Assignment 1 - Networks Lab

15CS30042 - Sree Theerdha
15CS30008 - Buridi Sree Aditya

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 wires shark tool.
   '$wget --no-proxy http://10.5.20.128:8000/pic1.png '
2. Used **iperf** tool to create a UDP packet and sent it to UDP server. The monitoring tool wireshark is used to analyse the packets.
   '$iperf -c 10.5.20.128 -u -b 64k**'**

Observations:

TCP case:

Application layer : HTTP
Transport layer : TCP
Network layer : IP version 4

UDP case:

Application layer : -
Transport layer : UDP
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 == 10.5.20.128 && 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: In all the cases, 2 SYN,ACK packets at the beginning, 1 TCP "ok" packet at the end, and 1 TCP ACK packets at the end are observed. Moreover, there was an ACK packet after each of the received packets.

Pic 1 : 20 data packets
Pic 2 : 6087 data packets
Pic 3 : 231 data packets
Pic 4 : 1365 data packets
Pic 5 : 292 data packets

<u>Justification :</u> since the pictures are of different sizes and using a TCP protocol, number of data packets are different in all the cases.

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

Some packet sizes for each of the pics in bytes are
Pic 1 : 74,66,217,8258,2962,4410 etc.
Pic 2 : 74,66,217,1514,7306,2962 etc.
Pic 3 : 74,66,217,83,5362,1514 etc.
Pic 4 : 74,66,217, 240,10202,1514,7306 etc
Pic 5 : 74,66,217,29962,1514,4410 etc

<u>Details:</u>

<u>Pic1:</u>

Wireshark · Packet Lengths · ass1_TCP1

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	46	1092.91	66	8258	0.9584	100%	0.4600	44.505
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	25	66.64	66	74	0.5209	54.35%	0.2500	44.505
80-159	1	83.00	83	83	0.0208	2.17%	0.0100	44.513
160-319	2	227.50	217	238	0.0417	4.35%	0.0200	44.507
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	1	1108.00	1108	1108	0.0208	2.17%	0.0100	44.550
1280-2559	11	1514.00	1514	1514	0.2292	23.91%	0.1100	44.524
2560-5119	4	3686.00	2962	4410	0.0833	8.70%	0.0400	44.530
5120 and greater	2	7782.00	7306	8258	0.0417	4.35%	0.0200	44.524

Display filter: ip.addr == 10.5.20.128 Apply

Copy Save as... Close

<u>Pic2:</u>

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Packet Lengths	10120	1699.42	66	8754	0.9060	100%	1.9200	190.491
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	3973	66.39	66	78	0.3557	39.26%	0.7700	190.542
80-159	60	89.15	83	94	0.0054	0.59%	0.3200	190.503
160-319	1	217.00	217	217	0.0001	0.01%	0.0100	180.178
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	3323	1514.00	1514	1514	0.2975	32.84%	0.6000	190.487
2560-5119	2005	3423.29	2584	4410	0.1795	19.81%	0.5200	189.711
5120 and greater	758	6641.22	5858	8754	0.0679	7.49%	0.2100	190.898

Display filter: ip.addr == 10.5.20.128 Apply

Copy Save as… Close

Pic3:

Wireshark · Packet Lengths · ass1_TCP3

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Packet Lengths	381	1648.88	66	8754	1.4762	100%	1.7700	3.651
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	149	66.11	66	74	0.5773	39.11%	0.7200	3.561
80-159	1	83.00	83	83	0.0039	0.26%	0.0100	3.533
160-319	2	228.00	217	239	0.0077	0.52%	0.0200	3.531
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	1	913.00	913	913	0.0039	0.26%	0.0100	3.785
1280-2559	112	1514.00	1514	1514	0.4339	29.40%	0.5500	3.572
2560-5119	100	3396.40	2962	4410	0.3874	26.25%	0.5500	3.642
5120 and greater	16	6732.00	5362	8754	0.0620	4.20%	0.0900	3.595

Display filter: ip.addr == 10.5.20.128 Apply

Copy Save as… Close

Pic4:

Wireshark · Packet Lengths · ass1_TCP4

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	2310	1805.75	66	10202	0.9581	100%	1.8700	6.897
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	913	66.04	66	78	0.3787	39.52%	0.7400	6.880
80-159	32	85.91	83	86	0.0133	1.39%	0.3100	7.125
160-319	2	228.50	217	240	0.0008	0.09%	0.0200	5.792
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	689	1514.00	1514	1514	0.2858	29.83%	0.6000	6.915
2560-5119	454	3433.18	2962	4410	0.1883	19.65%	0.4600	6.926
5120 and greater	220	6845.27	5858	10202	0.0913	9.52%	0.2300	7.501

Display filter: ip.addr == 10.5.20.128 Apply

Copy Save as... Close

Pic5:

Wireshark · Packet Lengths · ass1_TCP5

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	482	1802.03	66	8754	1.2881	100%	1.6300	3.346
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	189	66.08	66	74	0.5051	39.21%	0.6800	3.315
80-159	1	83.00	83	83	0.0027	0.21%	0.0100	3.301
160-319	1	217.00	217	217	0.0027	0.21%	0.0100	3.296
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	158	1514.00	1514	1514	0.4222	32.78%	0.5400	3.336
2560-5119	90	3360.21	2601	4410	0.2405	18.67%	0.3300	3.360
5120 and greater	43	7306.00	5858	8754	0.1149	8.92%	0.1800	3.566

Display filter: ip.addr == 10.5.20.128 Apply

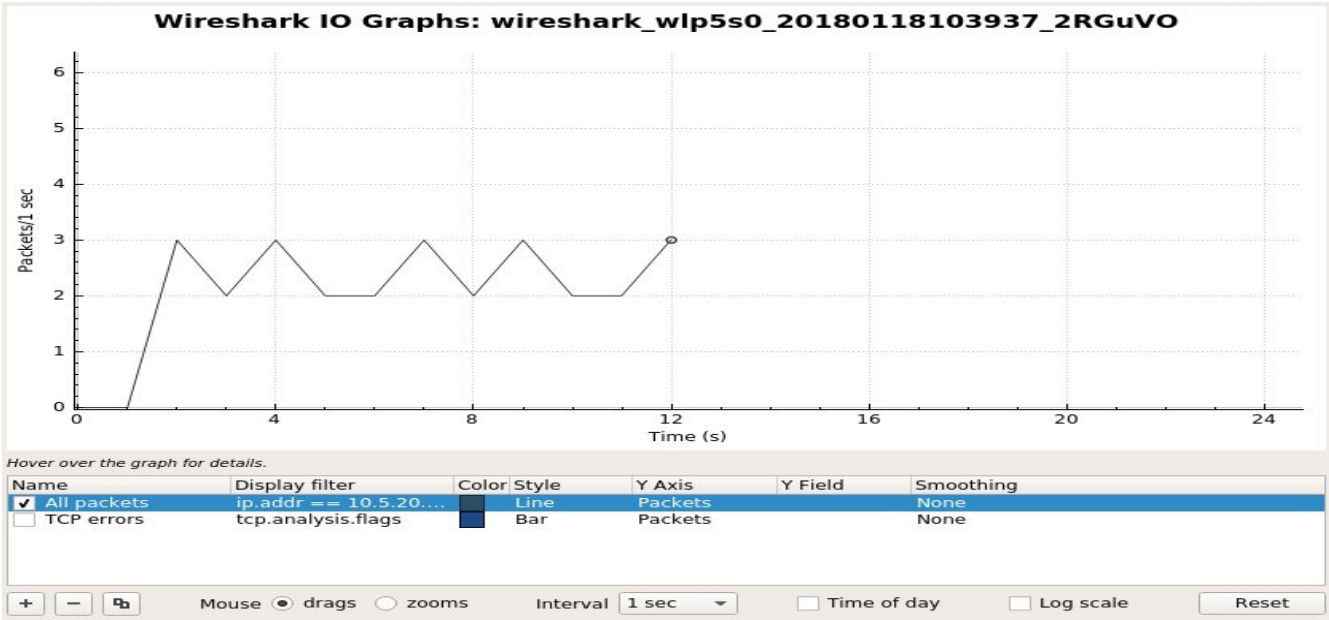
Copy Save as... Close

b) All 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.

