CS349 ASSIGNMENT 2

Rashi Singh
170101052

Network Protocol Analysis Of Dropbox Using Wireshark

Please find the traces [here]

Q1 : PROTOCOLS USED IN DIFF LAYERS

```
Protocol
▾ Frame
  ▾ Ethernet
    ▾ Internet Protocol Version 4
      ▾ User Datagram Protocol
          Dropbox LAN sync Discovery Protocol
          Domain Name System
      ▾ Transmission Control Protocol
          Transport Layer Security
```

Figure 1: Protocol Hierarchy used by DropBox

Network Layer

IPv4 : Internet Protocol Version 4

IPv4 is a **packet-switched**, network-layer protocol. It provides a **logical connection** between network devices by providing **identification** for each device and **routing** data among them over the underlying network. IP uses best effort delivery, i.e. it does not guarantee that packets would be delivered to the destined host, but it will do its best to reach the destination. Internet Protocol version 4 uses **32-bit logical** address.

IPv4 : Packet Structure

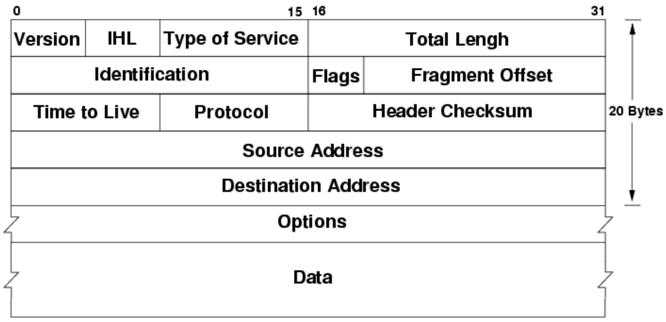


Figure 2: IPv4 Header Structure

```
▾ Internet Protocol Version 4, Src: 162.125.19.131, Dst: 10.16.0.46
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xf753 (63315)
  ▸ Flags: 0x0000
    ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 62
    Protocol: TCP (6)
    Header checksum: 0xc532 [validation disabled]
    [Header checksum status: Unverified]
    Source: 162.125.19.131
    Destination: 10.16.0.46
```

Figure 3: WireShark IPv4 layer packet details

VERSION	- (4 bit) Version of the Internet Protocol used
HEADER LENGTH	- (4 bit) Length of the IP header in 32 bit increments. **Min length of IP header is 20 bytes, so with 32 bit increments, min value of IHL is 5. **Max value of IHL is 15 so with 32 bit increments, max length of IP header is 60 bytes
TYPE OF SERVICE	- Provides an indication of the abstract parameters of the quality of service desired.
TOTAL LENGTH	- (16 bit) Length of the datagram (in bytes), including internet header and data.
IDENTIFICATION	- An identifying value assigned by the sender to help assemble the fragments of a datagram
FLAGS	- (3 bit) Various Control Flags
FRAGMENT OFFSET	- (13 bit) Indicating where in the datagram this fragment belongs.
TIME TO LIVE	- (8 bit) The max time the datagram is allowed to remain in the internet system.
PROTOCOL	- Next level protocol used in the data portion of the internet datagram.
HEADER CHECKSUM	- A checksum on the header only.
SOURCE/DEST ADDRESS	- (32 bit) Addr of the source/destination respectively

Transport Layer

TCP : Transport Control Protocol

TCP is a **connection oriented** protocol which offers **end-to-end packet delivery**. TCP ensures reliability by sequencing bytes with a forwarding **acknowledgement** number that indicates to the destination, the next byte the source expect to receive.It retransmits the bytes not acknowledged with in specified time period . **Dropbox** mainly used TCP to send application data mainly due its **reliability** feature

TCP : Packet Structure

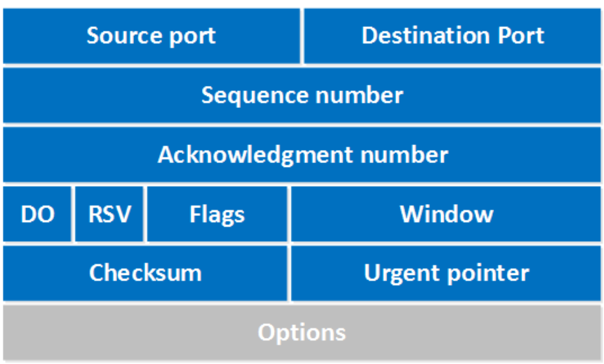


Figure 4: TCP header structure

```
▾ Transmission Control Protocol, Src Port: 443, Dst Port: 38288, Seq: 102, Ack: 95, Len: 0
    Source Port: 443
    Destination Port: 38288
    [Stream index: 18]
    [TCP Segment Len: 0]
    Sequence number: 102    (relative sequence number)
    [Next sequence number: 102    (relative sequence number)]
    Acknowledgment number: 95    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 513]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x1c9d [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▾ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 75]
      [The RTT to ACK the segment was: 0.000225906 seconds]
```

