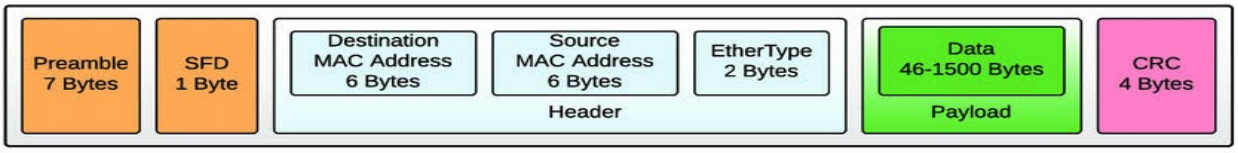
CS349 Networks Lab	Name	Roll Number
Assignment 2	Udbhav Chugh	170101081

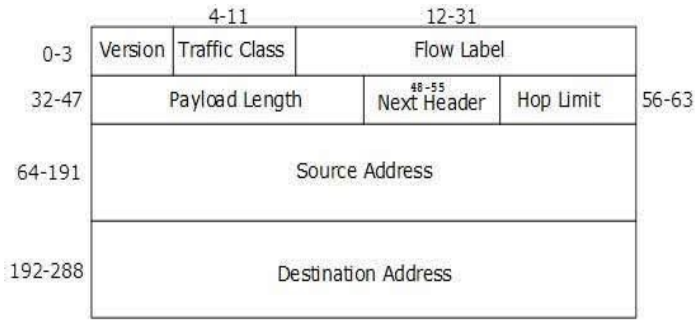
Note: All data is collected by playing Red Ball on https://red-ball4.com/red-ball-4
Drive: https://drive.google.com/drive/folders/1wiBqE5hApqlDvyJK3m_53bF4xgYQqt-4?usp=sharing
Note: The game was played on my Mobile data network and all other tabs were closed on my browser to reduce the number of irrelevant packets. First I looked at the DNS request and response packets corresponding to red-ball4 to figure out the IP address of the website server. Then I filtered the packets with this IP address either as source or destination to analyse the necessary packets.

1) Protocol Used at different layers:

- Data Link Layer:
  Ethernet: The Ethernet frame begins with a Preamble and SFD (start frame delimiter), both of which work at the physical layer. The preamble consists of a 56-bit (seven-byte) pattern of alternating 1 and 0 bits, allowing devices on the network to synchronize their receiver clocks, providing bit-level synchronization, SFD is the eight-bit (one-byte) value that marks the end of the preamble. The Destination and Source MAC addresses are MAC addresses of the receiving and the sending machine respectively, the EtherType field is the only one which differs between 802.3 and Ethernet II (Type of Ethernet). Data is the data to be sent and CRC(Cyclic Redundancy Check) is a CRC-32 polynomial code for error detection.

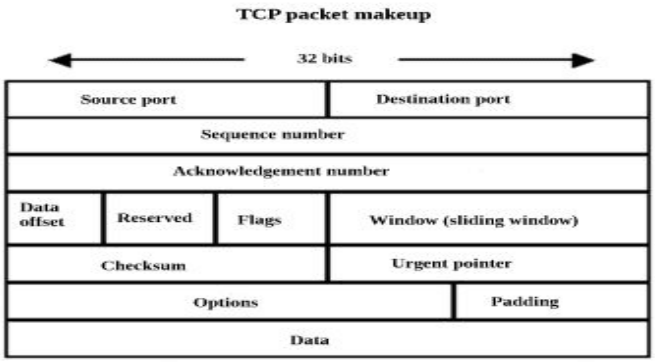


- Network Layer:
  Internet Protocol Version 6: IPv6 is a widely used protocol over different kinds of networks. IP version 6 is the new version of Internet Protocol, which is better than IP version 4 in terms of complexity and efficiency. IPv6 is a 128-Bit IP address. It is a connectionless protocol used in packet-switched layer networks. Version is 4-bit version number of Internet Protocol (= 6). Traffic class is the 8-bit traffic class field. Flow label is the 20-bit field. Payload length is the 16-bit unsigned integer, which is the rest of the packet that follows the IPv6 header, in octets. Next header is the 8-bit selector. It identifies the type of header that immediately follows the IPv6 header. Hop limit is the 8-bit unsigned integer which is decremented by one by each node that forwards the packet. The packet is discarded if the hop limit is decremented to zero. Source address is 128 bits which is the address of the initial sender of the packet. Destination address is 128 bits which is the address of the intended recipient of the packet.

