ITS 413 Internet technologies and application

By:

Thanyatorn Parapuntakul(5222791824)

Tanapoom Kongarsa(5222792509)

Chavamon Srisak(5222790305)

28 march 2012

By submitting this report all members of the group listed above agree that each member has contributed approximately equal amounts to designing and performing experiments, as well as to preparing this report. All members agree that this report accurately reflects the experiments conducted by the group members, and is their own work (not works of other groups).

Sirindhorn International Institute of Technology

Thammasat University

Content

Aims

1. To study and understand about UDP and TCP
2. To find out what factor will impact on TCP performance
3. To understand about the sessions
4. To study to use command in terminal

Network Diagram

Figure1 : Wireless network

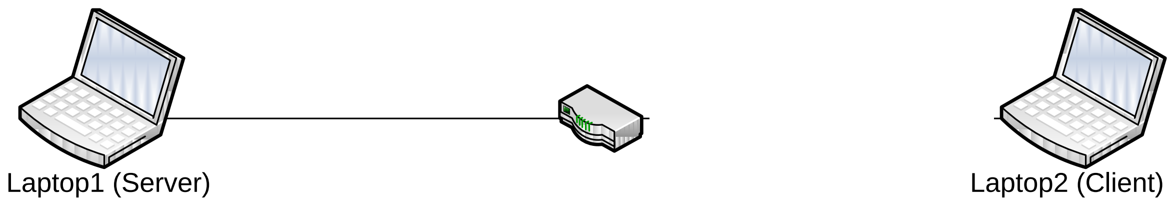
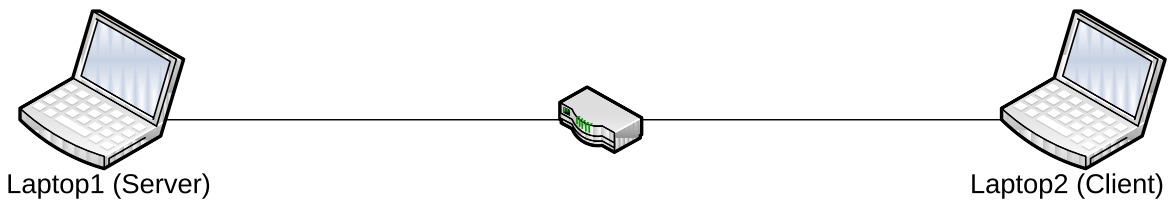


Figure2 : Wired network



Equipment Specifications

Server

Toshiba Protege T230

Processor intel core i3 cpu 1.33GHz

Broadcom WLAN 802.11bgn

Ubuntu linux

Client

Dell Inspiron n-series

Processor intel core i7 cpu 2.0GHz

Intel Centrino Wireless-N 1030

Ubuntu linux

Router

Linksys wrt54gl

Version Backfire

10.03.1 with the Broadcom Linux 2.4 kernel

LAN cable

Buffalo CAT-6 Flat Network cable

Cable type: UTP-4

Straight Cable

Connector Type: RJ-45

Connector

Cable: Twisted

Experiment

There are four experiment that we perform in this assignment

1.TCP vs UDP

2. Impact of packet drops on TCP performance

3. Multiple TCP sessions

4. TCP sessions vs UDP sessions

Experiment 1 TCP vs UDP

In this experiment we will design an experiment differentiate function of TCP and UDP by changing some parameter and study on that

Experiment 1.1 TCP vs UDP with changing delay

In this experiment, we will try to differentiate TCP and UDP by changing delay and collect the output to study

So, first we going to do is to open TCP server on computer A

```
Iperf -s
```

Next use **tc** on the computer B

```
sudo tc qdisc add dev eth0 root netem delay 100ms
```

This code will tell that the packet will delay for 100ms on interface eth0(wire)

Now let run Iperf on computer B

```
Iperf –c IP –t 60s
```

But if we want to change package delay we have to delete the old one first before adding a new one

We also use this command to delete it

```
sudo tc qdisc del dev eth0 root netem delay 100ms
```

