Assignment 1 Networks Lab(CS39006) Spring 2020

Submitted By -Group 18

Prakhar Bindal-17CS10036

Gaurav Goyal-17CS30013

Problem Statement : Using Wireshark for analysing network packet traces

1) **Steps-**

1) UDP Client (with 28 Kbps) for UDP performance measurement using iperf :
iperf -c 10.5.18.163 -u -b 28000
This command will send UDP packets to the iperf server running at 10.5.18.163. You can specify the time for which you want to send UDP packets. Please check iperf documentation.
2) TCP Client:
wget --no-proxy http://10.5.18.163:8000/1.jpg
This command uses the HTTP protocol to download the image file 1.jpg from the web server running
at 10.5.18.163. You can change the URI to access 1.jpg to 7.jpg.

Observation:

UDP Case:

Application Layer- NIL
Transport Layer- UDP
Network Layer- IP Version 4

TCP Case:

Application Layer- HTTP
Transport Layer- TCP
Network Layer- IP Version 4

Justification: In case of TCP, we used wget which is an application layer tool sending HTTP requests and iperf is a transport layer tool with -u flag specifying to send UDP packets.

2) a)

Steps:

1."ip.addr=10.5.18.163 && ip.addr=client_ip" is used in the filter to monitor only the Packets that are concerned without our experiment
2. Client_id found using the $ifconfig command
3. I/O graphs are obtained from wireshark menu Menu->Statistics->IO Graphs

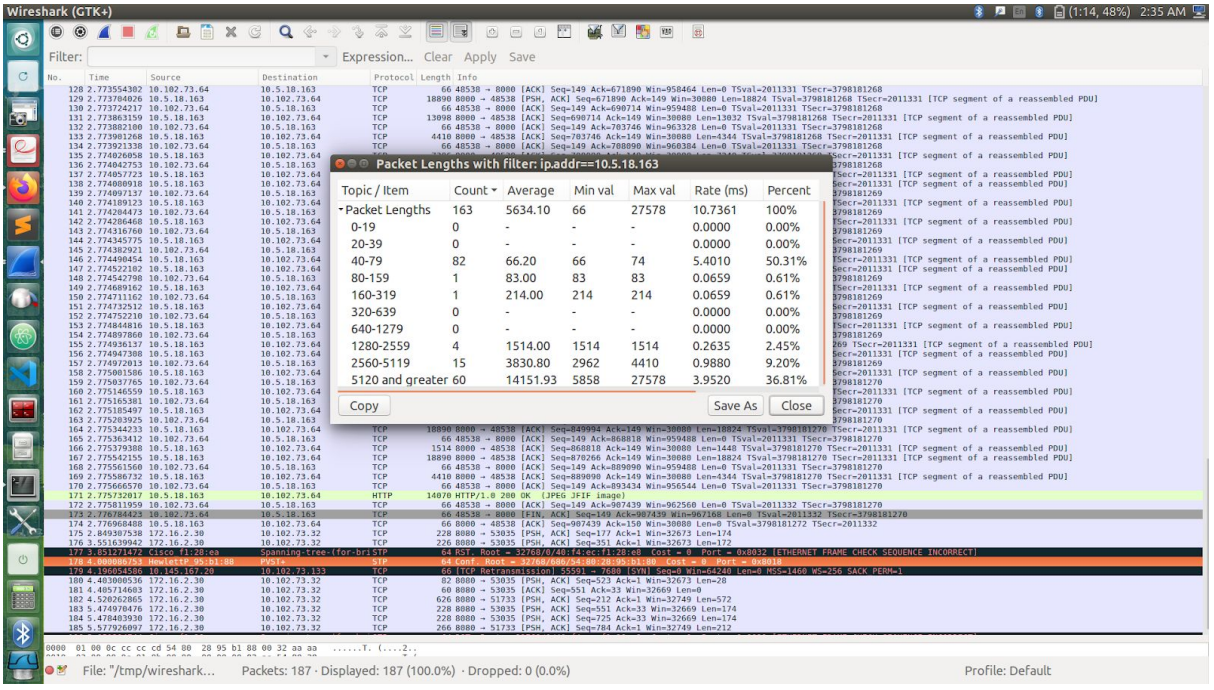
Observation: In all the cases 2 SYN and ACK Packets are observed at the beginning. 1 HTTP "OK" packet and 1 TCP "OK" packet were observed at the end. There was an ACK packet after each of the received packets.

Pic1-186 data packets
Pic2- 621 data packets
Pic3- 1362 data packets
Pic4- 845 data packets
Pic5- 855 data packets

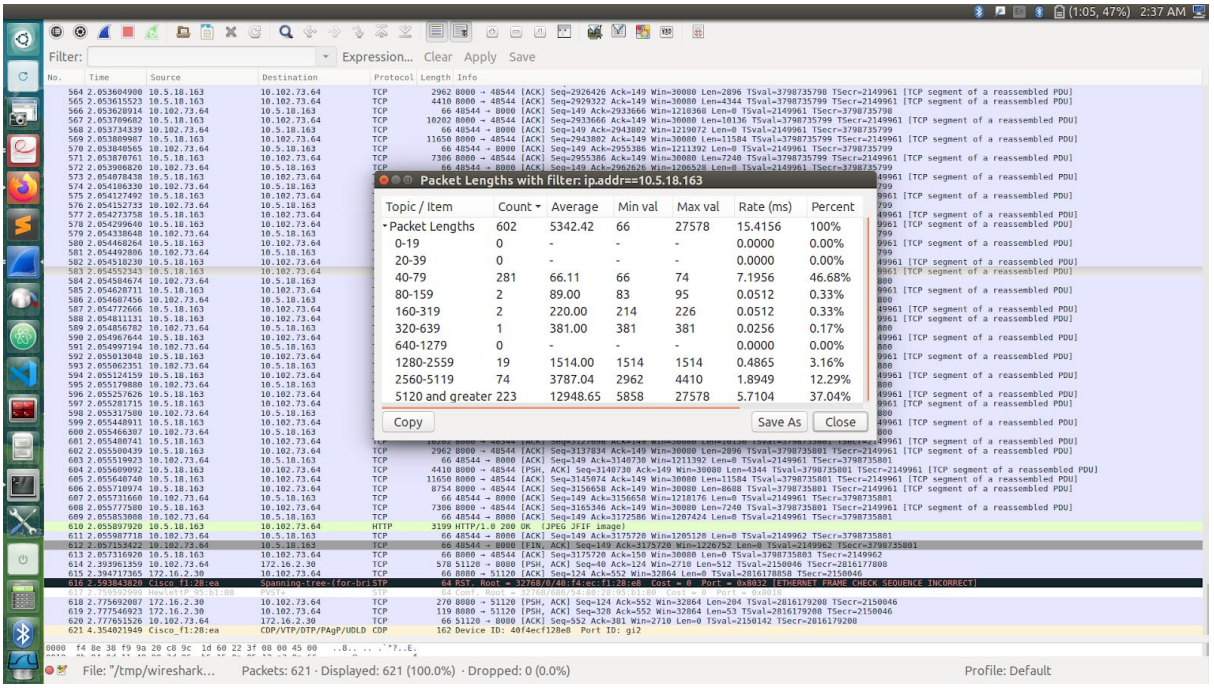
Justification: Since the pictures are of different sizes and are using a TCP protocol,number of data packets are different in all cases.
No,all the packets are not of same size and there were various sizes ranging from few 60's to a few thousands.Generally the ACK packets are of less size than data packets.
Some packet sizes for each of the pic in bytes are:
Pic1-60,66,82,228,252,1514,18890 etc
Pic2-60,66,119,270,7386,8654,11297 etc
Pic3-66,73016,9472,18654 etc
Pic4-66,119,2962,3412,4572 etc
Pic5-66,105,119,136,2964,4410 etc

