CS349 ASSIGNMENT 02

ANIKET RAJPUT (170101007)

Drive Link for Traces:

https://iitgoffice-my.sharepoint.com/:f:/g/personal/anike170101007_iitg_ac_in/Enwx9FmwV69DrvLmWKI5jXQBAiIlvjxcq1Z685iRa2_XVg?e=kUxexu

01. PROTOCOLS USED AT DIFFERENT LAYERS BY THE APPLICATION

DATA LINK LAYER

- **ARP –** Used for discovering the MAC address associated with an IP(IPv4) address. '**Hardware and Protocol type**' specifies the network link protocol type and the internetwork protocol for which ARP request is intended. '**Opcode**' gives the operation performed: 1 for request and 2 for reply.

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Cisco_74:60:43 (ec:44:76:74:60:43)
    Sender IP address: 10.19.4.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.19.6.223
```

- **SSDP –** Used for discovering and advertising services and devices on a TCP/IP network. It is the basis of Universal Plug and Play protocol. It uses the HTTP method 'NOTIFY' to announce establishment or withdrawal of services and 'M-SEARCH' method to discover available services.

```
Simple Service Discovery Protocol
NOTIFY * HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): NOTIFY * HTTP/1.1\r\n]
  Request Method: NOTIFY
  Request URI: *
  Request Version: HTTP/1.1
Host: 239.255.255.250:1900\r\n
NT: uuid:cf81577f-f32c-4b50-878b-8f3789b69d54\r\n
NTS: ssdp:alive\r\n
Location: http://10.19.6.31:2869/upnphost/udhisapi.dll?content=uuid:cf81577f-f32c-4b50-878b-8f3789b69d54\r\n
USN: uuid:cf81577f-f32c-4b50-878b-8f3789b69d54\r\n
Cache-Control: max-age=1800\r\n
Server: Microsoft-Windows/10.0 UPnP/1.0 UPnP-Device-Host/1.0\r\n
OPT:"http://schemas.upnp.org/upnp/1/0/"; ns=01\r\n
01-NLS: 28e2c9ce1707c58919e2026cad60219f\r\n
\r\n
[Full request URI: http://239.255.255.250:1900*]
```

NETWORK LAYER

- **ICMPv6 –** Used to send error messages and operational information about success or failure during communication. Version 6 is used for IPv6 addresses. It consists of a '**header**' and a '**payload**'. Header has 3 fields. '**Type**' indicates error or information message. '**Code**' indicates the level of detail of message. '**Checksum**' provides a level of integrity verification. '**Payload**' is the data or message.

```
Internet Control Message Protocol v6
   Type: Multicast Listener Report Message v2 (143)
   Code: 0
   Checksum: 0xa200 [correct]
   [Checksum Status: Good]
   Reserved: 0000
   Number of Multicast Address Records: 1
 > Multicast Address Record Changed to exclude: ff02::1:ff92:ccf7
```

- **IGMP –** Used on IPv4 networks to establish multicast group memberships by host and adjacent routers. '**Type**' depends on the version on IGMP used (2 and 3 in the experiment). '**Max Resp Time**' specifies the required responsiveness of replies.

```
Internet Group Management Protocol
   [IGMP Version: 2]
   Type: Membership Report (0x16)
   Max Resp Time: 0.0 sec (0x00)
   Checksum: 0xfa04 [correct]
   [Checksum Status: Good]
   Multicast Address: 239.255.255.250
```

TRANSPORT LAYER

- **TCP –** Used to establish and maintain a network conversation through which application programs can exchange data. It defines how systems send packets to each other. '**Sequence number**' gives the sequence of data byte depending on acknowledged ACK. '**Acknowledgement number**' is next sequence number of ACK expected. '**Data Offset**' specifies the size of TCP header. Contains 9 1-bit '**flag**' bits. '**Window size**' gives number of window sized units to be received. '**Checksum**' is used for error checking.

- **TLS –** It runs on top of TCP and is designed to provide communication security over the network. TLS is used by websites to secure all communications between their servers and web browsers. TLS provides privacy and data integrity.