c) Throughput using Wireshark

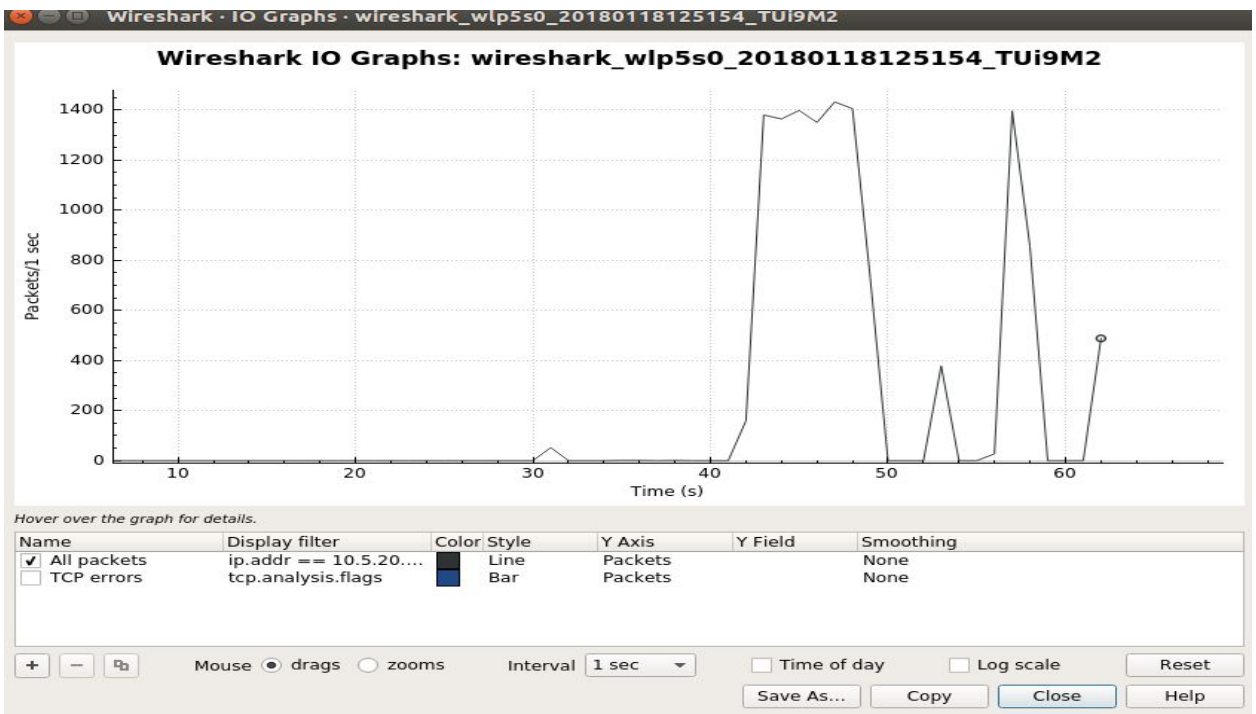
UDP:



TCP CASE:

Steps:

1. The images are requested using wget tool one after the other in a sequential manner.



Observation: 5 elevations in the graphs corresponds to 5 pictures transferred from server to client. Sequence of peaks correspond to sequence in which pics are requested.

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)

Steps:

1. Run the command 'iperf -c 10.5.20.128 -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 == 10.5.20.128 && ip.src == client_ip'.
3. Client ip is looked using ifconfig.

