

Computer Networks Lab

Assignment No. 2

Submitted By:

Ashish Kumar Poddar

Roll No.: 150123049

Mathematics and Computing

IIT Guwahati

Date - 18th February, 2018

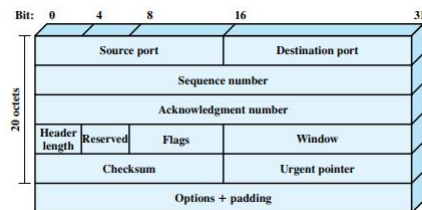
The trace files had size larger than 2 MB and thus I have uploaded them [here](#).

Question 1

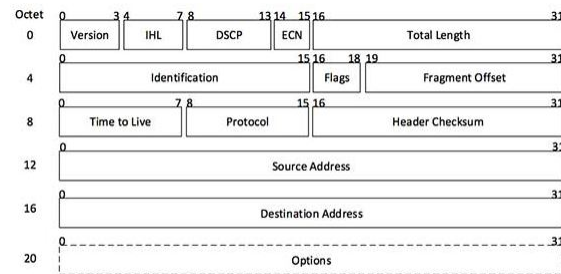
The application I was supposed to do my project on was Live Sports Streaming. I used the website www.sonyliv.com to do so. While streaming live sports, a number of protocols are used. From traces, I could figure out the TCP, UDP and the IPv4 protocols being used in the streaming process. these protocols belong to the transport layer. The Address Resolution Protocol was also used in our application.

Transmission Control Protocol

TCP provides transport layer services to applications. It is connection-oriented i.e. it establishes a connection before transmitting data. The frame format for TCP is the following



(a) Question 1
TCP Header



(b) Question 1
IP Header

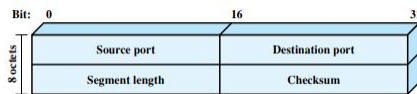
The source port and the destination port specify the source and the destination port respectively. The sequence and acknowledgement number specify the respective properties for the exchanged packets. The Data offset shows where the data begins. The Control bit

carries various control information. Window is the number of octets the sender can send without waiting for an acknowledgement while the checksum is used for errors. The urgent pointer points to the sequence number of the octet following the urgent data. Options may occupy a space at the end of the TCP header while Padding ensures that the header ends and data begins on a 32 bit boundary.

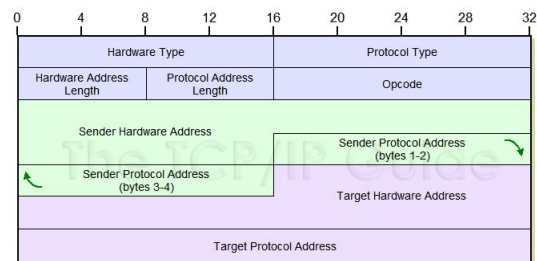
IPv4 Protocol

IPv4 takes data segments from the transport layer and divides it into packets. IP packet encapsulates data unit received from above and adds its own header information. The frame format contains many relevant information. Version defines the version no. of the IP protocol used. IHL shows the IP header length. DSCP is the type of service while ECN carries information about congestion in the route. The Total Length is the length of the entire IP packet. Fragment offset tells us the exact position of the fragment in the IP packet. Time to Live is the TTL value of the packet. Protocol specifies the layer the packet belongs to, specifically the TCP, ICMP etc. Checksum is for error detection. The source and destination address are self explanatory. Options are only used if IHL is greater than 5.

User Datagram Protocol



(a) Question 1
UDP Header



(b) Question 1
ARP Header

UDP, in contrast to TCP, takes a minimalistic approach. It does not guarantee preservation of sequence, delivery or protection against duplication. It is connectionless and implements checksum as error control.