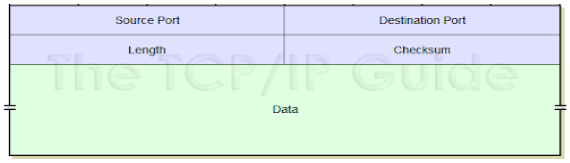


- Transport layer
  Transmission Control Protocol: TCP is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. The fields in

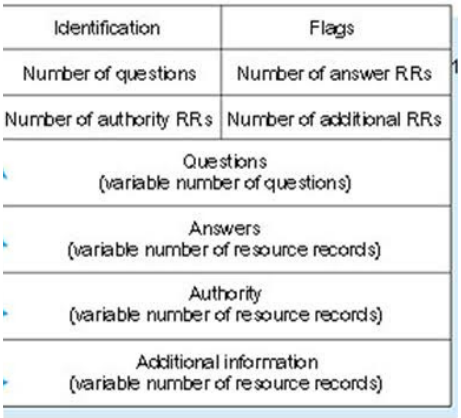
TCP packet header are: **Source Port** and **Destination Port** fields (16 bits each) identify the end points of the connection. **Sequence number** (32 bits) is number assigned to the first byte of data in the current message. **Acknowledgement** number (32 bits) is the sequence number of the next packet expected by the sender. **Data offset** stores the total size of a TCP header in multiples of four bytes, a set of six standard and three extended control **flags** manage data flow in specific situations. **Reserved** in TCP headers always has a value of zero and serves the purpose of aligning the total header size as a multiple of four bytes. **Window** size regulates how much data can be sent to a receiver before requiring an acknowledgment in return. **Checksum** is used for error detection and correction. **Urgent Pointer** field is often set to zero and ignored. **Options** field specifies the various TCP options and **Data field** contains upper layer information.



User Datagram Protocol: UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. **Source Port** is a 2 Byte long field used to identify port number of source. **Destination Port** is a 2 Byte long field, used to identify the port of destined packet. **Length** is a 16-bit field that specifies the length of UDP including header and the data. **Checksum** is used to verify that the end to end data has not been corrupted.

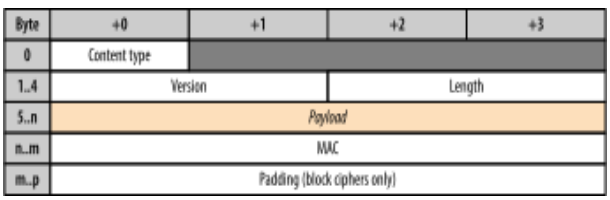


- Application Layer:

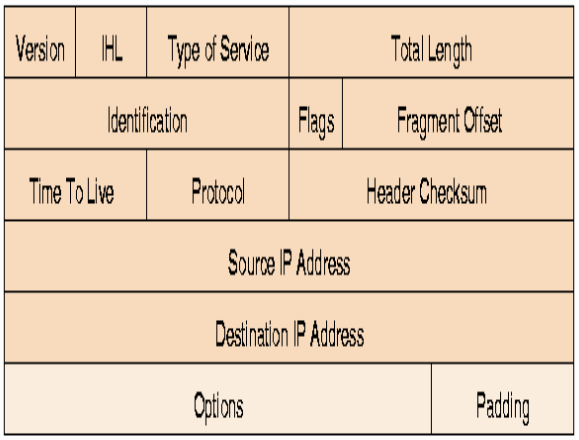


Domain Name System: DNS is a distributed database implemented in a hierarchy of DNS servers, and an application-layer protocol that allows hosts to query the distributed database. The first 12 bytes is the header section. The **identification** field is a 16-bit number that identifies the query. **Flags** in the flag field include query/reply flag, and authoritative flag. The next four fields indicate the number of occurrences of the four types of data sections that follow. The **question** section contains information about the query that is being made. The **answer** section contains the resource records for the name that was originally queried. The **authority** section contains records of other authoritative servers. The **additional** section contains other helpful records.

TLSv1.3 (SSL) and HTTP : TLSv1.3 provides security in communication by encrypting the application data.The basic unit of data in **SSL**(Secure Socket Layer) is a record. Each record consists of a five-byte record header, followed by data.
The record format is **Type** (Handshake, Application Data, Alert and Change Cipher Spec), **Version** and **Length**. **MAC** is the message authentication code.



