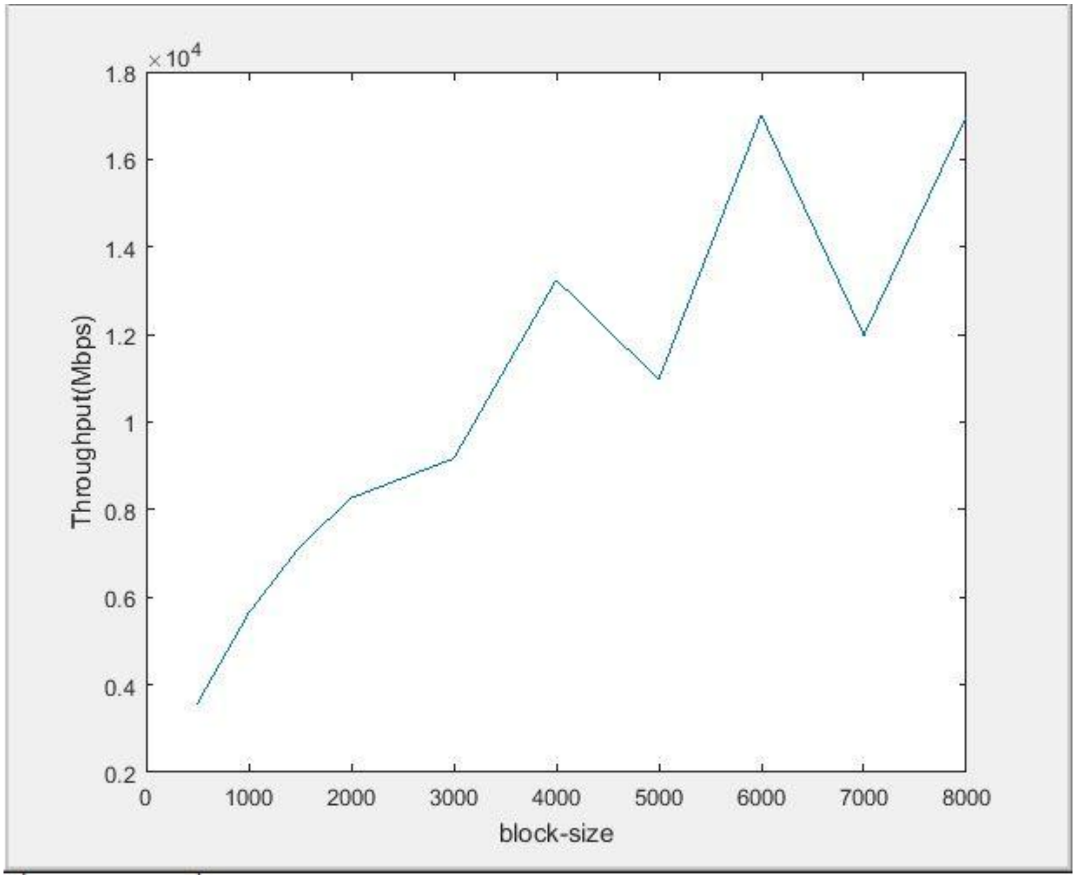
LAB 3 – Sri Yogesh Dorbala - 0029086240

1) Throughput increases with the increase in block size. There is no definite block size for which the speed is optimal. As observed from the graph, throughput keeps on increasing with block size. Below readings are taken by calculating an average of 8-10 operations on each block size (hence the random peaks at some stages).

