CS 349 NETWORKS LAB
Assignment 2
PRIYANSHU SINGH 170101049

Application: Dropbox

Q.1

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	177	100.0	40502	1,480	0	0	0
Ethernet	100.0	177	6.1	2478	90	0	0	0
Internet Protocol Version 4	100.0	177	8.7	3540	129	0	0	0
Transmission Control Protocol	100.0	177	85.1	34484	1,260	87	8576	313
Transport Layer Security	50.8	90	74.6	30229	1,105	90	30229	1,105

Protocols used by application at different layers (as seen from the traces) are: -

1. **TLSv1.2 (Transport Layer Security): Session Layer**

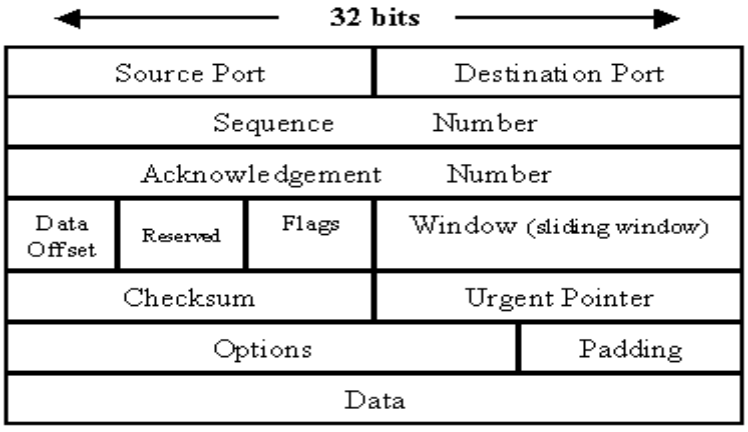
   TLS protocol sits between the **Application Layer** and the **Transport Layer**. It aims primarily to provide privacy and data integrity between two or more communicating computer applications using **cryptographic security**. The connection is *reliable* because each message transmitted includes a message integrity check using a **message authentication code** to prevent undetected loss or alteration of the data during transmission.

   The basic unit of data is a record. Each record consists of a five-byte record header, followed by data.

   - Record Format: Type (uint8), Version(uint16), Length(uint16)
   - Record Type: Handshake (22, 0x16), Change Cipher Spec (20, 0x14), Alert (21, 0x15), Application Data (23, 0x17)
   - Record Version: The record version is a 16-byte value and is formatted in network order.
   - Record Length: The record length is a 16-byte value and is formatted in network order.

2. **TCP (Transmission Control Protocol): Transport Layer**

   TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. The Transmission Control Protocol provides a communication service at an intermediate level between an application program and the Internet Protocol. At the transport layer, TCP handles all handshaking and transmission details and presents an abstraction of the network connection to the application typically through a network socket interface. The TCP packet format consists of these fields:



   - **Source Port** and **Destination Port fields** (16 bits each) identify the end points of the connection.
   - **Sequence Number field** (32 bits) specifies the number assigned to the first byte of data in the current message
   - **Acknowledgement 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.
   - **Data Offset (a.k.a. Header Length) field** (variable length) tells how many 32-bit words are contained in the TCP header.
   - **Reserved field** (6 bits) must be zero. This is for future use.
   - Flags field (6 bits) - Contains one of URG, ACK, PSH, RST, SYN, FIN flags.

- **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. Options have up to three fields: Option-Kind (1byte), Option-Length (1 byte), Option-Data(variable).
- **Data field** (variable length) contains upper-layer information.

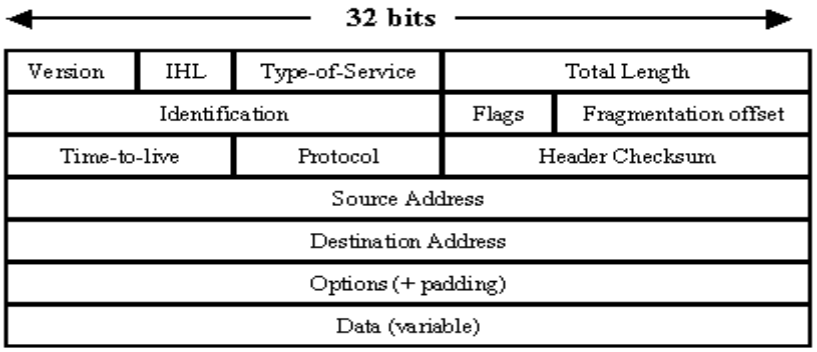
3. **DB-LSP-DISC (Transport Layer)**

It stands for Dropbox LAN Sync Protocol. It is a Dropbox feature that speeds syncing dramatically when the file exists on your Local Area Network (LAN). It sends a UDP broadcast packet in the local network at periodic time. The source and destination port of this packet is set to 17500. This UDP packet has some payload attached to it for identify itself to the receiver. It contains a JavaScript object notation.

