Assignment 1 - Networks Lab

17CS10058 - Gurjot Singh Suri
17CS10022 - Kumar Abhishek

1) Steps :

1. Used wget tool to create a http request to the TCP server to download various pictures in the server and analysed the packets in the wireshark tool.
'wget --no-proxy http://10.5.18.163:8000/1.jpg'
2. Used iperf tool to create a UDP packet and sent it to UDP server. The monitoring tool wireshark is used to analyse the packets.
'iperf -c 10.5.18.163 -u -b 28000 '

Protocols:
TCP case :
Application layer : HTTP
Transport layer : TCP
Network layer : IPv4
UDP case :
Application layer : None
Transport layer : UDP
Network layer : IPv4

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 filter to monitor only the packets that are concerned with our experiment in wireshark.
2. Repeat the steps of question 1.
3. client_ip is found using the 'ifconfig' command.
4. I/O graphs are obtained from wireshark Menu->Statistics->IO Graphs

Observation:

Pic 1 : 982 tcp packets
Pic 2 : 3421 tcp packets
Pic 3 : 7500 tcp packets
Pic 4 : 3559 tcp packets
Pic 5 : 3607 tcp packets

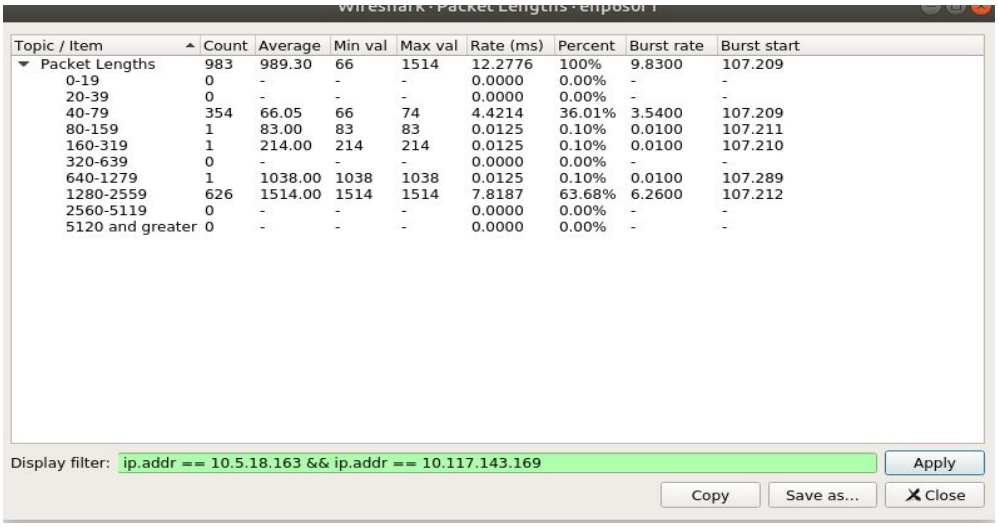
Justification : Since the pictures are of different sizes and using a TCP protocol, number of data packets are different in all the cases. It is clear that pic 3 has the largest size.

No, all the packets are not of the same size and there were various sizes ranging from 60s to a few thousands.

Some packet sizes for each of the pics in bytes are
Pic 1 : 74,66,214,83,1514,1038 etc.
Pic 2 : 74,66,214,83,1514,303 etc.
Pic 3 : 74,66,214,83,1514,416,78,86,94 etc.
Pic 4 : 74,66,214,83,1514,516,78,86,94 etc
Pic 5 : 74,66,214,83,1514,1402 etc

Justification : The number of packets depend on the size of the data transferred and the speed of the connection. The packets have varying sizes because for TCP, packets other than the data packets are also transmitted and received (like acknowledgment, handshake, etc.) which have different sizes.