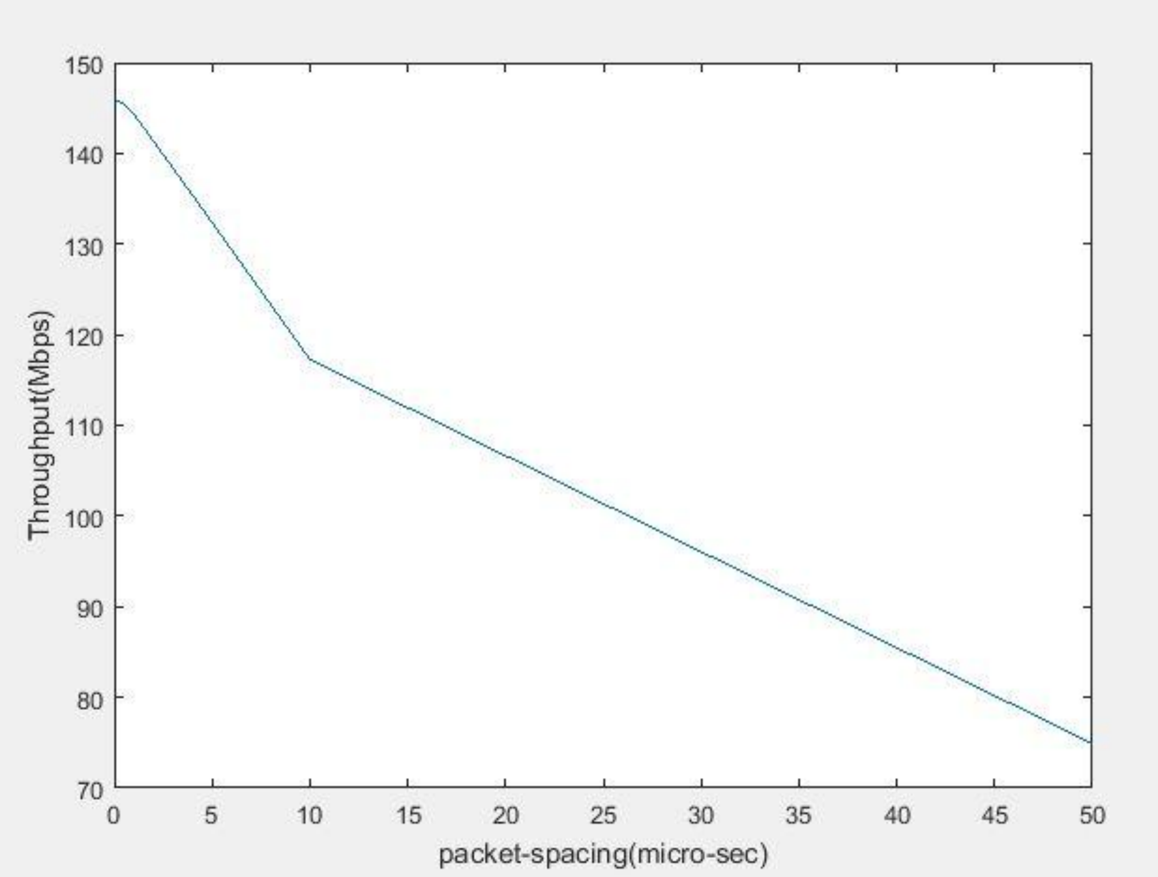
Block Size	Throughput(Mbps)	Completion Time (micro secs)
500	3572	6490
1000	5634	4115
1500	7156	3240
2000	8266	2805
3000	9168	2529
4000	13249	1750
5000	10973	2113
6000	17011	1363
7000	11982	1935
8000	16986	1365



The above results may vary at different times as there are many other minute factors involved in network speeds. Though there are a few fluctuations here and there, it is almost certain that throughput value keeps on increasing with increase in block size.

