CS342: ASSIGNMENT 2

RITWIK GANGULY – 180101067

Drive link to the folder containing the wireshark traces: https://drive.google.com/drive/folders/1c-Yr7Aw0Cnl09CccYtk9XN0nSbe6wg2S?usp=sharing

I ran the wireshark traces on my Mobile Wifi network at 3 different times of the day: morning, evening and night. Due to activation issues with Outlook desktop app, I used the traces after running outlook on google chrome after closing all other tabs so that they don't interfere with the packets generated. Since Outlook is a global website and has many servers around the world, IP-based filtering was difficult so I used the filters "tcp contains outlook", "ip contains outlook", "Dns contains outlook" and "udp contains outlook" to segregate the packets generated due to running of the outlook application.

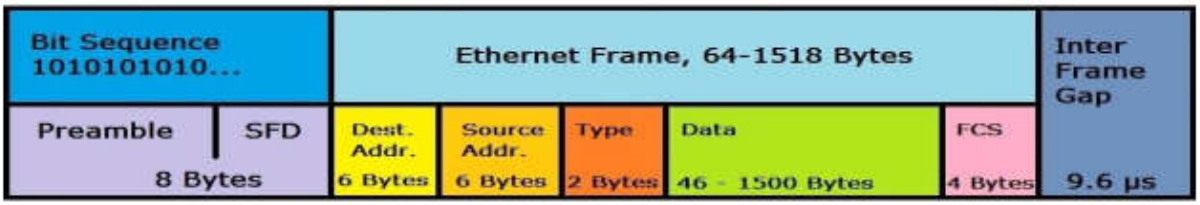
Answer 1:

I have identified the following protocols from the traces:

- **DATA LINK LAYER:**

  Ethernet II: *This is the protocol used in the physical data link layer.*

  - The frame begins with a **preamble** and **SFD** (Start frame delimiter). The **preamble** consists of a 56-bit pattern of alternating 1 and 0 bits, providing bit-level synchronization. **SFD** is the 8-bit value that marks the end of the preamble. **Type** denotes the type of connection. The **FCS** (Frame check sequence) is a cyclic redundancy check that allows the detection of corrupted data.

