CS342:Networks Lab Assignment – 2 **Annapurne Krishna**

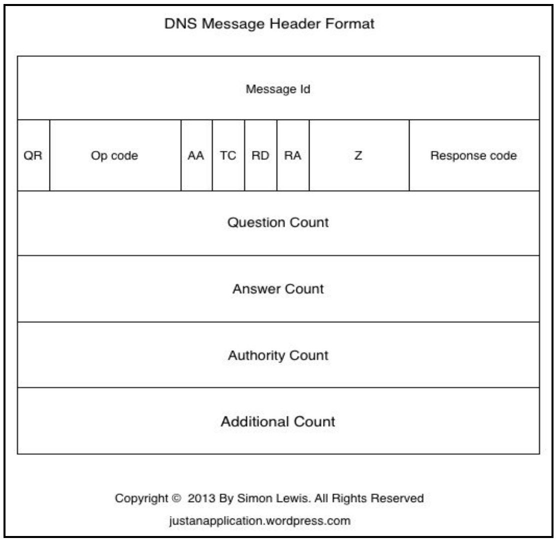
Application - Microsoft Teams **Roll No: 180101009**

Traces:https://drive.google.com/drive/folders/1SD8XiPRYE_mLQsA2dNmLj1HGi1zljXOu?usp=sharing

Q.1 Protocols used:

A) Application Layer Protocols:

a) DNS:



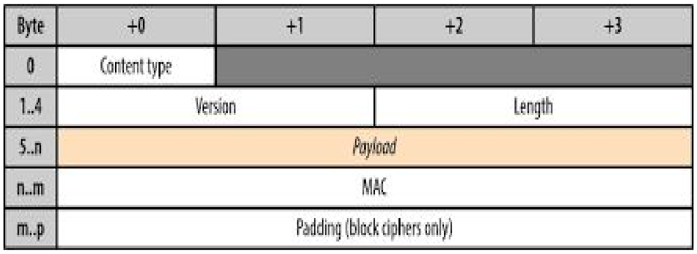
DNS Message Header Format

Message Id							
QR	Op code	AA	TC	RD	RA	Z	Response code
Question Count							
Answer Count							
Authority Count							
Additional Count							

Copyright © 2013 By Simon Lewis. All Rights Reserved
justanapplication.wordpress.com

Message id- Used to match request/reply packets.
OR- 1 bit field that specifies if message if request (0) or response (1)
Op code- 4 bits specifying request type
AA- stands for Authoritative Answer, 1 bit field specifying that the responding name server is authoritative (1) or from cache (0)
TC- Truncation - specifies that this message was truncated
RD- Recursion Desired : this bit directs the name server to pursue the query recursively
RA-*Recursion* Available :1bit, denotes whether recursive query support is available in the name server (1) or not (0)
Z- Reserved for future use
Response code- Response code - this 4 bit field is set as part of responses.
0 means No error condition;
1 means Format error - The name server was unable to interpret the query;
2 means Server failure -The name server was unable to process this query due to a problem with name server;
3 means Name Error - Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist;
4 means Not Implemented - The name server does not support the requested kind of query;
5 means Refused - The name server refuses to perform the specified operation for policy reasons.
Question count- an unsigned 16 bit integer specifying the number of entries in the question section
Answer count- unsigned 16 bit int specifying the number of resource records in the answer section.
Authority count-unsigned 16 bit int specifying number of name server resource records in authority records section.
Additional count- it specifies the number of resource records in the additional records section.

b) TLSv1.0 and TLSv1.2:

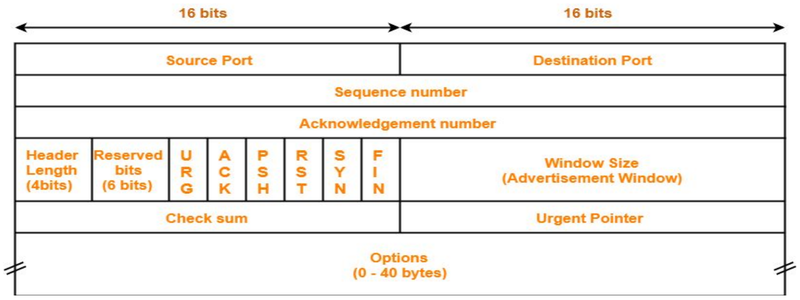


Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	Payload			
n..m	MAC			
m..p	Padding (block ciphers only)			

Each record consists of a five-byte record header, followed by data. **Content Type** can be of four types viz.Handshake, Change Cipher Spec, Alert, Application Data; **Version** is 16-bit value formatted in network order; **Length** is 16-bit value; **MAC** is up to 20 bytes for TLSv1.0 and up to 32 bytes for TLSv1.2.

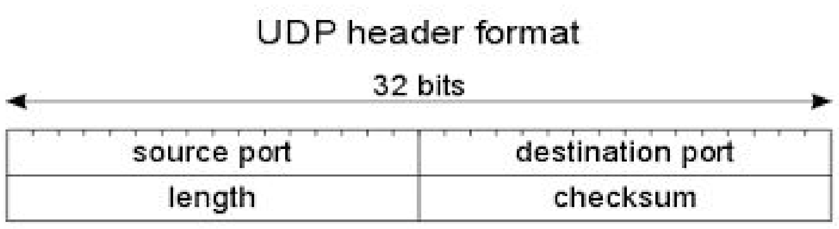
B) Transport Layer Protocols:

a) TCP



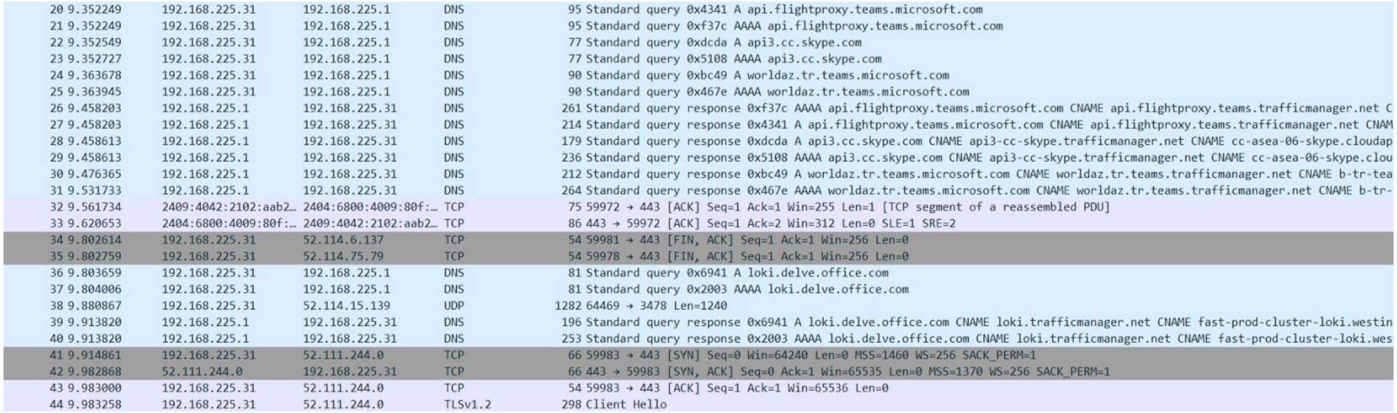
Source Port-It identifies the port of the sending application.**Destination Port-**It identifies the port of the receiving application.**Sequence Number-**This field contains the sequence number of the first data byte
Acknowledgement Number-It contains sequence number of the data byte that receiver expects to receive next from the sender **Header Length-**It contains the length of TCP header. **Reserved Bits-**These 6 bits are reserved. **Control flags-**These are 6 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc. Their function is: **URG**: Urgent pointer is valid **ACK**: Acknowledgement number is valid(used in case of cumulative acknowledgement) **PSH**: Request for push **RST**: Reset the connection **SYN**: Synchronize sequence numbers.**FIN**: Terminate the connection **Window Size-** It contains the size of the receiving window of the sender.It advertises how much data (in bytes) the sender can receive without acknowledgement. **Checksum-**It verifies the integrity of data in the TCP payload.**Urgent Pointer-**It indicates how much data in the current segment counting from the first data byte is urgent. **Options-** can be used to include support for special acknowledgment and window scaling algorithms.

b) UDP

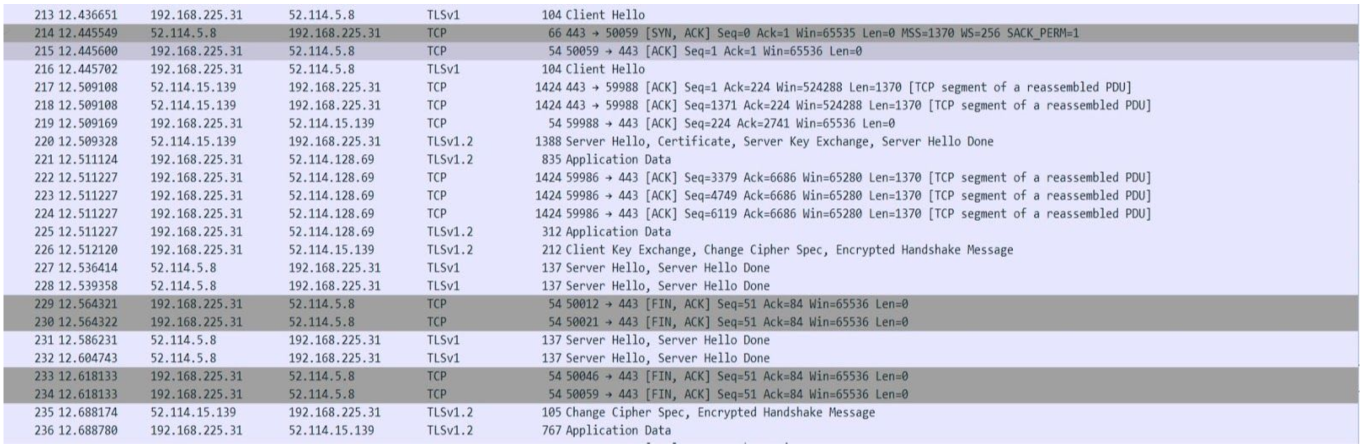


Source port- The port of the device sending the data.
Destination Port- It is 2 byte long field, used to identify the port of destined packet.
Length- It is the length in octets of this user datagram including header and the data
Checksum- The checksum allows the receiving device to verify the integrity of the packet header and payload.