Details-

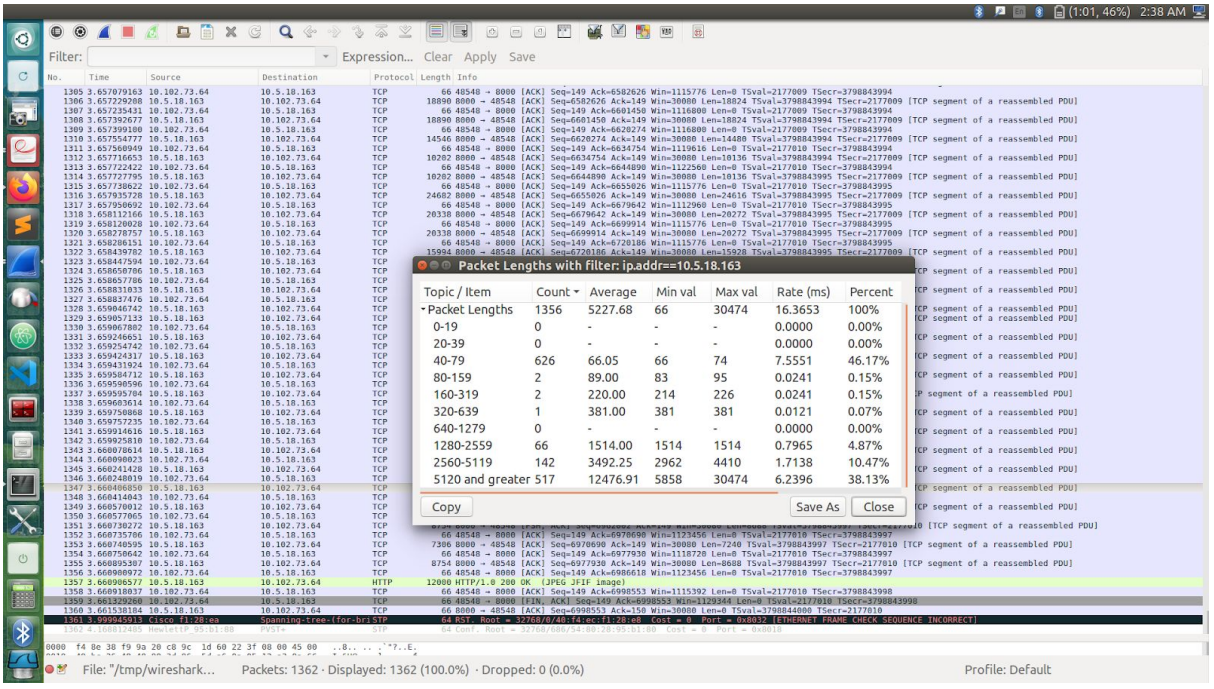
Pic-1



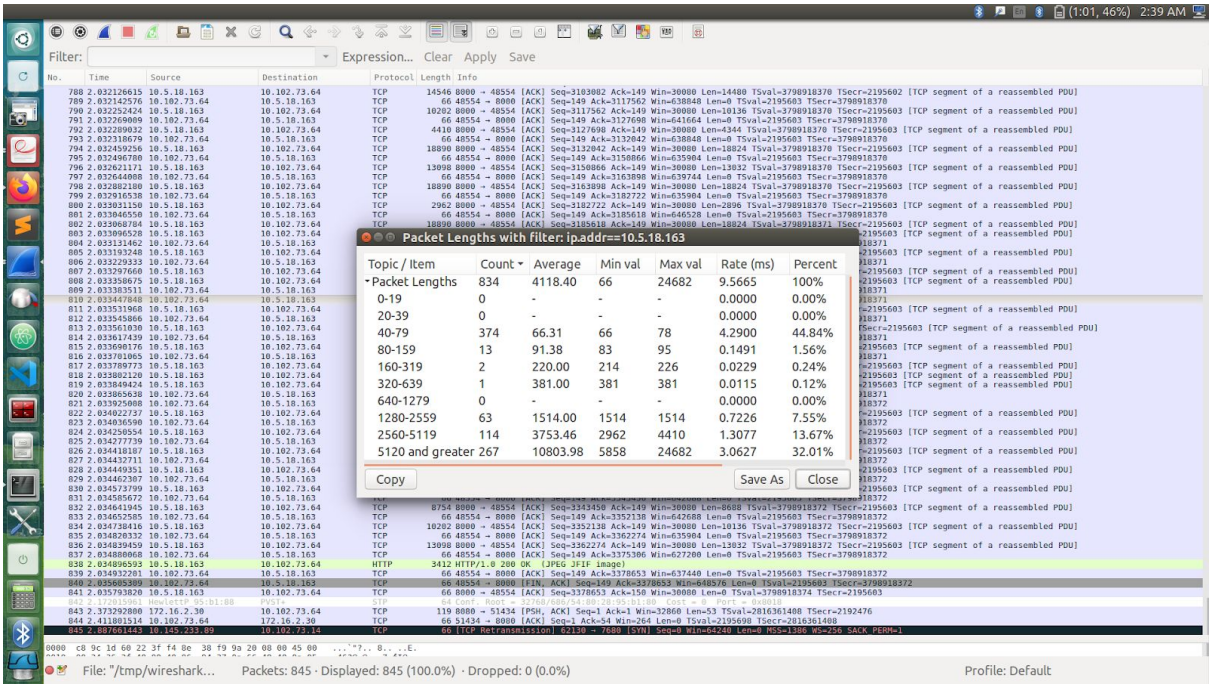
Pic-2


Filter:
Expression... Clear Apply Save
Packet Lengths with filter: ip.addr==10.5.18.163
Topic / Item | Count | Average | Min val | Max val | Rate (ms) | Percent
Packet Lengths | 602 | 5342.42 | 66 | 27578 | 15.4156 | 100%
0-19 | 0 | - | - | - | 0.0000 | 0.00%
20-39 | 0 | - | - | - | 0.0000 | 0.00%
40-79 | 281 | 66.11 | 66 | 74 | 7.1956 | 46.68%
80-159 | 2 | 89.00 | 83 | 95 | 0.0512 | 0.33%
160-319 | 2 | 220.00 | 214 | 226 | 0.0512 | 0.33%
320-639 | 1 | 381.00 | 381 | 381 | 0.0256 | 0.17%
640-1279 | 0 | - | - | - | 0.0000 | 0.00%
1280-2559 | 19 | 1514.00 | 1514 | 1514 | 0.4865 | 3.16%
2560-5119 | 74 | 3787.04 | 2962 | 4410 | 1.8949 | 12.29%
5120 and greater | 223 | 12948.65 | 5858 | 27578 | 5.7104 | 37.04%
Copy
Save As
Close
File: "/tmp/wireshark...
Packets: 621 · Displayed: 621 (100.0%) · Dropped: 0 (0.0%)
