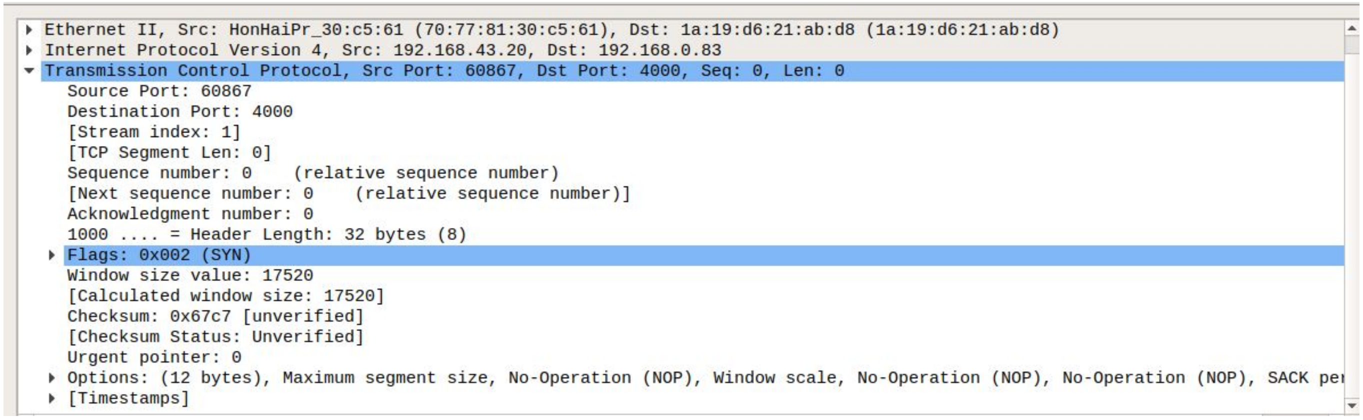
# CS342 Assignment 2

## Online game : AirMech

**1)**List of protocols used by the application at different layers as detected from the traces.

## TRANSPORT LAYER:

### ❖ TCP (Transmission Control Protocol):

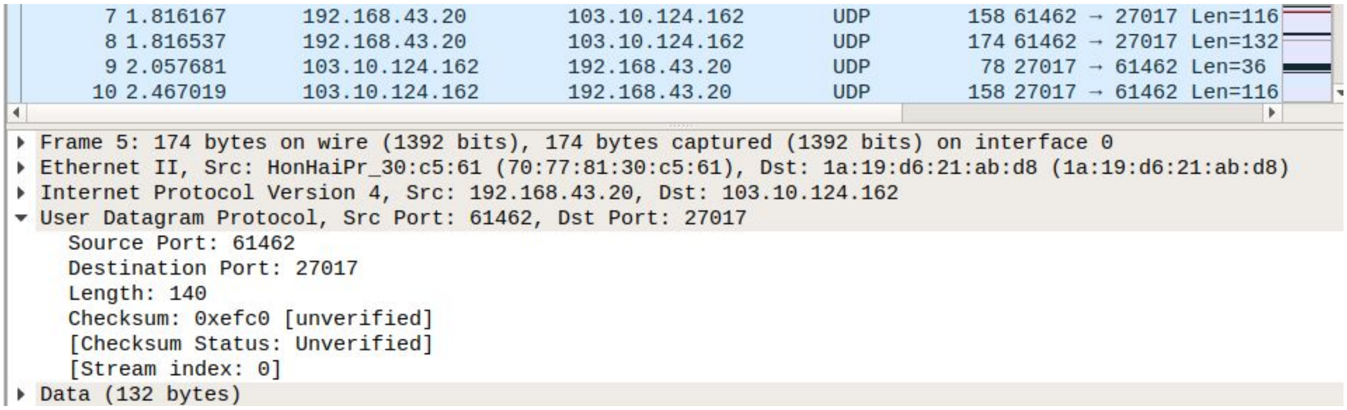


```
▸ Ethernet II, Src: HonHaiPr_30:c5:61 (70:77:81:30:c5:61), Dst: 1a:19:d6:21:ab:d8 (1a:19:d6:21:ab:d8)
▸ Internet Protocol Version 4, Src: 192.168.43.20, Dst: 192.168.0.83
▾ Transmission Control Protocol, Src Port: 60867, Dst Port: 4000, Seq: 0, Len: 0
    Source Port: 60867
    Destination Port: 4000
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x002 (SYN)
    Window size value: 17520
    [Calculated window size: 17520]
    Checksum: 0x67c7 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK pe
  ▸ [Timestamps]
```