```
Transmission Control Protocol, Src Port: 443, Dst Port: 49877, Seq: 3703, Ack: 126081, Len: 0
   Source Port: 443
   Destination Port: 49877
   [Stream index: 6]
   [TCP Segment Len: 0]
   Sequence number: 3703    (relative sequence number)
   Sequence number (raw): 581108095
   [Next sequence number: 3703    (relative sequence number)]
   Acknowledgment number: 126081    (relative ack number)
   Acknowledgment number (raw): 1332370898
   0101 .... = Header Length: 20 bytes (5)
 > Flags: 0x010 (ACK)
   Window size value: 967
   [Calculated window size: 123776]
   [Window size scaling factor: 128]
   Checksum: 0x7cfc [unverified]
   [Checksum Status: Unverified]
   Urgent pointer: 0
 v [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 319]
      [The RTT to ACK the segment was: 0.005902000 seconds]
      [iRTT: 0.000765000 seconds]
 v [Timestamps]
      [Time since first frame in this TCP stream: 1.283356000 seconds]
      [Time since previous frame in this TCP stream: 0.000001000 seconds]
```

- **UDP –** It is an alternative communications protocol to TCP, used primarily for establishing low-latency and loss-tolerating connections between applications on the internet. Packet consist of '**header**' and '**data**'. '**Length**' specifies length in bytes of header and data. '**Checksum**' is used for error-correction.

```
User Datagram Protocol, Src Port: 54915, Dst Port: 54915
   Source Port: 54915
   Destination Port: 54915
   Length: 271
   Checksum: 0x21fc [unverified]
   [Checksum Status: Unverified]
   [Stream index: 9]
 > [Timestamps]
Data (263 bytes)
```

APPLICATION LAYER

- **DHCP –** It provides quick, automatic and central management for the distribution of IP addresses within a network and also used to configure the subnet mask, default gateway and DNS server information on device.
- **DNS –** It translates domain names to IP addresses so browsers can load internet resources. A DNS message consists of a '**header**' and 4 sections. '**Header**' contains fields of '**identification**', '**flags**', '**no. of questions'**, '**no. of authority resource records (RR)'** and '**no. of additional RR's**'. DNS specifies information elements for network resources which are recorded in resource records. Each record has a name, type and class specified. '**TTL**' gives the count of seconds for which RR stays valid.

