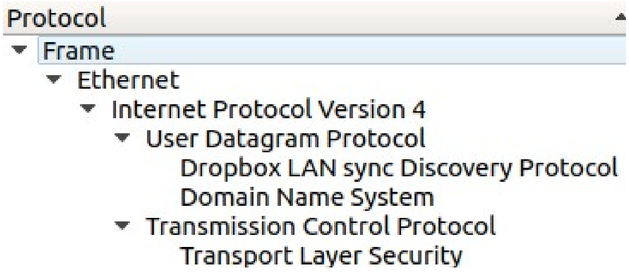
CS349 ASSIGNMENT 2

Network Protocol Analysis Using Wireshark

Rashi Singh
170101052

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
```

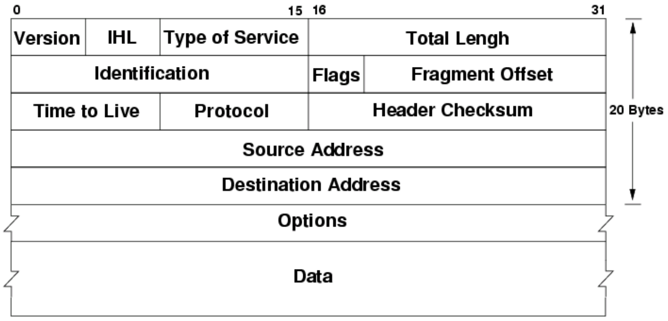
Figure 1: Protocol Hierarchy used by DropBox

Network Layer

IPv4 : Internet Protocol Version 4

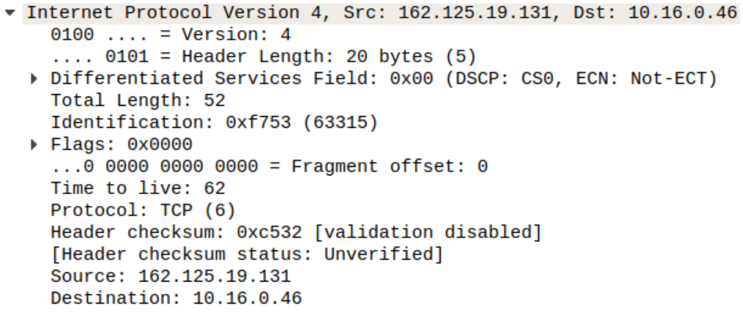
IPv4 is a **packet-switched,** network-layer protocol. It provides a **logical connection** between network devices by providing **identification** for each device and **routing** data among them over the underlying network. IP uses best effort delivery, i.e. it does not guarantee that packets would 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



0		15	16	31
Version	IHL	Type of Service	Total Lengh	
Identification			Flags	Fragment Offset
Time to Live	Protocol		Header Checksum	
Source Address				
Destination Address				
Options				
Data				

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.

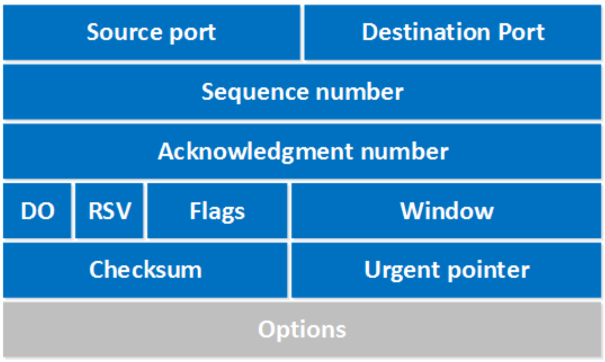
SOURCE/DEST ADDRESS - (32 bit) Addr of the source/destination respectively

Transport Layer

TCP : Transport Control Protocol

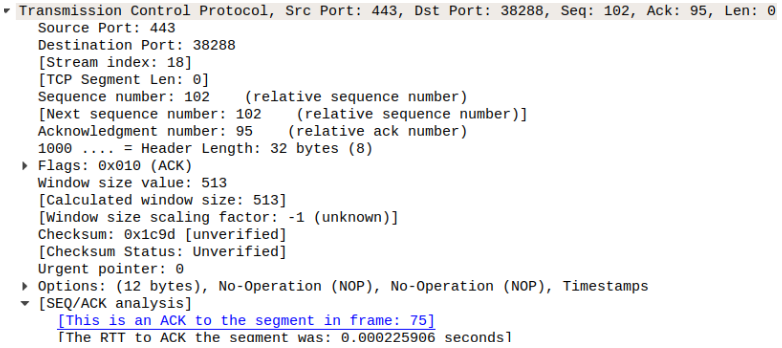
TCP is a **connection oriented** protocol which offers **end-to-end packet delivery**. TCP ensures reliability by sequencing bytes with a forwarding **acknowledgement** number that indicates to the destination, the next byte the source expect to receive.It retransmits the bytes not acknowledged with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

TCP : Packet Structure



Source port		Destination Port	
Sequence number			
Acknowledgment number			
DO	RSV	Flags	Window
Checksum		Urgent pointer	
Options			

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]
```