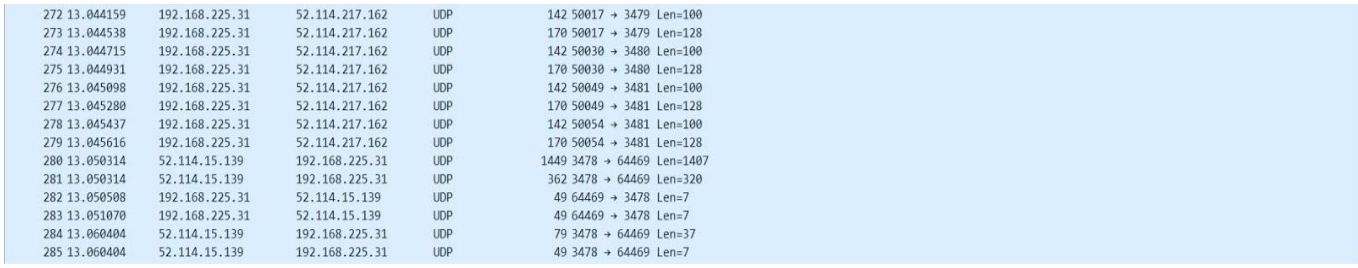
Observations: **Following packets were observed while joining a meeting on MS teams.**



```
20 9.352249   192.168.225.31    192.168.225.1     DNS       95 Standard query 0x4341 A api.flightproxy.teams.microsoft.com
21 9.352249   192.168.225.31    192.168.225.1     DNS       95 Standard query 0xf37c AAAA api.flightproxy.teams.microsoft.com
22 9.352549   192.168.225.31    192.168.225.1     DNS       77 Standard query 0xdcda A api3.cc.skype.com
23 9.352727   192.168.225.31    192.168.225.1     DNS       77 Standard query 0x5108 AAAA api3.cc.skype.com
24 9.363678   192.168.225.31    192.168.225.1     DNS       90 Standard query 0xbc49 A worldaz.tr.teams.microsoft.com
25 9.363945   192.168.225.31    192.168.225.1     DNS       90 Standard query 0x467e AAAA worldaz.tr.teams.microsoft.com
26 9.458203   192.168.225.1     192.168.225.31    DNS      261 Standard query response 0xf37c AAAA api.flightproxy.teams.microsoft.com CNAME api.flightproxy.teams.trafficmanager.net C
27 9.458203   192.168.225.1     192.168.225.31    DNS      214 Standard query response 0x4341 A api.flightproxy.teams.microsoft.com CNAME api.flightproxy.teams.trafficmanager.net CNAM
28 9.458613   192.168.225.1     192.168.225.31    DNS      179 Standard query response 0xdcda A api3.cc.skype.com CNAME api3-cc-skype.trafficmanager.net CNAME cc-asea-06-skype.cloudap
29 9.458613   192.168.225.1     192.168.225.31    DNS      236 Standard query response 0x5108 AAAA api3.cc.skype.com CNAME api3-cc-skype.trafficmanager.net CNAME cc-asea-06-skype.clou
30 9.476365   192.168.225.1     192.168.225.31    DNS      212 Standard query response 0xbc49 A worldaz.tr.teams.microsoft.com CNAME worldaz.tr.teams.trafficmanager.net CNAME b-tr-tea
31 9.531733   192.168.225.1     192.168.225.31    DNS      264 Standard query response 0x467e AAAA worldaz.tr.teams.microsoft.com CNAME worldaz.tr.teams.trafficmanager.net CNAME b-tr-
32 9.561734   2409:4042:2102:aab2… 2404:6800:4009:80f:… TCP   75 59972 → 443 [ACK] Seq=1 Ack=1 Win=255 Len=1 [TCP segment of a reassembled PDU]
33 9.620653   2404:6800:4009:80f:… 2409:4042:2102:aab2… TCP   86 443 → 59972 [ACK] Seq=1 Ack=2 Win=312 Len=0 SLE=1 SRE=2
34 9.802614   192.168.225.31    52.114.6.137      TCP       54 59981 → 443 [FIN, ACK] Seq=1 Ack=1 Win=256 Len=0
35 9.802759   192.168.225.31    52.114.75.79      TCP       54 59978 → 443 [FIN, ACK] Seq=1 Ack=1 Win=256 Len=0
36 9.803659   192.168.225.31    192.168.225.1     DNS       81 Standard query 0x6941 A loki.delve.office.com
37 9.804006   192.168.225.31    192.168.225.1     DNS       81 Standard query 0x2003 AAAA loki.delve.office.com
38 9.880867   192.168.225.31    52.114.15.139     UDP     1282 64469 → 3478 Len=1240
39 9.913820   192.168.225.1     192.168.225.31    DNS      196 Standard query response 0x6941 A loki.delve.office.com CNAME loki.trafficmanager.net CNAME fast-prod-cluster-loki.westin
40 9.913820   192.168.225.1     192.168.225.31    DNS      253 Standard query response 0x2003 AAAA loki.delve.office.com CNAME loki.trafficmanager.net CNAME fast-prod-cluster-loki.wes
41 9.914861   192.168.225.31    52.111.244.0      TCP       66 59983 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
42 9.982868   52.111.244.0      192.168.225.31    TCP       66 443 → 59983 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1370 WS=256 SACK_PERM=1
43 9.983000   192.168.225.31    52.111.244.0      TCP       54 59983 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
44 9.983258   192.168.225.31    52.111.244.0      TLSv1.2  298 Client Hello
```