And then perform every step again

But in our group use the shell script file run in the terminal

```
m='ms'

f=1

s=10



for j in {1..4}

do

        for i in 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
1700 1800 1900 2000 2100 2200 2300 2400 2500 2600 2700 2800 2900 3000

        do

        echo "================= Delay: "${i} ms"=========" >> outass1tcpwdelay-
${f}.txt

        sudo tc qdisc add dev eth0 root netem delay ${i}${m}

        sudo tc qdisc show dev eth0 >> outass1tcpwdelay-${f}.txt

     iperf -c 192.168.1.248 >> outass1tcpwdelay-${f}.txt

     sleep 5s

        sudo tc qdisc del dev eth0 root netem delay ${i}${m}

        let  i=i+10

        done

        echo "File "${f}" successfully generated"

        let  f=f+1

done
```

doing the same with UDP

first, we have to open UDP server on computer A

```
iperf –s -u
```

and also use this shell script file to run

```
m='ms'

f=1

s=10



for j in {1..4}

do

        for i in 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700
1800 1900 2000 2100 2200 2300 2400 2500 2600 2700 2800 2900 3000

        do

        echo "================== Delay: "${i} ms"==========" >> outass1udpwdelay-${f}.txt

        sudo tc qdisc add dev eth0 root netem delay ${i}${m}

        sudo tc qdisc show dev eth0 >> outass1udpwdelay-${f}.txt

    iperf -c 192.168.1.248 -u >> outass1udpwdelay-${f}.txt

    sleep 5s

        sudo tc qdisc del dev eth0 root netem delay ${i}${m}

        let  i=i+10

        done

        echo "File "${f}" successfully generated"

        let  f=f+1

done
```