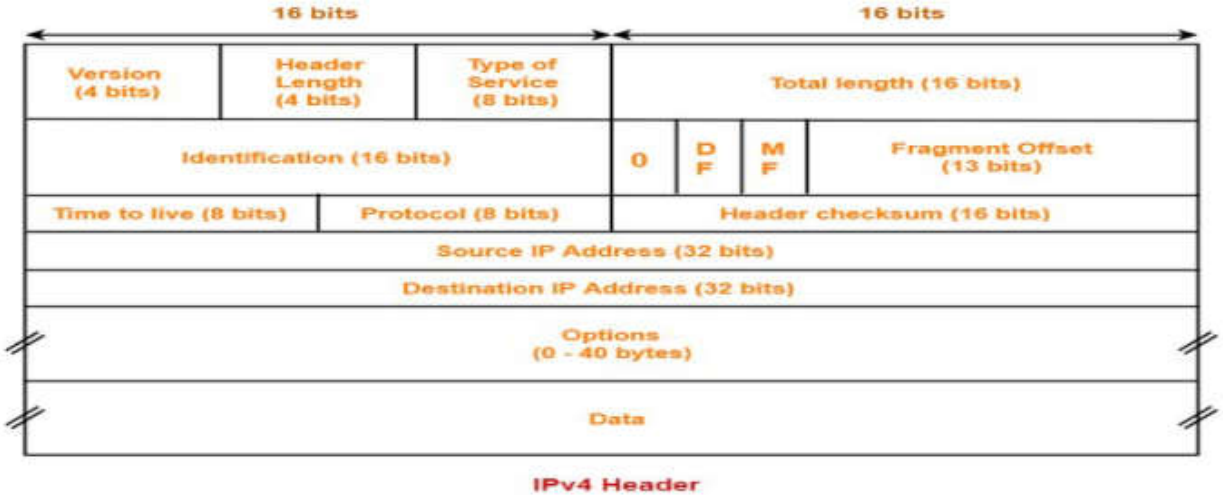


  - It contains information about the **Source** and **Destination MAC addresses** which are unique identifiers assigned to the NICs present in the machines and these are globally unique. **Destination** indicates the address of the destination adapter. **Source** indicates the address of the source adapter. **Type** indicates the upper layer protocol to be used. A 0 **IG Bit** indicates unicast MAC address while a 1 indicates multicast address. An **LG Bit** distinguishes vendor assigned (0) and administratively assigned (1) MAC addresses.

```
Ethernet II, Src: IntelCor_fe:44:99 (a0:51:0b:fe:44:99), Dst: XiaomiCo_5e:7c:04 (d8:32:e3:5e:7c:04)
  Destination: XiaomiCo_5e:7c:04 (d8:32:e3:5e:7c:04)
      Address: XiaomiCo_5e:7c:04 (d8:32:e3:5e:7c:04)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: IntelCor_fe:44:99 (a0:51:0b:fe:44:99)
      Address: IntelCor_fe:44:99 (a0:51:0b:fe:44:99)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

- **NETWORK LAYER:**

  IPv4: *This is the used protocol in the network layer.*



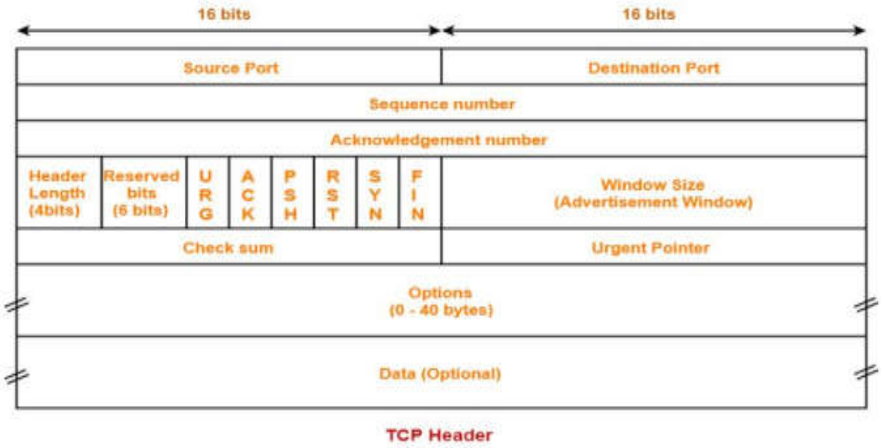
IPv4 Header

- **Total length** field defines the entire packet size in bytes, including header and data. The **identification** field is primarily used for uniquely identifying the group of fragments of a single IP datagram. **Fragment** specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram.
- **Header length** is 20 bytes and shown as 5 because the jump is 4 bytes each. **DCP:CS0** indicates best-effort delivery service and **ECN: Not-ECT** implies non-ECN-Capable Transport. **Packet size** is 64 bytes. **TTL** (Time to live) is 128 hops. UDP is used as **protocol** for the layer above. **Header checksum** is the technique used for error detection of packet headers. **Source and Destination** contain the IP address of my laptop and the application respectively.

```
Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.92
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 234
    Identification: 0x2f6d (12141)
  > Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x32e8 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.43.1
    Destination: 192.168.43.92
```

❖ **TRANSPORT LAYER:**

TCP (Transmission Control Protocol): *This protocol is used in the transport layer.*



TCP Header

- **Sequence Number field** (32 bits) specifies the number assigned to the first byte of data in the current message. **Acknowledgment Number field** (32 bits) contains the value of the next sequence number that the sender of the segment is expecting to receive if the ACK control bit is set. The Flags is a set of six values after the reserved bits. **Window field** (16 bits) specifies the size of the sender's receive window (that is, buffer space available for incoming data). **Checksum field** (16 bits) indicates whether the header was damaged in transit. **Urgent pointer field** (16 bits) points to the first urgent data byte in the packet. **Options field** (variable length) specifies various TCP options. The **data field** (variable length) contains upper-layer information.

- The **Source, Destination, Header length** are the same as explained before. **Flags** PSH and ACK are set. ACK refers to the acknowledgment from another device upon receiving. **PSH** is an indication by the sender that, if the receiving machine's TCP implementation has not yet provided the data it's received to the code that's reading the data, it should do so at that point. The **Window Size Value** indicates buffer space at one end for receiving. **Scaling Factor** refers to the multiplier for window size displayed.