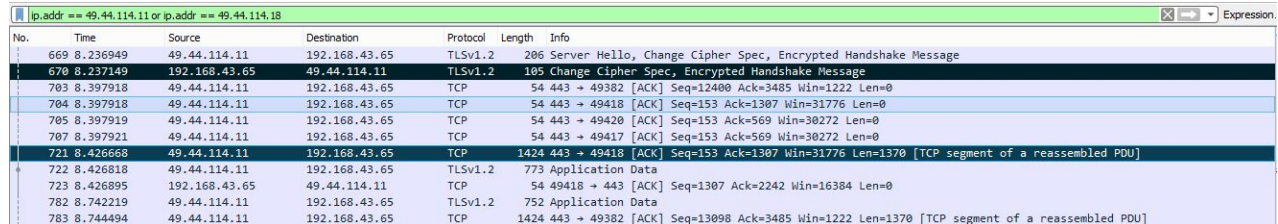
The header describes the source port, the destination port and the checksum for error control. The segment length indicates the total length of the header and the data.

Address Resolution Protocol

The ARP or the Address Resolution Protocol is an IP Protocol used to map IP addresses to the hardware addresses used by a data link protocol. It acts as an interface between the OSI network layer and the OSI link layer. The frame format is shown below. It is not used by our application specifically but it is necessary to use our application. HLN and PLN specify the hardware and protocol type, while HLA and PLA specify the respective address lengths. SHA and SPA give us the sender hardware and protocol address while THA and TPA give us the same for the target side.

Question 2

(a). **TCP Packets** - We will first look at the TCP headers. The screenshot lists a number of packets and we will be looking at the highlighted one (number 721).

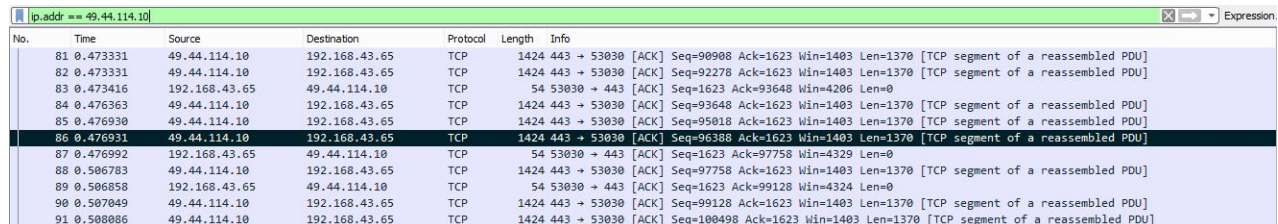


No.	Time	Source	Destination	Protocol	Length	Info
669	8.236949	49.44.114.11	192.168.43.65	TLSv1.2	206	Server Hello, Change Cipher Spec, Encrypted Handshake Message
670	8.237149	192.168.43.65	49.44.114.11	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
703	8.397918	49.44.114.11	192.168.43.65	TCP	54	443 → 49382 [ACK] Seq=12400 Ack=3485 Win=1222 Len=0
704	8.397918	49.44.114.11	192.168.43.65	TCP	54	443 → 49418 [ACK] Seq=153 Ack=1307 Win=31776 Len=0
705	8.397919	49.44.114.11	192.168.43.65	TCP	54	443 → 49420 [ACK] Seq=153 Ack=569 Win=30272 Len=0
707	8.397921	49.44.114.11	192.168.43.65	TCP	54	443 → 49417 [ACK] Seq=153 Ack=569 Win=30272 Len=0
721	8.426668	49.44.114.11	192.168.43.65	TCP	1424	443 → 49418 [ACK] Seq=153 Ack=1307 Win=31776 Len=1370 [TCP segment of a reassembled PDU]
722	8.426818	49.44.114.11	192.168.43.65	TLSv1.2	773	Application Data
723	8.426895	192.168.43.65	49.44.114.11	TCP	54	49418 → 443 [ACK] Seq=1307 Ack=2242 Win=16384 Len=0
782	8.742219	49.44.114.11	192.168.43.65	TLSv1.2	752	Application Data
783	8.744494	49.44.114.11	192.168.43.65	TCP	1424	443 → 49382 [ACK] Seq=13098 Ack=3485 Win=1222 Len=1370 [TCP segment of a reassembled PDU]

Figure 3: Question 2
TCP and TLS Data Packets

The source and the destination are 49.44.114.11 and 192.168.43.65 respectively. The Protocol is TCP. Length shows the total length of the TCP Packet, 1424 bits here. Info shows the source and the destination port of the packet as 443 and 49418 respectively. Seq and Ack show the relative sequence and acknowledgement number of the packet, here 153 and 1307 respectively. Win is the window for the packet in bits, here 31776. Len, 1370, is the length of the TCP Payload.