Profile: Default

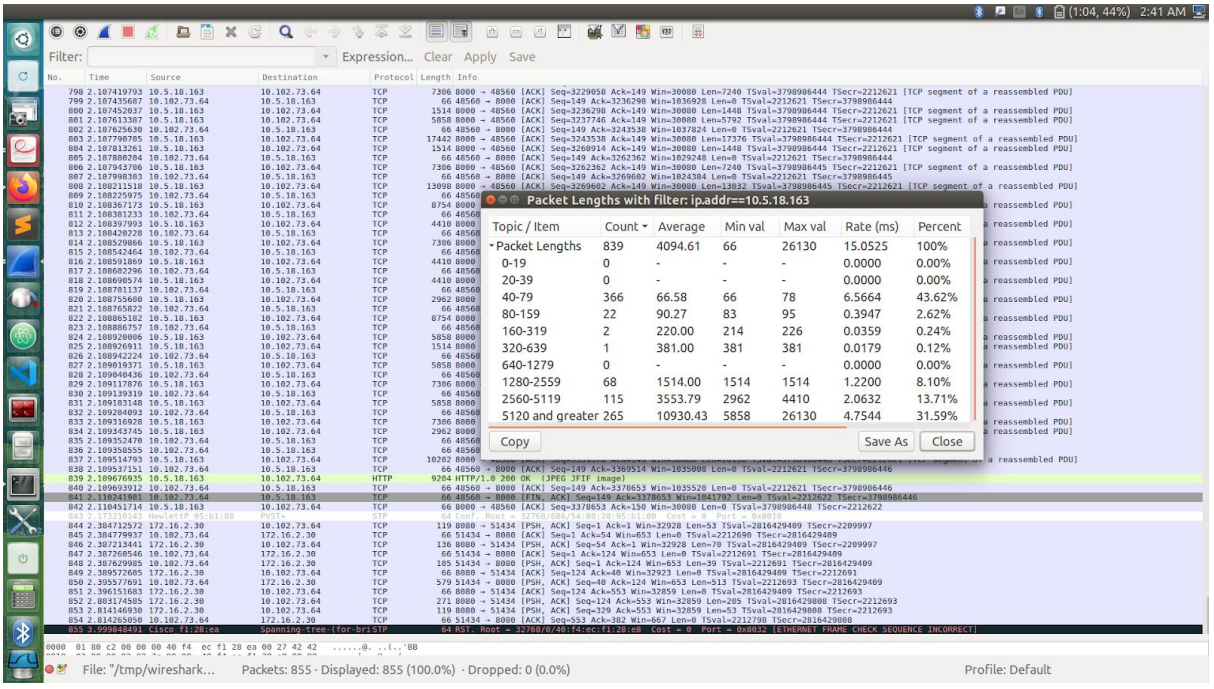
Pic-3


Filter:
Expression... Clear Apply Save
Packet Lengths with filter: ip.addr==10.5.18.163
Topic / Item | Count | Average | Min val | Max val | Rate (ms) | Percent
Packet Lengths | 1356 | 5227.68 | 66 | 30474 | 16.3653 | 100%
0-19 | 0 | - | - | - | 0.0000 | 0.00%
20-39 | 0 | - | - | - | 0.0000 | 0.00%
40-79 | 626 | 66.05 | 66 | 74 | 7.5551 | 46.17%
80-159 | 2 | 89.00 | 83 | 95 | 0.0241 | 0.15%
160-319 | 2 | 220.00 | 214 | 226 | 0.0241 | 0.15%
320-639 | 1 | 381.00 | 381 | 381 | 0.0121 | 0.07%
640-1279 | 0 | - | - | - | 0.0000 | 0.00%
1280-2559 | 66 | 1514.00 | 1514 | 1514 | 0.7965 | 4.87%
2560-5119 | 142 | 3492.25 | 2962 | 4410 | 1.7138 | 10.47%
5120 and greater | 517 | 12476.91 | 5858 | 30474 | 6.2396 | 38.13%
Copy
Save As
Close
File: "/tmp/wireshark...
Packets: 1362 · Displayed: 1362 (100.0%) · Dropped: 0 (0.0%)
Profile: Default