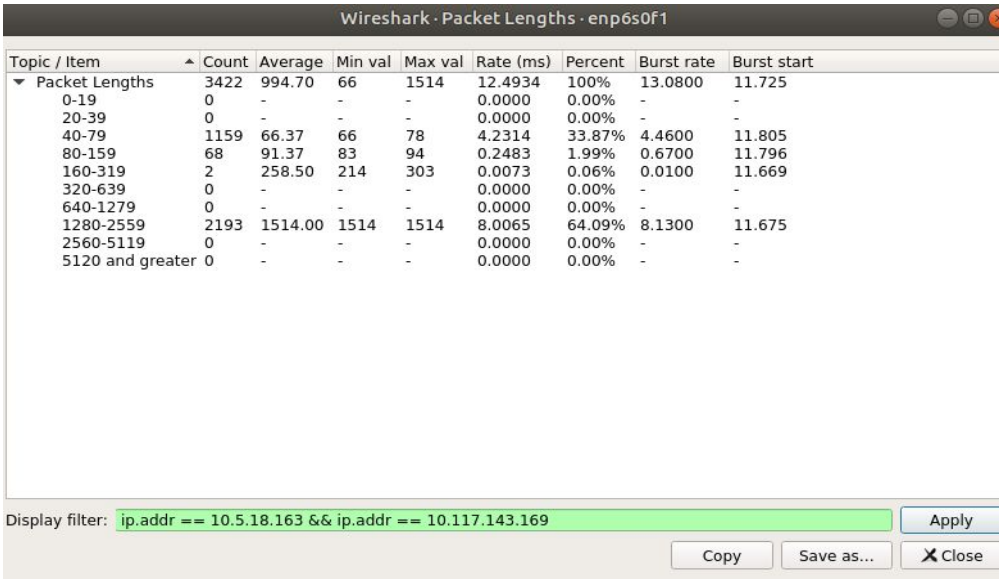
Pic 1:



Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	983	989.30	66	1514	12.2776	100%	9.8300	107.209
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	354	66.05	66	74	4.4214	36.01%	3.5400	107.209
80-159	1	83.00	83	83	0.0125	0.10%	0.0100	107.211
160-319	1	214.00	214	214	0.0125	0.10%	0.0100	107.210
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	1	1038.00	1038	1038	0.0125	0.10%	0.0100	107.289
1280-2559	626	1514.00	1514	1514	7.8187	63.68%	6.2600	107.212
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter: ip.addr == 10.5.18.163 && ip.addr == 10.117.143.169 Apply
Copy Save as... Close

Pic 2:



Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	3422	994.70	66	1514	12.4934	100%	13.0800	11.725
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	1159	66.37	66	78	4.2314	33.87%	4.4600	11.805
80-159	68	91.37	83	94	0.2483	1.99%	0.6700	11.796
160-319	2	258.50	214	303	0.0073	0.06%	0.0100	11.669
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	2193	1514.00	1514	1514	8.0065	64.09%	8.1300	11.675
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter: ip.addr == 10.5.18.163 && ip.addr == 10.117.143.169 Apply
Copy Save as... Close

Pic 3:

Wireshark · Packet Lengths · enp6s0f1

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	7510	998.38	66	1514	12.5664	100%	13.8500	55.710
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	2574	66.50	66	78	4.3071	34.27%	5.1900	55.775
80-159	100	88.13	83	94	0.1673	1.33%	0.5900	55.767
160-319	1	214.00	214	214	0.0017	0.01%	0.0100	55.642
320-639	2	505.00	416	594	0.0033	0.03%	0.0100	55.783
640-1279	1	986.00	986	986	0.0017	0.01%	0.0100	55.779
1280-2559	4832	1514.00	1514	1514	8.0853	64.34%	8.1300	55.645
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter: ip.addr == 10.5.18.163 && ip.addr == 10.117.143.169 Apply

Copy Save as... Close

Pic 4:

Wireshark · Packet Lengths · enp6s0f1

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	3632	996.84	66	1514	12.4936	100%	13.0800	8.600
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	1211	66.17	66	78	4.1657	33.34%	4.4500	8.671
80-159	86	86.99	83	94	0.2958	2.37%	0.8500	8.660
160-319	1	214.00	214	214	0.0034	0.03%	0.0100	8.535
320-639	1	516.00	516	516	0.0034	0.03%	0.0100	8.825
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	2333	1514.00	1514	1514	8.0252	64.23%	8.1300	8.540
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter: ip.addr == 10.5.18.163 && ip.addr == 10.117.143.169 Apply

Pic 5:

Wireshark · Packet Lengths · enp6s0f1

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	3692	976.13	66	1514	12.8135	100%	13.9000	12.205
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	1304	66.35	66	78	4.5257	35.32%	5.2400	12.265
80-159	67	92.52	83	94	0.2325	1.81%	0.6600	12.256
160-319	2	216.00	214	218	0.0069	0.05%	0.0100	12.128
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	2319	1513.89	1362	1514	8.0483	62.81%	8.1300	12.130
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