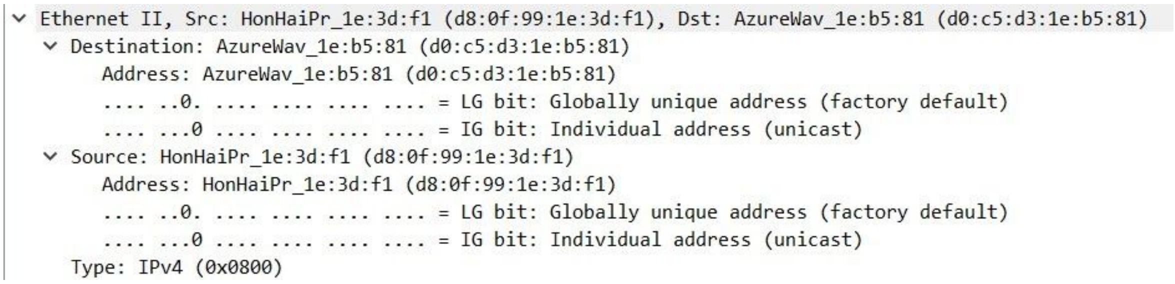
```
213 12.436651  192.168.225.31  52.114.5.8      TLSv1    104 Client Hello
214 12.445549  52.114.5.8      192.168.225.31  TCP       66 443 → 50059 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1370 WS=256 SACK_PERM=1
215 12.445600  192.168.225.31  52.114.5.8      TCP       54 50059 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
216 12.445702  192.168.225.31  52.114.5.8      TLSv1    104 Client Hello
217 12.509108  52.114.15.139   192.168.225.31  TCP     1424 443 → 59988 [ACK] Seq=1 Ack=224 Win=524288 Len=1370 [TCP segment of a reassembled PDU]
218 12.509108  52.114.15.139   192.168.225.31  TCP     1424 443 → 59988 [ACK] Seq=1371 Ack=224 Win=524288 Len=1370 [TCP segment of a reassembled PDU]
219 12.509169  192.168.225.31  52.114.15.139   TCP       54 59988 → 443 [ACK] Seq=224 Ack=2741 Win=65536 Len=0
220 12.509328  52.114.15.139   192.168.225.31  TLSv1.2 1388 Server Hello, Certificate, Server Key Exchange, Server Hello Done
221 12.511124  192.168.225.31  52.114.128.69   TLSv1.2  835 Application Data
222 12.511227  192.168.225.31  52.114.128.69   TCP     1424 59986 → 443 [ACK] Seq=3379 Ack=6686 Win=65280 Len=1370 [TCP segment of a reassembled PDU]
223 12.511227  192.168.225.31  52.114.128.69   TCP     1424 59986 → 443 [ACK] Seq=4749 Ack=6686 Win=65280 Len=1370 [TCP segment of a reassembled PDU]
224 12.511227  192.168.225.31  52.114.128.69   TCP     1424 59986 → 443 [ACK] Seq=6119 Ack=6686 Win=65280 Len=1370 [TCP segment of a reassembled PDU]
225 12.511227  192.168.225.31  52.114.128.69   TLSv1.2  312 Application Data
226 12.512120  192.168.225.31  52.114.15.139   TLSv1.2  212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
227 12.536414  52.114.5.8      192.168.225.31  TLSv1    137 Server Hello, Server Hello Done
228 12.539358  52.114.5.8      192.168.225.31  TLSv1    137 Server Hello, Server Hello Done
229 12.564321  192.168.225.31  52.114.5.8      TCP       54 50012 → 443 [FIN, ACK] Seq=51 Ack=84 Win=65536 Len=0
230 12.564322  192.168.225.31  52.114.5.8      TCP       54 50021 → 443 [FIN, ACK] Seq=51 Ack=84 Win=65536 Len=0
231 12.586231  52.114.5.8      192.168.225.31  TLSv1    137 Server Hello, Server Hello Done
232 12.604743  52.114.5.8      192.168.225.31  TLSv1    137 Server Hello, Server Hello Done
233 12.618133  192.168.225.31  52.114.5.8      TCP       54 50046 → 443 [FIN, ACK] Seq=51 Ack=84 Win=65536 Len=0
234 12.618133  192.168.225.31  52.114.5.8      TCP       54 50059 → 443 [FIN, ACK] Seq=51 Ack=84 Win=65536 Len=0
235 12.688174  52.114.15.139   192.168.225.31  TLSv1.2  105 Change Cipher Spec, Encrypted Handshake Message
236 12.688780  192.168.225.31  52.114.15.139   TLSv1.2  767 Application Data
```



```
272 13.044159  192.168.225.31  52.114.217.162  UDP   142 50017 → 3479 Len=100
273 13.044538  192.168.225.31  52.114.217.162  UDP   170 50017 → 3479 Len=128
274 13.044715  192.168.225.31  52.114.217.162  UDP   142 50030 → 3480 Len=100
275 13.044931  192.168.225.31  52.114.217.162  UDP   170 50030 → 3480 Len=128
276 13.045098  192.168.225.31  52.114.217.162  UDP   142 50049 → 3481 Len=100
277 13.045280  192.168.225.31  52.114.217.162  UDP   170 50049 → 3481 Len=128
278 13.045437  192.168.225.31  52.114.217.162  UDP   142 50054 → 3481 Len=100
279 13.045616  192.168.225.31  52.114.217.162  UDP   170 50054 → 3481 Len=128
280 13.050314  52.114.15.139   192.168.225.31  UDP  1449 3478 → 64469 Len=1407
281 13.050314  52.114.15.139   192.168.225.31  UDP   362 3478 → 64469 Len=320
282 13.050508  192.168.225.31  52.114.15.139   UDP    49 64469 → 3478 Len=7
283 13.051070  192.168.225.31  52.114.15.139   UDP    49 64469 → 3478 Len=7
284 13.060404  52.114.15.139   192.168.225.31  UDP    79 3478 → 64469 Len=37
285 13.060404  52.114.15.139   192.168.225.31  UDP    49 3478 → 64469 Len=7
```

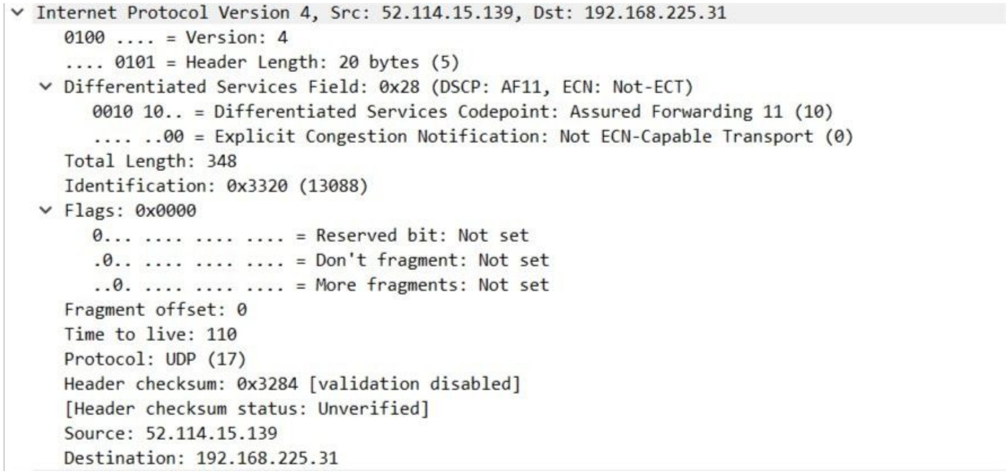
TIme is the time elapsed since the starting of packets capture
Source- IP address of the sender of packets
Destination- IP address of the receiver of packets
Protocol- is the protocol(of the highest layer) that the wireshark could identify
Length- is the length of the packet
Info- is a brief information contained in the packet decoded by wireshark

1. Ethernet II - It contains the physical MAC address of the devices communicating. Destination is my HP Device and the source is the Switch to which my device is connected. Source is always Unicast. Destination is Unicast in this case. In both of them, it is Globally Unique Address and not a Local Address.