Figure 5: WireShark TCP packet details

SOURCE/DEST PORT	: (16 bit) Fields to identify the end points of the connection
SEQUENCE #	: (32 bit) Number assigned to the first byte of data in the current message

ACKNOWLEDGEMENT # : (32 bit) Value of the next sequence # that the sender of the segment is expecting to receive
DATA OFFSET : Specifies how many 32-bit words are contained in the TCP header.
RESERVED : (6 bit) Must be zero. This is for future use.
FLAGS : (6 bit) URG, ACK, PSH, RST, SYN, FIN
WINDOW : (16 bit) Specifies the size of the sender's receive window
CHECKSUM : (16 bit) Indicates whether the header was damaged in transit.
URGENT : pointer (16 bit) Points to the first urgent data byte in the packet.
OPTIONS : (variable length) Specifies various TCP options.
DATA : (variable length) Contains upper-layer information.

UDP : User Datagram Protocol

```
▾ User Datagram Protocol, Src Port: 17500, Dst Port: 17500
    Source Port: 17500
    Destination Port: 17500
    Length: 201
    Checksum: 0xfbcb [unverified]
    [Checksum Status: Unverified]
    [Stream index: 3]
  ▸ [Timestamps]
▸ Dropbox LAN sync Discovery Protocol
```

Figure 6: WireShark UDP packet details

UDP is **connectionless** and **unreliable** protocol. It doesn't require making a connection with the host to exchange data. Since UDP is unreliable protocol, there is no mechanism for ensuring that data sent is received

UDP is used by applications that typically transmit **small** amount of **data** at one time

Src/Dest Port, **Length**, **Checksum** - UDP Packet Structure

DB-LSP-DISC : DropBox LAN Sync Discovery Protocol

Dropbox LAN Sync (**DB-LSP**, a diff protocol) is a feature that allows you to **download files** from **other computers** on your network, saving time and bandwidth compared to downloading them from Dropbox servers.

Without LAN Sync, these requests would be queued up and sent to the block server, which would return block data.

For LAN sync to work, the **discovery engine** is responsible for **finding machines** on the network that we can sync with (i.e., machines which have access to namespaces in common with ours). To do this, each machine **periodically sends** and **listens** for **UDP broadcast packets** over port **17500**.

```
▾ Dropbox LAN sync Discovery Protocol
  ▾ JavaScript Object Notation
    ▾ Object
      ▸ Member Key: version
      ▸ Member Key: port
      ▸ Member Key: host_int
      ▸ Member Key: displayname
      ▸ Member Key: namespaces
```

Figure 7: WireShark DB-LSP-DISC packet details

The packet contains **VERSION** of protocol used by PC, **TCP PORT** of the server (17500), **HOST_INT** : a random identifier for the UDP packet to be identified by the receiver, the **NAMESPACES**** supported.

*****Namespaces** are the primitive behind Dropbox's permissions model. They can be thought of as a directory with specific permissions. Every account has a namespace which represents its personal Dropbox account*

DNS : Domain Name System :

```
▾ Domain Name System (response)
    Transaction ID: 0x5c4b
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 1
  ▸ Queries
  ▸ Answers
```

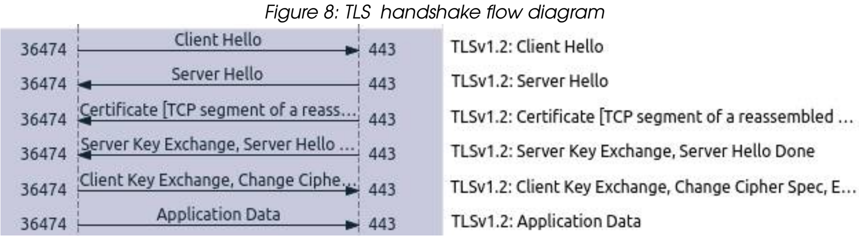
The Domain Name System is a **hierarchical** and **decentralized naming system** for computers, services, or other resources connected to the Internet or a private network.