```
Domain Name System (response)
   Transaction ID: 0x5ad5
 > Flags: 0x8180 Standard query response, No error
   Questions: 1
   Answer RRs: 2
   Authority RRs: 4
   Additional RRs: 1
 v Queries
   > api.dailymotion.com: type A, class IN
 v Answers
   > api.dailymotion.com: type CNAME, class IN, cname dmapi.api-aws.dailymotion.com
   > dmapi.api-aws.dailymotion.com: type A, class IN, addr 195.8.215.129
 v Authoritative nameservers
   > api-aws.dailymotion.com: type NS, class IN, ns ns-1385.awsdns-45.org
   > api-aws.dailymotion.com: type NS, class IN, ns ns-771.awsdns-32.net
   > api-aws.dailymotion.com: type NS, class IN, ns ns-137.awsdns-17.com
   > api-aws.dailymotion.com: type NS, class IN, ns ns-1706.awsdns-21.co.uk
 v Additional records
   > ns-137.awsdns-17.com: type A, class IN, addr 205.251.192.137
   [Request In: 45]
   [Time: 0.135053000 seconds]
```

02. OBSERVED VALUES (FROM UPLOAD TRACE OF NOON)

- **ARP**

The '**Destination**' field in this case '**Broadcast**' as this protocol determines the MAC address of a given IP by request all systems in the local network. The '**Source**' field is the address of sender of this request. My system uses '**IPv4**' '**Protocol Type**'. The '**Opcode**' distinguishes the request and reply packets. The '**Target MAC address**' in this case is 00:00:00:00:00:00 as it is to be determined.

- **SSDP**

It employs two HTTP methods M-SEARCH and NOTIFY. During M-SEARCH most common '**Destination**' address is '239.255.255.250' which is used for multicast DNS. The '**Source**' field is the address of sender of

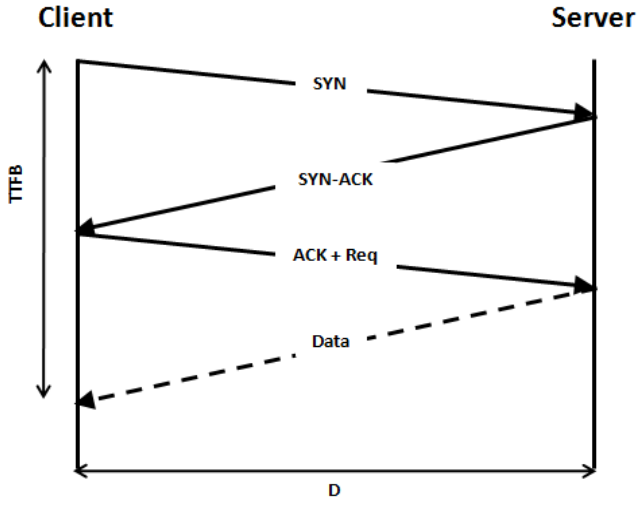
this request. It uses UDP '**Protocol**' and the most common '**Destination Port**' is 1900 which is used for Universal Plug n Play devices. '**Time To Live**' field is associated with these packets to ensure duration of their validity.

- **ICMPv6**
  The most common '**Destination**' address associated with packets of this protocol is found as 'ff02::16' which is for Multicast Listener Discovery (MLD) capable router and is used by IPv6 routers. These packets have and associated '**Hop Limit**' with them. They are used in IPv6 '**Protocol**'. These packets also have '**Padding bits**' associated with them and '**Router Alert**' field for routers to examine the contents of an IP packet more closely.
- **IGMP**
  These packets have same function as ICMPv6 but they use **IPv4 Protocol**.
- **TCP**
  For the given trace most common '**destination and source ports'** were 443 and 49877. These packets had '**acknowledgement number**' associated with them to keep check on lost frames. '**Timestamps**' are used to keep record of time since first frame and previous frame in the stream. These packets also have '**payload**' associated with them. These packets also have flags to distinguish between duplicate acknowledgement packets.
- **TLS**
  These packets have same properties as TCP packets along with a security layer. There is field depicting the '**Encrypted Application Data**'.
- **UDP**
  10.19.6.200 and 10.19.7.255 are most common source and destination addresses in UDP packets. The '**Protocol Type**' of these packets is IPv4 and these packets have '**Data**' associated with them.