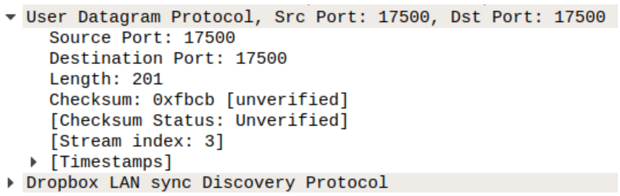
Figure 5: WireShark TCP packet details

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

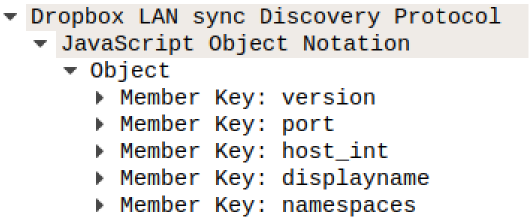
Src/Dest Port, **Length**, **Checksum** - UDP Packet Structure

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**.



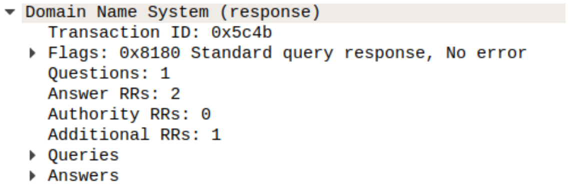
```
▾ Dropbox LAN sync Discovery Protocol
  ▾ JavaScript Object Notation
    ▾ Object
      ▸ Member Key: version
      ▸ Member Key: port
      ▸ Member Key: host_int
      ▸ Member Key: displayname
      ▸ Member Key: namespaces
```

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
```

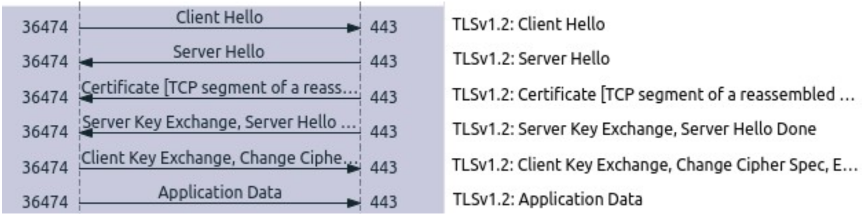
The Domain Name System is a **hierarchical** and **decentralized naming system** for computers, services, or other resources connected to the Internet or a private network.

Please refer to ..... for the packet details

B/w Application, Transport Layer

TLSv1.2 : Transport Layer Security

Figure 8: TLS packet exchange flow