Fields of the packets belonging to TCP protocol :

- **Source Port**: The source port is the next available client number assigned to the user machine, for the router to know which user to send back responses to.
- **Destination Port**: This number identifies the port of the receiving application
- **Sequence Number**: The sequence number is the byte number of the first byte of data in the TCP packet sent (also called a TCP segment).
- **Ack Number**: The sequence number of the next byte the receiver expects to receive
- **Header Length**: Specifies the size of the TCP header in 32-bit words.
- **Flags:** Flags are used to indicate a particular state of connection or to provide some additional useful information like troubleshooting purposes or to handle a control of a particular connection. Most commonly used flags are "SYN", "ACK" and "FIN".
- **Window Size**: Denotes how much data (in bytes) the receiving device is willing to receive at any point in time.
- **Checksum**: The value that represents the number of bits in a transmission message and is used to detect high-level errors within data transmissions.
- **Urgent Pointer:**The urgent pointer is a 16 bit value in the TCP header located after the checksum that is set when the TCP urgent flag is 1.

### ❖ UDP(User Datagram Protocol): The UDP header has a fixed length of 8 bytes



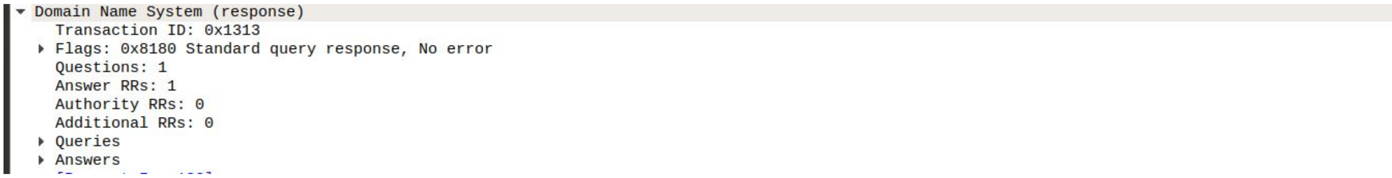
```
     7 1.816167      192.168.43.20       103.10.124.162      UDP        158 61462 → 27017 Len=116
     8 1.816537      192.168.43.20       103.10.124.162      UDP        174 61462 → 27017 Len=132
     9 2.057681      103.10.124.162      192.168.43.20       UDP         78 27017 → 61462 Len=36
    10 2.467019      103.10.124.162      192.168.43.20       UDP        158 27017 → 61462 Len=116

▸ Frame 5: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_30:c5:61 (70:77:81:30:c5:61), Dst: 1a:19:d6:21:ab:d8 (1a:19:d6:21:ab:d8)
▸ Internet Protocol Version 4, Src: 192.168.43.20, Dst: 103.10.124.162
▾ User Datagram Protocol, Src Port: 61462, Dst Port: 27017
    Source Port: 61462
    Destination Port: 27017
    Length: 140
    Checksum: 0xefc0 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
▸ Data (132 bytes)
```

- **Source Port:** The port of the device sending the data. This field can be set to zero if the destination computer doesn't need to reply to the sender.
- **Destination Port**: The port of the device receiving the data
- **Length:** Length is the length in octets of this user datagram including this header and data
- **Checksum:** checksums are often used to verify data integrity but are not relied upon to verify data authenticity.Although unlike in TCP, UDP does not provide any guarantee of package delivery.
- **Timestamp**: It ensures that the endpoints keep the current value of RTT between them

