Problem 1

In this question, a client and server have been executed in the same machine. The packets are sent across client and server using the ping app (Problem 1, Lab 2).

The following are the details of the packets sent –

SERVER :-

Private IP – 192.168.1.1 , Port Used by the ping app - 55555

MAC Address – 62:b5:6d:a1:0d:bb

```
veth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.1  netmask 255.255.255.0  broadcast 192.1
68.1.255
        inet6 fe80::60b5:6dff:fea1:dbb  prefixlen 64  scopeid 0x
20<link>
        ether 62:b5:6d:a1:0d:bb  txqueuelen 1000  (Ethernet)
        RX packets 2161  bytes 137918 (137.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3967  bytes 505259 (505.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions
 0
```

CLIENT :-

Private IP – 192.168.1.2 , port used by the ping app - 36250

MAC Address – 06:c2:5f:25:ea:64

```
veth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.1
68.1.255
        inet6 fe80::4c2:5fff:fe25:ea64  prefixlen 64  scopeid 0x
20<link>
        ether 06:c2:5f:25:ea:64  txqueuelen 1000  (Ethernet)
        RX packets 3967  bytes 505259 (505.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2161  bytes 137918 (137.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions
 0
```

The following image depicts the Manner in which packets were sent and received using the ping application (Problem 1 Lab 2) –

```
pod3-3 53 $ veth "./mypingcli.bin 192.168.1.2 192.168.1.1 55555"
ping with packet : 1234
ping with packet : 1235
Received a packetReceived Packet 1234 from server
Round Trip Time for the packet 1234 is 1000.000000
ping with packet : 1236
Received a packetReceived Packet 1235 from server
Round Trip Time for the packet 1235 is 1000.000000
ping with packet : 1237
Received a packetReceived Packet 1236 from server
Round Trip Time for the packet 1236 is 1000.000000
ping with packet : 1238
Received a packetReceived Packet 1237 from server
Round Trip Time for the packet 1237 is 1000.000000
ping with packet : 1239
Received a packetReceived Packet 1238 from server
Round Trip Time for the packet 1238 is 1000.000000
ping with packet : 1240
Received a packetReceived Packet 1239 from server
Round Trip Time for the packet 1239 is 1000.000000
```

Analysis of the log file obtained by sniffing these packets. (please find a snippet of the log file below)

For the first packet (packet sent by client and received by the server)

```
data 71 $ xxd testlogfile
00000000: d4c3 b2a1 0200 0400 0000 0000 0000 0000  ................
00000010: 0000 0400 0100 0000 2423 6a61 ff5b 0a00  ........$#ja.[..
00000020: 2f00 0000 2f00 0000 62b5 6da1 0dbb 06c2  /.../...b.m.....
00000030: 5f25 ea64 0800 4500 0021 2c8c 4000 4011  _%.d..E..!,.@.@.
00000040: 8aec c0a8 0102 c0a8 0101 8d9a d903 000d  ................
00000050: 8372 3132 3334 0125 236a 617a 580a 002f  .r1234.%#jazX../
00000060: 0000 002f 0000 0062 b56d a10d bb06 c25f  .../...b.m.....
00000070: 25ea 6408 0045 0000 212d 2a40 0040 118a  %.d..E..!-*@.@..
00000080: 4ec0 a801 02c0 a801 018d 9ad9 0300 0d83  N...............
00000090: 7231 3233 3501 2523 6a61 cf5c 0a00 2e00  r1235.%#ja.\....
000000a0: 0000 2e00 0000 06c2 5f25 ea64 62b5 6da1  ........_%.db.m.
000000b0: 0dbb 0800 4500 0020 332f 4000 4011 844a  ....E.. 3/@.@..J
000000c0: c0a8 0101 c0a8 0102 d903 8d9a 000c 8371  ...............q
000000d0: 3132 3334 2623 6a61 7958 0a00 2f00 0000  1234&#jayX../...
000000e0: 2f00 0000 62b5 6da1 0dbb 06c2 5f25 ea64  /...b.m....._%.d
000000f0: 0800 4500 0021 2dd9 4000 4011 899f c0a8  ..E..!-.@.@.....
00000100: 0102 c0a8 0101 8d9a d903 000d 8372 3132  .............r12
00000110: 3336 0126 236a 612e 590a 002e 0000 002e  36.&#ja.Y.......
00000120: 0000 0006 c25f 25ea 6462 b56d a10d bb08  ....._%.db.m....
00000130: 0045 0000 2033 eb40 0040 1183 8ec0 a801  .E.. 3.@.@......
00000140: 01c0 a801 02d9 038d 9a00 0c83 7131 3233  ............q123
00000150: 3527 236a 617b 580a 002f 0000 002f 0000  5'#ja{X../.../..
```

From the file, we can see that for sending the packet with the data 1234 from source - client to destination - server,

The **destination address** (highlighted in green) is **62:b5:6d:a1:0d:bb**, which matches the MAC address of the server (indicated above).
The **source address** (highlighted in red) is **06:c2:5f:25:ea:64**, which matches the MAC address of the client (indicated above).