```
Transmission Control Protocol, Src Port: 443, Dst Port: 63222, Seq: 1, Ack: 518, Len: 1370
    Source Port: 443
    Destination Port: 63222
    [Stream index: 66]
    [TCP Segment Len: 1370]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 4241500583
    [Next sequence number: 1371    (relative sequence number)]
    Acknowledgment number: 518    (relative ack number)
    Acknowledgment number (raw): 4267055052
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 2047
    [Calculated window size: 524032]
    [Window size scaling factor: 256]
    Checksum: 0xffe8 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (1370 bytes)
    [Reassembled PDU in frame: 5800]
    TCP segment data (1370 bytes)
```

UDP (User Datagram Protocol): This is another protocol that is used in this layer.

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. **Source Port** is a 2 Byte long field used to identify port number of sources. **Destination Port** is a 2 Byte long field, used to identify the port of destined packet. **Length** is a 16-bit field that specifies the length of UDP including header and the data. **Checksum** is used to verify that the end to end data has not been corrupted.

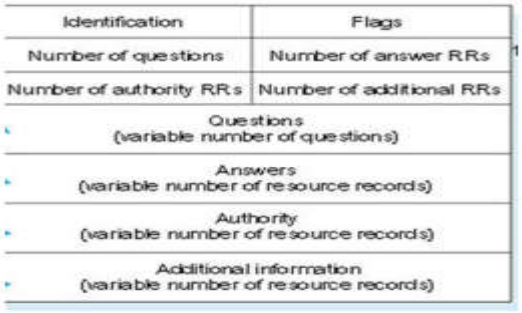
- The **Source Port** and **Destination Port** fields identify the endpoints of the connectionless interface. **Urgent** pointer field points to the first urgent data byte in the packet. **Options** field specifies various TCP options. **Length** indicates length of UDP including header and data. **Checksum** field indicates whether the header was damaged in transit.

```
User Datagram Protocol, Src Port: 53, Dst Port: 58310
    Source Port: 53
    Destination Port: 58310
    Length: 184
    Checksum: 0x2c47 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 64]
  > [Timestamps]
```

❖ **APPLICATION LAYER:**

Domain Name System (DNS): *This is a distributed database implemented in a hierarchy of DNS servers. and an application-layer protocol that allows hosts to query the distributed database.*

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

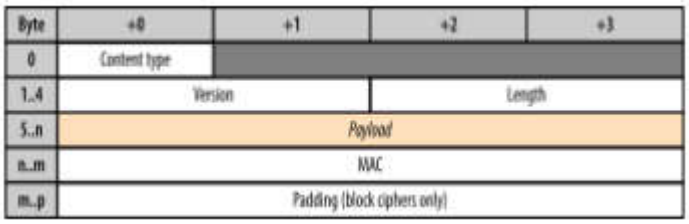



```
Domain Name System (query)
    Transaction ID: 0xa2f0
  Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  Queries
    > outlook.office.com: type A, class IN
    [Response In: 6859]
```

- **Transaction ID** is the 16-bit field to track queries and responses to queries. **Flags** control the content of the next four states: Questions, Answer RRs, Authority RRs and Additional RRs. **Questions c**ontain information about the query that is being made. 1 indicates that the query type is a question. **Answer** contains the resource records for the name that was originally queried (0). **Authority** contains records of other authoritative servers (0). **Additional** contains other helpful records. (0)

TLSv1.2 (SSL) and HTTP: *TLSv1.2 provides security in communication by encrypting the application data. The basic unit of data in SSL is a record. Each record consists of a 5-byte record header, followed by data.*

- The record format is **Type** (Handshake, Application Data, Alert and Change Cipher Specifications), **Version** and **Length. MAC** is the message authentication code.