(b). **TLS Packets** - We take a look at the TLS or the SSL Packet. The entry highlighted dark in the above image (number 670) is our TLS packet. The Source and Destination are 132.168.43.65 and 49.44.114.11 respectively. The Protocol field shows that it is an TLS Packet. The Length of the packet here is 105. The info specifies that this TLS packet is an Encrypted Handshake Message.



No.	Time	Source	Destination	Protocol	Length	Info
81	0.473331	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=90908 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]
82	0.473331	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=92278 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]
83	0.473416	192.168.43.65	49.44.114.10	TCP	54	53030 → 443 [ACK] Seq=1623 Ack=93648 Win=4206 Len=0
84	0.476363	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=93648 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]
85	0.476930	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=95018 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]
86	0.476931	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=96388 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]
87	0.476992	192.168.43.65	49.44.114.10	TCP	54	53030 → 443 [ACK] Seq=1623 Ack=97758 Win=4329 Len=0
88	0.506783	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=97758 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]
89	0.506858	192.168.43.65	49.44.114.10	TCP	54	53030 → 443 [ACK] Seq=1623 Ack=99128 Win=4324 Len=0
90	0.507049	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=99128 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]
91	0.508086	49.44.114.10	192.168.43.65	TCP	1424	443 → 53030 [ACK] Seq=100498 Ack=1623 Win=1403 Len=1370 [TCP segment of a reassembled PDU]

Figure 4: Question 2
IPv4 Packets

(c). **IPv4 Packets** - Here, the version is 4, the Header Length is 20 bytes and the total length of the packet is 1410 while the Fragment Offset is 0. The Flag, 0x02, signifies don't fragment condition while the TTL value is 58. The protocol number is 6 (TCP). The Identification is 0x82fa. The Checksum value is 0x2934 while the Source and Destination IP Addresses are 40.44.114.10 and 192.168.43.65.

(d). **UDP Packets** - UDP Packets are only used as dns packets in out data exchange. Since DNS is only for the IP Address lookup, we don't need it run our application but we need it to find out our required IP Address. Figure 5 shows the highlighted packet number 4055.

No.	Time	Source	Destination	Protocol	Length	Info
4010	53.528008	2405:204:b00b:2...	2405:200:800::1	DNS	102	Standard query 0x9345 A players.brightcove.net
4011	53.528027	2405:204:b00b:2...	2405:200:800::1	DNS	102	Standard query 0x8e8f AAAA players.brightcove.net
4024	53.662679	2405:200:800::1	2405:204:b00b:25bd::	DNS	186	Standard query response 0x3306 AAAA assets-sonyliv.sportz.io CNAME assets-sonyliv.sportz.io.edgekey.net CNAME...
4047	53.715691	2405:200:800::1	2405:204:b00b:25bd::	DNS	180	Standard query response 0x8e8f AAAA players.brightcove.net CNAME players.brightcove.net.edgekey.net CNAME e95...
4048	53.715692	2405:200:800::1	2405:204:b00b:25bd::	DNS	202	Standard query response 0x0817 A assets-sonyliv.sportz.io CNAME assets-sonyliv.sportz.io.edgekey.net CNAME e2...
4055	53.716015	192.168.43.1	192.168.43.65	DNS	182	Standard query response 0x0817 A assets-sonyliv.sportz.io CNAME assets-sonyliv.sportz.io.edgekey.net CNAME e2...
4057	53.716017	192.168.43.1	192.168.43.65	DNS	227	Standard query response 0x3306 AAAA assets-sonyliv.sportz.io CNAME assets-sonyliv.sportz.io.edgekey.net CNAME...
4060	53.761644	2405:200:800::1	2405:204:b00b:25bd::	DNS	196	Standard query response 0x9345 A players.brightcove.net CNAME players.brightcove.net.edgekey.net CNAME e9573...