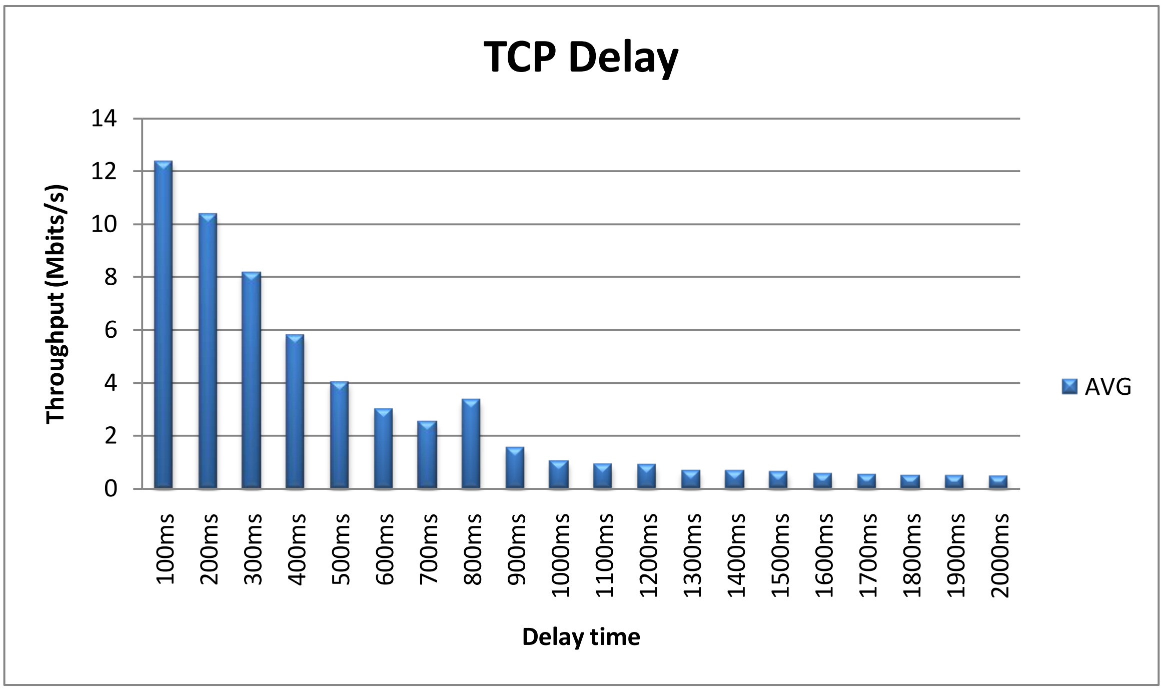


Figure : graph that show our average result for TCP delay

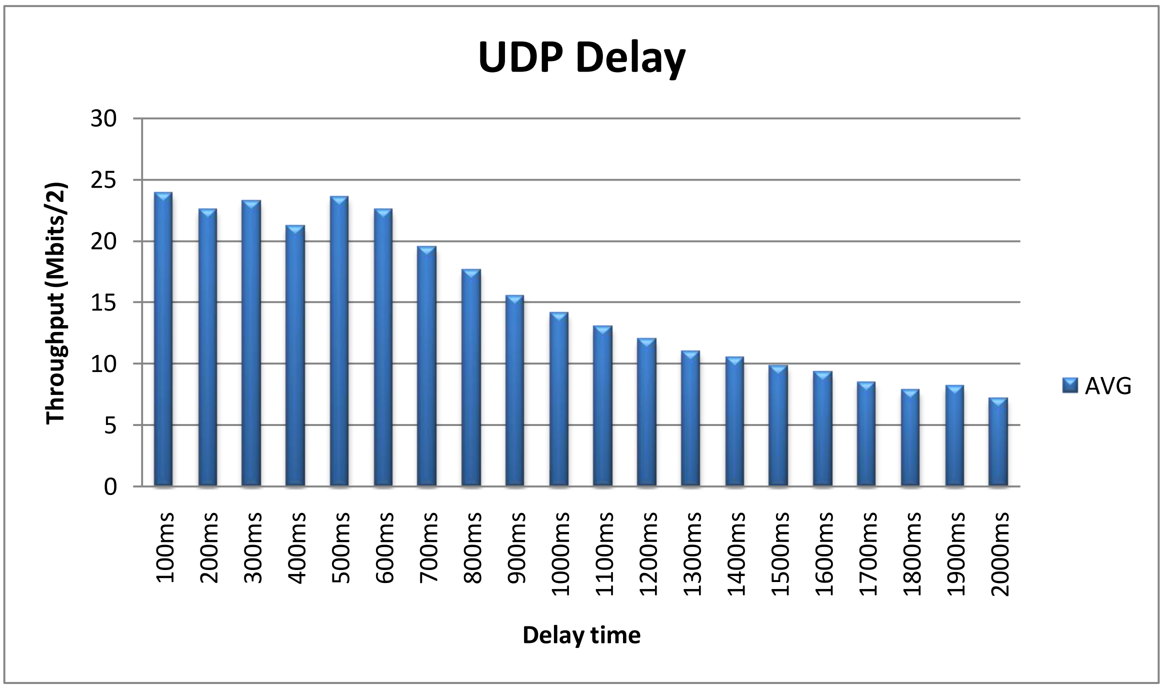


Figure : graph that show our average result for UDP delay

<u>Summary</u>

As you can see from the graph that show the throughput of UDP will get higher than the throughput of TCP it cause of the mechanism of TCP lead to result like this

<u>Experiment 1.2</u> TCP vs UDP with changing jitter

So, now in this experiment we will do the same as last experiment is to differentiate between UDP and TCP but now changing with jitter