```
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 78
    Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 74
        Version: TLS 1.2 (0x0303)
      > Random: c8057c44e9a3820894d36c8acd167272b6135bc5e64165e9…
        Session ID Length: 0
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Compression Method: null (0)
        Extensions Length: 34
      > Extension: renegotiation_info (len=1)
      > Extension: server_name (len=0)
      > Extension: ec_point_formats (len=4)
      > Extension: session_ticket (len=0)
      > Extension: status_request (len=0)
      > Extension: application_layer_protocol_negotiation (len=5)
```

HTTP is used in the application layer. It is the underlying protocol for application data transfer over the web which is encrypted by the TLS layer during transportation to provide data security. Hence no standalone http packets are received, instead they are all encrypted with TLS. **Length** indicates the length of the packet sent (including header).

Answer 2:

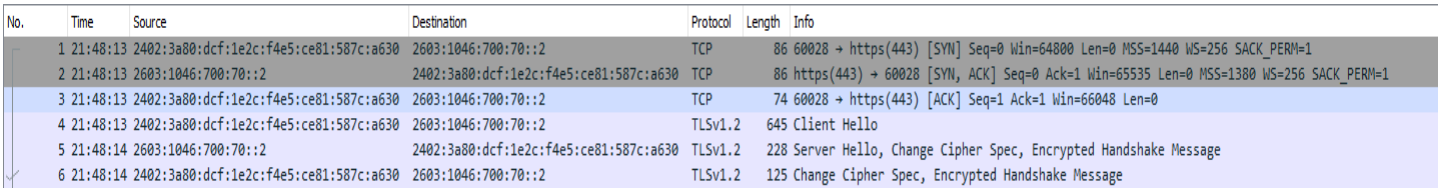
Some of the applications of the Outlook client are as follows:

- Login with Password
- Loading the inbox initially
- Refreshing Inbox
- Send mail
- Compose mail and save to drafts
- Create an outlook group
- Create a personal folder
- Move/Copy emails from one folder to another
- Download attachments in the mails
- Move emails to trash, i.e. deleting the mails
- Logout from the application

The protocols used in the above applications are:

- **Ethernet II:** Being the most widely used data link layer protocol, it is used most in the application since it has a reliability rate of data transfer coupled with good bandwidth and security. It also allows proper error handling, which is essential in composing and sending mails.
- **Transmission Control Protocol:** TCP is used for data communications between the user client and the outlook servers. It ensures safe transport of data from source to destination without any interceptions in between. If for some reason, the packers sent to the receiver don't get an acknowledgment to the sender, an appropriate error message will be sent to the sender so that he/she may send it again if required. Besides, it also ensures ordered delivery of the packets at the receiver's end if they arrive out of order. It also takes care of proper error handling to an extent since there is little scope for affording data loss over email.
- **IPv4:** This is a connection less protocol which allows data communication over packet switched networks. It is coupled with TCP because TCP is solely compatible with IP at the Network layer.
- **TLSv1.2:** Outlook client is secured by a TLS certificate. It is a SSL protocol encrypting application data and preventing hackers to gain access to important information that might lead to breaches in security. Login credentials etc. are safely transmitted from sender to receiver without granting access to any third party because of the presence of this protocol.
- **DNS:** This is used to map domain names to IP addresses, indicating the server's address (either MAC or IP) the client is trying to connect in order to use the application. DNS helps to translate numbered IP or MAC addresses to user-readable domains like google.com instead of 172.217.10.142.

Answer 3:



No.	Time	Source	Destination	Protocol	Length	Info
1	21:48:13	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TCP	86	60028 → https(443) [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS=256 SACK_PERM=1
2	21:48:13	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	86	https(443) → 60028 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1380 WS=256 SACK_PERM=1
3	21:48:13	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TCP	74	60028 → https(443) [ACK] Seq=1 Ack=1 Win=66048 Len=0
4	21:48:13	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TLSv1.2	645	Client Hello
5	21:48:14	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TLSv1.2	228	Server Hello, Change Cipher Spec, Encrypted Handshake Message
6	21:48:14	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TLSv1.2	125	Change Cipher Spec, Encrypted Handshake Message

LOADING THE INBOX INITIALLY:

On loading the inbox for the first time, two handshaking procedures take place:

- 3-way TCP handshaking (first three in the above screenshot): This creates a connection between my PC and outlook server in a TCP/IP network. It happens in three steps between ports 60028(my PC) and 443(server) ->
  - SYN: My PC sends a packet to the server with required syn and informs the server of the starting of the communication.
  - SYN, ACK: The server returns a response with SYN_ACK bit 1. ACK signifies that the server has acknowledged the client's (my PC) SYN and the SYN of this indicates the server's sequence number with which it would start the communication.
  - ACK: My PC then sends a message with the ACK bit as 1 and acknowledges the server's response. Thus, the connection is established between the client and the server.
- TLS handshaking (the last 3): This makes the above established connection secure. "Client Hello" initiates the session with the server, the server responds with a "Server hello" message containing the server certificate and the client responds with a client key and then a secure connection is established.

No.	Time	Source	Destination	Protocol	Length	Info
34	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=10071 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
35	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=11451 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
36	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=12831 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
37	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=14211 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
38	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=15591 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
39	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=16971 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