Please refer Table 5: DNS - Domain Name System for the packet details

B/w Application, Transport Layer

TLSv1.2 : Transport Layer Security

Figure 8: TLS handshake flow diagram



It is a cryptographic protocol, developed from the generalized version of SSL(Secure Socket Layer)(now deprecated). It provides 3 essential services to the applications running above it

***TLS requires a reliable transport. Hence, it uses TCP*

Verification of validity of identity : **AUTHENTICATION** a)
Detection of msg tampering, forgery : **DATA INTEGRITY** b)
A mechanism to obfuscate what is sent from one host to another : **ENCRYPTION** c)

PACKETS for TLSv1.2

The common parameters present in diff. kinds of TLS packets are :

a) **VERSION** : 16 byte version
b) **LENGTH** : 16 byte record length
c) **CONTENT TYPE** : The type of TLS packet.

Some of the common types are :

1) **HANDSHAKE** – Please refer Q4 TLS HANDSHAKE for more info.
2) **APPLICATION_DATA** – This type of TLS packet has an additional field k/a **Encrypted Application Data**, which is the actual encrypted data to be sent.

```
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 324
```

Figure 9: Common parameters in TLS packets

```
▾ TLSv1.2 Record Layer: Application Data Protocol: http2
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 88
    Encrypted Application Data: 00000000000000015ea2f1420
```

3) **CHANGE_CIPHER_SPEC** – Used to **change** the **encryption** being used by the client and server. The **message** tells the peer that the sender wants to **change** to a **new set of keys,** which are then created from information exchanged by the handshake protocol.

```
▾ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
```

Q2 : OBSERVED PACKET VALUES :

1

FIELD	Src,Dst	Total Length	Flags	Time To Live (TTL)	Protocol	Header Checksum
VALUE	10.16.0.46, 162.125.82.1	168	0x4000	64	TCP	0x7618
EXPLANATION	This packet is from my PC to 162.125.82.1 (www.dropbox-dns .com)	Since the min header length is 20 bytes, the amount of payload is 148 bytes	0(Reserved bit) 1(Don't Fragment bit) 0(More Fragments) 0 0000 0000 0000(Fragment Offset)	This particular packet is allowed to remain in the network for at max 64 hops	The next level protocol is TCP	This represents the checksum value calculated for the header part only

Table 1: IPv4 - Internet Protocol Version 4

2

FIELD	Sequence_#	Acknowledgment #	Flags	Window_Size	Urgent Pointer	Checksum
VALUE	102	95	0x010	513	0	0x1c9d [unverified]
EXPLANATION	The first bytes of this packet is numbered 102	Since the acknowledgement flag bit is set, this number is valid and represents that the receiver expects to receive packet with seq number 95	000(Reserved bit) 0(Nonce) 0(Congestion Window Reduced) 0(ECN-Echo) 0(Urgent) 1(Acknowledgment) 0(Push) 0(Reset) 0(Syn) 0(Fin)	This is the size of the sender's receive window	Since the Urgent Flag bit is not set, this pointer shows the default value 0, otherwise, it would have pointed to the first urgent byte in the packet	This represents the checksum value calculated for the complete packet. This value has not been verified either by wireshark or the dest

Table 2: TCP - Transport Control Protocol

3

FIELD	Source Port	Destination Port	Length	Checksum
VALUE	17500	17500	201	0xfbcb [unverified]
EXPLANATION	This port is reserved for LAN sync discovery which uses UDP	This port is reserved for LAN sync discovery which uses UDP	This is the total length of the packet including the payload	This represents the checksum value calculated for the complete packet. This value has not been verified either by wireshark or the dest

Table 3: UDP - User Datagram Protocol

4

FIELD	Version	Destination Port	Host_Int	DisplayName	Namespaces
VALUE	2	17500	611837924...		12760xxxxx
EXPLANATION	Version of the DB-LSP-DISC protocol used	This port is reserved for LAN sync discovery which uses UDP	The identifier for the packet to be identified by the receiver	By default the display name is an empty string	The array of the namespaces supported. In this case, there is only one namespace([key,value] pair), with the value mentioned above