## APPLICATION LAYER

### ❖ DNS (Domain Name system):



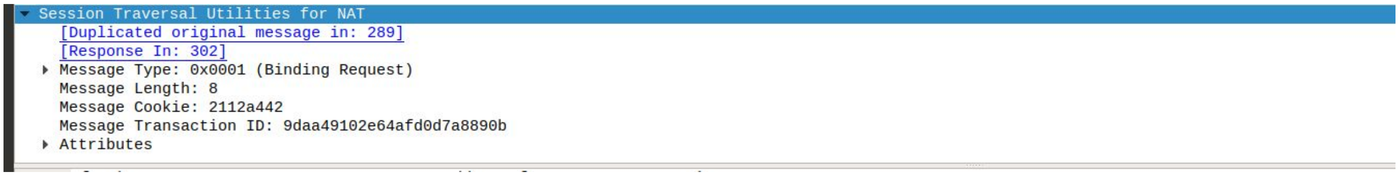
```
▾ Domain Name System (response)
    Transaction ID: 0x1313
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▸ Queries
  ▸ Answers
```

- **Transaction ID:** Enables easy tracking of queries and query responses
- **Queries**:This denotes the list of query structures
- **Flags**: Entails information like whether the packet is recursive or not, whether it is query or response etc.
- **Questions:**
- **Answer RR's :**
- **Authority RR's:** are type NS records pointing to name servers closer to the target name in the naming hierarchy.
- **Additional RR's**

### ❖ TLSv1.2 (Transport Layer Security)

- **Content Type:** Type of the content being carried by TLS
- **Application Data Protocol:** The protocol containing the application data
- **Version:** Version of TLS being used(1.2)
- **Encrypted Application:** Data associated with the encrypted application
- **Length:** Length of the data being transferred

### ❖ STUN( Session Traversal Utilities for NAT):



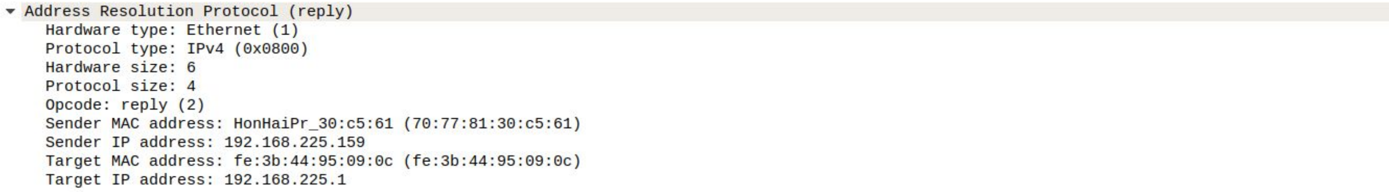
```
▾ Session Traversal Utilities for NAT
    [Duplicated original message in: 289]
    [Response In: 302]
  ▸ Message Type: 0x0001 (Binding Request)
    Message Length: 8
    Message Cookie: 2112a442
    Message Transaction ID: 9daa49102e64afd0d7a8890b
  ▸ Attributes
```

- **Message Type:** STUN messages are TLV (type-length-value) encoded using big endian (network ordered) binary. All STUN messages start with a STUN header, followed by a STUN payload.
- **Length:** Indicates the total length of the STUN payload in bytes but does not include the 20 bytes header.
- **Cookie:**

➢ **Transaction ID:** Is used to correlate requests and responses.

## LINK LAYER

### ❖ ARP(Address Resolution Protocol):



```
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: HonHaiPr_30:c5:61 (70:77:81:30:c5:61)
    Sender IP address: 192.168.225.159
    Target MAC address: fe:3b:44:95:09:0c (fe:3b:44:95:09:0c)
    Target IP address: 192.168.225.1
```

ARP finds the hardware address, also known as Media Access Control (MAC)  address, of a host from its known IP address.

➢ **Hardware Type** : specifies the type of hardware used for the network transmitting the Address Resolution Protocol (ARP) message. Ethernet is the common Hardware Type and the value for Ethernet is 1.

➢ **Protocol Type:** The address resolution protocol (arp) is a protocol used by the Internet Protocol (IP) [RFC826], specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol.

➢ **Hardware Size**: Refers to the amount of bytes in MAC address(6 bytes)

➢ **Protocol Size**: Refers to the amount of bytes in an IPV4 address(4 bytes)

➢ **OpCode**: Specifies the nature of the message. 1 for ARP request and 2 for ARP reply

➢ **Sender MAC/IP address**: MAC and IP addresses of the sender of the ARP request

➢ **Target MAC/IP address**: MAC and IP addresses of the receiver of the request.

