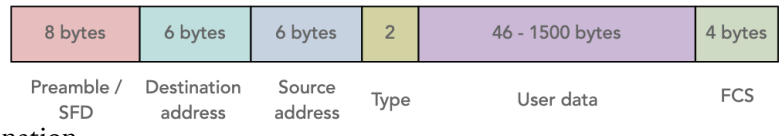
Parth Bakare
180101056
CS342 : Networks Lab, Assignment-2

Wireshark traces: https://rb.gy/sjdvv6

Question 1.

(a) Data link layer

Ethernet II



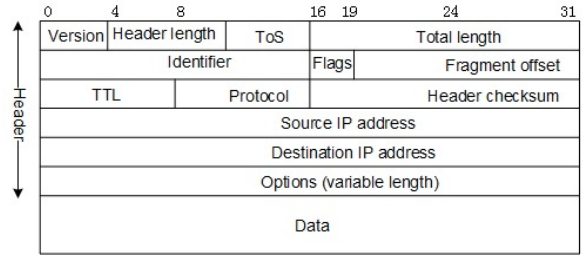
Preamble and SFD (start frame delimiter): Beginning of the packet, enables bit synchronization between the source and the destination.
Destination address: MAC address of the destination machine.
Source address: MAC address of the source machine.
Type: Identifies the ethernet type.
Data: Data to be transferred.
CRC (cyclic redundancy check): Detects errors in the data.

```
▼ Ethernet II, Src: ee:01:ee:d0:75:d2 (ee:01:ee:d0:75:d2), Dst: Apple_cc:75:12 (30:35:ad:cc:75:12)
  ▶ Destination: Apple_cc:75:12 (30:35:ad:cc:75:12)
  ▶ Source: ee:01:ee:d0:75:d2 (ee:01:ee:d0:75:d2)
    Type: IPv4 (0x0800)
```

Destination address *30:35:ad:cc:75:12* (my laptop),
Source address *ee:01:ee:d0:75:d2* (outlook.com's server), Type *IPv4*

(b) Network layer

Internet Protocol Version 4



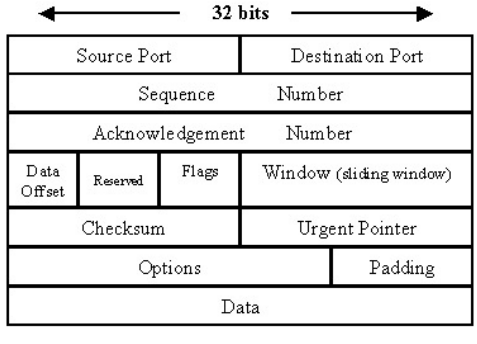
Version: IP version used.
Header length: Length of the header.
ToS (type of service): DSCP (differentiated services code point) and ECN (explicit congestion notification, route congestion information)
Total length: Length of the IPv6 packet.
Identifier: Identifies the original IP packet if it gets fragmented.
Fragment offset: Position of the fragment in the original packet.
TTL (time to live): Maximum number of hops allowed for the packet.
Protocol: Protocol used in next layer.
Header checksum: Used for error detection.
Source and Destination addresses: Addresses of the packet originator and the recipient respectively.

```
▼ Internet Protocol Version 4, Src: 49.44.204.73, Dst: 192.168.43.128
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x28 (DSCP: AF11, ECN: Not-ECT)
    Total Length: 1410
    Identification: 0x535e (21342)
  ▶ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 53
    Protocol: TCP (6)
    Header checksum: 0x0352 [validation disabled]
    [Header checksum status: Unverified]
    Source: 49.44.204.73
    Destination: 192.168.43.128
```

Version *4*, Header length 20 *bytes*,
Flags *0x4000* (don't fragment), Fragment offset *0*,
TTL *53*, Protocol *TCP*, Header checksum *0x0352*,
Source *49.44.204.73*, Destination *192.168.43.128*

(c) Transport layer

Transmission Control Protocol



Source and Destination ports: End points of the connection.
Sequence number: Records the order of the data.
Acknowledgement number: Sequence number of the next packet.
Data offset: Number of 32-bit words in the header.
Reserved: Has the value 0, makes the total size a multiple of 4.
Flags: 6 flags, 1-bit each(URG, ACK etc).
Window: Size of the receiving window of the sender.
Checksum: Indicates if the data was corrupted.
Urgent pointer: Points to first urgent data byte.
Options: Specifies various TCP options.
Data: Upper-layer information.

```
▼ Transmission Control Protocol, Src Port: 49703, Dst Port: 443, Seq: 1, Ack: 1, Len: 574
    Source Port: 49703
    Destination Port: 443
    [Stream index: 11]
    [TCP Segment Len: 574]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 1400680583
    [Next sequence number: 575    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 2723643741
    0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 4096
    [Calculated window size: 262144]
    [Window size scaling factor: 64]
    Checksum: 0xf780 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
    TCP payload (574 bytes)
```

Source port 49703, Destination port *443*,
Sequence no. *1*, Acknowledgement no. *1*,
Flags *PSH, ACK* (data should be passed to the application as soon as possible, acknowledgement no. is valid),
Window size *4096*,
Checksum *0xf780*, Urgent pointer *0*

User Datagram Protocol
Source and Destination ports: End points of the connection.
Length: Length of the packet including the header and the data.
Checksum: Indicates if the data was corrupted.