Figure 5: Question 2
UDP Data Packets

The packet shows the source and destination IP addresses as 192.168.43.1 and 192.168.43.65 respectively. The protocol shows it as DNS. The length shows the length of the entire Packet. The info shows that this is the DNS lookup request for the assets-sonyliv.sportz.io website.

(e). **ARP Packets** - This is used to find the MAC addresses from the IP Addresses of the server. Figure 6 shows the highlighted entry we are going to look at. We can see that the Source and the Destination show the IP addresses of the sender and the receiver respectively. The Protocol shows that it is an ARP Packet while the Length displays the length of the entire packet. The info displays the resolved MAC address.

No.	Time	Source	Destination	Protocol	Length	Info
17	0.770101	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f
793	13.130697	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f
1790	25.489995	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f
2375	37.849785	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f
3764	50.209997	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f
4898	62.569722	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f
6962	74.929541	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f
8589	87.290534	IntelCor_18:c1:8f	XiaomiCo_c6:02:59	ARP	42	192.168.43.65 is at 2c:6e:85:18:c1:8f

Figure 6: Question 2
ARP Data Packets

Question 3

A range of messages may be exchanged on attempting to use various functionalities of our application. We will discuss what happens when we play, pause or stop a video.

No.	Time	Source	Destination	Protocol	Length	Info
262	2.425043	192.168.43.65	49.44.114.18	TCP	66	64441 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
282	2.476343	49.44.114.18	192.168.43.65	TCP	66	443 → 64437 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1370 SACK_PERM=1 WS=32
283	2.476344	49.44.114.18	192.168.43.65	TCP	66	443 → 64440 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1370 SACK_PERM=1 WS=32
284	2.476344	49.44.114.18	192.168.43.65	TCP	66	443 → 64441 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1370 SACK_PERM=1 WS=32
285	2.476345	49.44.114.18	192.168.43.65	TCP	66	443 → 64439 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1370 SACK_PERM=1 WS=32
290	2.476899	192.168.43.65	49.44.114.18	TCP	54	64437 → 443 [ACK] Seq=1 Ack=1 Win=16384 Len=0
292	2.476973	192.168.43.65	49.44.114.18	TCP	54	64440 → 443 [ACK] Seq=1 Ack=1 Win=16384 Len=0

Figure 7: Question 3
TCP Handshake

(a). **play** - When we first start the video, a three way TCP handshake ensues. First, the host who wants to stream the live video sends SYN packets to the server. The server then replies with SYN, ACK packets. As a final step of this TCP handshake, the host again replies with an ACK and the connection is set. After this TCP handshake, a TLS handshake takes place, which begins with a number of CLient Hello and Server Hello messages and uses Certificate exchanges as well. A client Key Exchange takes place and a Server Hello Done completes the TLS Handshake.

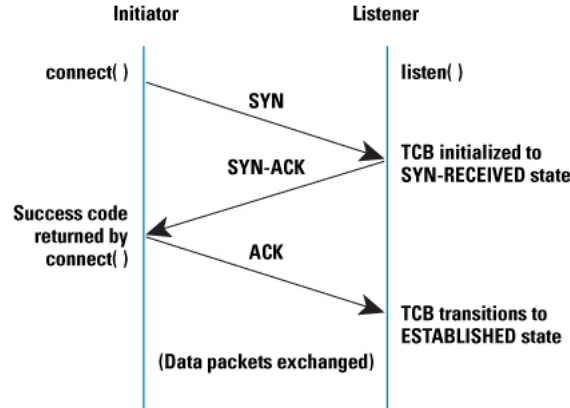


Figure 8: Three-Way TCP Handshake
The sequence of messages arriving from the TCP Handshake

(b). **pause** - Pausing a connection causes a stream of Keep Alive packets to be transmitted. Keep-alives can be used to verify that the computer at the remote end of a connection is still available. TCP keep-alives can be sent once every KeepAliveTime.