```
36474  --- Client Hello ----------------------------->  443   TLSv1.2: Client Hello
36474  <-- Server Hello ------------------------------  443   TLSv1.2: Server Hello
36474  <-- Certificate [TCP segment of a reass... ----  443   TLSv1.2: Certificate [TCP segment of a reassembled ...
36474  <-- Server Key Exchange, Server Hello ... -----  443   TLSv1.2: Server Key Exchange, Server Hello Done
36474  --- Client Key Exchange, Change Ciphe... ----->  443   TLSv1.2: Client Key Exchange, Change Cipher Spec, E...
36474  --- Application Data ------------------------->  443   TLSv1.2: Application Data
```

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

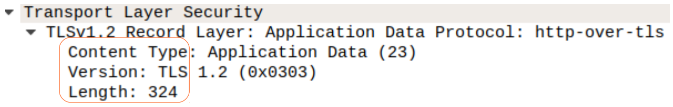
b) **LENGTH** : 16 byte record length

c) **CONTENT TYPE** : The type of TLS packet.

Some of the common types are :

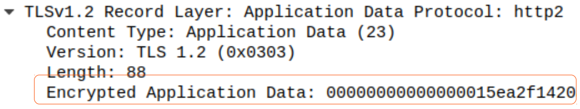
1) **HANDSHAKE** – Please refer Q4 HANDSHAKE for more info.

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.



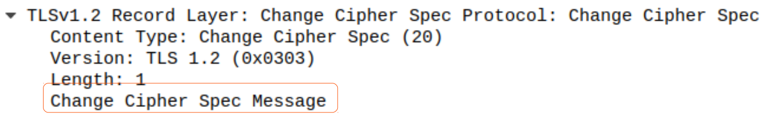
```
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 324
```

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 :

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

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

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

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

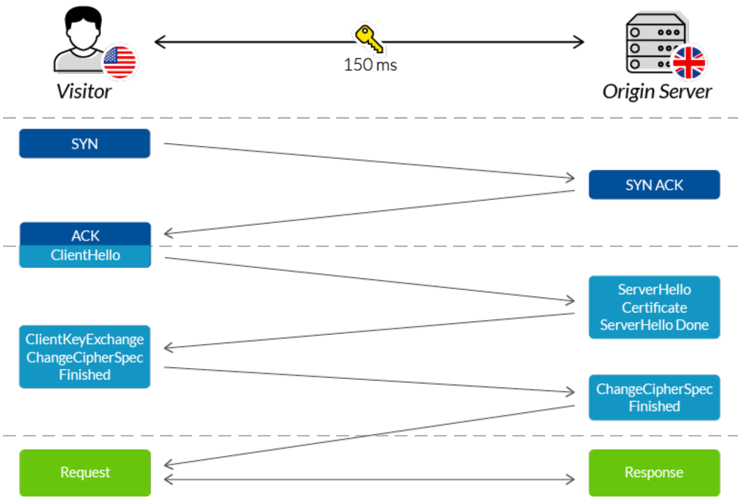
Table 4: TLSv1.2 - Transport Layer Security

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

Q3 : PROTOCOLS FOR IMP DROPBOX FUNCTIONS :

Q4 : DROPBOX FUNCTIONALITIES, MSGS, HANDSHAKES :

HANDSHAKE - The process that **kicks off a communication session** that uses TLS encryption. As shown in Figure 8: TLS packet exchange flow, the handshake takes place in the following seq :



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

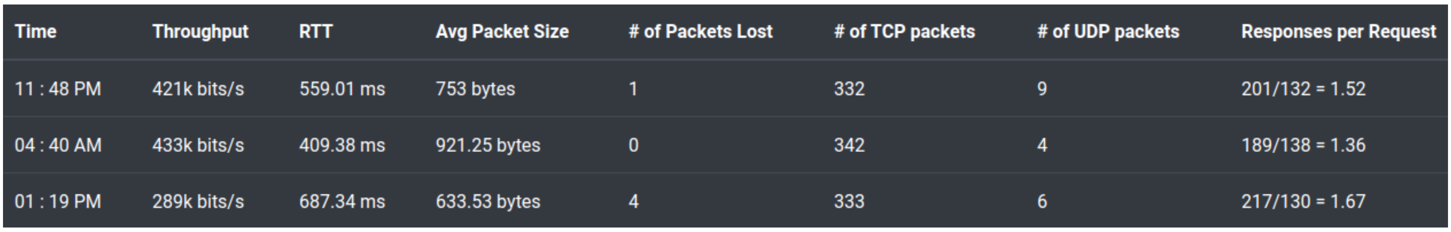
b) verify each other

c) establish the encryption algorithms they will use

d) agree on session keys

In WireShark, Client and Server Hello

Q5 : TRACE STATISTICS AT DIFF TIMES :

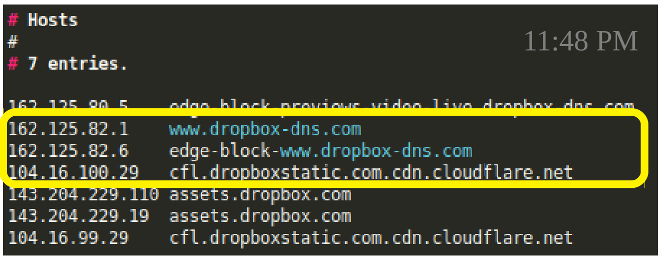


Time	Throughput	RTT	Avg Packet Size	# of Packets Lost	# of TCP packets	# of UDP packets	Responses per Request
11 : 48 PM	421k bits/s	559.01 ms	753 bytes	1	332	9	201/132 = 1.52
04 : 40 AM	433k bits/s	409.38 ms	921.25 bytes	0	342	4	189/138 = 1.36
01 : 19 PM	289k bits/s	687.34 ms	633.53 bytes	4	333	6	217/130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

//TODO: GRAPH LAGA DIYO MANN KRE TO

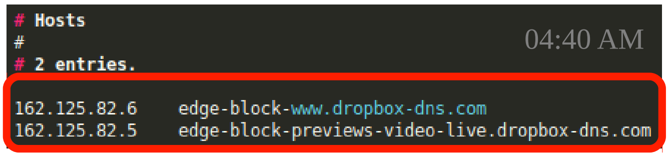
//TODO: METHOD TO CALCULATE BHI LIKH DIYO

Q6 : RESOLVED HOSTS :



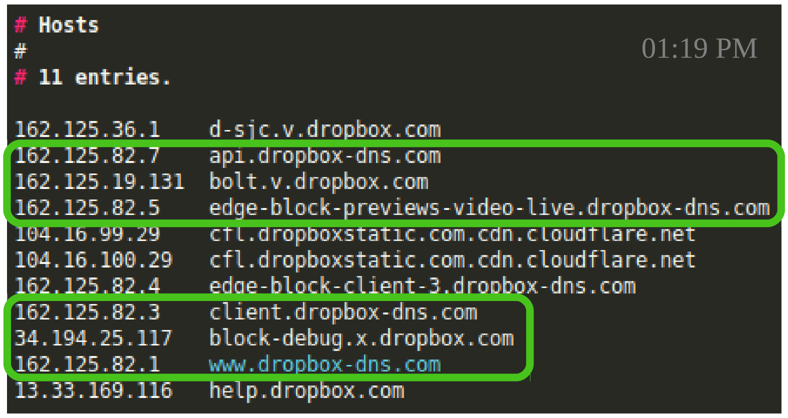
11:48 PM

```
# 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
```