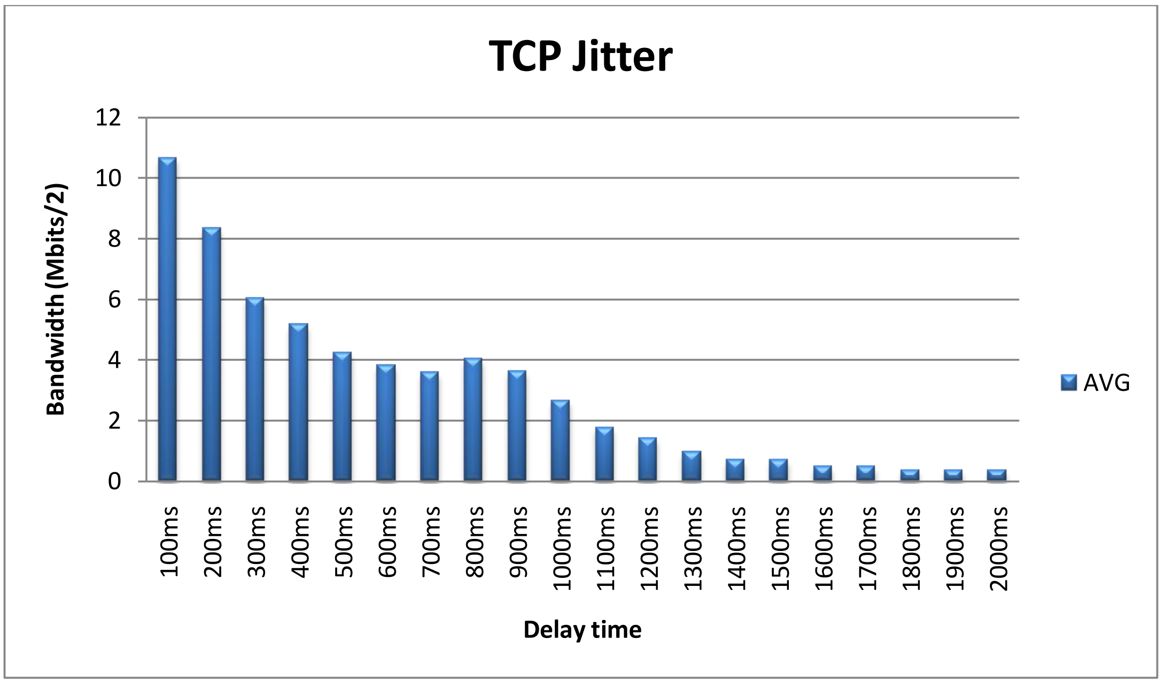


Figure : graph that show our average result for TCP jitter

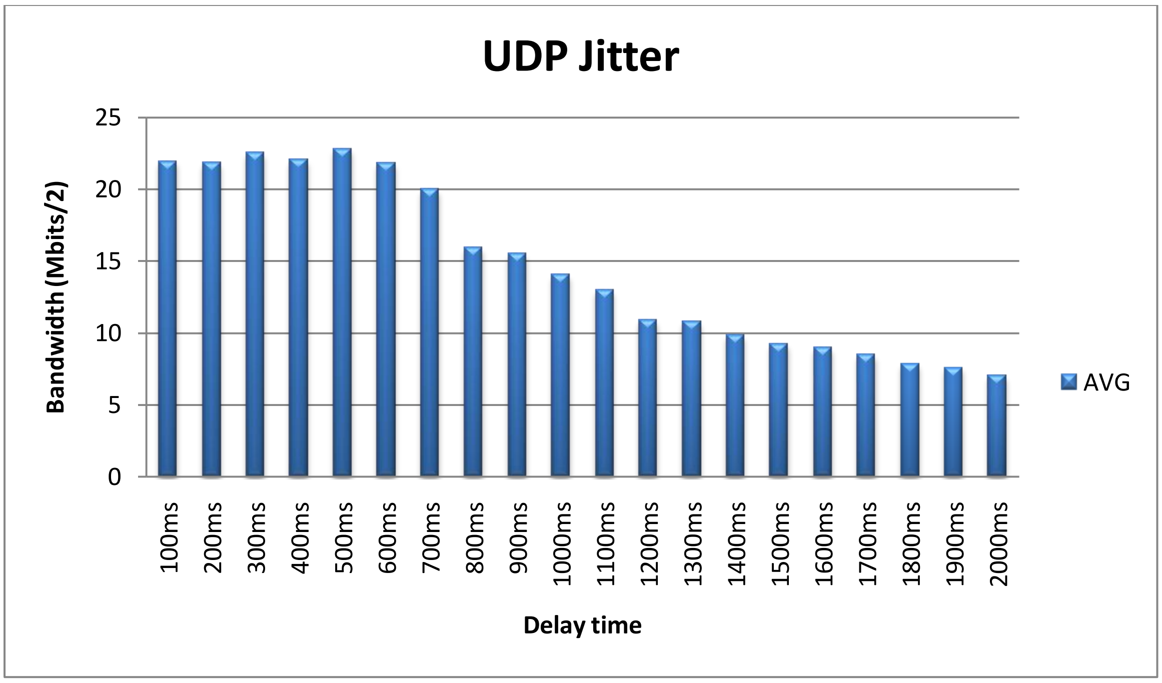


Figure : graph that show our average result for UDP jitter

Summary

We can summary like the same as last experiment UDP have higher bandwidth because of it’s didn’t care about reliable so it’s just sent the packet that lead to the higher throughput

Conclusion

On this experiment we can conclude that the higher throughput of UDP with compare to TCP throughput is cause of UDP didn’t have a reliable so it’s didn’t care that the packet will loss or duplicate it just send the packet to the destination but in TCP it’s have a mechanism that can make it reliable such as error control so, if we want to send all packet to the destination perfectly please use TCP but if we talk about the speed UDP also have higher speed but packet maybe lost.