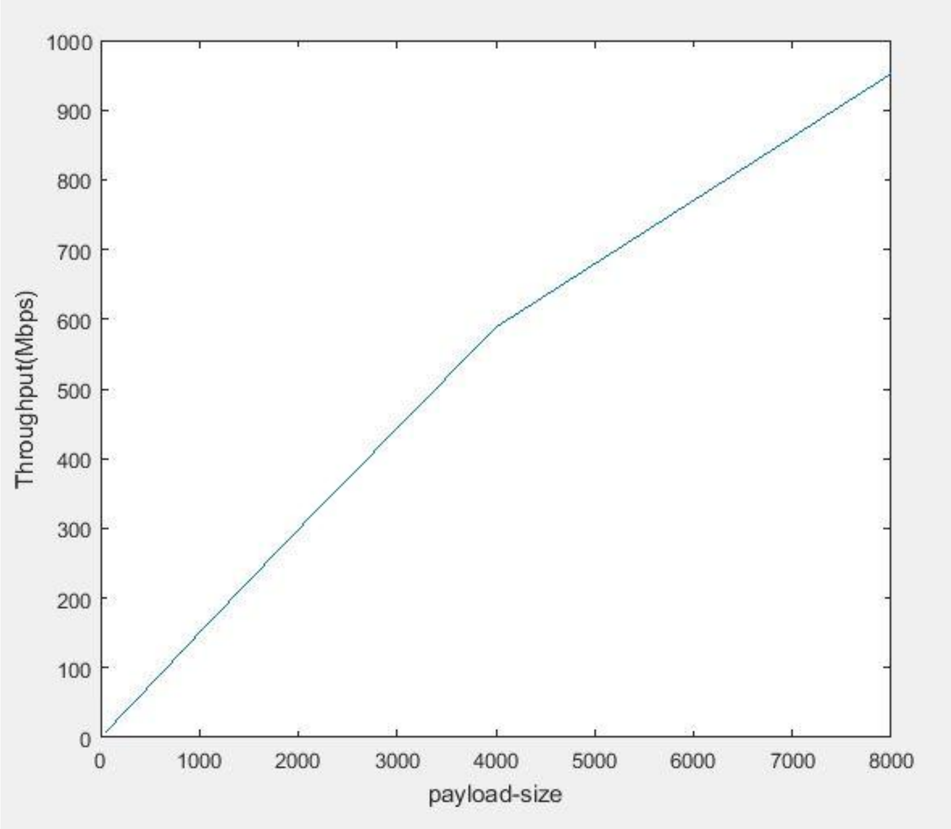
2) Throughput observed at both the sender and the receiver are the same

Packet-Spacing(micro-sec)	Sender(Throughput-Mbps)	Receiver(Throughput-Mbps)
50	74.892	74.856
10	117.245	117.235
5	132.38	132.256
1	144.345	144.015
0.5	145.471	145.239
0.1	145.823	145.536



As packet spacing is reduced from 50 to 0.1 micro seconds throughput value increases from 74 to 145 Mbps as shown both from the graph and the table. This is expected because when the time spacing between two consecutive packets decrease, the completion time decreases and therefore the throughput increases.

Payload-Size	Sender(Throughput-Mbps)
50	7.456
500	74.563
1000	150.239
2000	298.481
4000	588.439
8000	952.231



As the payload size increases, more number of bits are carried per transaction. This makes the time taken to complete the file transfer to reduce. This drop in completion time results in a better throughput yield.

3) After capturing the packets, the *pcap* file generated can be viewed in a friendly manner by using Wireshark. One of the 10 packets that got captured is shown below after expanding in Wireshark utility.

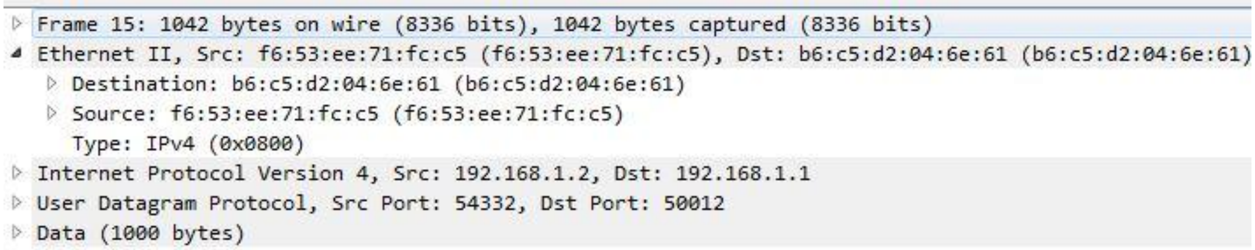
An Ethernet frame has in order: 6-octet destination MAC address (which from the pic below is b6:c5:d2:04:6e:61), 6-octet source MAC address(f6:53:ee:71:fc:c5) followed by a 2-byte field which in the pic shown below is '*08 00'* which is used to denote IPv4 type.

This *type* value right after source and destination MAC addresses helps in distinguishing the type of Ethernet frame at hand. If the value of the type is less than 1500 then it represents 'IEEE 802.3', else it represents 'DIX' (the 2-bit value in IEEE 802.3 is length field which stores the length of the payload. Since the payload length for an Ethernet frame cannot be greater than 1500, the 2-bit value after source and destination MAC addresses in IEEE 802.3 cannot be greater than 1500). As the value here is hex *'08 00'* which represents type IPv4(which is greater than 1500), the Ethernet frame is of type DIX here.