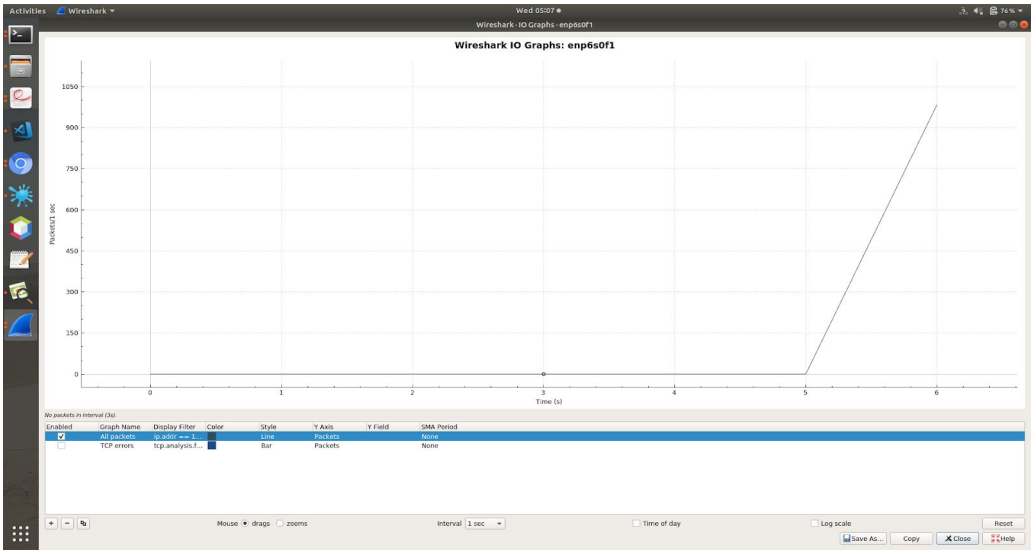
Display filter: ip.addr == 10.5.18.163 && ip.addr == 10.117.143.169 Apply

Copy Save as... Close

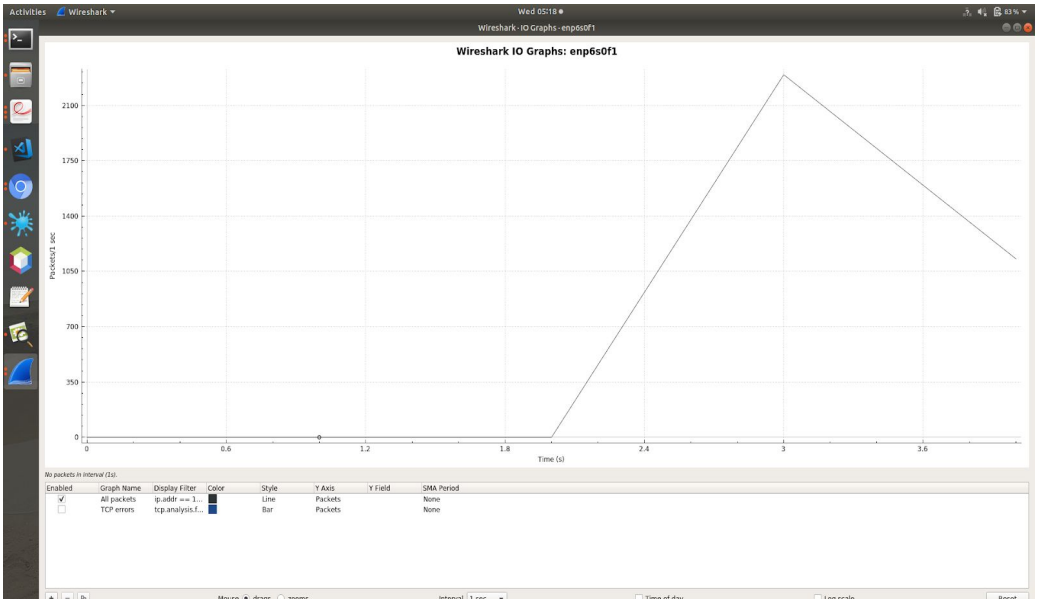
b)
Yes, all UDP packets transferred were found to be of the same size.
Packet size : 1512 bytes.
The packets are of the same size because for transmission through UDP UDP client, only the data packets are transferred and not any other packets(like handshake, acknowledgement, etc.). UDP client will divide the data into packets of the same size.