03. PROTOCOLS

When we are **uploading or downloading** something we are sending or receiving data in respective cases. For uploading and downloading of videos the main protocols used at **transport layer** are **TCP**, **TLS** and **UDP**.

TCP (Transmission Control Protocol) enables the smooth flow of data and its automatic recovery. It detects errors and acknowledge sent and received data. It provides sequencing of data packets, flow control and error checking to ensure message delivery. TCP is used is by internet applications such as WWW, FTP (File Transfer Protocol), P2P etc. TCP also helps in network congestion avoidance. It works on the principle of **3-way handshake**. For one application to send data to another, the two processes must first handshake i.e. they must send some segments to each other to establish the parameters of data transfer. The client application process first informs the client transport layer to establish a connection. The client first sends a TCP segment and then server replies with a TCP segment. Then client sends a third TCP segment. This connection-establishment procedure is known as 3-way handshake. The client passes a stream of data. TCP directs it to a send buffer and this chunk of data will pass through the network layer.

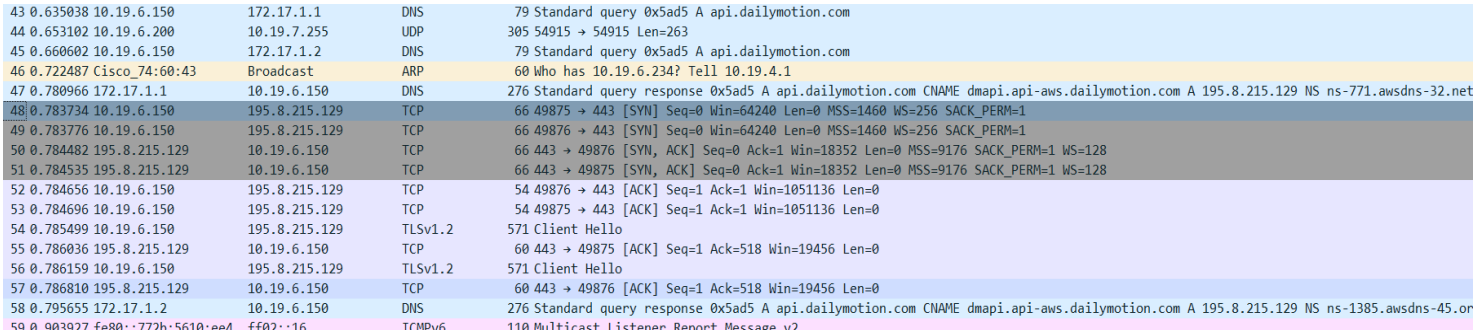


TLS (Transport Layer Security) protocol aims to provide privacy and data integrity between two or more communication computer applications. It creates a private, secure connection between a client and server. It encrypts the data transmitted. It also uses the handshake protocol to establish connection between hosts.

UDP (User Datagram Protocol) uses a simple model of communication. Due to lack of handshaking dialogues, there exist an unreliability in the network. There are no protective measures from disorders, nondelivered and duplicate protection. There is no extensive error checking mechanism. But it provides a faster connection than TCP.

At **Internet layer**, **IP** (Internet Protocol) is a useful protocol. It is a protocol for routing and addressing packets of data so that they can travel across networks and arrive at the correct destination. Every device of domain that connects to the Internet is assigned an IP address, and as packets are directed to the IP address attached to them, data arrives where it is needed.

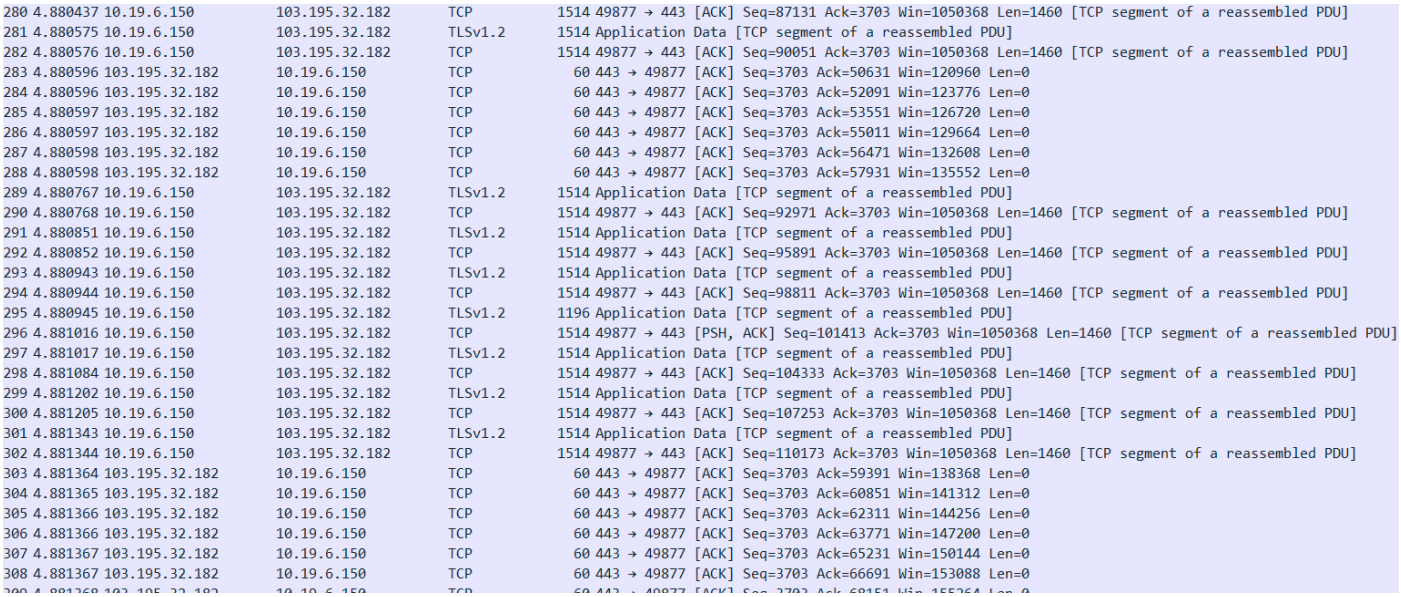
04. (a) HANDSHAKING SEQUENCE



43	0.635038	10.19.6.150	172.17.1.1	DNS	79 Standard query 0x5ad5 A api.dailymotion.com
44	0.653102	10.19.6.200	10.19.7.255	UDP	305 54915 → 54915 Len=263
45	0.660602	10.19.6.150	172.17.1.2	DNS	79 Standard query 0x5ad5 A api.dailymotion.com
46	0.722487	Cisco_74:60:43	Broadcast	ARP	60 Who has 10.19.6.234? Tell 10.19.4.1
47	0.780966	172.17.1.1	10.19.6.150	DNS	276 Standard query response 0x5ad5 A api.dailymotion.com CNAME dmapi.api-aws.dailymotion.com A 195.8.215.129 NS ns-771.awsdns-32.net
48	0.783734	10.19.6.150	195.8.215.129	TCP	66 49875 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
49	0.783776	10.19.6.150	195.8.215.129	TCP	66 49876 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