Hypertext Transfer Protocol is used in the application layer. A simple request message from a client computer consists of a request line to get a required resource, headers (Eg, Accept-Language: EN), an empty line and a message body which is optional. A simple response message from the server consists of HTTP Status Code, Headers (Eg, Accept-Language: EN), an empty line and a message body which is optional. TLSv1.3 acts as a security wrapper for transportation which along with http forms the https protocol that provides end-to-end communications security over networks and is widely used for internet communications.



2) Protocol Used at different layers:

- Data Link Layer:
  Ethernet II: It contains the information about the Source and Destination MAC addresses which are unique identifiers assigned to the Network Interface Controllers (NICs) present in the machines and these are globally unique.

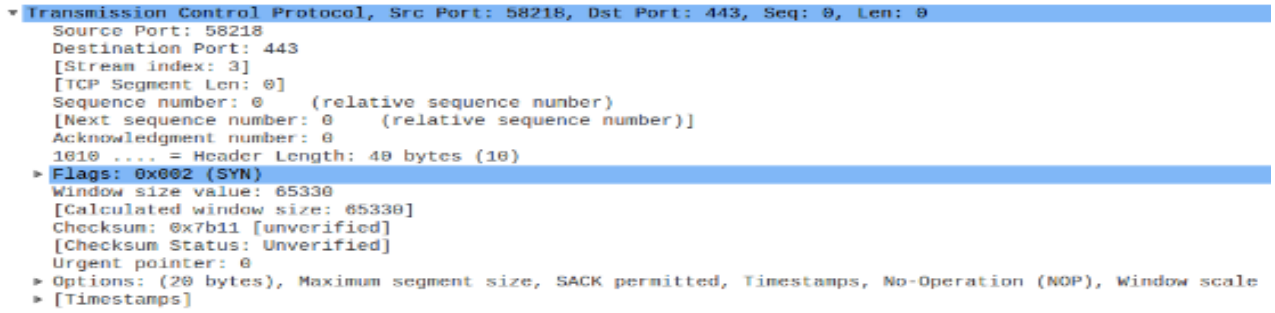
```
▾ Ethernet II, Src: 3a:89:2c:4a:5e:64 (3a:89:2c:4a:5e:64), Dst: 3a:89:2c:a4:1c:9f (3a:89:2c:a4:1c:9f)
  ▾ Destination: 3a:89:2c:a4:1c:9f (3a:89:2c:a4:1c:9f)
      Address: 3a:89:2c:a4:1c:9f (3a:89:2c:a4:1c:9f)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: 3a:89:2c:4a:5e:64 (3a:89:2c:4a:5e:64)
      Address: 3a:89:2c:4a:5e:64 (3a:89:2c:4a:5e:64)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv6 (0x86dd)
```

  Destination(here, My PC) indicates the address of the destination adapter. Source indicates the address of the source adapter. Type indicates the upper layer protocol to be used. An IG bit of 0 indicates that this is a unicast MAC address while an IG bit of 1 indicates a multicast or broadcast address(here, IG bit of source and destination is 0). An LG bit distinguishes vendor assigned and administratively assigned MAC addresses. When we administratively change the MAC address of your device, this bit is set to 1. Otherwise, it is 0.(Here, LG bit of Source is and of destination is 1)
- Network Layer:
  IPv6: The image for IPV6 packet from wireshark is shown. The description is given below:

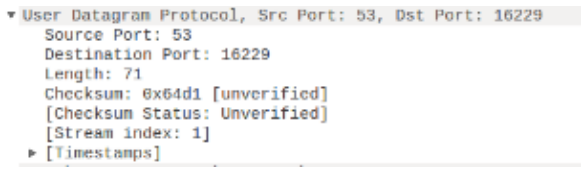
```
▾ Internet Protocol Version 6, Src: fe80::106f:6410:2695:eafa, Dst: fe80::6bae:6b3e:b6f3:ec1d
    0110 .... = Version: 6
  ▾ .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
      .... 0000 00.. .... .... .... .... .... = Differentiated Services Codepoint: Default (0)
      .... .... ..00 .... .... .... .... .... = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    .... .... .... 1101 1000 0000 0101 0111 = Flow Label: 0xd8057
    Payload Length: 71
    Next Header: UDP (17)
    Hop Limit: 255
    Source: fe80::106f:6410:2695:eafa
    Destination: fe80::6bae:6b3e:b6f3:ec1d
```