## INTERNET LAYER

### ❖ ICMPv6:

➢ **Payload Length**: The IPv6 packet payload is the combination of the IPv6 extension headers and the upper layer PDU.

➢ **Next Header**: Next Header indicates type of extension header(if present) immediately following the IPv6 header. Whereas In some cases it indicates the protocols contained within upper-layer packet, such as TCP, UDP

➢ **Hop Limit**: Indicates the maximum number of links over which the **IPv6** packet can travel before being discarded. The size of this field is 8 bits.

➢ **Source:**Source Address is 128-bit IPv6 address of the original source of the packet.

➢ **Destination**:Destination Address (128-bits) : Destination Address field indicates the IPv6 address of the final destination

2)The important functionalities associated with the game are:

| | |
|---|---|
| Start Game | DNS, TLSv1.2, TCP |
| Game Play( Data flow, maintaining state) | TCP,UDP, Ethernet II |
| End Game | TCP |

### a)DNS(Domain Name System):

- Most modern online games employ a client-server model.
- Our Device, the DNS client, i.e our system issues a DNS request to the game server, providing the hostname, which in our case, is api.steampowered.com. The request is received by a DNS resolver, which is responsible for finding the correct IP address for that hostname.
- The DNS resolver looks for a DNS name server that holds the IP address for the hostname in the DNS request.When the resolver reaches the authoritative DNS name server for “example.com”, it receives the IP address and other relevant details, and returns it to the DNS client. The DNS request is now resolved.The DNS client device can connect to the server directly using the correct IP address.

### b)TLSv1.2(Transport Layer Security):

- The TLS record protocol provides connection security, and the TLS handshake protocol enables the client and server to authenticate each other and to negotiate security keys before any data is transmitted.
- The TLS handshake is a multi-step process. A basic TLS handshake involves the client and server sending “hello” messages, and the exchange of keys, cipher message and a finish message. The multi-step process is what makes TLS flexible enough to use in different applications because the format and order of exchange can be modified.

### c)TCP(Transmission Control Protocol):

- TCP is a connection oriented protocol that ensures that a connection is established and maintained until application programs from both client and server ends have finished exchanging messages.(By sending [SYN] and [FIN] messages.
- It determines how to break application data into packets that networks can deliver, sends packets to and accepts packets from the network layer, manages flow control and -- because it is meant to provide error-free data transmission -- handles retransmission of dropped or garbled packets and acknowledges all packets that arrive.

### d)UDP(User Datagram Protocol) :

- **UDP** (User Datagram Protocol) is a communications protocol that is primarily used for establishing low-latency and loss-tolerating connections between applications on the internet. It has lower reliability in comparison to TCP
- Majorly used as part of application data flow, an example of how udp is used in the gaming arena is sending position information.the position of a character is sent several times a

second and it doesn't matter whether a packet is lost as another one will be sent again shortly.

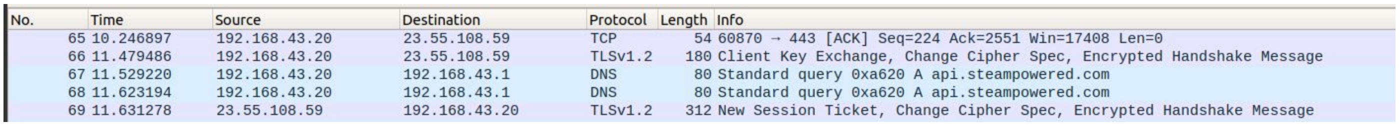
### e)Ethernet II(Link Layer Protocol):

- The Ethernet is a local area network (LAN) set of protocols which serves the physical and data link layers.

3)The message sequences as observed during START and END functionalities are as follows:

### Starting Game:

1)A **DNS request** is made by the client(our PC) for resolving the hostname of the DNS server to an IP address.