<u>Experiment 2</u> Impact of packet drops on TCP performance

In this experiment we investigate the impact in performance of TCP due to the packet loss. By compared between wire and wireless connection and **tc** and **iptables** function

<u>Experiment2.1</u>: Use **iptables** to drop packet on wire network

In this experiment we focus on dropping packet by using **iptables** on wire network to make the firewall rule that can detect the packet that coming into the server

First let

rules **iptables** to randomly dropping packet for 5% on computer A

```
sudo iptables -A INPUT –m statistic –mode random –probability 0.05 –j DROP
```

Then, run TCP server on computer A

```
Iperf  -s
```

Finally, run TCP test with **iperf** on computer B

```
Iperf -c IP -t 60s
```

But we have to increase the dropping rate of **iptables** rule so, we have to delete the old rules first before make the new rule using command

```
sudo iptables -D INPUT  1
```

Then make the new rule with increase dropping rate and perform all of this step again

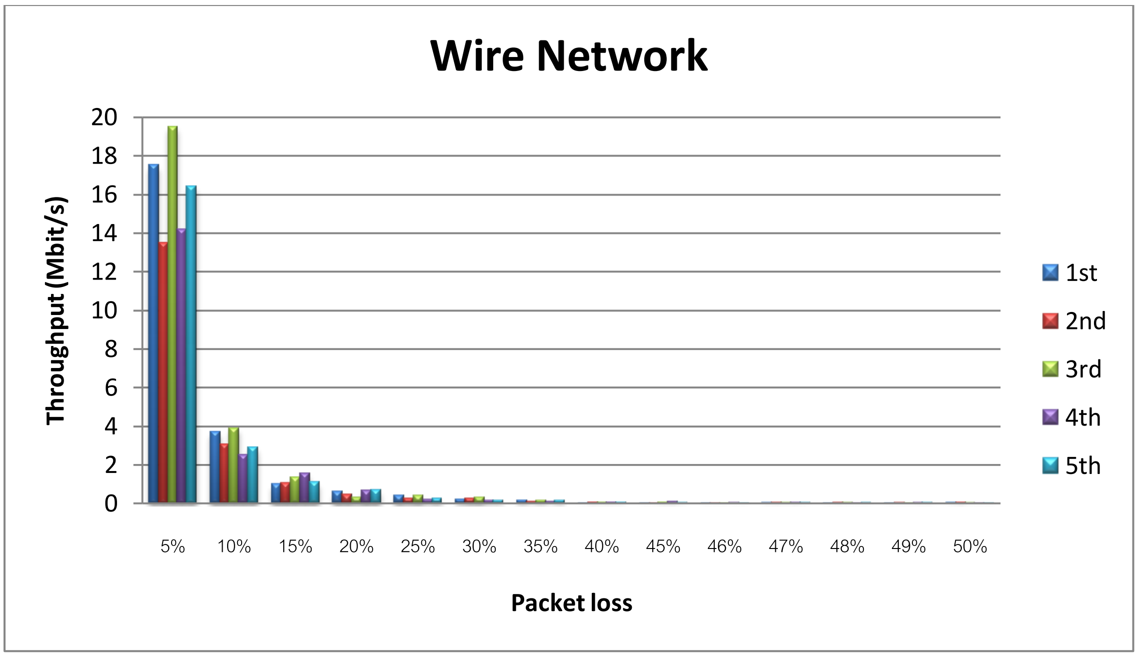


Figure : graph that show our result for use **iptables** to dropping packet with wire network(5 times)