Table 4: DB-LSP-DISC - DropBox LAN Sync Discovery Protocol

5

FIELD	Transaction_ID	Questions	Answer RRs	Flags	Queries	Answers
VALUE	0x4320	1	3	0x8180	uc8f8...dl.dropbox usercontent.com : type A, class IN	uc8f8...dl.dropboxusercontent.com: type CNAME, class IN, cname dl.dropboxusercontent.com
EXPLANATION	16 bit ID to track queries and synchronize responses.	No of questions in a single query	No of answer records. Hence, there were 3 aliases before getting the IP address	1(Message is a Response) 0000(Opcode : Std query) 0 (Server not authority for domain) 0(Msg is not Truncated) 1(Do query recursively) 1(Recursion possible by server) 0(Reserved) 0(Answer authenticated : Not set) 0(Non authenticated data) 0000(Reply code : No error)	A : host address, CNAME : alias	The above field just shows the first answer record. Each of the 3 records gives the cname (aliases) for the query/resolved_name.

Table 5: DNS - Domain Name System

6

FIELD	Version	Content Type	Handshake Type	Cipher Suite	Compression Method	Extensions Length
VALUE	TLS 1.2	Handshake	Server Hello	TLS_ECDHE_RSA_WITH AES_128_GCM_SHA256 (0xc02f)	null (0)	32
EXPLANATION	This is the version of TLS protocol	TLS packets are of multiple types like application data, handshake, change_cipher_spec etc. This TLS packet is a handshake	Handshakes are of multiple types like server, client hello, key exchanges, certificate etc. This one is a server hello	This is the encryption method agreed upon by both the client and the server. RSA uses asymmetric encryption to create the session key.	This refers to the compression done prior to the encryption. In this case, no compression has been done	Length of the extensions block

Table 6: TLSv1.2 - Transport Layer Security

Q3 : PROTOCOLS FOR IMP DROPBOX FUNCTIONS :

Dropbox heavily relies on the **amazon EC2** and **AWS services** for the various functions it provides. It mainly uses the data centers based in USA to store the data. It has only been a couple of years that Dropbox started working on developing its own cloud services to shift from the Amazon cloud services.

In dropbox, there are three major operations :

a) **Storage** of data on cloud

b) **Retrieval** of data on client's machine

c) Continuous **syncing** of data between server, client.

***Please note that the protocols are in orange color.*

STORAGE / RETRIEVAL :

FUNCTIONS :

This includes **download, upload, open, share** functions.

REASON OF USING TCP OVER UDP -

DropBox mainly uses **TCP** as its transport layer protocol because of its reliability, as it is a connection-based protocol.

Data communication via TCP can be divided into 2 parts

a) HANDSHAKE -

The figure on the right shows the **3-way TCP handshake**, which is initiated by the client by sending the **SYN** packet, followed by different types of packet conversations between the client and the server. Please refer TCP HANDSHAKE for the detailed view of the TCP handshake. TCP handshake is followed by **TLS** handshake. DropBox uses TLS as its cryptographic protocol. Please refer TLS HANDSHAKE for the detailed view of the TLS handshake.