Pic-4


Packet Lengths with filter: ip.addr==10.5.18.163
Topic / Item | Count | Average | Min val | Max val | Rate (ms) | Percent
Packet Lengths | 834 | 4118.40 | 66 | 24682 | 9.5665 | 100%
0-19 | 0 | - | - | - | 0.0000 | 0.00%
20-39 | 0 | - | - | - | 0.0000 | 0.00%
40-79 | 374 | 66.31 | 66 | 78 | 4.2900 | 44.84%
80-159 | 13 | 91.38 | 83 | 95 | 0.1491 | 1.56%
160-319 | 2 | 220.00 | 214 | 226 | 0.0229 | 0.24%
320-639 | 1 | 381.00 | 381 | 381 | 0.0115 | 0.12%
640-1279 | 0 | - | - | - | 0.0000 | 0.00%
1280-2559 | 63 | 1514.00 | 1514 | 1514 | 0.7226 | 7.55%
2560-5119 | 114 | 3753.46 | 2962 | 4410 | 1.3077 | 13.67%
5120 and greater | 267 | 10803.98 | 5858 | 24682 | 3.0627 | 32.01%
Copy
Save As
Close
Packets: 845 · Displayed: 845 (100.0%) · Dropped: 0 (0.0%)
Profile: Default

Pic-5

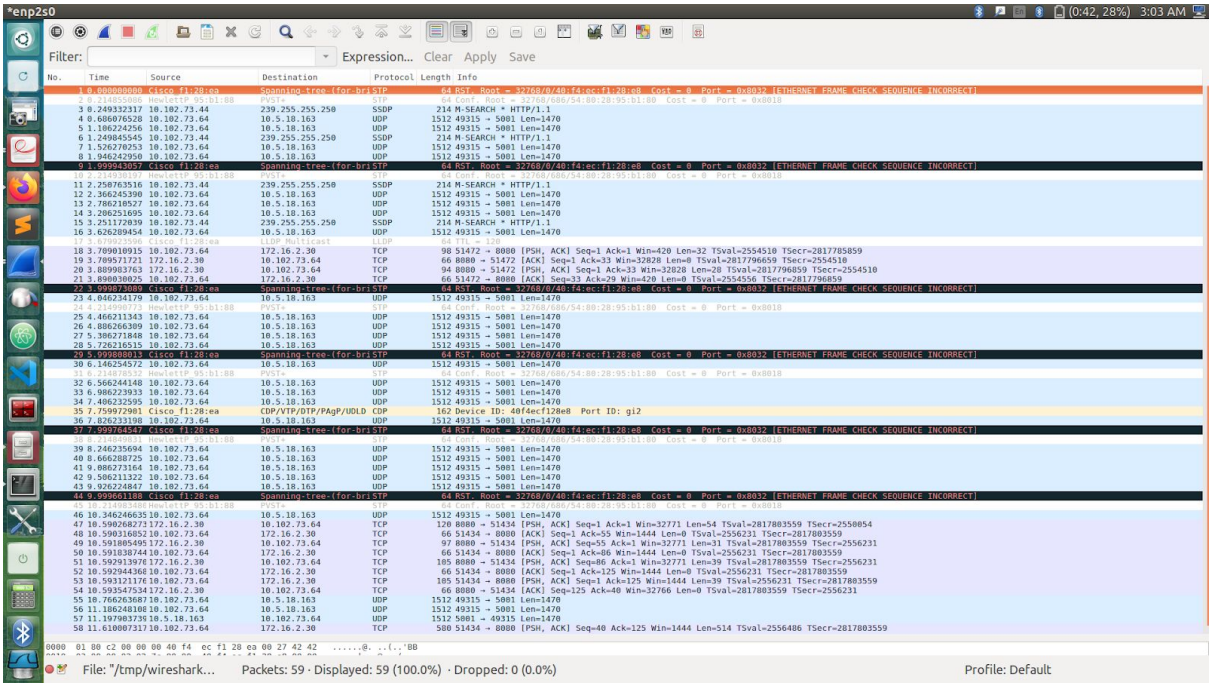

Packet Lengths with filter: ip.addr==10.5.18.163
Topic / Item | Count | Average | Min val | Max val | Rate (ms) | Percent
Packet Lengths | 839 | 4094.61 | 66 | 26130 | 15.0525 | 100%
0-19 | 0 | - | - | - | 0.0000 | 0.00%
20-39 | 0 | - | - | - | 0.0000 | 0.00%
40-79 | 366 | 66.58 | 66 | 78 | 6.5664 | 43.62%
80-159 | 22 | 90.27 | 83 | 95 | 0.3947 | 2.62%
160-319 | 2 | 220.00 | 214 | 226 | 0.0359 | 0.24%
320-639 | 1 | 381.00 | 381 | 381 | 0.0179 | 0.12%
640-1279 | 0 | - | - | - | 0.0000 | 0.00%
1280-2559 | 68 | 1514.00 | 1514 | 1514 | 1.2200 | 8.10%
2560-5119 | 115 | 3553.79 | 2962 | 4410 | 2.0632 | 13.71%
5120 and greater | 265 | 10930.43 | 5858 | 26130 | 4.7544 | 31.59%
Copy
Save As
Close
Packets: 855 · Displayed: 855 (100.0%) · Dropped: 0 (0.0%)
Profile: Default

2.b)

All the UDP Packets are of same size- 1512 bytes length per packet.