Observations:

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

(i) 64 Kbps

Data transfer = 80.4 k Bytes
Uplink throughput = 64.0 kbps
Downlink throughput = 64.3 kbps
Datagrams Sent = 58

(ii) 128 Kbps

Data transfer = 158 k Bytes
Uplink throughput = 128 kbps
Downlink throughput = 130 kbps
Datagrams Sent = 112

(iii) 256 Kbps

Data transfer = 314 k Bytes
Uplink throughput = 256.0 kbps
Downlink throughput = 256.0 kbps
Datagrams Sent = 221

(iv)512 Kbps

Data transfer = 627 k Bytes
Uplink throughput = 512 kbps
Downlink throughput = 520 kbps
Datagrams Sent = 439

(v)1024 Kbps

Data transfer = 1.22 MB
Uplink throughput = 1.03Mbps
Downlink throughput = 1.03Mbps
Datagrams Sent = 874

(vi) 2048 Kbps

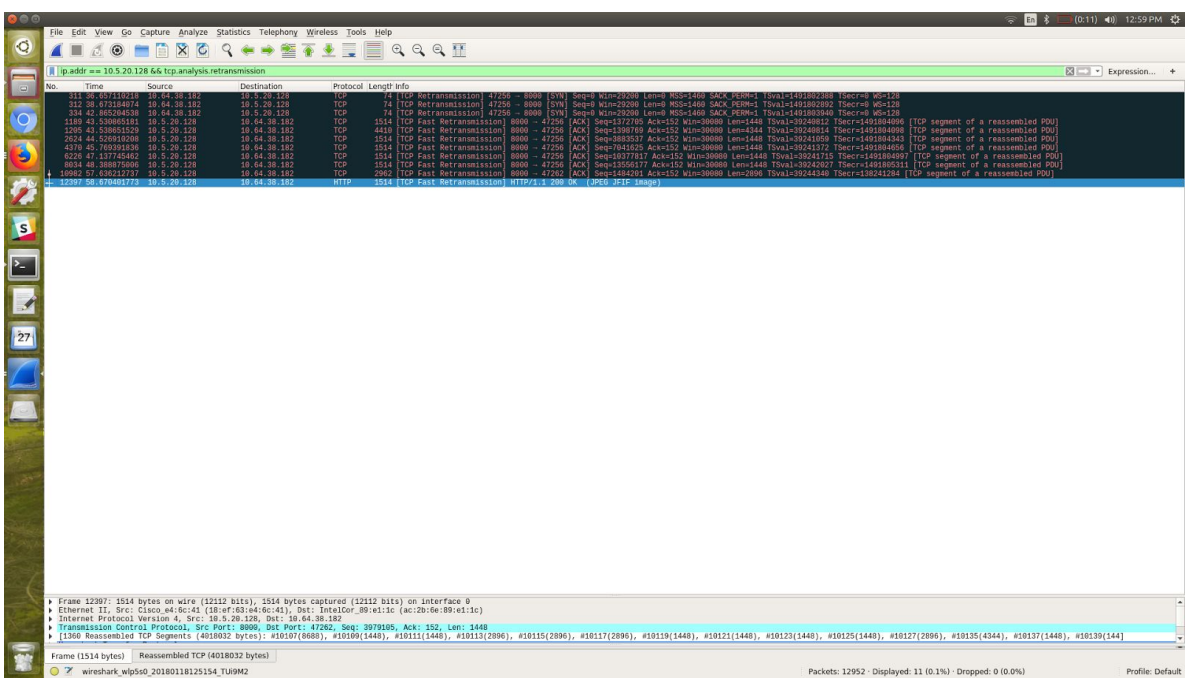
Data transfer = 2.44 MB
Uplink throughput = 2.05 Mbps
Downlink throughput = 2.08Mbps
Datagrams Sent = 1745

<u>Justification:</u> It is clear that the uplink throughput 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 throughput equal to bandwidth.

3)

<u>Steps:</u>

1. ‘ip.addr == 10.5.20.128 && 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.

