully processed the client's request and agrees to communicate with client with HTTP version 1.1.



```
 699 5.087829    10.150.39.89     202.141.80.24    HTTP      365 CONNECT hotstar-sin.gravityrd-services.com:443 HTTP/1.0
 758 5.184151    202.141.80.24    10.150.39.89     HTTP       93 HTTP/1.1 200 Connection established
 760 5.184458    10.150.39.89     202.141.80.24    TLSv1.2   501 Client Hello
 798 5.281349    202.141.80.24    10.150.39.89     TLSv1.2  1450 Server Hello
 821 5.374907    202.141.80.24    10.150.39.89     TLSv1.2   364 Server Key Exchange, Server Hello Done
 822 5.375023    10.150.39.89     202.141.80.24    TCP        54 55789 → 3128 [ACK] Seq=759 Ack=5894 Win=261632 Len=0
 823 5.376633    10.150.39.89     202.141.80.24    TLSv1.2   147 Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request
 824 5.377743    202.141.80.24    10.150.39.89     TCP        54 3128 → 55789 [ACK] Seq=5894 Ack=852 Win=16768 Len=0
 825 5.383917    202.141.80.24    10.150.39.89     HTTP      855 HTTP/1.1 302 Moved Temporarily
 826 5.383976    10.150.39.89     202.141.80.24    TCP        54 55790 → 3128 [ACK] Seq=2760 Ack=1287 Win=260608 Len=0
 827 5.384922    10.150.39.89     202.141.80.24    HTTP      634 GET http://ad.doubleclick.net/activity;dc_pre=CL_Co4iUt9kCFYYQaAoddSEHUQ;src=5977010;type=hotst00;cat=hotst0;ord=7472681198017;gtm=G2d? HTTP/1.1
 828 5.386993    202.141.80.24    10.150.39.89     TCP        54 3128 → 55790 [ACK] Seq=1287 Ack=3340 Win=25856 Len=0
 829 5.410214    202.141.80.24    10.150.39.89     HTTP      952 HTTP/1.1 302 Moved Temporarily
 830 5.410312    10.150.39.89     202.141.80.24    TCP        54 55792 → 3128 [ACK] Seq=598 Ack=899 Win=261120 Len=0
 831 5.419712    10.150.39.89     10.150.39.255    NBNS       92 Name query NB HTTPS<00>
 832 5.427860    202.141.80.24    10.150.39.89     TLSv1.2    60 Change Cipher Spec
 833 5.427932    10.150.39.89     202.141.80.24    TCP        54 55785 → 3128 [ACK] Seq=636 Ack=3134 Win=261632 Len=0
 834 5.428512    202.141.80.24    10.150.39.89     TLSv1.2    99 Encrypted Handshake Message
```

Following this, another handshake is done to set up the secure connection with the TLSv1.2 protocol. The packets associated with this are highlighted blue in the image

- The client sends a "Client hello" message to the server, along with the client's random value and supported cipher suites.
- The server responds by sending a "Server hello" message to the client, along with the server's random value.
- The server sends its certificate to the client for authentication and server sends the "Server hello done" message.
- The client creates a random Pre-Master Secret and encrypts it with the public key from the server's certificate, sending the encrypted Pre-Master Secret to the server.
- The server receives the Pre-Master Secret. The server and client each generate the Master Secret and session keys based on the Pre-Master Secret. The client sends "Change cipher spec" notification to server to indicate that the client will start using the new session keys for hashing and encrypting messages.
- Server receives "Change cipher spec" and switches its record layer security state to symmetric encryption using the session keys. Acknowledges using "New session Ticket"
- Client and server can now exchange application data over the secured channel they have established. All messages sent from client to server and from server to client are encrypted using session key.