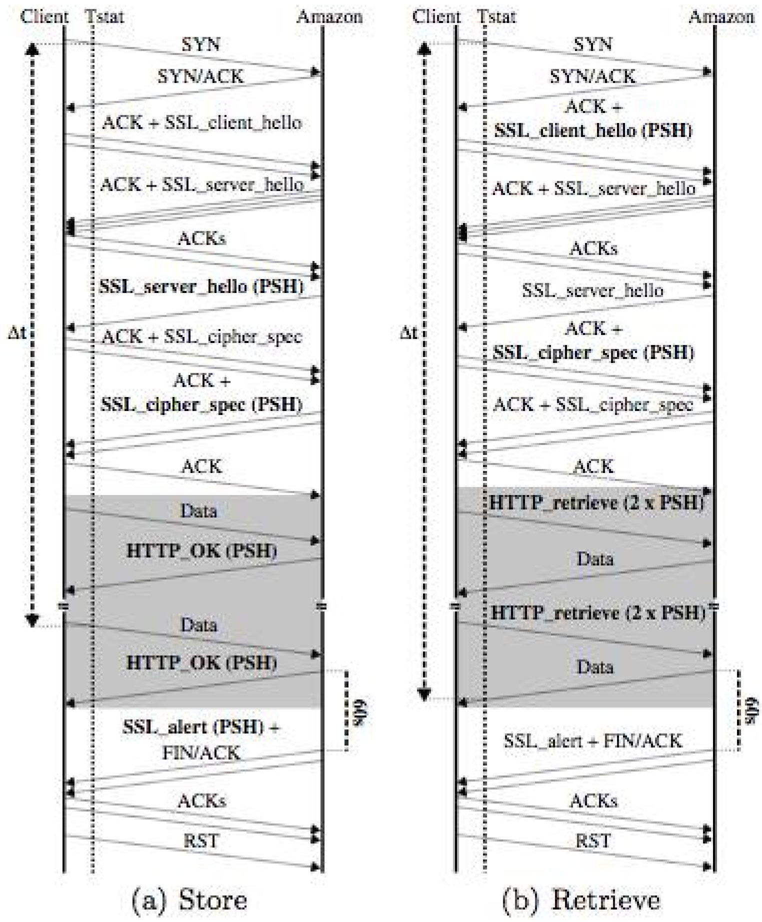


Figure 10: Typical Flows in Storage Operations

b) DATA TRANSFER -

Data is transferred from client to server (for **storage**) and server to client (for **retrieval**). After completion of transfer, a TCP packet is sent with **FIN** flag indicating it had finished the transfer. This is responded with a TCP packet containing ACK and the storage/retrieval process is finished.

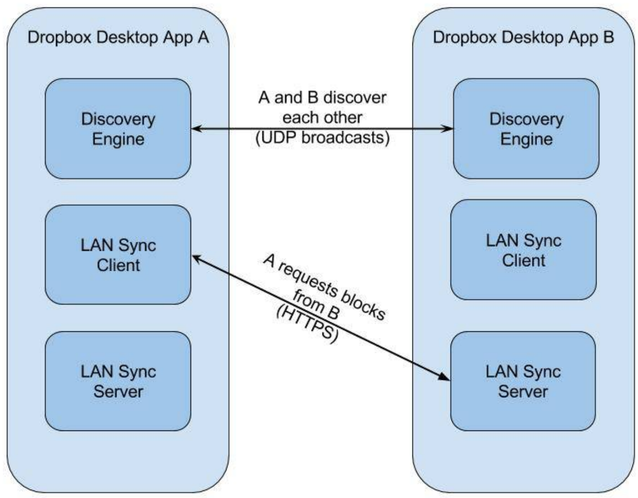
SYNCHRONIZATION :

FUNCTIONS :

This includes the **syncing** of the **downloadable dropbox directory** with the online dropbox files. This also includes the syncing of the **shared folders** between multiple clients.

NEED (REASON) OF SYNCHRONIZATION :

Imagine that you are at home or at your office, and someone on the same network as you adds a file to a shared folder that you are a part of. Without LAN Sync, their computer would upload the file to Dropbox, and then you would download the file from Dropbox. With LAN Sync, you can download the file straight from their computer.



Therefore, Dropbox uses **Dropbox LAN Sync** protocol (**DB_LSP**), which allows you to **download files** from **other computers** on your network, saving time and bandwidth compared to downloading them from Dropbox servers.

LAN Sync only syncs actual file data, and not file metadata, such as filenames. By ensuring that all metadata is received from Dropbox's servers, we can ensure that everyone is always in a consistent state. LAN sync only works with computers that are on the same subnet, or broadcast address.

For LAN sync to work, the **discovery engine** is responsible for **finding machines** on the network that we can sync with. Dropbox achieves this by using **DROPBOX LAN SYNC DISCOVERY** protocol (**DB_LSP_DISC**) To do this, each machine **periodically sends** and **listens** for **UDP broadcast packets** over port **17500**.

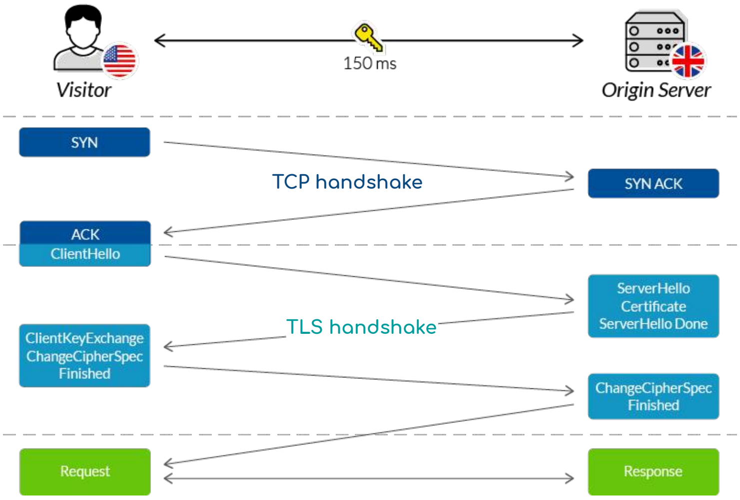
Q4 : DROPBOX FUNCTIONALITIES, MSGS, HANDSHAKES :

a) **STORAGE & RETRIEVAL** : As mentioned in Q3, Dropbox uses **TCP** for retrieval and storage. TCP(Transmission Control Protocol) is responsible for synchronization and initiating connection. It is basically responsible for establishing and maintaining a network conversation via which dropbox can exchange data between client and server. Please refer to TCP and TCP HANDSHAKE for more detail.