40	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=18351 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
41	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=19731 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
42	21:48:15	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TCP	74	60028 → 443 [ACK] Seq=13509 Ack=21111 Win=66048 Len=0
43	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=21111 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
44	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TLSv1.2	1454	Application Data, Application Data

REFRESHING THE INBOX:

On refreshing the inbox, server sends data to the outlook client. Application data with TCP IP headers make up a segment. The client starts by sending acknowledgement packets on successfully receiving data from the server. Even though the packets arrive out of order, TCP looks through them and puts them in an ordered fashion before delivering the final response to the client, that is, my PC.

66	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=51471 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
67	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=52851 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
68	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=54231 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
69	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=55611 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
70	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=56991 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
71	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=58371 Ack=13509 Win=523776 Len=1380 [TCP segment of a reassembled PDU]
72	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TLSv1.2	451	Application Data, Application Data, Application Data, Application Data
73	21:48:15	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TCP	74	60028 → 443 [ACK] Seq=13509 Ack=60128 Win=66048 Len=0
74	21:48:15	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TLSv1.2	3629	Application Data
75	21:48:15	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TLSv1.2	3283	Application Data
76	21:48:15	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TLSv1.2	3674	Application Data
77	21:48:15	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	2603:1046:700:70::2	TLSv1.2	3674	Application Data
78	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	74	443 → 60028 [ACK] Seq=60128 Ack=16269 Win=524288 Len=0
79	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	74	443 → 60028 [ACK] Seq=60128 Ack=18444 Win=524288 Len=0
80	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	74	443 → 60028 [ACK] Seq=60128 Ack=20273 Win=524288 Len=0
81	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=60128 Ack=20273 Win=524288 Len=1380 [TCP segment of a reassembled PDU]
82	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=61508 Ack=20273 Win=524288 Len=1380 [TCP segment of a reassembled PDU]
83	21:48:15	2603:1046:700:70::2	2402:3a80:dcf:1e2c:f4e5:ce81:587c:a630	TCP	1454	443 → 60028 [ACK] Seq=62888 Ack=20273 Win=524288 Len=1380 [TCP segment of a reassembled PDU]

SENDING MAIL:

When a mail is to be sent, my PC first sends the data to the server in the form of TCP packets which are encrypted using a shared secret key. This is then acknowledged by the server through an ACK message. There are TLS packets in between the TCP packets which indicates and ensures that the data transfer is successful and secure between the client and the server. These TCP packets may take different routes to travel from source to destination, subject to network congestion and load balancing, and may arrive at the destination out of order, but TCP ensures the rearrangement of the packets to the original order.

Handshaking:

The handshaking process takes place in order to establish rules for communication when a computer tries to connect with another device. In addition to exchanging protocol information, handshaking also verifies the quality or bandwidth of the connection, as well as any secure authority that may be required to complete the connection between devices. Handshaking takes place only when the connection is established with a particular server IP. Handshaking messages were found only in case of loading the inbox initially because the protocol for communication is only configured when the connection is established.

Answer 4:

STATISTICS ->

	12:00 AM	4:00 AM	10:00 PM
THROUGHPUT (BYTES/SEC)	70k	1.4K	53K
ROUND TRIP TIME (ms)	53	48	43
AVG. PACKET SIZE (BYTES)	653	134	850
NO OF PACKETS LOST	0	0	0
NO OF TCP PACKETS	14537	128	2872
NO OF UDP PACKETS	2119	628	0
NO OF RESPONSES PER REQUEST SENT	1.43	1.12	1.53

Answer 5:

To achieve uniformity in experiment results, I performed wireshark traces multiple times during the day. All the experiences were performed using my mobile data. Two server IP addresses were observed: 13.107.18.11 and 40.100.141.162. Outlook is a website having huge traffic at almost all times of the day. Hence, they use multiple servers to fasten up the data transfer which happens due to **Load Balancing** of data across servers since there is little **network congestion & increased reliability.** Different servers are also helpful in ensuring reliability since there is **no single point of failure**. Even if some server experiences some issues, others can provide data to the client without any sort of interruptions.

On checking the TLS packets and the DNS queries made more carefully, I found that during the course of the various applications of the outlook client, connections were made to many different servers which were not directly related to the outlook client, but performed additional tasks. For example:

- google.com: Since this is my default search engine, whenever I typed the website address, it connects to google to show my default search suggestions.
- fonts.googleapis.com: This is required for the display of the webpages in the default google fonts.
- google-analytics.com: Google Analytics is a web analytics service offered by Google that tracks and reports website traffic and hence, always runs in the background when any website is accessed.