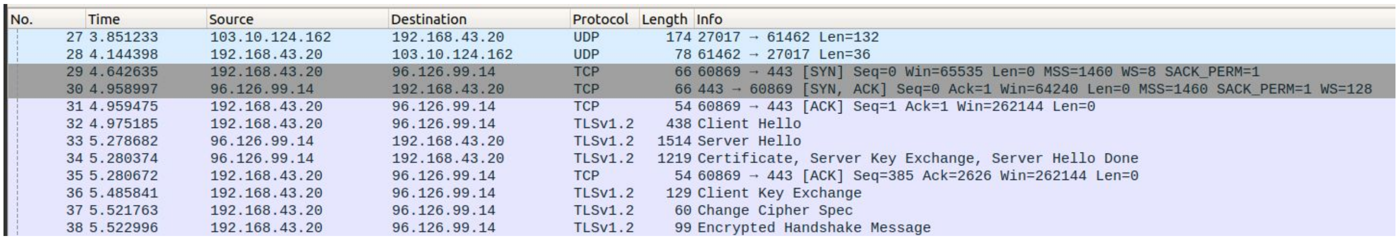


| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 65 | 10.246897 | 192.168.43.20 | 23.55.108.59 | TCP | 54 | 60870 → 443 [ACK] Seq=224 Ack=2551 Win=17408 Len=0 |
| 66 | 11.479486 | 192.168.43.20 | 23.55.108.59 | TLSv1.2 | 180 | Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message |
| 67 | 11.529220 | 192.168.43.20 | 192.168.43.1 | DNS | 80 | Standard query 0xa620 A api.steampowered.com |
| 68 | 11.623194 | 192.168.43.20 | 192.168.43.1 | DNS | 80 | Standard query 0xa620 A api.steampowered.com |
| 69 | 11.631278 | 23.55.108.59 | 192.168.43.20 | TLSv1.2 | 312 | New Session Ticket, Change Cipher Spec, Encrypted Handshake Message |

2)**TCP handshaking**

A three way TCP handshake follows:

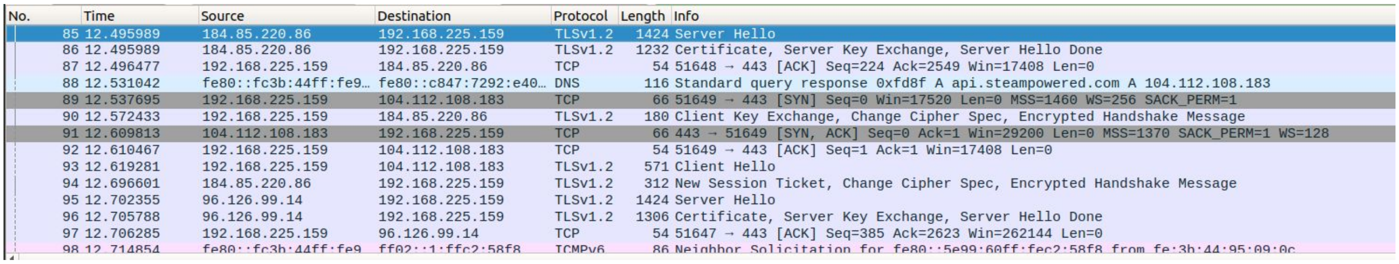
- A SYN frame containing request is sent to the destination server from the client
- The server sends an ACK packet for acknowledging the request. It also sends a SYN packet in order to synchronize the sequence number.
- An ACK frame is sent to the server from the client acknowledging the above packet.



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 27 | 3.851233 | 103.10.124.162 | 192.168.43.20 | UDP | 174 | 27017 → 61462 Len=132 |
| 28 | 4.144398 | 192.168.43.20 | 103.10.124.162 | UDP | 78 | 61462 → 27017 Len=36 |
| 29 | 4.642635 | 192.168.43.20 | 96.126.99.14 | TCP | 66 | 60869 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 SACK_PERM=1 |
| 30 | 4.958997 | 96.126.99.14 | 192.168.43.20 | TCP | 66 | 443 → 60869 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128 |
| 31 | 4.959475 | 192.168.43.20 | 96.126.99.14 | TCP | 54 | 60869 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 32 | 4.975185 | 192.168.43.20 | 96.126.99.14 | TLSv1.2 | 438 | Client Hello |
| 33 | 5.278682 | 96.126.99.14 | 192.168.43.20 | TLSv1.2 | 1514 | Server Hello |
| 34 | 5.280374 | 96.126.99.14 | 192.168.43.20 | TLSv1.2 | 1219 | Certificate, Server Key Exchange, Server Hello Done |
| 35 | 5.280672 | 192.168.43.20 | 96.126.99.14 | TCP | 54 | 60869 → 443 [ACK] Seq=385 Ack=2626 Win=262144 Len=0 |
| 36 | 5.485841 | 192.168.43.20 | 96.126.99.14 | TLSv1.2 | 129 | Client Key Exchange |
| 37 | 5.521763 | 192.168.43.20 | 96.126.99.14 | TLSv1.2 | 60 | Change Cipher Spec |
| 38 | 5.522996 | 192.168.43.20 | 96.126.99.14 | TLSv1.2 | 99 | Encrypted Handshake Message |