  Hop limit is the maximum number of hops (255) the packet can make before dying out. Payload Length indicates the length of the IPv6 payload (71). Next header tells type of extension header(if present) immediately following the IPv6 header. Source IP is the IP of the sender, here it is **feae:106f:6419:2695:eafa**. Destination IP is the IP of the receiver, here it is **fe80:6bae:6b3e:b6f3:ec1d**.
- Transport Layer:
  TCP: The Source Port(58218) and Destination Port(443) fields (16 bits each) identify the endpoints of the connection. In this case, the source port is 58218 and destination port is 443. Sequence Number (32 bits) specifies the number assigned to the first byte of data in the current message. Acknowledgement Number (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. Header Length (variable length) tells how many bytes are contained in the TCP

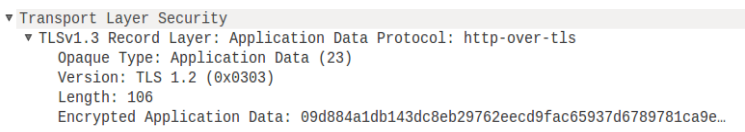
header. Flags field (6 bits) contains various flags. Here SYN means Synchronizes sequence numbers to initiate a connection. Window(here, 65330) field (16 bits) specifies the size of the sender's receive window (that is, buffer space available for incoming data). Checksum (here, 0x7b11) field (16 bits) 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.



UDP: The Source Port(53) and Destination Port(16229) fields (16 bits each) identify the endpoints of the connectionless interface. Length(here, 71) indicates length of UDP including header and data. Checksum (here, 0x64d1) 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.

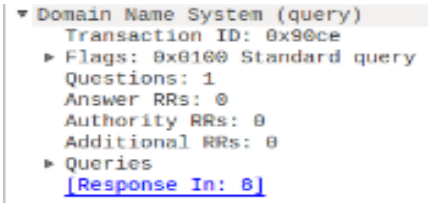


- Application Layer:
  TLSv1.3 (Transport Layer Security) is the secure socket layer version used by the RedBall server. HTTP is the underlying protocol for application data transfer over the web which is encrypted by the TLS layer during transportation to provide data security. Hence, no stand-alone HTTP packets are received from the RedBall server. They are all encrypted with TLS. Application Data Protocol indicates the application layer protocol used with TLS (here, http). Opaque refers to uninterrupted data (here the data is Application data). Version indicates the TLS version used. Length indicates the length of the packet sent (including header). Encrypted Application data shows the data transferred in encrypted format.



The HTTP request (without) TLS is seen from an external server which monitors Flash applications (core.mochibot.com). Request method indicates the desired action to be performed for a given resource(here, POST).Host URL : code.mochibot.com is the query string. HTTP Version is 1.1 User-Agent: Chrome browser running on Ubuntu. Connection : Keep-alive denotes persistence of TCP connection used by HTML. Accept Language:US-English

DNS query is done to get the IP mapping for the domain. Transaction ID is the 16 bit field to track queries and responses to queries (here, 0x90ce). Flags are controls the content of the



next four states-Questions, Answer, Authority, Additional. **Questions** contain information about the query that is being made. 1 indicates that the query type is a question. **Answer** contains the resource records for the name that was originally queried.(0) **Authority** contains records of other authoritative servers.(0) **Additional** contains other helpful records.(0)

3) Protocol Helping in functioning of application and reason for using them:

RedBall4 is an https website, i.e, the website is secured by an SSL certificate (now TLS). Hence it uses TLSv1.3 (SSL) as a security wrapper for transportation with HTTP at the Application Layer.

- HTTP is the underlying protocol used by the World Wide Web and defines how messages are formatted and transmitted.
- TLSv1.3 is a Secure Sockets Layer protocol which encrypts the application data and prevents hackers who snoop packets to get any important information.
- TLSv1.3 provides end-to-end communications security over networks and is widely used for internet communications.

DNS protocol is used initially to query for the IPv6 address of the RedBall4 website.

- DNS is used to translate domain names into IP Addresses, which computers can understand. It responds with the IP address of the server.
- It allows users to have human-readable domains while ensuring a mapping to the IP addresses corresponding to the domain names.

At the Transport Layer, TCP(Transmission Control Protocol) is used mainly for communication by the client and RedBall server to send application data. Some UDP packets are also observed when querying and getting DNS response to get IP for the gaming site.