The next 2 bytes (highlighted in blue) indicate the type field which confirms that the frame is a **DIX frame**.

The segment highlighted in purple is the application layer payload that the mypingcli program has sent. This corresponds to 12341 (1234 and 1 which is appended for time delay).

The first 4 bytes of the UDP header are highlighted in yellow.
The hex value of the first 2 bytes – **8d9a** corresponds to the **client port** (decimal value - **36250**).
The hex value of the next 2 bytes – **d903** corresponds to the **server port** (decimal value - **55555**).

The IP segment of the next frame starts from where there is a yellow line in the image. We can notice here that there is no CRC field available in the ethernet frame.

To verify the findings, the following image shows the analysis by tcpdump. Manual inspection and tcpdump analysis give same results.

```
pod3-3 68 $ tcpdump -r - < testlogfile
reading from file -, link-type EN10MB (Ethernet)
20:56:04.678911 IP remote.36250 > host.55555: UDP, length 5
20:56:05.678010 IP remote.36250 > host.55555: UDP, length 5
20:56:05.679119 IP host.55555 > remote.36250: UDP, length 4
20:56:06.678009 IP remote.36250 > host.55555: UDP, length 5
20:56:06.678190 IP host.55555 > remote.36250: UDP, length 4
20:56:07.678011 IP remote.36250 > host.55555: UDP, length 5
20:56:07.678188 IP host.55555 > remote.36250: UDP, length 4
20:56:08.678011 IP remote.36250 > host.55555: UDP, length 5
20:56:08.678184 IP host.55555 > remote.36250: UDP, length 4
20:56:09.678010 IP remote.36250 > host.55555: UDP, length 5
20:56:09.678185 IP host.55555 > remote.36250: UDP, length 4
20:56:09.782174 ARP, Request who-has host tell remote, length
 28
```

Problem 2

Performance with respect to file size :-

In the same lab –

Category	Block Size	File Size	Time Taken (milliseconds)	Throughput (Megabits/seconds)
Small Files (tens of KB)	512 bytes	17.4 KB	3 msec	46 Mbps
	1024 bytes	17.4 KB	2 msec	69 Mbps
	2048 bytes	17.4 KB	2 msec	69 Mbps
	4096 bytes	17.4 KB	1 msec	139 Mbps
Large Files (tens of MB)	512 bytes	30.116 MB	276 msec	893.85 Mbps
	1024 bytes	30.116 MB	276 msec	893.85 Mbps
	2048 bytes	30.116 MB	276 msec	893.85 Mbps
	4096 bytes	30.116 MB	273 msec	903.67 Mbps

In different labs –

Category	Block Size	File Size	Time Taken (milliseconds)	Throughput (Megabits/seconds)
Small Files (tens of KB)	512 bytes	17.4 KB	3 msec	46 Mbps
	1024 bytes	17.4 KB	2 msec	69 Mbps
	2048 bytes	17.4 KB	2 msec	69 Mbps
	4096 bytes	17.4 KB	2 msec	69 Mbps
Large Files (tens of MB)	512 bytes	30.116 MB	279 msec	884.24 Mbps
	1024 bytes	30.116 MB	297 msec	830.65 Mbps
	2048 bytes	30.116 MB	278 msec	887.42 Mbps
	4096 bytes	30.116 MB	277 msec	892.32 Mbps

As it can be seen, the performance becomes better on increasing the block size. Also, the throughput is better I the same lab. In different labs, the throughput becomes slightly lower. In small files there is not much of a difference with the block size because the number of IO operations is anyways less for small files.

Problem 3

Performance of **large file (30.116 MB)** on **different lab** machines in different labs when placed in tmp folder is –
Round Trip Time – 274 msec
Throughput – 900 Mbps

Performance of **large file (30.116 MB)** on **same lab** machines in same lab when placed in tmp folder is –
Round Trip Time – 264 msec
Throughput – 910 Mbps

The performance is slightly better in this case because the file to be copied is cached and it is faster to fetch such files.

Therefore, on placing the file in tmp folder, the copying becomes faster.