B) **SYNCHRONIZATION** : This is the protocol of Dropbox which is mainly responsible for syncing folders across multiple devices connection over a local networks 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.

For the sequence of messages, please refer : https://drive.google.com/open?id=15ZMkkRxl-6Ma0JKR9dCx3T3PaGdZZ2Qi

The following **handshake messages** were observed in case of storage and retrieval functionalities of Dropbox:



TCP HANDSHAKE : TCP, UDP operate at the transport layer. Any **TCP connection** bootstraps with a **3-way handshake**. In other words **TCP** is a **connection-oriented** protocol and the client has to establish a connection with the server prior to the data transmission. Hence, **Dropbox** uses **TCP** primarily for **reliable** packet transfer.

Before the data transmission begins between the client and the server, each party has to exchange the foll :

a) **Starting** packet **sequence** numbers

b) Many other connection specific parameters.

The handshake happens in the following manner :

```
Sequence number: 0    (relative sequence number)
[Next sequence number: 0    (relative sequence number)]
Acknowledgment number: 0
1010 .... = Header Length: 40 bytes (10)
Flags: 0x002 (SYN)
```

1) The client initiates the TCP 3-way handshake, by sending the **SYN** packet (SYN flag bit is set in the TCP packet). The SYN packet includes a randomly picked sequence # by the client,

```
Sequence number: 0    (relative sequence number)
[Next sequence number: 0    (relative sequence number)]
Acknowledgment number: 1    (relative ack number)
1010 .... = Header Length: 40 bytes (10)
Flags: 0x012 (SYN, ACK)
```

2) Once the server receives the initial message from the client, it too picks its own random sequence # & passes it back in **SYN ACK** packet. Add 1 to the client sequence # found in SYN packet to derive acknowledgement #.

```
Sequence number: 1    (relative sequence number)
[Next sequence number: 1    (relative sequence number)]
Acknowledgment number: 1    (relative ack number)
1000 .... = Header Length: 32 bytes (8)
Flags: 0x010 (ACK)
```

3) To complete the handshake, the client will send **ACK** packet to the server to acknowledge the SYN ACK packet it received from the server. It adds 1 to the initial client sequence # to get the new sequence # and one to the server sequence # found in the SYN ACK packet to get the acknowledgement #.

TLS HANDSHAKE - This is the process that **kicks off a communication session** that uses TLS encryption. The TLS handshake happens after the TCP handshake. The TLS handshake includes **three subprotocols** :

HANDSHAKE protocol - Responsible for building an **agreement** between the client and the server on **cryptographic keys** to be used to protect the application data.

CHANGE_CIPHER_SPEC protocol - Both the client and the server precede the *Change Cipher Spec* protocol to indicate to the other party that it's going to **switch** to a **cryptographically secured channel** for further communication.

ALERT protocol - Responsible for **generating alerts** and communicating them to the parties involved in the TLS connection. For ex : revoked certificate alert

The basic seq of steps followed in the handshake are :

a) the two communicating sides exchange messages to acknowledge each other

b) verify each other

c) establish the encryption algorithms they will use

d) agree on session keys

However, the **exact steps** within a TLS handshake will vary depending upon the kind of **key exchange algorithm** used and the **cipher suites** supported by both sides. For example, in Figure 8: TLS handshake flow diagram, **Diffie-Hellman** handshake is used, even though the mot common type of handshake is the RSA handshake.

RSA KEY EXCHANGE -

the **client** generates a **symmetric key**, **encrypts** it with the **server's public key**, and sends it to the server to use as the symmetric key for the established session. In turn, the **server** uses its **private key** to **decrypt** the sent symmetric key and the key-exchange is complete.

The RSA handshake works, but has a critical weakness: the **same public-private key pair** is used both to **authenticate** the server and to **encrypt** the symmetric session key sent to the server. As a result, if an attacker gains access to the server's private key and listens in on the exchange, then they can decrypt the the entire session.