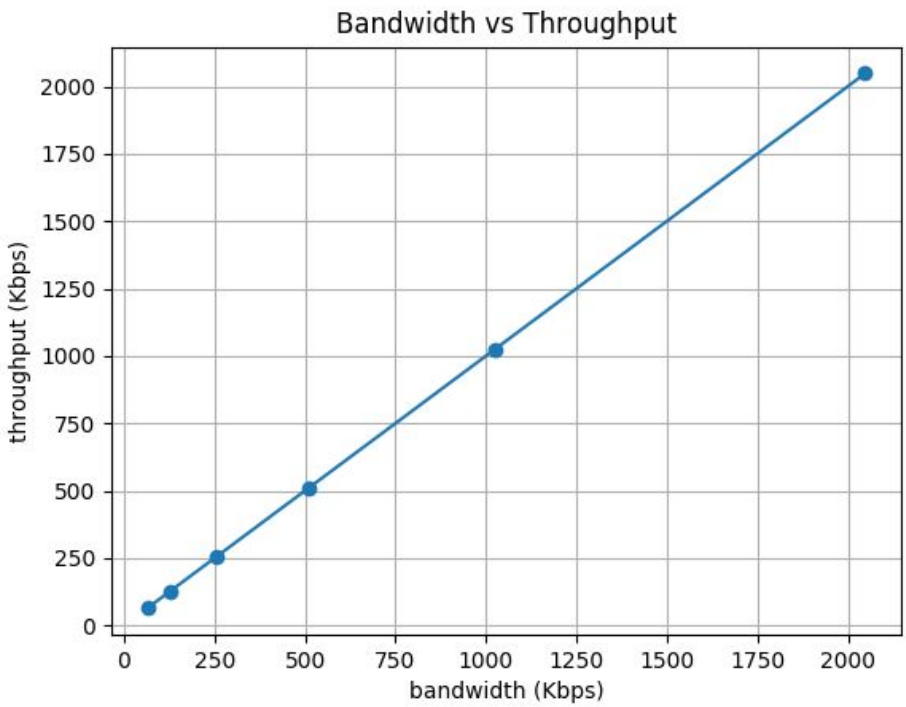


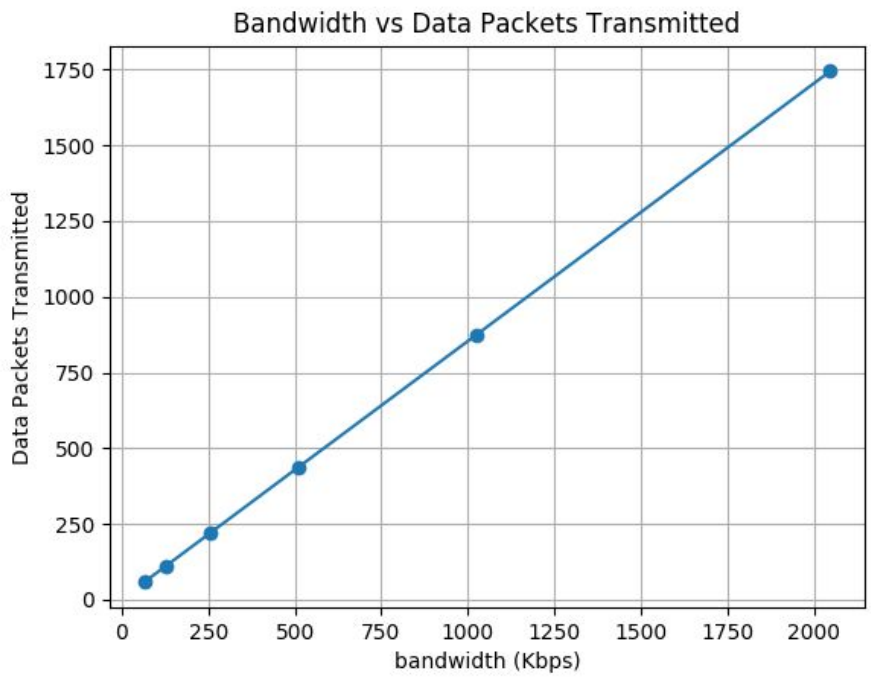
<u>Observation</u>: Number of retransmissions packets lost depend on the traffic and the strength of the network connection.

4)a)

<u>Steps:</u>

1. Used a python script and matplotlib python library to do necessary plotting.





b)

Observations:

- UDP uplink throughput is almost equal to the bandwidth specific using iperf. That means the data rate is almost equal to the uplink 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.
- 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.