c)
TCP throughput

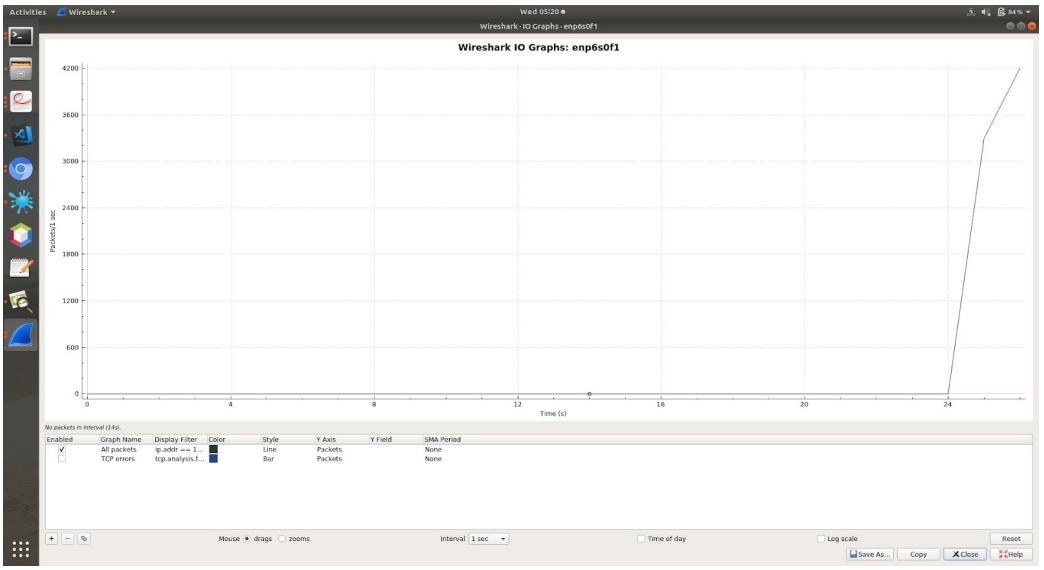
Pic 1:



Pic 2:

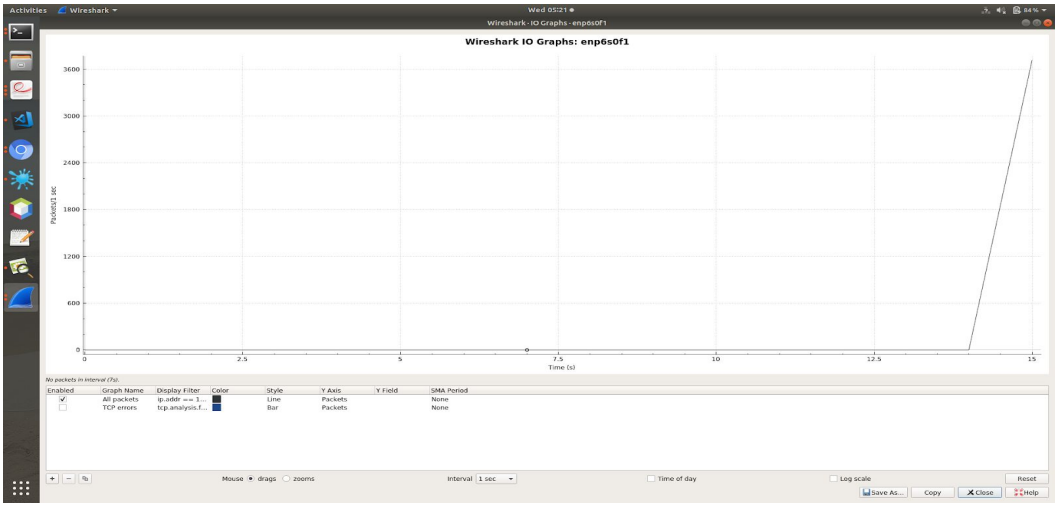


Pic 3:



Pic 4:

Pic 5:



UDP throughput

(d)

Steps:

1. Run the command ‘iperf -c 10.5.18.163 -u -b bandwith_value -r ’ for getting both uplink throughput and downlink throughput.
2. The No of datagrams sent can be looked in the wireshark after applying the necessary filter ‘ip.dst == 10.5.18.163 && ip.src == client_ip’.
3. Client ip is looked using ifconfig.