50	0.784482	195.8.215.129	10.19.6.150	TCP	66 443 → 49876 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=128
51	0.784535	195.8.215.129	10.19.6.150	TCP	66 443 → 49875 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=128
52	0.784656	10.19.6.150	195.8.215.129	TCP	54 49876 → 443 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
53	0.784696	10.19.6.150	195.8.215.129	TCP	54 49875 → 443 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
54	0.785499	10.19.6.150	195.8.215.129	TLSv1.2	571 Client Hello
55	0.786036	195.8.215.129	10.19.6.150	TCP	60 443 → 49875 [ACK] Seq=1 Ack=518 Win=19456 Len=0
56	0.786159	10.19.6.150	195.8.215.129	TLSv1.2	571 Client Hello
57	0.786810	195.8.215.129	10.19.6.150	TCP	60 443 → 49876 [ACK] Seq=1 Ack=518 Win=19456 Len=0
58	0.795655	172.17.1.2	10.19.6.150	DNS	276 Standard query response 0x5ad5 A api.dailymotion.com CNAME dmapi.api-aws.dailymotion.com A 195.8.215.129 NS ns-1385.awsdns-45.or
59	0.903927	fe80::772b:5610:ee4	ff02::16	ICMPv6	110 Multicast Listener Report Message v2

The packets from 48 onwards are involved in the '**three-way handshaking**'. These packets are exchanged to establish the TCP connection. The '**info**' of these packets suggests the parameters defined for the communication between the hosts such as MSS (Maximum Segment Size) and WS (Window Scale). SYN is sent by my system to host and SYN/ACK is received from the host by my system indicating the formation of TCP connection. On receiving the SYN/ACK from host, final ACK is sent to the host by my system.

(b) PACKETS ACKNOWLEDGEMENT



280	4.880437	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=87131 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
281	4.880575	10.19.6.150	103.195.32.182	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
282	4.880576	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=90051 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
283	4.880596	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=50631 Win=120960 Len=0
284	4.880596	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=52091 Win=123776 Len=0
285	4.880597	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=53551 Win=126720 Len=0
286	4.880597	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=55011 Win=129664 Len=0
287	4.880598	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=56471 Win=132608 Len=0
288	4.880598	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=57931 Win=135552 Len=0
289	4.880767	10.19.6.150	103.195.32.182	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
290	4.880768	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=92971 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
291	4.880851	10.19.6.150	103.195.32.182	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
292	4.880852	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=95891 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
293	4.880943	10.19.6.150	103.195.32.182	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
294	4.880944	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=98811 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