- TCP is a connection-oriented reliable data transfer protocol and involves handshaking between the client and the server.
- Http (secured with TLS) uses TCP in my application to send server data without loss. TCP ensures data integrity, proper sequencing and avoidance of duplicate delivery.
- TCP also ensures proper error handling and flow control mechanisms to minimize the error loss rate which is needed in my application (RedBall) as while playing the game, we cannot afford in between data loss.
- UDP is an alternative communications protocol to TCP used primarily for establishing low-latency and loss-tolerating connections between applications on the internet. DNS query to fetch the IP address of Red Ball server uses UDP as the underlying protocol.

IPv6 is used at the Network Layer as TCP works over Internet protocol.

- IPv6 is a network layer protocol that enables data communications over a packet switched network. IPv6 has an increased address space (128 bits) allowing for an almost limitless number of unique IP addresses. It is referred to as "next gen internet" and hence RedBall uses IPv6 address.
- The size of the IPv6 address space makes it less vulnerable to malicious activities such as IP scanning.
- IPv6 packets can support a larger payload than IPv4 packets resulting in increased throughput and transport efficiency.

Ethernet (II) is used at the data link layer.

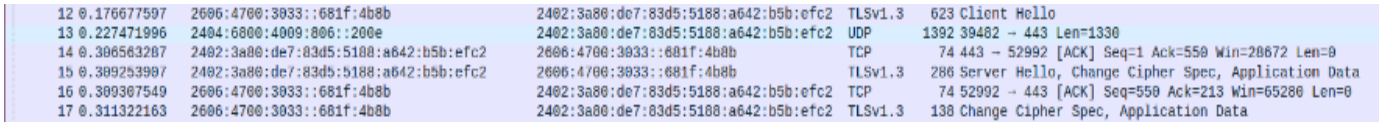
- It is one of the most widely used and reliable link layer protocols.
- It has a well defined preamble for synchronization and CRC field for error detection.
- It ensures reliable data transfer between 2 network devices (i.e on a link) on the path of the packet.

4) Sequence of messages

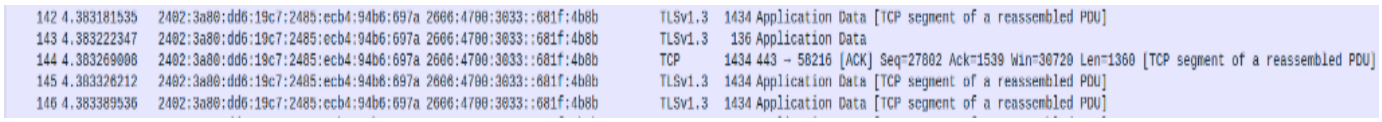
- Establishment of Connection with Gaming Application (Handshaking):
  Step 1 (SYN), Client (my PC) establishes a connection with the RedBall server, so it sends a

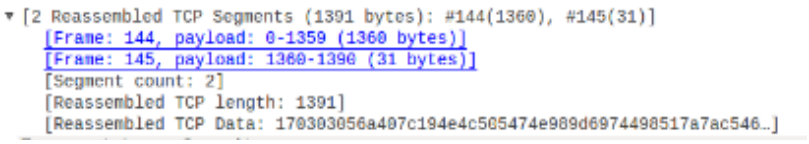
segment with SYN(Synchronize Sequence Number) which informs the server that the client is likely to start communication and the sequence number. Step 2 (SYN + ACK), Server responds to the client request with SYN-ACK signal bits set. The ACK signifies the response of the segment it received and SYN signifies with what sequence number it is likely to start the segments with. Step 3 (ACK), the client acknowledges the response of the server and thus establish a reliable connection with which they start the actual data transfer. The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is thus established.

- Application layer handshaking starts with client sending a "Client Hello" to the server to which server responds with a "Server Hello" to complete the handshake. This occurs at the start of the site of Red Ball.



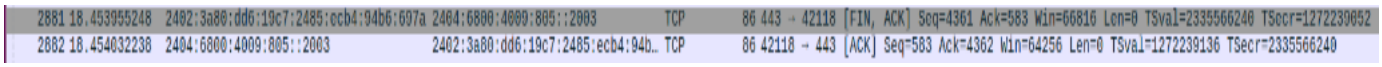
- Playing the Game: Application data with TCP and IP headers make up a segment. During the duration of the game (Clicking Start, choosing level and playing), different TCP segments carrying data arrive at the client machine and they may be out of order (as they may take different paths for reducing congestion and load balancing implemented by IPv6 protocol at Network layer). Once all these segments arrive, they are re-assembled and the data is given to the application layer (out of order data is never used).