4. **IPv4 (Internet Protocol): Network Layer**

IPv4 is a connectionless protocol for use on packet-switched networks. It operates on a best effort delivery model. The encapsulated data is referred to as IP Payload. IP header contains all the necessary information to deliver the packet at the other end. IPv4 packet details are as follows:

- **Version field** (4 bits) indicates the version of IP currently used.
- **IP Header Length (IHL) field** (4 bits) indicates how many 32-bit words are in the IP header.
- **Type-of-service field** (8 bits) specifies how a particular upper-layer protocol would like the current datagram to be handled.
- **Total Length field** (16 bits) specifies the length of the entire IP packet, including data and header, in bytes.
- **Identification field** (16 bits) contains an integer that identifies the current datagram.
- **Flags field** (4 bits; one is not used) controls whether routers are allowed to fragment a packet and indicates the parts of a packet to the receiver.
- **Time-to-live field** (8 bits) maintains a counter that gradually decrements to zero, at which point the datagram is discarded. This keeps packets from looping endlessly.
- **Protocol field** (8 bits) indicates which upper-layer protocol receives incoming packets after IP processing is complete.
- **Header Checksum field** (16 bits) helps ensure IP header integrity.
- **Source and Destination address field (**32 bits each) specifies sending and receiving node.
- **Options field** (32 bits) allows IP to support various options, such as security.
- **Data field** (32 bits) contains upper-layer information.

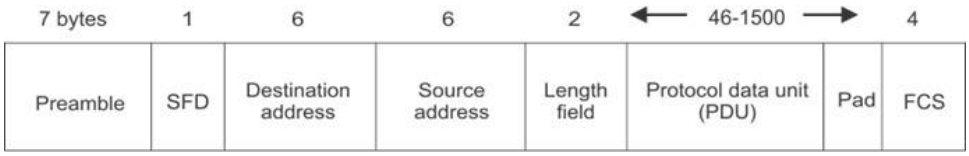


5. **Ethernet II: Data Link Layer**

Data link layer protocols are point to point protocols. The data link layer is concerned with local delivery of frames between devices on the same LAN. The Frame format consists of:

- **Preamble** (7 bytes): This is a stream of bits (alternating 1's and 0's) used to allow the transmitter and reciever to synchronize their communication.
- **Start of frame delimiter (SFD) –** This is a 1-Byte field which is always set to 10101011.
- **Source and Destination address (**6 bytes): contains MAC addresses of sending and receiving machines.
- **Length –** Length is a 2-Byte field, which indicates the length of entire Ethernet frame.
- **Data –** This is the place where actual data is inserted, also known as **Payload**.

- **FCS** (4 bytes): This field contains the Frame Check Sequence (FCS) which is calculated using a Cyclic Redundancy Check (CRC). The FCS allows Ethernet to detect errors in the Ethernet frame and reject the frame if it appears damaged.



Q.2

Here are the observed values for Packet number **112**. All the fields have been explained in Question 1.

TLSv1.2

Field	Value
Record Type	Application Data
Version	TLS 1.2
Length	34
Encrypted Application Data	Cf3af0a0aa0b05.....

TCP

Field	Value
Source Port	443
Destination Port	33198
Sequence Number	1226
Acknowledgment Number	1364
Header Length	32 bytes (multiple of 8 bytes)
Flags	0x018 (PSH, ACK)
Window size value	1232
Checksum	0x72ec (unverified)
Urgent Pointer	0
Options	12 bytes (Timestamps, No-operation NOP)

DB-LSP-DISC

Field	Value
Host_int	826070237118762321770693322540849814 59
Version	[2,0]
Display Name	(NULL)
Port	17500
Namespaces	3001767440

IPv4

Field	Value
Version	4
Header Length	20 bytes (multiple of 4 bytes)
Type of service field	0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length	91
Identification	0x86ab
Flags	0x0000
Time to live	61
Protocol	TCP
Checksum	0x30af (unverified)
Destination Port	maa03s31-in-f14.1e100.net (216.58.196.174)
Source Port	10.12.31.78