16 bits each

Source port	Destination port
Length	Checksum

```
▼ User Datagram Protocol, Src Port: 53, Dst Port: 57033
    Source Port: 53
    Destination Port: 57033
    Length: 299
    Checksum: 0xc218 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 39]
  ▶ [Timestamps]
```

Source port *53*, Destination port *57033*,
Length *299*, Checksum *0xc218*

(d) Application layer

Transport Layer Security
Type: Handshake, Application Data, Alert, Change Cipher Spec.
Version: TLS version.
Length: Packet size.
MAC: Message Authentication Code.

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

```
▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 569
    ▶ Handshake Protocol: Client Hello
```

Type *Handshake*, Version *TLS 1.0(0x0301)*,
Length *569*

Domain Name System
Identification: Identifies the query.
Flags: Query/Reply flag, Authoritative flag etc.
Question section: Information about the query being made.
Answer section: Resource records for the name originally queried.
Authority section: Records of other authoritative servers.
Additional section: Other helpful records.

Identification	Flags
Number of questions	Number of answer RRs
Number of authority RRs	Number of additional RRs
Questions (variable number of questions)	
Answers (variable number of resource records)	
Authority (variable number of resource records)	
Additional information (variable number of resource records)	

```
▼ Domain Name System (response)
    Transaction ID: 0x4ac2
  ▶ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 10
    Authority RRs: 0
    Additional RRs: 0
  ▶ Queries
  ▶ Answers
    [Request In: 10884]
    [Time: 0.231424000 seconds]
```

Transaction ID *0x4ac2*, Flags *0x8180*, Questions *1*, Answer RRs *10*, Authority RRs *0*, Additional RRs *0*

Question 2.

Some common functionalities:
(a) Opening the website in the browser
(b) Logging into the account
(c) Sending an email
(d) Deleting an email
(e) Downloading an attachment
(f) Refreshing the inbox
(g) Logging out of the account
(h) Closing the application tab in the browser

Ethernet II is a widely used data link layer protocol. Its preamble enables synchronization and its CRC field detects errors which makes it very good at correcting errors. It is very reliable, secure and provides higher speeds than other protocols.

IPv4 is a connection-less protocol used for packet switched networks like the internet. It is not very reliable and often sends data packets out of order, so a mechanism must be present which can reassemble these jumbled packets, which is what TCP provides.

TCP is used at almost every step in all outlook functionalities. TCP establishes a connection between two points through handshakes and enables data transfers through this link. TCP is very reliable, and even if a packet gets lost during an exchange, TCP sends it again (the ACK feature supports this). TCP is capable of reassembling jumbled packets and correcting errors too, which is of utmost importance in an email service.

UDP is a connection-less protocol and provides faster data transfer than TCP but is not very reliable or secure. It is mostly used to transfer small individual packets (like DNS requests). Outlook uses DNS queries to fetch IP addresses, which are made using UDP.

TLS provides network security and protects private information by encrypting data. For example, TLS is used to send private data like account password in outlook by encrypting the password so that it would not be found out even if malicious hackers sniff the data and catch hold of it.

DNS is used to map domain names to IP addresses, and returns the IP address of the server to which the client needs to connect. DNS makes it easier for users and machines to understand domain names and IP addresses.

Question 3.

(a) TCP 3-way handshake

192.168.43.128	13.107.18.11	TCP	78 52813 → 443 [SYN, ECN, CWR] S
13.107.18.11	192.168.43.128	TCP	66 443 → 52813 [SYN, ACK, ECN] S
192.168.43.128	13.107.18.11	TCP	54 52813 → 443 [ACK] Seq=1 Ack=1

The TCP 3-way handshake sets up the connection between the server(outlook.office.com) and the client (my laptop). Here, 52813 is the client port and 443 is the server port.
(i) The client sends an SYN (synchronize) packet to the server which informs the server that a connection is about to be established, and the sequence number is sent as well.
(ii) The server sends an ACK (acknowledgement), which signifies the response of the segment and an SYN to Inform about the sequence number for the starting segments.
(iii) The client sends an ACK to establish a stable connection, and data transfer can begin.

(b) TLS handshake

192.168.43.128	13.107.18.11	TLSv1.2	571 Client Hello
13.107.18.11	192.168.43.128	TCP	54 443 → 52813 [ACK] Seq=1 A
13.107.18.11	192.168.43.128	TCP	1424 443 → 52813 [ACK] Seq=1 A
13.107.18.11	192.168.43.128	TCP	1424 443 → 52813 [ACK] Seq=137
192.168.43.128	13.107.18.11	TCP	54 52813 → 443 [ACK] Seq=518
13.107.18.11	192.168.43.128	TCP	1424 443 → 52813 [ACK] Seq=274
13.107.18.11	192.168.43.128	TLSv1.2	338 Server Hello, Certificate

The client sends a “Client Hello” message to the server using TLS, and the server responds with a “Server Hello” message, along with the server certificate (for authentication). The TLS session is now established and application data can be exchanged between the server and the data.