1770	12.635147	2400:cb00:2048:1::...	2405:204:b204:3863::...	TLSv1.2	1294	Application Data [TCP segment of a reassembled PDU]
1771	12.635312	2405:204:b204:3863::...	2400:cb00:2048:1::6...	TCP	74	59971 → 443 [ACK] Seq=1107 Ack=9522 Win=16896 Len=0
1772	12.682171	34.214.37.134	192.168.43.65	TCP	54	443 → 59918 [FIN, ACK] Seq=1 Ack=2 Win=27136 Len=0
1773	12.682470	192.168.43.65	34.214.37.134	TCP	54	59918 → 443 [ACK] Seq=2 Ack=2 Win=16384 Len=0
1774	12.683120	34.214.37.134	192.168.43.65	TCP	54	443 → 59919 [FIN, ACK] Seq=1 Ack=2 Win=27136 Len=0
1775	12.683124	49.44.100.145	192.168.43.65	TCP	1424	443 → 59974 [ACK] Seq=22017 Ack=1075 Win=31360 Len=1370 [TCP segment of a reassembled PDU]
1776	12.683385	192.168.43.65	34.214.37.134	TCP	54	59919 → 443 [ACK] Seq=2 Ack=2 Win=16384 Len=0

Figure 9: Question 3
FIN Handshake

(c). **stop** - On closing down the connection, FIN packets are used to orderly close down the connection. The host sends FIN packets to the server and then waits for an ACK and a FIN as a reply. It then sends a final ACK to close down the connection. This is thus a way Handshake If the ACK does not arrive before timeout, then it sends a RST packets for immediately tearing down the connection without waiting for an ACK.

Question 4

(a). **UDP DNS** - The Domain Name System or DNS packets finds out the IP Address for a particular website. In effect, a DNS lookup is needed to resolve a website address, or

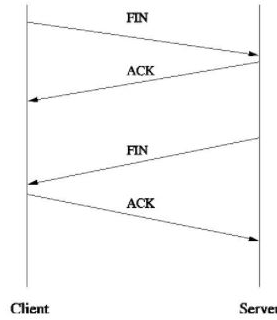


Figure 10: Four-Way FIN Handshake
The sequence of messages arriving from the FIN Handshake

a domain name, into an IP Address so our computer can access the website. Without the DNS, we would have to type in the IP address of the website we want to access and seeing as it would be cumbersome to remember the IP Addresses of all the websites, DNS is very important.

(b). TCP Data Packets - The TCP data packets are the ones that actually exchange the data that enables us to see the live match streaming. Without the TCP data packets, no data exchange would take place and thus it can be said to be the most important part of our data connection.

(c). TLS Packets - SSL is a computer networking protocol for securing connections between network application clients and servers. TLS, or transport layer security is just an updated and more secure form of SSL. So, TLS and SSL are basically security protocols without which we will be exposed to MITM(Man-in-the-middle) and other forms of cyber attacks.

(d). ARP Packets - ARP recovers hosts MAC or physical addresses from an IP address. Our system maintains a table which maps IP Addresses to MAC Addresses. If it does not save the MAC address, it has to repeatedly broadcast the message rather than sending packets only to our specific receiver.

Question 5

File	Throughput	RTT	Packet Size	Packets Lost	UDP Packets	TCP Packets	Ratio
data2	271308.344 bits/s	183.171 ms	930 bytes	0 %	178	13610	2.989
data3	281461.222 bits/s	144.073 ms	1038 bytes	0 %	12	9508	2.869
data4	241874.347 bits/s	117.812 ms	849 bytes	0 %	187	17505	2.259

Question 6

In one of my data collection sessions, I found that I was receiving relevant data from two sources, 40.44.114.11 and 49.44.114.18.

(a). The primary reason for this can be load balancing. When one server starts receiving a lot of requests, it transfers some of its users to the other servers. This may be one of the primary reasons of having different sources.

(b). Websites have to deal with a number of requests from a lot of servers. So, instead of one big server which can satisfy all the incoming requests, a number of servers is much cheaper. This is one of the reasons why multiple servers are used.

(c). Another reason for multiple servers is that if one of the servers fail, it would not impact the overall performance and thus websites prefer to operate on multiple servers.