3)**TLSv1.2:**

The client(192.168.225.159) and the game server(96.12.782355) exchange hellos. This is accompanied by the corresponding ack messages. The client requests the server to change encryption via a Change Cipher Spec request, to change the encryption during handshaking. A connection is finally established when the server responds to this request



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 85 | 12.495989 | 184.85.220.86 | 192.168.225.159 | TLSv1.2 | 1424 | Server Hello |
| 86 | 12.495989 | 184.85.220.86 | 192.168.225.159 | TLSv1.2 | 1232 | Certificate, Server Key Exchange, Server Hello Done |
| 87 | 12.496477 | 192.168.225.159 | 184.85.220.86 | TCP | 54 | 51648 → 443 [ACK] Seq=224 Ack=2549 Win=17408 Len=0 |
| 88 | 12.531042 | fe80::fc3b:44ff:fe9… | fe80::c847:7292:e40… | DNS | 116 | Standard query response 0xfd8f A api.steampowered.com A 104.112.108.183 |
| 89 | 12.537695 | 192.168.225.159 | 104.112.108.183 | TCP | 66 | 51649 → 443 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 90 | 12.572433 | 192.168.225.159 | 184.85.220.86 | TLSv1.2 | 180 | Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message |
| 91 | 12.609813 | 104.112.108.183 | 192.168.225.159 | TCP | 66 | 443 → 51649 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1370 SACK_PERM=1 WS=128 |
| 92 | 12.610467 | 192.168.225.159 | 104.112.108.183 | TCP | 54 | 51649 → 443 [ACK] Seq=1 Ack=1 Win=17408 Len=0 |
| 93 | 12.619281 | 192.168.225.159 | 104.112.108.183 | TLSv1.2 | 571 | Client Hello |
| 94 | 12.696601 | 184.85.220.86 | 192.168.225.159 | TLSv1.2 | 312 | New Session Ticket, Change Cipher Spec, Encrypted Handshake Message |
| 95 | 12.702355 | 96.126.99.14 | 192.168.225.159 | TLSv1.2 | 1424 | Server Hello |
| 96 | 12.705788 | 96.126.99.14 | 192.168.225.159 | TLSv1.2 | 1306 | Certificate, Server Key Exchange, Server Hello Done |
| 97 | 12.706285 | 192.168.225.159 | 96.126.99.14 | TCP | 54 | 51647 → 443 [ACK] Seq=385 Ack=2623 Win=262144 Len=0 |
| 98 | 12.714854 | fe80::fc3b:44ff:fe9 | ff02::1:ffc2:58f8 | ICMPv6 | 86 | Neighbor Solicitation for fe80::5e99:60ff:fec2:58f8 from fe:3b:44:95:09:0c |

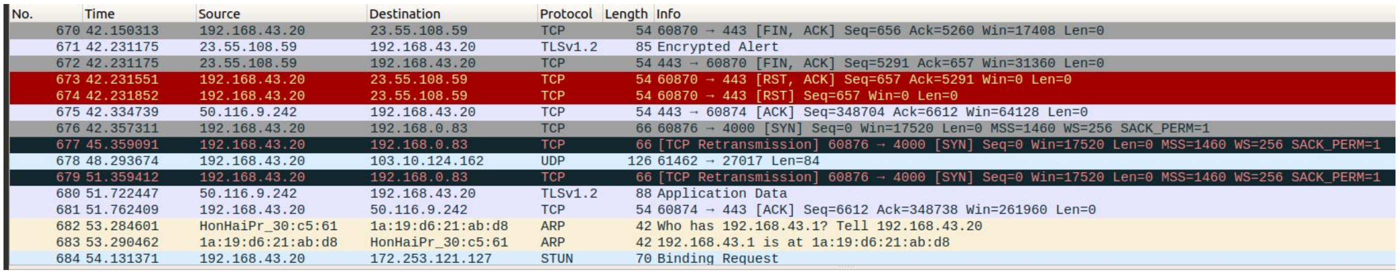
### Ending Game:

**TCP:**

In a very similar fashion to sending [SYN], [SYN, ACK] and [ACK] frames while starting the game, TCP also ensures a smooth finishing to the message exchange between the client and the server. The client sends a request to the server containing a [FIN,ACK] frame in order to terminate the present connection

The server acknowledges the request and sends back a [FIN,ACK] containing packet

The client sends an ACK frame packet acknowledging the termination and the connection closes.



```
No.   Time       Source             Destination        Protocol Length Info
  670 42.150313  192.168.43.20      23.55.108.59       TCP        54 60870 → 443 [FIN, ACK] Seq=656 Ack=5260 Win=17408 Len=0
  671 42.231175  23.55.108.59       192.168.43.20      TLSv1.2    85 Encrypted Alert
  672 42.231175  23.55.108.59       192.168.43.20      TCP        54 443 → 60870 [FIN, ACK] Seq=5291 Ack=657 Win=31360 Len=0
  673 42.231551  192.168.43.20      23.55.108.59       TCP        54 60870 → 443 [RST, ACK] Seq=657 Ack=5291 Win=0 Len=0
  674 42.231852  192.168.43.20      23.55.108.59       TCP        54 60870 → 443 [RST] Seq=657 Win=0 Len=0
  675 42.334739  50.116.9.242       192.168.43.20      TCP        54 443 → 60874 [ACK] Seq=348704 Ack=6612 Win=64128 Len=0
  676 42.357311  192.168.43.20      192.168.0.83       TCP        66 60876 → 4000 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
  677 45.359091  192.168.43.20      192.168.0.83       TCP        66 [TCP Retransmission] 60876 → 4000 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
  678 48.293674  192.168.43.20      103.10.124.162     UDP       126 61462 → 27017 Len=84
  679 51.359412  192.168.43.20      192.168.0.83       TCP        66 [TCP Retransmission] 60876 → 4000 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
  680 51.722447  50.116.9.242       192.168.43.20      TLSv1.2    88 Application Data
  681 51.762409  192.168.43.20      50.116.9.242       TCP        54 60874 → 443 [ACK] Seq=6612 Ack=348738 Win=261960 Len=0
  682 53.284601  HonHaiPr_30:c5:61  1a:19:d6:21:ab:d8  ARP        42 Who has 192.168.43.1? Tell 192.168.43.20
  683 53.290462  1a:19:d6:21:ab:d8  HonHaiPr_30:c5:61  ARP        42 192.168.43.1 is at 1a:19:d6:21:ab:d8
  684 54.131371  192.168.43.20      172.253.121.127    STUN       70 Binding Request
```

4)

| Statistics | Time | | |
|---|---|---|---|
| | 14:30 | 19:00 | 22:40 |
| Throughput(Bytes/sec) | 3523 | 1463 | 1480 |
| RTT(ms) | 263 | 527 | 429 |
| Packet Size(B) | 463 | 277 | 400 |
| Number of Packets Lost | 0 | 0 | 0 |
| UDP packets | 250 | 251 | 169 |
| TCP packets | 1616 | 1767 | 169 |
| Response Ratio | 0.17 | 0.42 | 10.57 |

5)Yes there indeed exist multiple source/destination IP addresses. They are listed below

| **Source** | **Destination** |
|---|---|
| 192.168.225.159 | 50.116.9.242 |
| 50.116.9.242 | 103.10.124.164 |
| 50.116.14.9 | 172.253.121.127 |
| 50.116.25.154 | 192.168.225.159 |
| 50.116.57.237 | -------- |

Having multiple IP addresses on the same physical network is advantageous as:

- ➢ It will prevent traffic from being exchanged via the gateway
- ➢ Speeds things up and reduces the load.
- ➢ In order to use different public IP addresses to avoid firewalls or to avoid being blacklisted in SPAM filters.
- ➢ And also if one network fails, there won't be an interruption.