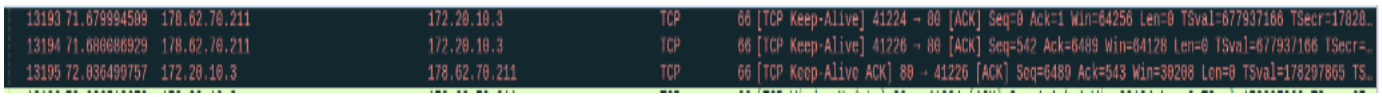


We can see in TCP info of Frame #146 that 2 frames(144,145) have been reassembled to provide the Application Layer (Secure Sockets Layer : TLSv1.3 protocol.

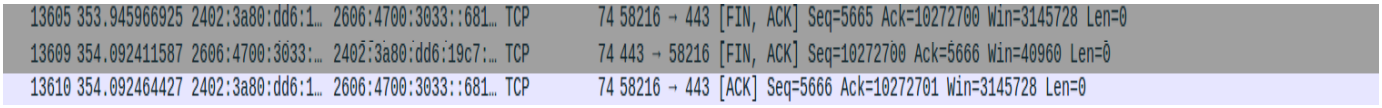
- Pausing the game: After pause is pressed, some application data still comes to the client machine (till the receiving buffer is filled) . Then the client sends a FIN (piggybacked with an ACK) to the server to stop the packet inflow and the server acknowledges.



But as this is pause functionality and user can click play any time, the client keeps sending TCP Keep-Alive to the server periodically and server sends TCP Keep-Alive ACKs in reply of these so that the connection is maintained. (Once play is pressed, client just sends a SYN and starts receiving data, server does not need to send a SYN and go through the handshake again as TCP Connection is intact. Note that in some cases, client sends as RST when paused and then TCP Connection has to be established when resumed.)



- Quitting the game and closing tab: The client sends a FIN acknowledgement to halt the data transmission (piggybacked by an ACK) which is then ACKed by the server which in turn sends a FIN acknowledgement to the client and client acknowledges it to complete the 4 way TCP Termination handshake. Note here, in the second line ACK=5666 indicates server has acknowledged the previous FIN packet sent by client and also incremented Ack number for the next Acknowledgement, i.e., the ACK=5665 and ACK=5666 are sent together by the server indicated by ACK=5666.



5) Statistics

	1:45 AM (Central library)	8:30 AM (Hostel)	7 PM (Central library)
Throughput (Bytes/s)	31,909.62	40,320.04	38,656.18
Round Trip Time (ms)	4.2	9.6	6.4
Avg. Packet size (Bytes)	844	1176	847
No. of packets lost	0	0	0
No. of TCP Packets	13,123	11,773	13,184
No. of UDP Packets	376	353	452
No. of Responses per Request sent	1.497 (7712/5149)	1.309 (4483/3423)	1.439 (7701/5351)

6) Multiple Hosts

I ran the experiment multiple times during the day. All the experiments were performed using my mobile data. Two server IP addresses were observed: 2606:4700:3033::681f:4b8b and 2606:4700:3031::681f:4a8b (6PM and 7PM file in the drive). Such websites use multiple servers as data transfer is faster because of Load Balancing of data across servers since there is little network congestion and increased reliability. Different servers are also helpful in ensuring reliability since there is no single point of failure. Since there are multiple paths to the user, even if some server experiences some issues, others can provide the data to the client without interruption.

On further analysis, while checking the TLS packets and DNS queries made during the time the game was loaded and played, I found that connections were made to many different servers while running the gaming website. These servers were not for the gaming application, but were called for certain specific additional tasks. Some of them include:

- fonts.googleapis.com : While searching for the game, the device would have loaded the Google fonts to display the search results.
- www.google.com : Google is my default search engine. Hence, my device connects to Google while showing search suggestions when the website url is typed.
- google-analytics.com: Google Analytics is a web analytics service offered by Google that tracks and reports website traffic.
- core.mochibot.com: MochiBot is used to monitor traffic for Flash applications and since RedBall runs on Flash, the device must have called this IP for Flash functionality monitoring.