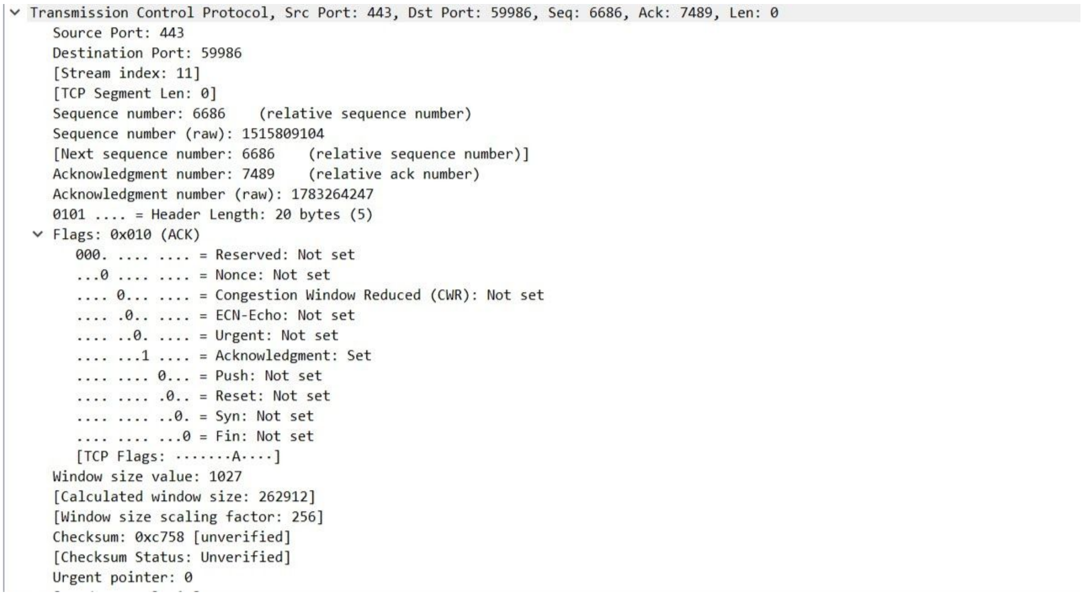
```
v Ethernet II, Src: HonHaiPr_1e:3d:f1 (d8:0f:99:1e:3d:f1), Dst: AzureWav_1e:b5:81 (d0:c5:d3:1e:b5:81)
  v Destination: AzureWav_1e:b5:81 (d0:c5:d3:1e:b5:81)
      Address: AzureWav_1e:b5:81 (d0:c5:d3:1e:b5:81)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: HonHaiPr_1e:3d:f1 (d8:0f:99:1e:3d:f1)
      Address: HonHaiPr_1e:3d:f1 (d8:0f:99:1e:3d:f1)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

2. IPv4 - Version header field is always 4 as we are using IPv4. Header length is 5 (which means 20 bytes because it counts in 4 Bytes word). Total length of the packet is 348 bytes.No flag is set. TTL is 110. The protocol contained in it is UCP. The packet is sent by MS Teams(52.114.15.139) to my device(192.168.225.31).



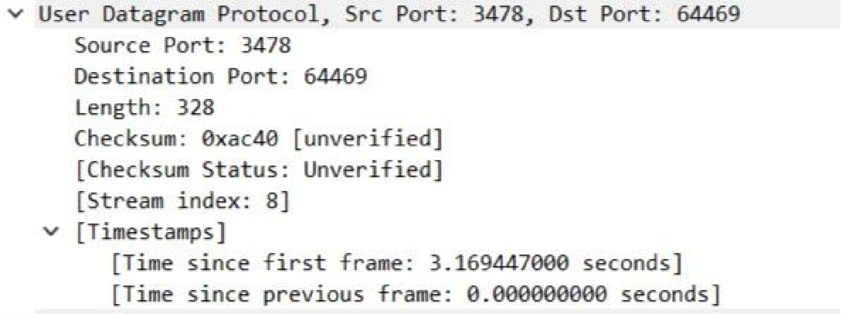
```
v Internet Protocol Version 4, Src: 52.114.15.139, Dst: 192.168.225.31
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  v Differentiated Services Field: 0x28 (DSCP: AF11, ECN: Not-ECT)
      0010 10.. = Differentiated Services Codepoint: Assured Forwarding 11 (10)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 348
    Identification: 0x3320 (13088)
  v Flags: 0x0000
      0... .... .... .... = Reserved bit: Not set
      .0.. .... .... .... = Don't fragment: Not set
      ..0. .... .... .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 110
    Protocol: UDP (17)
    Header checksum: 0x3284 [validation disabled]
    [Header checksum status: Unverified]
    Source: 52.114.15.139
    Destination: 192.168.225.31
```

3. TCP - The packet contains the Destination and the Source Port, TCP Stream index, sequence number and the acknowledgement number, Header length, Flags. In the following figure, the flags are set as 010 in (HexaDecimal) which corresponds to the Acknowledgement. Window size value is 1027. Checksum is used for error detection. Wireshark is remembering the value of the Window size scaling factor and presenting it again. Scaling factor shows the number of leftward bit shifts that should be used for an advertised window size.



```
v Transmission Control Protocol, Src Port: 443, Dst Port: 59986, Seq: 6686, Ack: 7489, Len: 0
    Source Port: 443
    Destination Port: 59986
    [Stream index: 11]
    [TCP Segment Len: 0]
    Sequence number: 6686    (relative sequence number)
    Sequence number (raw): 1515809104
    [Next sequence number: 6686    (relative sequence number)]
    Acknowledgment number: 7489    (relative ack number)
    Acknowledgment number (raw): 1783264247
    0101 .... = Header Length: 20 bytes (5)
  v Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A····]
    Window size value: 1027
    [Calculated window size: 262912]
    [Window size scaling factor: 256]
    Checksum: 0xc758 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
```

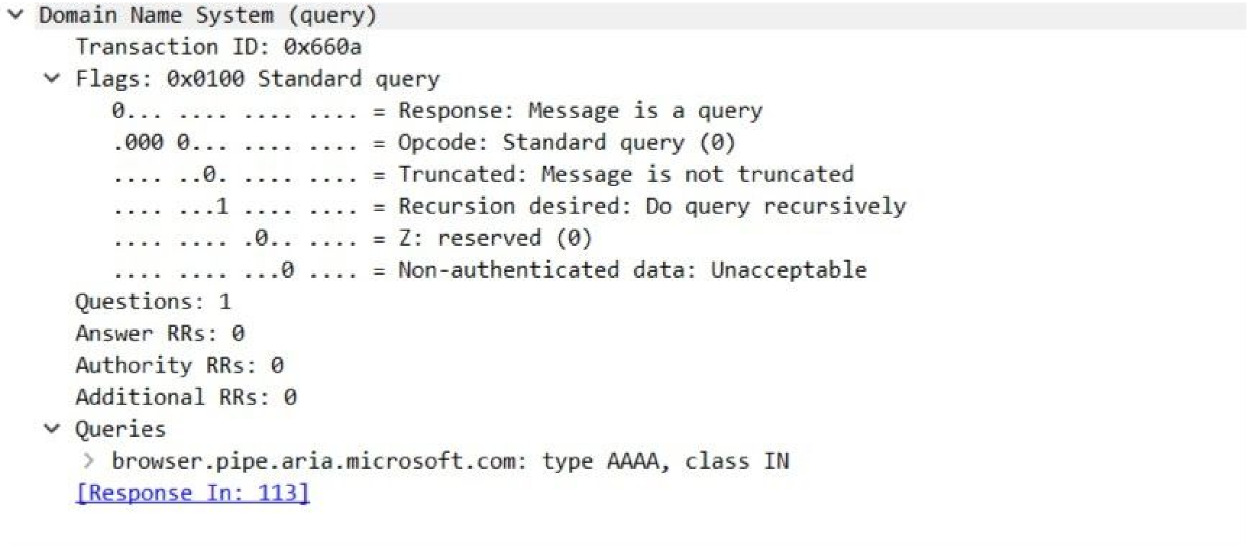
4. UDP- The packet contains source port, destination port, length of the packet which is 328 and checksum is 0xac40.