A similar handshaking, but less complex, also occurs periodically between client and server for secure transmission of data. It also occurs for resuming the session. If the hotstar page remains idle and then I use it again, the following handshake occurs:

- The client sends a "Client hello" message using the Session ID of the session to be resumed.
- The server checks its session cache for a matching Session ID. If a match is found, and the server is able to resume the session, it sends a "Server hello" message with the Session ID.
- Client and server exchange "Change cipher spec" messages
- Client and server can now resume application data exchange over the secure channel

4)

Ethernet: Ethernet is a link layer protocol in the TCP/IP stack, describing how networked devices can format data for transmission to other network devices on the same network segment, and how to put that data out on the network connection. This is essential for all types of applications.

IP: The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet. Essential for all types of applications

UDP:-applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network and prior communications are not required in order to set up communication channels or data paths.UDP uses a simple connectionless communication model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

DNS:- It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. By providing a worldwide, distributed directory service, the Domain Name System is an essential component of the functionality on the Internet. The Domain Name System delegates the responsibility of assigning domain names and mapping those names to Internet resources by designating authoritative name servers for each domain. Network administrators may delegate authority over sub-domains of their allocated name space to other name servers. This mechanism provides distributed and fault tolerant service and was designed to avoid a single large central database.The Domain Name System also specifies the technical functionality of the database service that is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in the DNS, as part of the Internet Protocol Suite.

TCP: It provides host-to-host connectivity at the Transport Layer. At the transport layer, the protocol handles all handshaking and transmission details and presents an abstraction of the network connection to the application. TCP is optimized for accurate delivery rather than timely delivery and hence used by web browsers to connect to the internet for peer to peer file sharing, email transfer etc. This is important for all kinds of internet applications but also that for a chat application that the sent message is free of errors.

SSL/TLSv1.2: SSL/TLS is used in every browser worldwide to provide https (http secure) functionality. Default usage in HTTPS is to verify server authenticity with trusted Certificate Authorities known by the browser, which is an important issue with chat applications so that private information about its client is secured.

HTTP: The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. It is a protocol that runs on top of the TCP/IP suite of protocols. Connecting from the browser to the server for the char application is important.

5)

time(24 hr format)	network	location	Ip address	Throughput (average bytes/s)	RTT (secs)	Packet size(bytes)	No. of packets lost	Number of udp packets	Number of tcp packets	number of responses received with respect to one request sent.
19:30	wifi	library	10.150.39.89	156000	0.001602	804.5	0	2254	83584	0.325465
8:00	wifi	dihing	192.168.43.56	134000	0.248079	908.5	0	1348	108951	0.871287
4:00	wifi	kapili	192.168.43.56	292000	0.138318	999.5	0	376	100046	0.942477
2:00	wifi	kapili	192.168.43.56	463000	0.108346	993.5	0	197	113566	0.822430
11:00	wifi	dihing	192.168.43.56	145000	0.072520	873.5	0	232	15229	0.949640

6)

An HTTP client normally uses the host name part of the URL to do a DNS lookup to connect to the requested host. Then it sends an HTTP GET of the rest of the URL. The host name is also included in that request. Now, if there is a proxy configured for a client it opens a connection to there and then sends the HTTP GET, with the complete URL, including the host name. The proxy then uses that information to set up its own connection with the intended host and forwards the HTTP GET. Any responses received are send back the originating client. Now, since, the campus is proxy authenticated network, all traffic is through this proxy server and thus we do not find the destination address of the original target server in the packets. But we can extract the DNS name of the target, header of HTTP request. Some of them are:

www.hotstar.com
staragvod3-vh.akamaihd.net
pagead2.googlesyndication.com
ma312-r.analytics.edgesuite.net
in-star.videoplaza.tv
hotstar.worldgravity.com
hotstar-sin.gravityrd-services.com
adunit.cdn.auditude.com
ad.doubleclick.net
ad.auditude.com

We can then do the name resolving by ourselves, but there is no 100% guarantee that the Proxy uses the same address. This is because it might use a different nameserver that returns a different IP address the same name server might respond with a different IP address to different clients there might be multiple IP addresses returned and you don't know which one the proxy picked

P.S.:- The google drive link for all the collected traces with wireshark is wireshark_traces