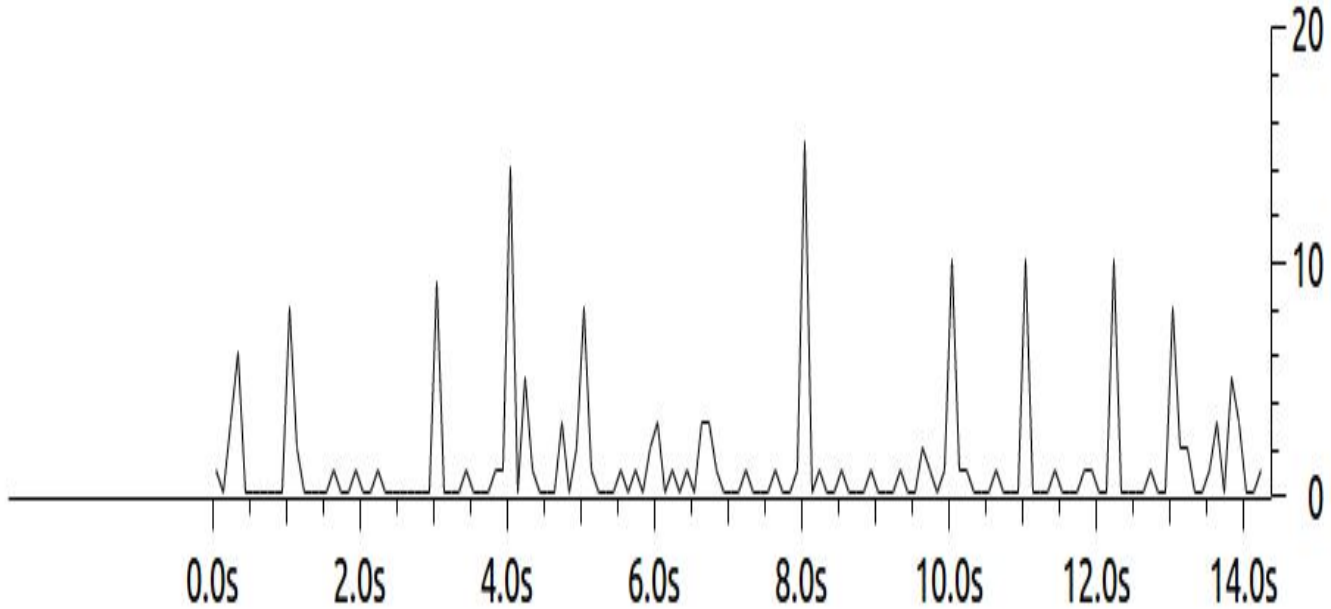
Justification-All packets are of same size because of the iperf server sending it that way.

Pic for Reference:

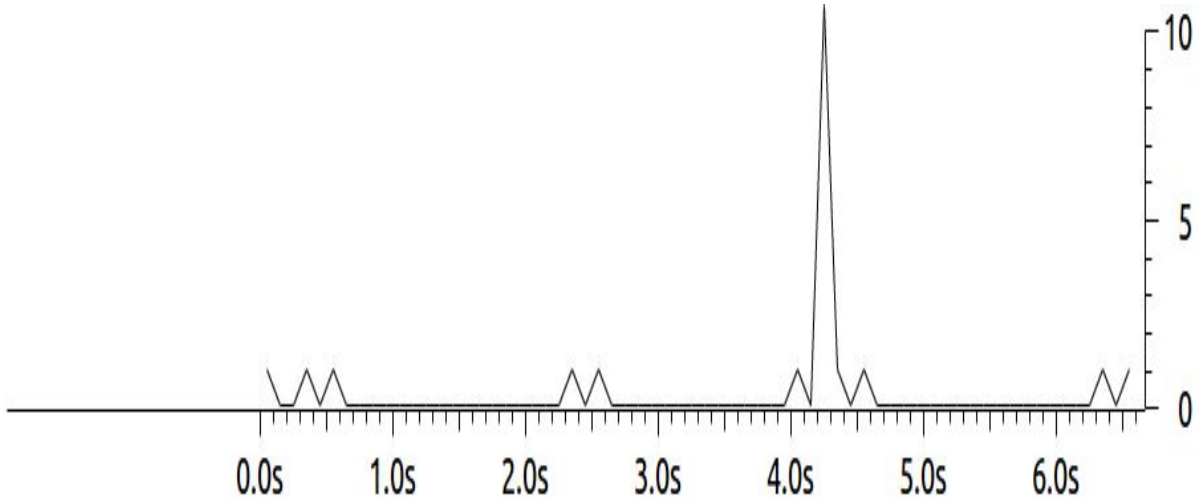


2.c) Throughout using Wireshark

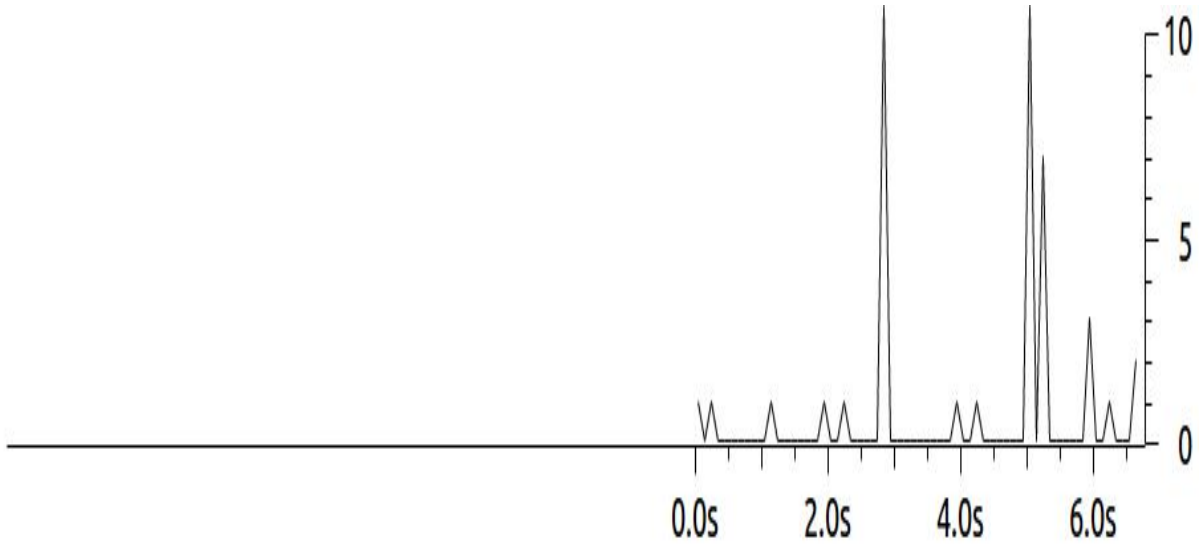
UDP Case:



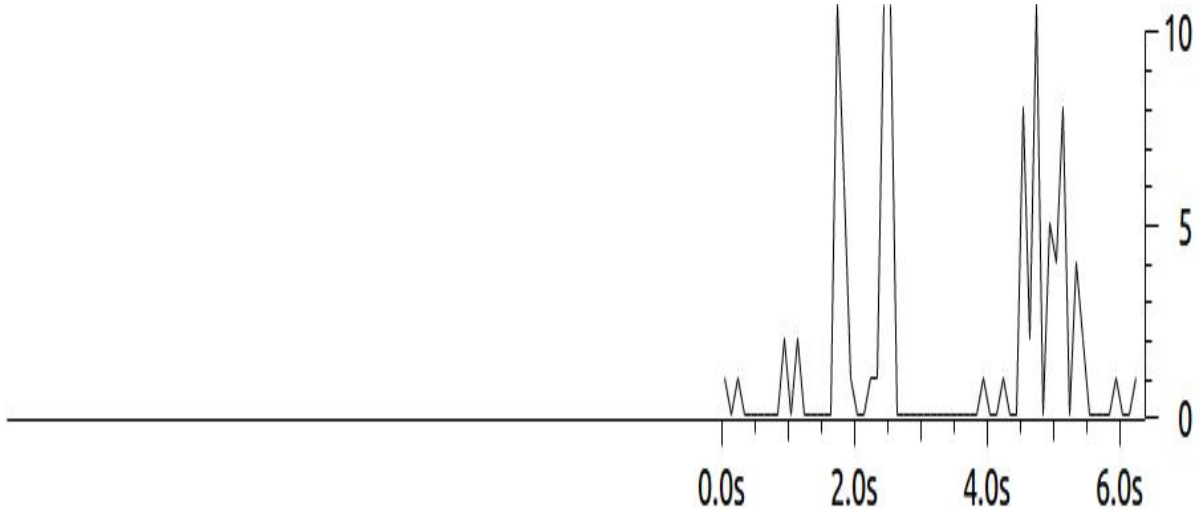
TCP Case:
Pic 1:



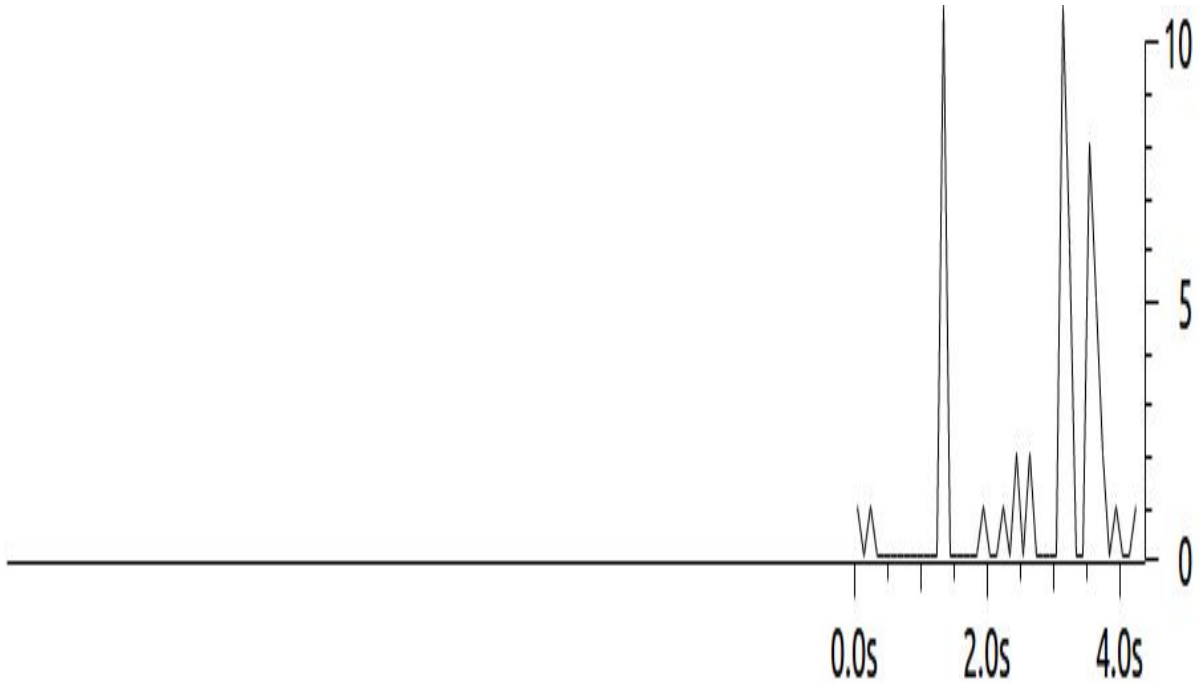
Pic 2:



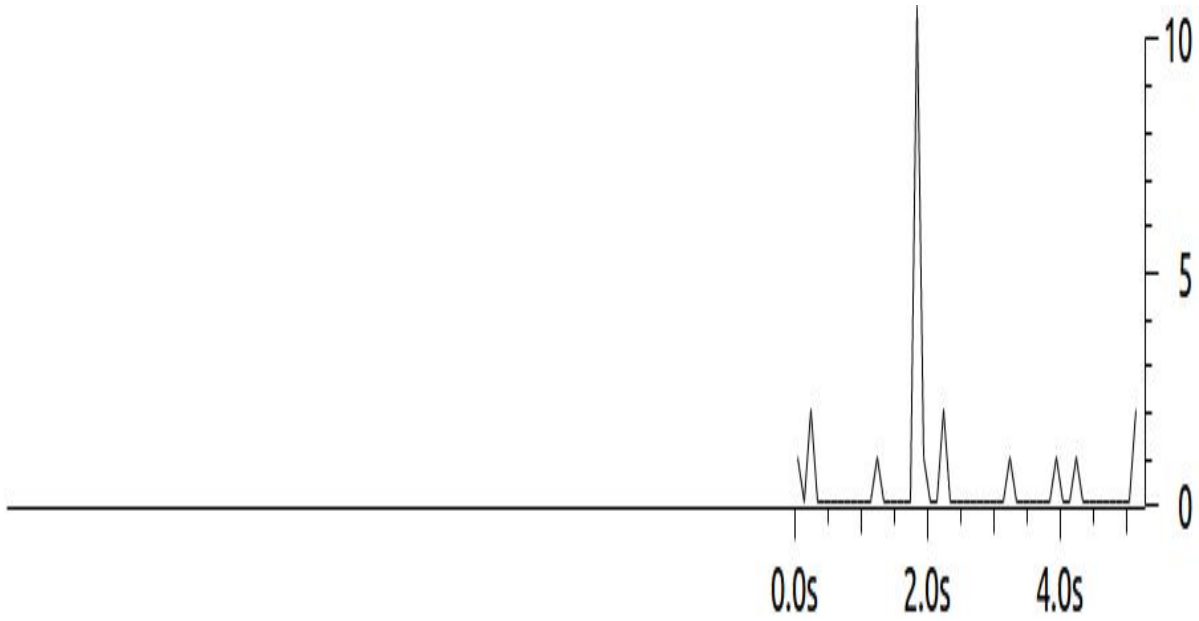
Pic 3:



Pic 4:



Pic 5:



Justification-Since,the request for all the pictures are sent one after the other,the set of packets for each image are received consecutively for 5 pictures. Since each picture is requested only after receiving the previous picture,each peak corresponds to one picture only.

d) The UDP throughout (amount of UDP data received per second) for the following cases of UDP traffic generation rates(bandwidth)

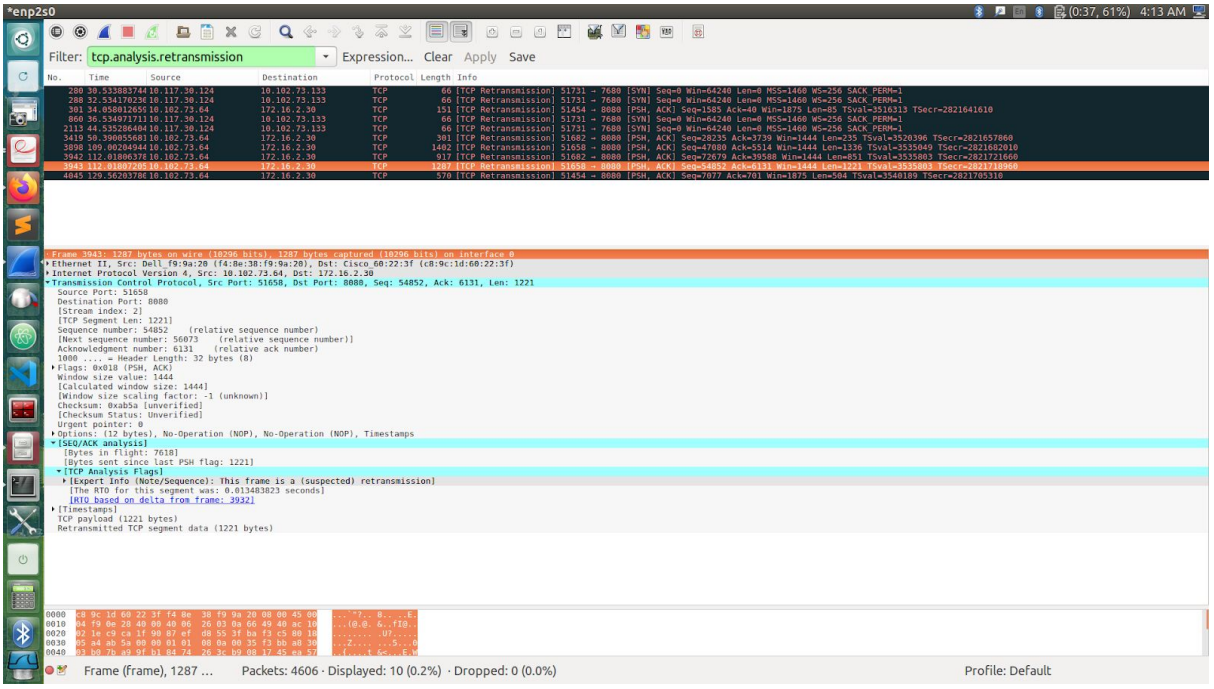
1. 64 Kbps
   Data Transfer=80.4 Kbytes
   Uplink throughout=64.0 kbps
   Downlink throughout=64.3 kbps
   Datagrams Sent-56

2. 128 kbps
   Data Transfer=158 Kbytes
   Uplink throughout=128 kbps
   Downlink throughout=130 kbps
   Datagrams Sent-110
3. 256 kbps
   Data Transfer=314 Kbytes
   Uplink throughout=256 kbps
   Downlink throughout=256 kbps
   Datagrams Sent-219
4. 512 kbps
   Data Transfer=627 Kbytes
   Uplink throughout=512 kbps
   Downlink throughout=519 kbps
   Datagrams Sent-437

5. 1024 kbps
   Data Transfer=1.22 MB
   Uplink throughout=1.02 Mbps
   Downlink throughout=1.02 Mbps
   Datagrams Sent-872
6. 2048 kbps
   Data Transfer=2.44 MB
   Uplink throughout=2.05 Mbps
   Downlink throughout=2.08 Mbps
   Datagrams Sent-1743

Justification-It is clear that the uplink throughout is limited by the network condition and the bandwidth of the host machines and the bandwidth used by the client.In most of the cases the network has sufficient capability to have uplink throughout equal to the bandwidth.

3)