TCP 3-way handshake and TLS handshake occur while opening outlook in the browser in the beginning.

(c) Sending an email

```
192.168.43.128   52.98.88.242     TLSv1.2    1179 Application Data
52.98.88.242     192.168.43.128   TCP          54 443 → 53701 [ACK] Seq=4690 Ack=4939 Win=524544 Len=0
52.98.88.242     192.168.43.128   TCP          54 443 → 53701 [ACK] Seq=4690 Ack=7434 Win=524544 Len=0
52.98.88.242     192.168.43.128   TCP        1424 443 → 53701 [ACK] Seq=4690 Ack=7434 Win=524544 Len=1370 [TCP segment of a reassembled PDU]
52.98.88.242     192.168.43.128   TCP        1424 443 → 53701 [ACK] Seq=6060 Ack=7434 Win=524544 Len=1370 [TCP segment of a reassembled PDU]
192.168.43.128   52.98.88.242     TCP          54 53701 → 443 [ACK] Seq=7434 Ack=7430 Win=260736 Len=0
52.98.88.242     192.168.43.128   TLSv1.2     500 Application Data
192.168.43.128   52.98.88.242     TCP          54 53701 → 443 [ACK] Seq=7434 Ack=7876 Win=261696 Len=0
192.168.43.128   52.98.88.242     TCP        1424 53701 → 443 [ACK] Seq=7434 Ack=7876 Win=262144 Len=1370 [TCP segment of a reassembled PDU]
192.168.43.128   52.98.88.242     TCP        1424 53701 → 443 [ACK] Seq=8804 Ack=7876 Win=262144 Len=1370 [TCP segment of a reassembled PDU]
192.168.43.128   52.98.88.242     TCP        1424 53701 → 443 [ACK] Seq=10174 Ack=7876 Win=262144 Len=1370 [TCP segment of a reassembled PDU]
192.168.43.128   52.98.88.242     TLSv1.2     207 Application Data
```

The client sends data to the server through TCP packets, and TLS ensures that the exchanged data is encrypted. Again, the data needs to be reassembled because the packets may arrive out of order (similar to the previous situation).

(d) Refreshing inbox

```
49.44.204.73     192.168.43.128   TLSv1.2    1424 Application Data
49.44.204.73     192.168.43.128   TLSv1.2    1424 Application Data [TCP segment of a reassembled PDU]
192.168.43.128   49.44.204.73     TCP          66 53204 → 443 [ACK] Seq=1534 Ack=24656 Win=129664 Len
49.44.204.73     192.168.43.128   TLSv1.2    1424 Application Data [TCP segment of a reassembled PDU]
49.44.204.73     192.168.43.128   TLSv1.2    1424 Application Data, Application Data
192.168.43.128   49.44.204.73     TCP          66 53204 → 443 [ACK] Seq=1534 Ack=27372 Win=126976 Len
192.168.43.128   49.44.204.73     TCP          66 [TCP Window Update] 53204 → 443 [ACK] Seq=1534 Ack=
```

```
▼ [2 Reassembled TCP Segments (1053 bytes): #11567(305), #11568(748)]
    [Frame: 11567, payload: 0-304 (305 bytes)]
    [Frame: 11568, payload: 305-1052 (748 bytes)]
    [Segment count: 2]
    [Reassembled TCP length: 1053]
    [Reassembled TCP Data: 17030304184062275e7f5774bf10740dc2c070c2192a8def…]
```

The server sends application data to the client when the inbox is refreshed, and the client respond by sending ACK packets. As shown in the image, the packets (PDU, protocol data unit) need to be reassembled because they might arrive out of order (by using different routes, to ensure load balancing).

Handshaking

Handshaking occurs just before a connection is established between the server and the client ie before launching the website. A connection was already setup before sending an email and before refreshing an inbox, so there were no handshaking sequences in these two functionalities. However, we will find handshakes in these functionalities if a different server (than the one with which our initial connection was setup) processes our requests.

Question 4.

	10:50 AM	12:30 PM	6:20 PM
Throughput (kilobytes/sec)	68	32	5.445
RTT (s)	0.0502	0.079	0.044
Average packet size (bytes)	692	656	499
No. of packets lost	0	0	0
No. of TCP packets	20611	17693	10025
No. of UDP packets	596	1986	2584
No. of responses per Request sent	11064/10146 = 1.090	9709/9969 = 0.974	5853/6755 = 0.866

Question 5.

IP addresses observed at different times of the day:

10:50 AM	*52.98.88.66*
12:30 PM	*40.100.141.162*
6:20 PM	*40.100.50.114*

Outlook has multiple servers setup in various locations around the world. A request is sent to any one server depending upon the network traffic and congestion, in a manner that distributes the requests uniformly. This is called load balancing, and it keeps the network traffic stable. The use of multiple servers also increases the reliability of the system, as even if a server fails, other servers will provide alternative routes between two points and keep the website functioning smoothly.