DIFFIE-HELLMAN KEY EXCHANGE -

This allows the client and server to negotiate a **shared secret without** explicitly **communicating** it in the handshake: the server's private key is used to sign and verify the handshake, but the established symmetric key never leaves the client or server and cannot be intercepted by a passive attacker even if they have access to the private key.

TLS HANDHSAKE WITH DIFFIE-HELLMAN KEY EXCHANGE -

a) **Client Hello** - The client sends a client hello message with the protocol version, the client random, and a list of cipher suites.

b) **Server Hello** - The server replies with its SSL certificate, its selected cipher suite, and the server random. As shown in the figure below, **CIPHER SUITE** - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 :

TLS	- the protocol
ECDHE_RSA	- authentication and key exchange algorithms
WITH_AES_128	- the encryption/decryption algorithm
GCM	- the mode used for scrambling the data so it can be securely used with the algorithm
SHA256	- message authentication code algorithm

```
▾ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 508
      Version: TLS 1.2 (0x0303)
    ▸ Random: 00101f3c69cf45a061335d5ed33976641b5d5a09d301cc2f…
      Session ID Length: 32
      Session ID: dbfafc3c0b2bdd8f99314efa40eb5a44b11eb82bee16e73c…
      Cipher Suites Length: 34
    ▸ Cipher Suites (17 suites)
      Compression Methods Length: 1
    ▸ Compression Methods (1 method)
```

Figure 11: Client Hello

```
▾ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 76
  ▾ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 72
      Version: TLS 1.2 (0x0303)
    ▸ Random: 5e2cb1b15e991b09daa86ade8fb8392eb2b7a8b0ad394986…
      Session ID Length: 0
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
      Compression Method: null (0)
      Extensions Length: 32
```

Figure 12: Server Hello

c) **Server's Certificate** -

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication. This message conveys the server's certificate chain to the client.

```
▾ Handshake Protocol: Certificate
    Handshake Type: Certificate (11)
    Length: 2730
    Certificates Length: 2727
  ▾ Certificates (2727 bytes)
      Certificate Length: 1516
    ▸ Certificate: 308205e8308204d0a003020102021003962ca843b69ac7aa…
      Certificate Length: 1205
    ▸ Certificate: 308204b130820399a003020102021004e1e7a4dc5cf2f36d…
```

```
  Handshake Type: Server Key Exchange (12)
  Length: 296
▾ EC Diffie-Hellman Server Params
    Curve Type: named_curve (0x03)
    Named Curve: x25519 (0x001d)
    Pubkey Length: 32
    Pubkey: d6523469f17b65eb40d4687a0a37d7680fc8b5de6c1c1b5b…
  ▸ Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)
    Signature Length: 256
    Signature: 3e36f718ea9069e3fb1958c0d5b63e214e2f1e3b9b967722…
```

d) **Server's Key Exchange Message** -

The ServerKeyExchange message is sent by the server only when the server Certificate message (if sent) does not contain enough data to allow the client to exchange a premaster secret. For ex in case of DHE_RSA, DHE_DSS, etc.

This message conveys **cryptographic information** to allow the client to **communicate** the **premaster secret**: a Diffie-Hellman public key with which the client can complete a key exchange (with the result being the premaster secret) or a public key for some other algorithm.

```
▾ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 4
  ▾ Handshake Protocol: Server Hello Done
      Handshake Type: Server Hello Done (14)
      Length: 0
```

ServerHelloDone indicates the end of the ServerHello and associated messages.

d) **Server's Key Exchange Message** -

With this message, the premaster secret is set, either by direct transmission of the RSA-encrypted secret or by the transmission of Diffie-Hellman parameters that will allow each side to agree upon the same premaster secret.

When the client is using an ephemeral Diffie-Hellman exponent, then this message contains the client's Diffie-Hellman public value.

Change Cipher Spec Message -

This is sent by the client, and the client copies the pending Cipher Spec (the new one) into the current Cipher Spec (the one that was previously used). Change Cipher Spec protocol exists in order to signal transitions in ciphering strategies.

```
▾ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 37
  ▾ Handshake Protocol: Client Key Exchange
      Handshake Type: Client Key Exchange (16)
      Length: 33
    ▾ EC Diffie-Hellman Client Params
        Pubkey Length: 32
        Pubkey: ace9eac27150502e14355b3748e9125b2abbe167ca83b778…
▾ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
▾ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 40
    Handshake Protocol: Encrypted Handshake Message
```

CIPHER SUITE : A cipher suite is a set of **encryption algorithms** for use in establishing a **secure** communications **connection**. (An encryption algorithm is a set of mathematical operations performed on data for making data appear random.) There are a number of cipher suites in wide use, and an essential part of the TLS handshake is agreeing upon which cipher suite will be used for that handshake.