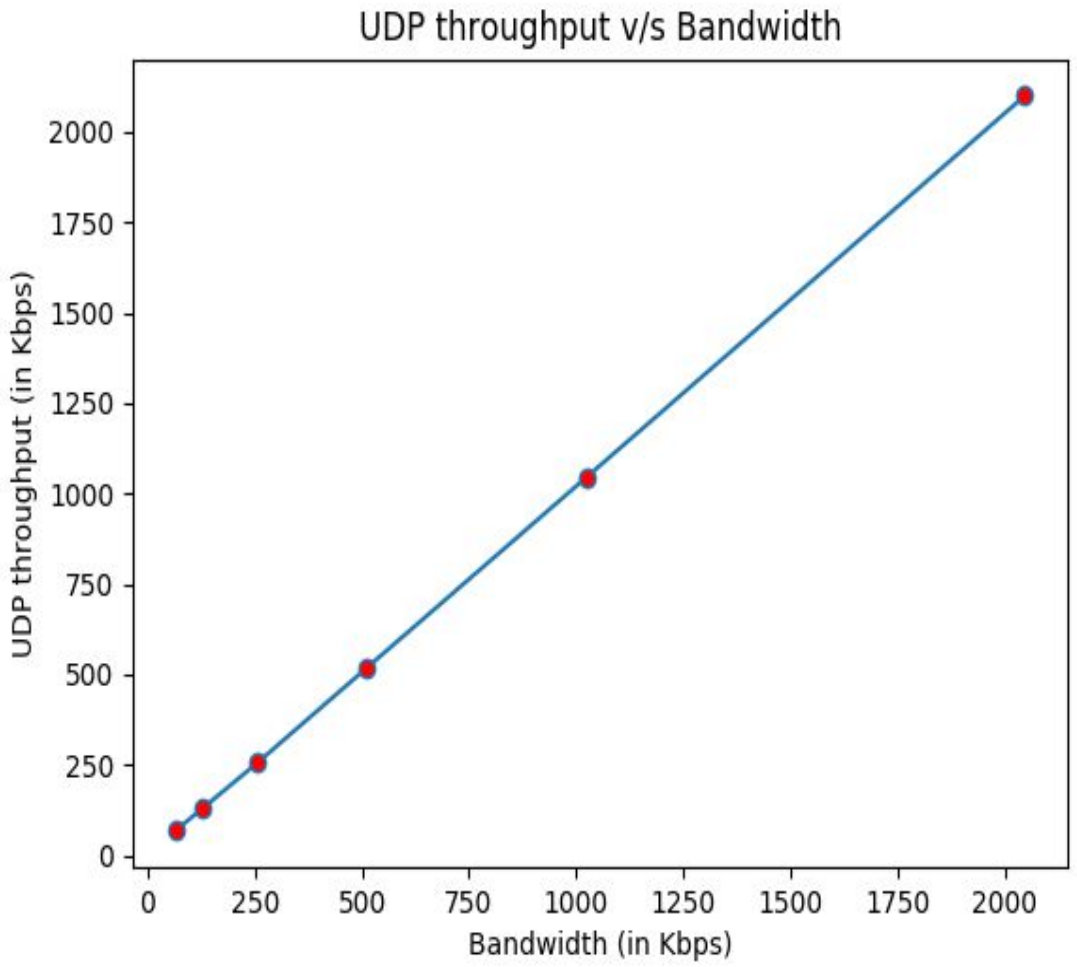


Observation: Number of retransmission packets lost depends on the traffic and the strength of network connection

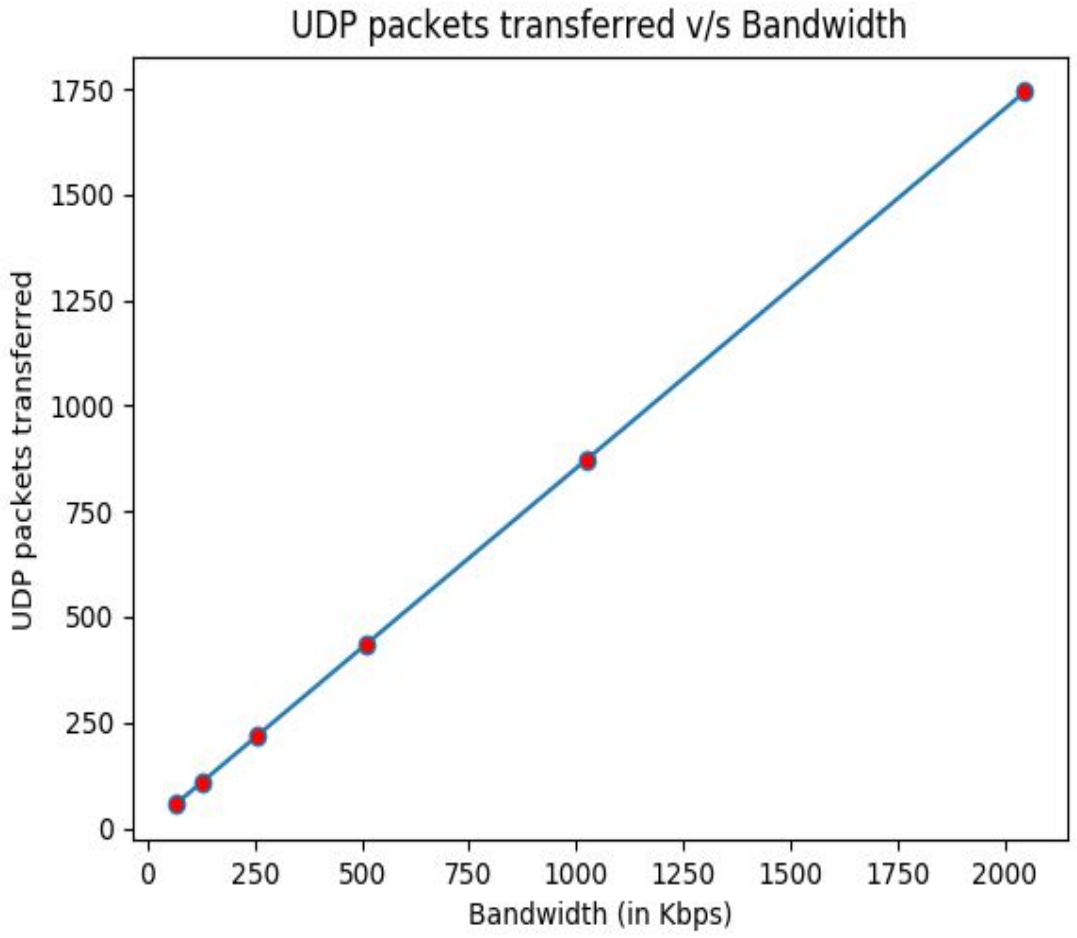
4) Plots

Using a *matplotlib* script, generate the graph that shows the relation between UDP throughput vs bandwidth, and also between the number of packets that are transmitted vs bandwidth.

UDP throughput vs UDP bandwidth



Number of UDP packet transmitted vs UDP bandwidth



Observations:

- UDP throughput is almost equal to the bandwidth specification using *iperf*. This shows the network is showing no latency at all. It can be observed that for very high data rate the throughput reaches a limiting value. This limiting value is due to network limitations.
- As bandwidth increases more number of packets were transferred in the same span of time. This can be observed by the increasing number of datagrams sent