Ethernet II

Field	Value
Destination	Cisco_bf:20:cd (f8:0b:cb:bf:20:cd)
Source	AsustekC_be:12:e9 (60:45:cb:be:12:e9)
Type	IPv4 (0x0800)]

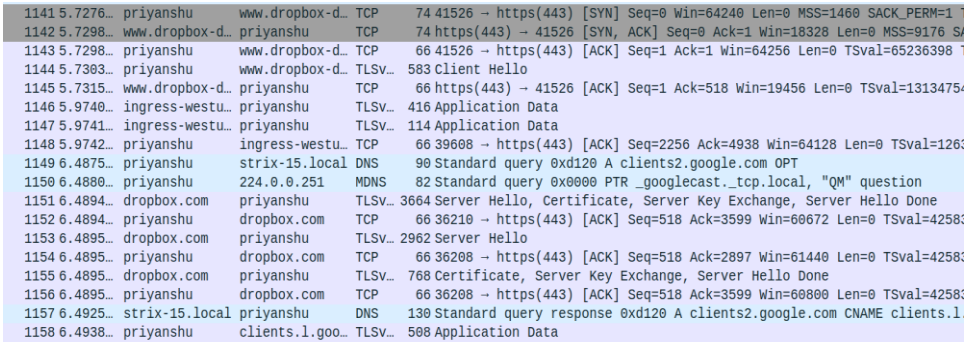
Q.3

Different protocols used during the application use are:

1. **TCP Connection Establishment (on opening of dropbox.com)**

To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way (or 3-step) handshake occurs:

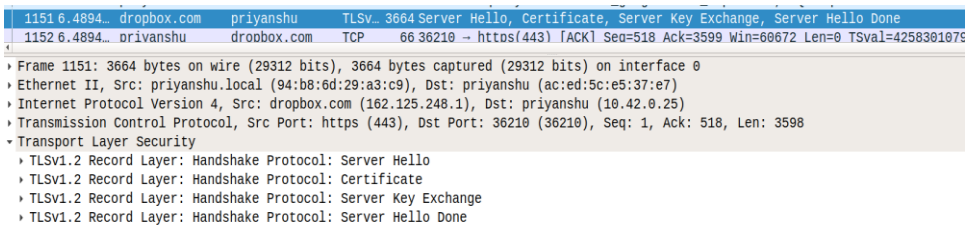
- **SYN**: The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.
- **SYN-ACK**: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.
- **ACK**: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1.



2. **Handshaking messages (on opening website after TCP connection)**

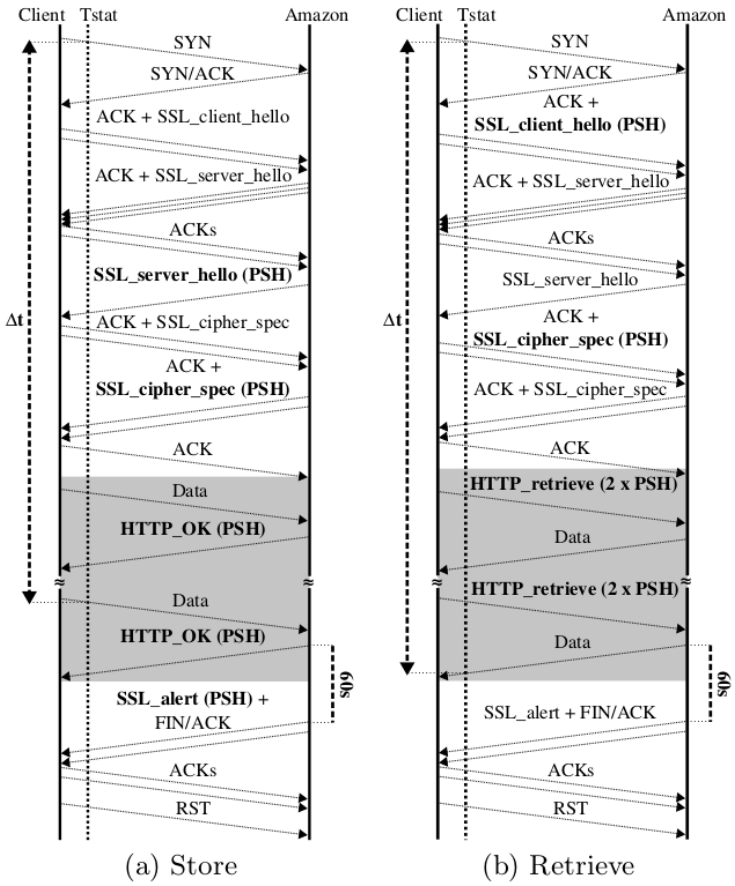
The Transport Layer Security (TLS) Handshake Protocol is responsible for the authentication and key exchange necessary to establish or resume secure sessions. When establishing a secure session, the Handshake Protocol manages the following:

- Client Hello is sent to the server which is responded back by Server Hello.
- The corresponding acknowledgements are also exchanged. This is followed by TLSv1.2 Change Cipher Spec Message to change the encryption during handshaking to switch to symmetric key encryption between client and server.
- Server responds correspondingly and finally connection is established to transfer data.
- Data is transferred from client to server (for storage) and server to client (for retrieval).



3. **Storage/Retrieval of Data (on uploading/downloading 185 MB file to dropbox)**

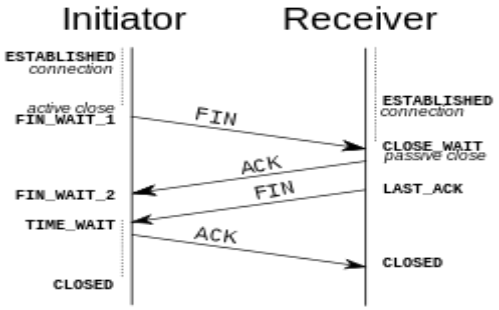
The Transport Layer Security (TLSv1.2) Application data Protocol is responsible for the transfer of data between client and server. Acknowledgement (ACK) is sent by client to server to notify server that the data packet is received.



4. **<u>TCP Connection Termination</u> (on closing dropbox tab on chrome)**

When the connection is closed, following protocols come into play:

- The connection termination phase uses a four-way handshake, with both side of the connection terminating independently.
- When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which the other end acknowledges with an ACK. The other endpoint also sends a FIN and expect a ACK from the first endpoint.
- After both FIN/ACK exchanges are concluded, the side which sent the first FIN before receiving one waits for a timeout before finally closing the connection, during which time the local port is unavailable for new connections.



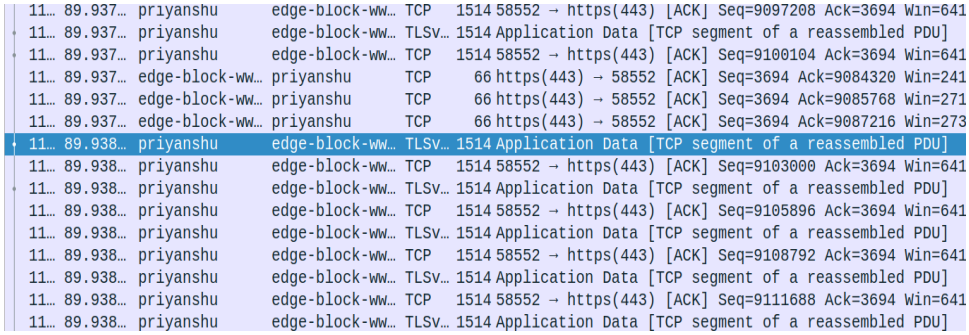
5. **<u>DB-LSP-SYNC</u> (for synchronization)**

This is done using Dropbox LAN SYNC Discovery Protocol. The DB-LSP DISC provides great sync of files across multiple devices, and into the cloud. When dropbox finds that you are on a LAN, it syncs directly between PCs. It sends a UDP broadcast packet in the local network. The source and destination port of the packet are set to 17500. This UDP packet has some payload attached to it for identify itself to the receiver. It seems like a dictionary as defined above.

Q.4

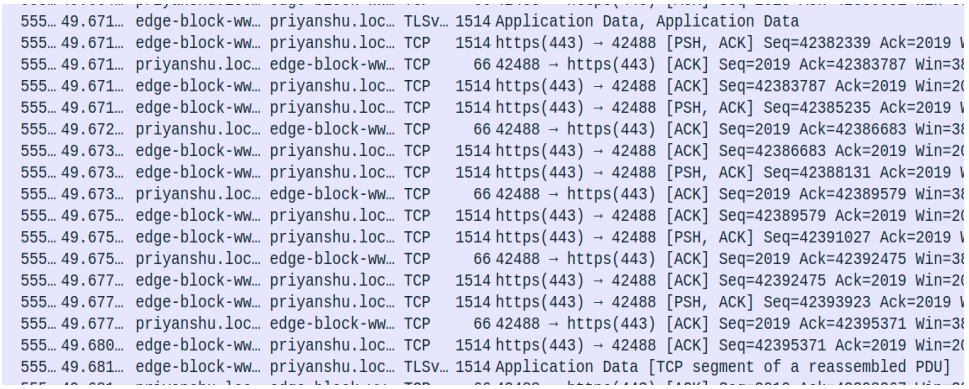
Some functionalities of the Application are:

1. **Uploading files on the cloud**



It can be seen that packets of data are being sent from my machine to dropbox server using TLSv1.2 protocol (for security) and dropbox server is acknowledging the receival of data.