Observations:

i) 64 Kbps: Throughput=68 Kbps, No of Datagrams=58
ii) 128 Kbps: Throughput=133 Kbps, No of Datagrams=112
iii) 256 Kbps: Throughput=265 Kbps, No of Datagrams=221
iv) 512 Kbps: Throughput=528 Kbps, No of Datagrams=439
v) 1024 Kbps: Throughput=1054 Kbps, No of Datagrams=874
vi) 2048 Kbps: Throughput=2106 Kbps, No of Datagrams=1745

3)

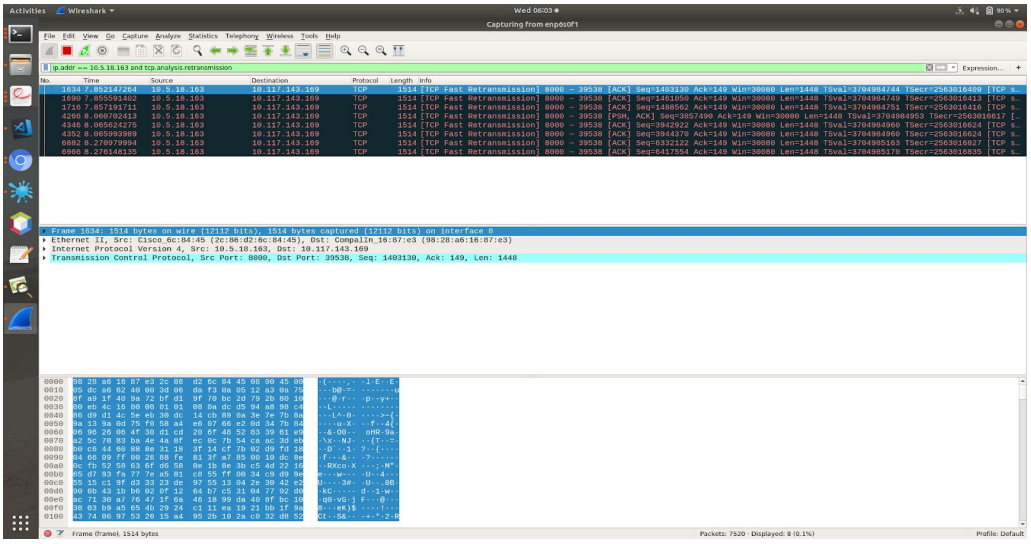
Steps:

1. ‘ip.addr == 10.5.18.163 && ip.addr == client_ip && tcp.analysis.retransmission’’ is used in filter to monitor only the packets that are concerned with our experiment and are retransmitted in wireshark.
2. Repeat the steps of question 1.
3. client_ip is found using the ‘$ ifconfig’ command.

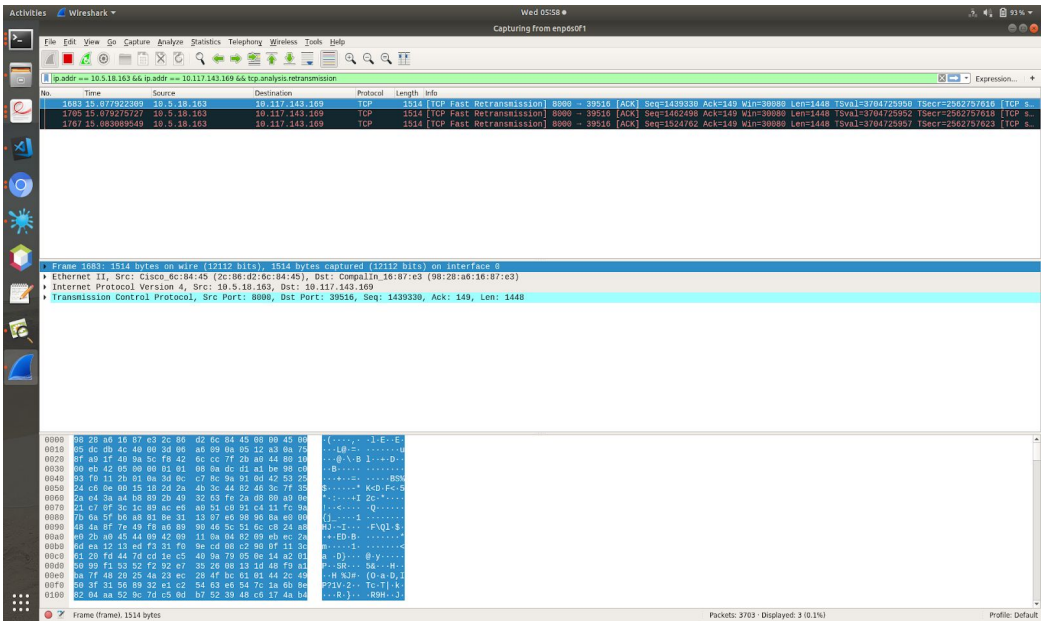
Observations:

Tcp packets were retransmitted for only pics 3 and 4 while for udp there was no retransmission.

Pic 3:
8 packets of size 1514 bytes

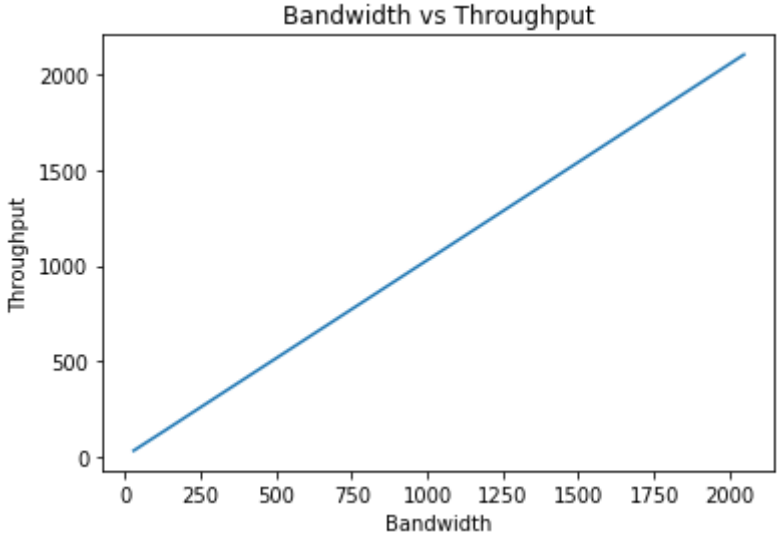


Pic 4:
3 packets of size 1514 bytes

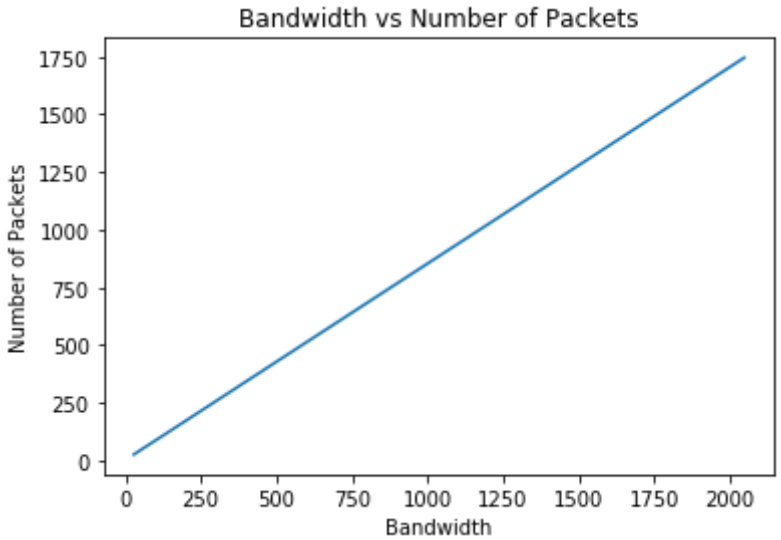


Justification:: Since pic3 and pic4 sizes are large, it takes significant amount of time to be transferred. Also, the number of packets transferred are also high. Due to this, there are higher chances of congestion which might lead to packets being dropped. The other pics' sizes are not too large and hence, no dropping and retransmission occurs.

4)
a)
Steps:
Used a python script and matplotlib python library to do necessary plotting.



4) b)



Observations:
● The graph of throughput vs bandwidth is linear. UDP throughput is almost equal to the bandwidth specified using iperf. That means the data rate is almost equal to the throughput. This shows the network is showing
no latency at all. It can be observed that for very high data rate the uplink throughput reaches a limiting value. This limiting value is the network limitation.
● The graph of number of packets vs bandwidth is linear. 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.