295	4.880945	10.19.6.150	103.195.32.182	TLSv1.2	1196 Application Data [TCP segment of a reassembled PDU]
296	4.881016	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [PSH, ACK] Seq=101413 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
297	4.881017	10.19.6.150	103.195.32.182	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
298	4.881084	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=104333 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
299	4.881202	10.19.6.150	103.195.32.182	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
300	4.881205	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=107253 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
301	4.881343	10.19.6.150	103.195.32.182	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
302	4.881344	10.19.6.150	103.195.32.182	TCP	1514 49877 → 443 [ACK] Seq=110173 Ack=3703 Win=1050368 Len=1460 [TCP segment of a reassembled PDU]
303	4.881364	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=59391 Win=138368 Len=0
304	4.881365	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=60851 Win=141312 Len=0
305	4.881366	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=62311 Win=144256 Len=0
306	4.881366	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=63771 Win=147200 Len=0
307	4.881367	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=65231 Win=150144 Len=0
308	4.881367	103.195.32.182	10.19.6.150	TCP	60 443 → 49877 [ACK] Seq=3703 Ack=66691 Win=153088 Len=0

The above screenshot portrays that during the upload a stream of packets is sent by my system and according to the TCP protocol it waits for the acknowledgement for the sent packets to be received from the host. If an acknowledgement for a particular packet is not received then a that packet will be sent again and the system will again wait for acknowledgement. The maximum number of continuous packets to be sent depends on the number of acknowledgements to be received and the maximum window size decided during establishment of TCP connection.

05. STATISTICS

	12:30		17:30		23:30	
	Upload	**Download**	**Upload**	**Download**	**Upload**	**Download**
Throughput (Packets/sec)	323.7	837.5	303.2	809	787.3	609.1
RTT (ms)	0.649	0.665	0.761	0.706	0.407	0.528
Avg. Packet size (Bytes)	1043	967	985	974	1002	945
No. of Packets Lost	0	0	0	0	0	0
No. of UDP packets	1931	2767	4189	2404	1427	5701
No. of TCP packets	78105	230407	81110	229526	82906	231901
No. of responses w.r.t 1 request	0.339	1.999	0.359	1.999	0.378	1.998

06. MULTIPLE SOURCES

In the given experiments during **uploading**, data was sent to **103.195.32.182** in the **noon** and **evening** while at **night** data was sent to **103.195.32.183**.

During **downloading**, data was received from **188.65.126.33** in the **noon**, **188.65.126.38** in the **evening** and at **night** data was received from **103.195.32.6**.

The reasons behind changing of IP address for a host can be –

- If ISP changes the center or router responsible for the address.
- Failure in ISP's network causing routing tables to rebuild.
- Load balancing across a server to distribute network traffic.
- DHCP lease time for the given IP address has expired.