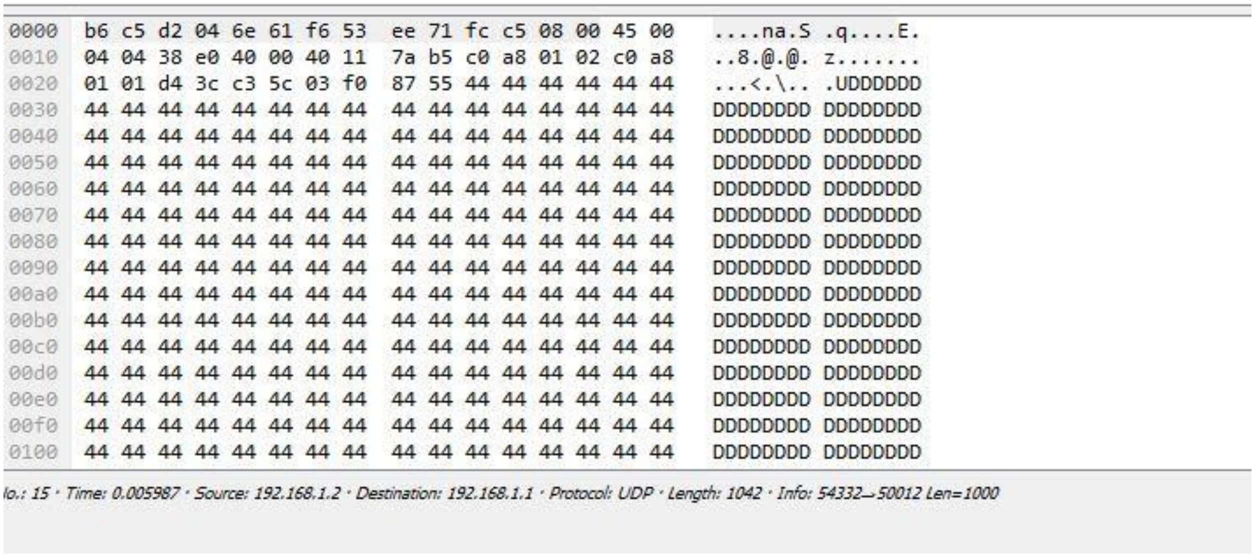
The first 4 bits of the IP packet header in this example after '08 00' is '01 00' denoted by 4. This informs us that the version of the IP is 4.

From the below division, we can get to the start of the UDP packet header. The first 4 bytes specify the source port ('d4 3c': 54332) and the destination port ('c3 5c': 50012). Once the UDP packet header is parsed we are left with payload information which in the below case are all represented by blocks of 4's (which is the hex for 'D') having a length of 1000 as shown below.

Wireshark utility helped discern the information about the different headers involved in the packet.



```
▷ Frame 15: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits)
◢ Ethernet II, Src: f6:53:ee:71:fc:c5 (f6:53:ee:71:fc:c5), Dst: b6:c5:d2:04:6e:61 (b6:c5:d2:04:6e:61)
   ▷ Destination: b6:c5:d2:04:6e:61 (b6:c5:d2:04:6e:61)
   ▷ Source: f6:53:ee:71:fc:c5 (f6:53:ee:71:fc:c5)
     Type: IPv4 (0x0800)
▷ Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.1
▷ User Datagram Protocol, Src Port: 54332, Dst Port: 50012
▷ Data (1000 bytes)
```



```
0000  b6 c5 d2 04 6e 61 f6 53  ee 71 fc c5 08 00 45 00   ....na.S .q....E.
0010  04 04 38 e0 40 00 40 11  7a b5 c0 a8 01 02 c0 a8   ..8.@.@. z.......
0020  01 01 d4 3c c3 5c 03 f0  87 55 44 44 44 44 44 44   ...<.\.. .UDDDDDD
0030  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
0040  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
0050  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
0060  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
0070  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
0080  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
0090  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
00a0  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
00b0  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
00c0  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
00d0  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
00e0  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
00f0  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
0100  44 44 44 44 44 44 44 44  44 44 44 44 44 44 44 44   DDDDDDDD DDDDDDDD
```

```
lo.: 15 · Time: 0.005987 · Source: 192.168.1.2 · Destination: 192.168.1.1 · Protocol: UDP · Length: 1042 · Info: 54332→50012 Len=1000
```