```
v User Datagram Protocol, Src Port: 3478, Dst Port: 64469
    Source Port: 3478
    Destination Port: 64469
    Length: 328
    Checksum: 0xac40 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 8]
  v [Timestamps]
      [Time since first frame: 3.169447000 seconds]
      [Time since previous frame: 0.000000000 seconds]
```

5. DNS

Every DNA packet has a unique transaction ID. Response flag is not set meaning that message is a query, opcode is reset meaning that it is a standard query, Truncated flag is reset hence message is not truncated, Recursive desired flag is set meaning that the response is given recursively, Z is reset meaning bits are reserved for future use. Non-authenticated data is set to 0 meaning that only authenticated data is acceptable.The number of entries in the questions list is 1. The number of resource records in the answer section is 0, the number of entries in the authority resource record list that were returned is 0. The number of resource records in the additional records section is 0. Query name is browser.pipe.aria.microsoft.com and the query type is AAAA. class is IN (internet), Response of the query is given in the packet numbered 10.



```
v Domain Name System (query)
    Transaction ID: 0x660a
  v 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
  v Queries
    > browser.pipe.aria.microsoft.com: type AAAA, class IN
    [Response In: 113]
```

Q2. Functionalities of the application:

a) **Join Meeting:** Protocols used- 1)TCP 2)UDP 3)DNS 4)TLS
b) **Post message:** Protocols used- 1)TCP 2)DNS 3)TLS
c) **Share Screen:** Protocols used- 1)TCP 2)UDP 3)DNS 4)TLS
d) **Start recording:** Protocols used- 1)TCP 2)UDP 3)DNS 4)TLS
e) **Turn camera on:** Protocols used- 1)TCP 2)UDP 3)TLS

DNS translates domain names to IP addresses so browsers can load Internet resources. Each device connected to the Internet has a unique IP address which other machines use to find the device. So first DNS requests server IPs which are required to use the features in Microsoft teams.

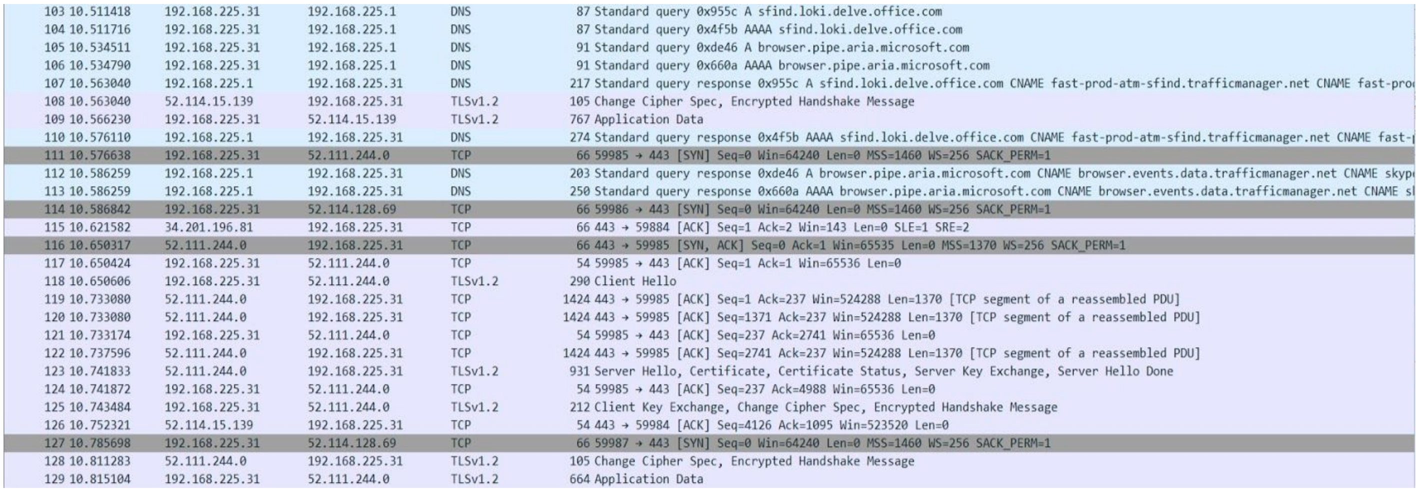
UDP is fastest at delivering packets on the network. It doesn't check if the packet is fully received and it is not sensitive to packet reorder. Once the connection is established then UDP is used to deliver packets in the live stream. UDP is preferred in Live video streaming. Since in video streaming if a packet is lost, we cannot wait for that packet to be transmitted again. The problem in UDP is that you must open many more ports on your network for it to work. And that is why, sometimes UDP is not recommended for communication outside of the company network. In these cases, the protocol used is TCP.

TCP or the Transmission Control Protocol is a communication protocol used to interconnect network devices on the internet. It facilitates the exchange of messages between computing devices in a network. TCP is much more secure, because it needs to open significantly fewer ports. Also, TCP makes sure that all packets are received at the destination, and if not, it will send them again.

The Transport Layer Security (**TLS**) protocol adds a layer of security on top of the TCP transport protocols TLS is a cryptographic protocol that provides end-to-end communications security over networks and is used for internet communications.

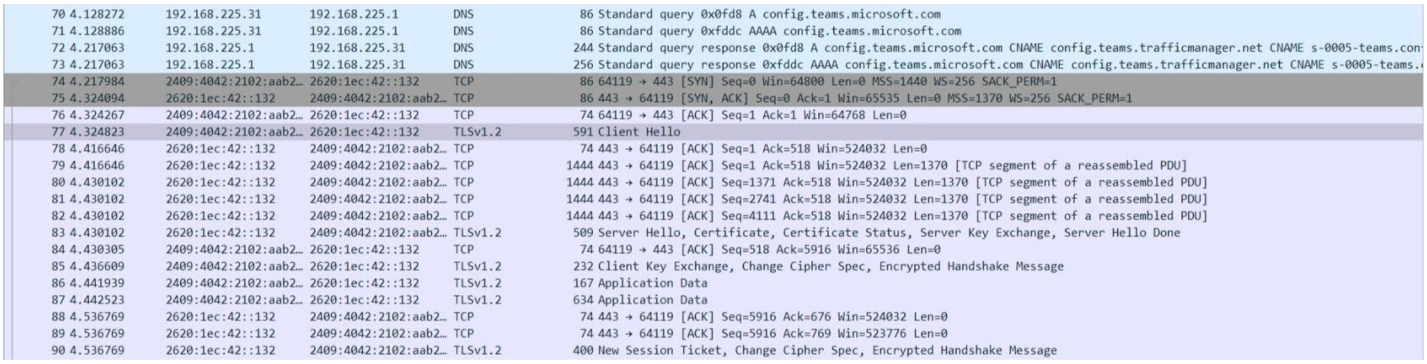
Q3. Sequence of messages exchanged:

a) While joining meeting:



```
103 10.511418   192.168.225.31   192.168.225.1    DNS        87 Standard query 0x955c A sfind.loki.delve.office.com
104 10.511716   192.168.225.31   192.168.225.1    DNS        87 Standard query 0x4f5b AAAA sfind.loki.delve.office.com
105 10.534511   192.168.225.31   192.168.225.1    DNS        91 Standard query 0xde46 A browser.pipe.aria.microsoft.com
106 10.534790   192.168.225.31   192.168.225.1    DNS        91 Standard query 0x660a AAAA browser.pipe.aria.microsoft.com
107 10.563040   192.168.225.1    192.168.225.31   DNS       217 Standard query response 0x955c A sfind.loki.delve.office.com CNAME fast-prod-atm-sfind.trafficmanager.net CNAME fast-pro
108 10.563040   52.114.15.139    192.168.225.31   TLSv1.2   105 Change Cipher Spec, Encrypted Handshake Message
109 10.566230   192.168.225.31   52.114.15.139    TLSv1.2   767 Application Data
110 10.576110   192.168.225.1    192.168.225.31   DNS       274 Standard query response 0x4f5b AAAA sfind.loki.delve.office.com CNAME fast-prod-atm-sfind.trafficmanager.net CNAME fast-
111 10.576638   192.168.225.31   52.111.244.0     TCP        66 59985 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
112 10.586259   192.168.225.1    192.168.225.31   DNS       203 Standard query response 0xde46 A browser.pipe.aria.microsoft.com CNAME browser.events.data.trafficmanager.net CNAME skyp
113 10.586259   192.168.225.1    192.168.225.31   DNS       250 Standard query response 0x660a AAAA browser.pipe.aria.microsoft.com CNAME browser.events.data.trafficmanager.net CNAME s
114 10.586842   192.168.225.31   52.114.128.69    TCP        66 59986 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
115 10.621582   34.201.196.81    192.168.225.31   TCP        66 443 → 59884 [ACK] Seq=1 Ack=2 Win=143 Len=0 SLE=1 SRE=2
116 10.650317   52.111.244.0     192.168.225.31   TCP        66 443 → 59985 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1370 WS=256 SACK_PERM=1
117 10.650424   192.168.225.31   52.111.244.0     TCP        54 59985 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
118 10.650606   192.168.225.31   52.111.244.0     TLSv1.2   290 Client Hello
119 10.733080   52.111.244.0     192.168.225.31   TCP      1424 443 → 59985 [ACK] Seq=1 Ack=237 Win=524288 Len=1370 [TCP segment of a reassembled PDU]
120 10.733080   52.111.244.0     192.168.225.31   TCP      1424 443 → 59985 [ACK] Seq=1371 Ack=237 Win=524288 Len=1370 [TCP segment of a reassembled PDU]
121 10.733174   192.168.225.31   52.111.244.0     TCP        54 59985 → 443 [ACK] Seq=237 Ack=2741 Win=65536 Len=0
122 10.737596   52.111.244.0     192.168.225.31   TCP      1424 443 → 59985 [ACK] Seq=2741 Ack=237 Win=524288 Len=1370 [TCP segment of a reassembled PDU]
123 10.741833   52.111.244.0     192.168.225.31   TLSv1.2   931 Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
124 10.741872   192.168.225.31   52.111.244.0     TCP        54 59985 → 443 [ACK] Seq=237 Ack=4988 Win=65536 Len=0
125 10.743484   192.168.225.31   52.111.244.0     TLSv1.2   212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
126 10.752321   52.114.15.139    192.168.225.31   TCP        54 443 → 59984 [ACK] Seq=4126 Ack=1095 Win=523520 Len=0
127 10.785698   192.168.225.31   52.114.128.69    TCP        66 59987 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
128 10.811283   52.111.244.0     192.168.225.31   TLSv1.2   105 Change Cipher Spec, Encrypted Handshake Message
129 10.815104   192.168.225.31   52.111.244.0     TLSv1.2   664 Application Data
```

b) While posting messages:



```
70 4.128272   192.168.225.31       192.168.225.1        DNS        86 Standard query 0x0fd8 A config.teams.microsoft.com
71 4.128886   192.168.225.31       192.168.225.1        DNS        86 Standard query 0xfddc AAAA config.teams.microsoft.com
72 4.217063   192.168.225.1        192.168.225.31       DNS       244 Standard query response 0x0fd8 A config.teams.microsoft.com CNAME config.teams.trafficmanager.net CNAME s-0005-teams.con
73 4.217063   192.168.225.1        192.168.225.31       DNS       256 Standard query response 0xfddc AAAA config.teams.microsoft.com CNAME config.teams.trafficmanager.net CNAME s-0005-teams.
74 4.217984   2409:4042:2102:aab2… 2620:1ec:42::132     TCP        86 64119 → 443 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS=256 SACK_PERM=1
75 4.324094   2620:1ec:42::132     2409:4042:2102:aab2… TCP        86 443 → 64119 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1370 WS=256 SACK_PERM=1
76 4.324267   2409:4042:2102:aab2… 2620:1ec:42::132     TCP        74 64119 → 443 [ACK] Seq=1 Ack=1 Win=64768 Len=0
77 4.324823   2409:4042:2102:aab2… 2620:1ec:42::132     TLSv1.2   591 Client Hello
78 4.416646   2620:1ec:42::132     2409:4042:2102:aab2… TCP        74 443 → 64119 [ACK] Seq=1 Ack=518 Win=524032 Len=0
79 4.416646   2620:1ec:42::132     2409:4042:2102:aab2… TCP      1444 443 → 64119 [ACK] Seq=1 Ack=518 Win=524032 Len=1370 [TCP segment of a reassembled PDU]
80 4.430102   2620:1ec:42::132     2409:4042:2102:aab2… TCP      1444 443 → 64119 [ACK] Seq=1371 Ack=518 Win=524032 Len=1370 [TCP segment of a reassembled PDU]
81 4.430102   2620:1ec:42::132     2409:4042:2102:aab2… TCP      1444 443 → 64119 [ACK] Seq=2741 Ack=518 Win=524032 Len=1370 [TCP segment of a reassembled PDU]
82 4.430102   2620:1ec:42::132     2409:4042:2102:aab2… TCP      1444 443 → 64119 [ACK] Seq=4111 Ack=518 Win=524032 Len=1370 [TCP segment of a reassembled PDU]
83 4.430102   2620:1ec:42::132     2409:4042:2102:aab2… TLSv1.2   509 Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
84 4.430305   2409:4042:2102:aab2… 2620:1ec:42::132     TCP        74 64119 → 443 [ACK] Seq=518 Ack=5916 Win=65536 Len=0
85 4.436609   2409:4042:2102:aab2… 2620:1ec:42::132     TLSv1.2   232 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
86 4.441939   2409:4042:2102:aab2… 2620:1ec:42::132     TLSv1.2   167 Application Data
87 4.442523   2409:4042:2102:aab2… 2620:1ec:42::132     TLSv1.2   634 Application Data
88 4.536769   2620:1ec:42::132     2409:4042:2102:aab2… TCP        74 443 → 64119 [ACK] Seq=5916 Ack=676 Win=524032 Len=0
89 4.536769   2620:1ec:42::132     2409:4042:2102:aab2… TCP        74 443 → 64119 [ACK] Seq=5916 Ack=769 Win=523776 Len=0
90 4.536769   2620:1ec:42::132     2409:4042:2102:aab2… TLSv1.2   400 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
```

Initial client to server communication-

Client hello:Typically, the first message in the TLS Handshake is the client hello message which is sent by the client to initiate a session with the server.

Server response to client-

Server hello: It is the server's response to the client.

Server Certificate:The server sends the client a list of certificates to authenticate itself. The server's certificate contains its public key.

Certificate Status:This message validates whether the server's X.509 digital certificate is revoked or not

Server Key Exchange:The message is optional and sent when the public key present in the server's certificate is not suitable for key exchange or if the cipher suite places a restriction requiring a temporary key. This key is used by the client to encrypt Client Key Exchange later in the process.

Client Certificate Request: This is optional and is sent when the server wants to authenticate the client

Server hello done:This message indicates the server is done and is awaiting the client's response.

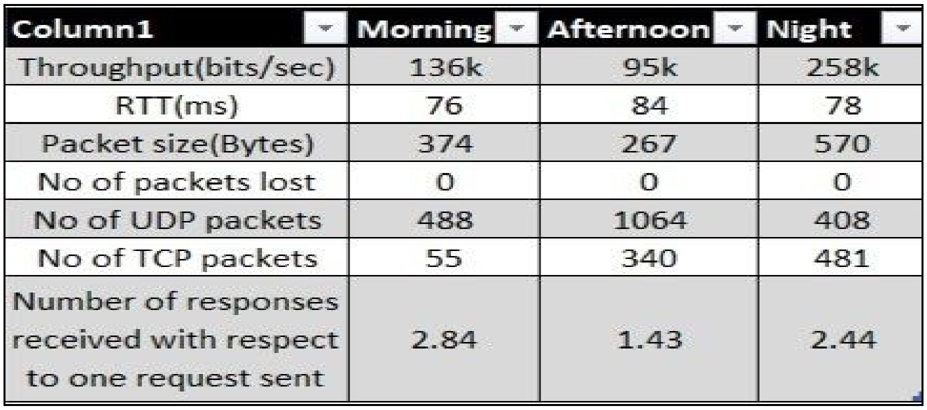
Client response to server-

<u>Client Key Exchange:</u>Until now, all the information sent between the client and server is unencrypted. Now the client receives the server’s public key and generates a new session key (aka pre-master key) encrypted with the public key and sends it to the server.
<u>Change Cipher Spec:</u> It is sent by both the client and server to notify the receiving party that subsequent records will be protected under the just negotiated CipherSpec and keys.
<u>Encrypted Handshake:</u> The client and the server sends to each other an encrypted message saying the key information is correct.

Handshaking sequence:

TCP CONNECTION HANDSHAKE : To establish a connection, each device must send a SYN and receive an ACK for it from the other device. Thus, conceptually, we need to have four control messages pass between the devices. However, it's inefficient to send a SYN and an ACK in separate messages when one could communicate both simultaneously. Thus, in the normal sequence of events in connection establishment, one of the SYNs and one of the ACKs is sent together by setting both of the relevant bits (a message sometimes called a SYN+ACK). This makes a total of three messages, and for this reason the connection procedure is called a three-way handshake.

Q4.



Column1 ▾	Morning ▾	Afternoon ▾	Night ▾
Throughput(bits/sec)	136k	95k	258k
RTT(ms)	76	84	78
Packet size(Bytes)	374	267	570
No of packets lost	0	0	0
No of UDP packets	488	1064	408
No of TCP packets	55	340	481
Number of responses received with respect to one request sent	2.84	1.43	2.44

.

Q5. Yes, the content is being sent/fetched by the application to/from the same or different destination(s)/source(s) Different IPs found were 52.114.75.79, 52.114.76.37, 52.114.3.203 and 52.114.14.55. The reason for having different servers is that the application uses more than one server to manage the overall network traffic. So when we send a connection request the server with least traffic is chosen. Also as we need a continuous strong connection for attending meetings live, the server may change depending on the traffic.