2. **Downloading files from the cloud**



It can be seen that packets of data are sent from dropbox server to our machine and then there is a lot of exchange of **ACK** and [**PSH, ACK**] going on. ACK means that the machine sending the packet with ACK is acknowledging data that it had received from the other machine. In TCP, once the connection is established, all packets sent by either side will contain an ACK, even if it's just re-acknowledging data that it's already acknowledged 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 (program, or library used by a program), it should do so at that point.

3. **Synchronization**

DB-LSP-SYNC is the protocol of Dropbox which is mainly responsible for syncing folders across multiple devices connection over a local network as well as server. For example, if the PC is connected over a LAN, and there are shared dropbox folders across machines, this protocol synchronizes all shared folders across machines and also synchronizes the content with the server.

Yes, there is a handshaking sequence in the application. It occurs while establishing a **TCP connection** from my machine to the dropbox server. To establish a connection, TCP uses a three-way handshake. Handshaking messages are also exchanged by the TLS protocol. The Transport Layer Security (TLS) Handshake Protocol is responsible for the authentication and key exchange necessary to establish or resume secure sessions. These handshaking sequences are explained in Q.3 (point 1,2)

Q.5

Statistics from Traces related to the core functionality (uploading/downloading) of dropbox is shown in the table below.

Property	Data 1(LAN morning)	Data 2(LAN afternoon)	Data 3(Mobile Data Night)
Throughput (A->B) (in kbps)	6695	5656	4748
Throughput (B->A) (in kbps)	78	64	176
RTT (in ms)	0.407	0.359	204.77
Avg. Packet Size (in bytes)	1212.61	1217.57	1004.56
No. of packets lost	3677 out of 1,62,478	3749 out of 1,66,064	1457 out of 2,02,051
TCP packets (A->B)	1,14,124	1,14,268	1,08,562
TCP packets (B->A)	29,368	28,639	60,536
Number of UDP packets	0	0	0
Response received wrt one packet sent i.e. packets B->A/packets A->B	0.257	0.250	0.557

Notes

1. TCP packets considered are only the packets that are involved in the uploading of data at dropbox.
2. In considering no of packets lost, all packets are considered which includes opening of dropbox and then uploading the data.
3. Throughput is considered only for the packets involved in the main part of application (i.e. uploading of data).
4. Throughput, Avg Packet Size, No of TCP and UDP packets are all taken from Statistics->Conversations tab.
5. No of packets lost are found by filters *tcp.analysis.retransmission* and *tcp.analysis.ack_lost_segment* .
6. RTT time is calculated by seeing the response time for ACK packet sent during the establishment of 1st TCP handshake.(using *tcp.flags.syn==1*)

Q.6

Yes, the whole content was being sent (either upload or download) from different IP addresses. The mutiple IPs are mentioned in the table below (list is not exhaustive)

Time	Multiple IP addresses
LAN Morning	162.125.82.6, 104.16.100.29, 162.125.248.1, 162.125.81.6
LAN Afternoon	162.125.82.6, 162.125.19.131, 162.125.82.5, 104.16.99.29
Mobile Data Night	162.125.248.1, 162.125.82.6, 162.125.81.5, 162.125.19.131, 162.125.81.6

Some of the reasons for multiple IP addresses are:

- Modern day websites deploy **load balancing** on servers, data is sent to clients from various IPs in most efficient manner. If you have multiple IP networks on the same physical/logical network/vlan it will prevent traffic from being exchanged via the gateway, speeding things up and reducing the load.
- **Round Robin DNS** mechanism for faster fetching of relevant pages by balancing the page requests across many servers may lead to multiple IP.
- Data can be sent to another IP to compensate for a host that's down at that moment by adding its IP address to another one.
- It is also due to the fact that different functionalities are assigned different IP addresses, i.e. IP for upload is different in some cases with the IP for download.