Q5 : TRACE STATISTICS AT DIFF TIMES

The values are taken from **open_file** (retrieval) functionality from DropBox after taking a look at resolved hosts and then applying the corresponding filters. The calculations were made as follows :

# of UDP packets, TCP packets : **Packets** under the UDP, TCP tab respectively from **Statistics** -> **Conversations.**

Avg Packet Size : **Bytes** / **Packets** from Statistics -> Conversations

# of Packets Lost : Added "**&& tcp.analysis.lost_segment**" to the filter

Responses per Request : **Packets B->A** / **Packets A->B** from Statistics -> Conversations. Here A is me and B is DropBox host

Throughput : **Bits/s** from **Statistics** -> **Conversations**

RTT : This can be found from the **RTT** of **ACK** corresponding to **SYN** of the three-way **TCP handshake**. This will give the intial RTT. RTT for the complete conversation can be seen from **Statistics** -> **TCP Stream Graphs** -> **Round Trip Time**.

Time	Throughput	RTT	Avg Packet Size	# of Packets Lost	# of TCP packets	# of UDP packets	Responses per Request
11 : 48 PM	421k bits/s	559.01 ms	753 bytes	1	332	9	201/132 = 1.52
04 : 40 AM	433k bits/s	409.38 ms	921.25 bytes	0	342	4	189/138 = 1.36
01 : 19 PM	289k bits/s	687.34 ms	633.53 bytes	4	333	6	217/130 = 1.67

Q6 : RESOLVED HOSTS :

11:48 PM

```
# Hosts
#
# 7 entries.

162.125.80.5     edge-block-previews-video-live.dropbox-dns.com
162.125.82.1     www.dropbox-dns.com
162.125.82.6     edge-block-www.dropbox-dns.com
104.16.100.29    cfl.dropboxstatic.com.cdn.cloudflare.net
143.204.229.110  assets.dropbox.com
143.204.229.19   assets.dropbox.com
104.16.99.29     cfl.dropboxstatic.com.cdn.cloudflare.net
```

04:40 AM

```
# Hosts
#
# 2 entries.

162.125.82.6     edge-block-www.dropbox-dns.com
162.125.82.5     edge-block-previews-video-live.dropbox-dns.com
```

01:19 PM

```
# Hosts
#
# 11 entries.

162.125.36.1     d-sjc.v.dropbox.com
162.125.82.7     api.dropbox-dns.com
162.125.19.131   bolt.v.dropbox.com
162.125.82.5     edge-block-previews-video-live.dropbox-dns.com
104.16.99.29     cfl.dropboxstatic.com.cdn.cloudflare.net
104.16.100.29    cfl.dropboxstatic.com.cdn.cloudflare.net
162.125.82.4     edge-block-client-3.dropbox-dns.com
162.125.82.3     client.dropbox-dns.com
34.194.25.117    block-debug.x.dropbox.com
162.125.82.1     www.dropbox-dns.com
13.33.169.116    help.dropbox.com
```

The hosts above have been obtained from **Statistics -> Resolved Addresses** from "**open file**" (retrieval) function of Dropbox. It is observed that Dropbox uses several domains to run the service. This can have several possible reasons :

sub-domain	Data-center	Description
client-lb/clientX	Dropbox	Meta-data
notifyX	Dropbox	Notifications
api	Dropbox	API control
www	Dropbox	Web servers
d	Dropbox	Event logs
dl	Amazon	Direct links
dl-clientX	Amazon	Client storage
dl-debugX	Amazon	Back-traces
dl-web	Amazon	Web storage
api-content	Amazon	API Storage

Figure 13: Domain Names used by different DropBox services

a) Each **domain** serves a **specific purpose**, like running the Dropbox product, providing communications, or hosting marketing materials. The figure on the right shows a few examples of the domain names used by DropBox. The official Dropbox domains can be checked from : **https://help.dropbox.com/accounts-billing/security/official-domains**

b) To compensate for a host that's down at that moment by adding its IP address to another one.

c) In order to reduce the load on one network adapter (interface) or load balance different request types.