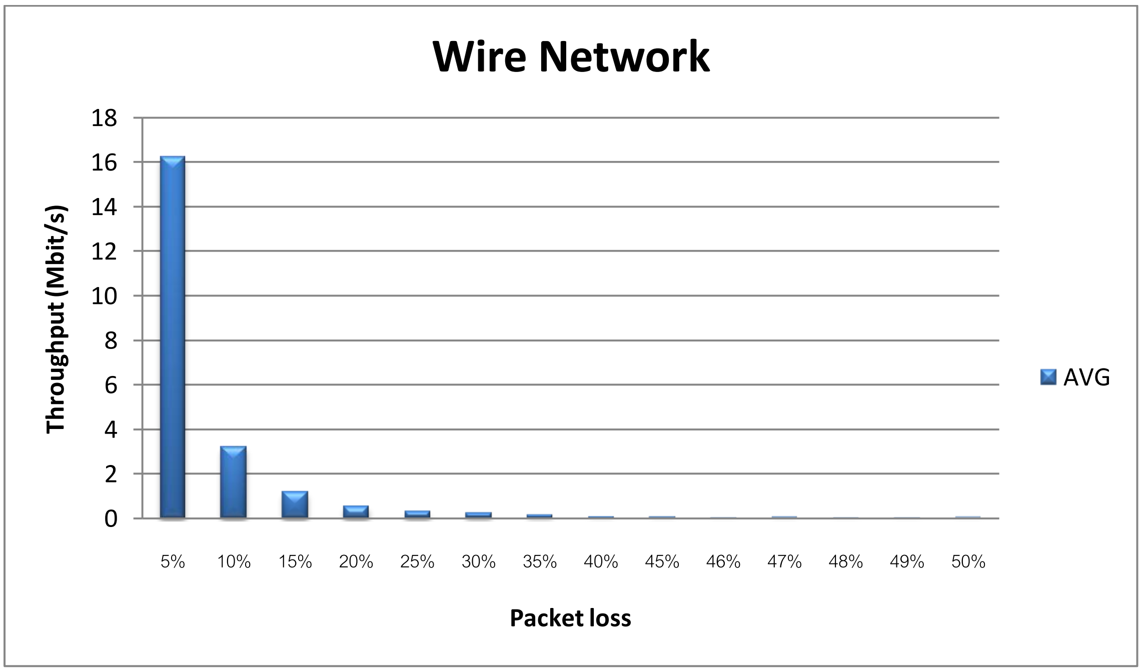


Figure : graph that show our result for use **iptables** to dropping packet with wire network(average value)

<u>Summary</u>

As you can see from the graph, **iptables** will randomly drop packet perfectly, first when the packet loss percentage was increased from 5% to 10% throughput will be dramatically dropped and as the packet loss percentage increased the throughput will increasingly drop down too.

<u>Experiment 2.2</u> Use **iptables** to drop packet on wireless network

In this experiment we will set every parameter and perform every step like the last experiment but this experiment will use wireless network for the test

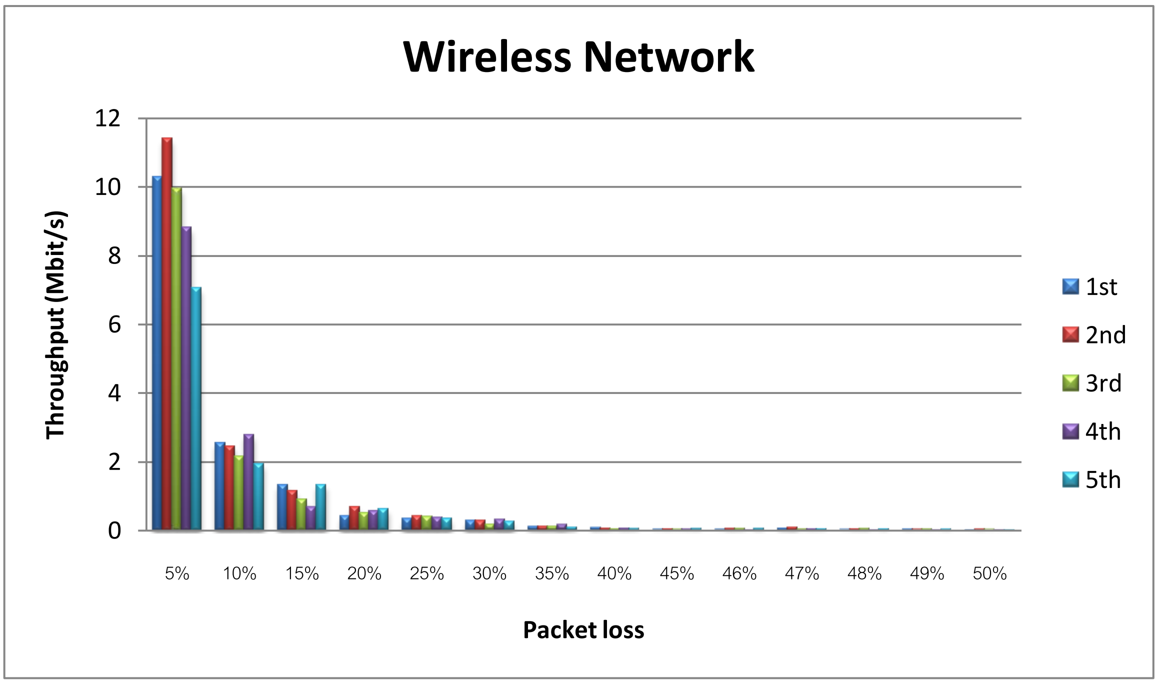


Figure : graph that show our result for use **iptables** to dropping packet with wireless network(5 times)

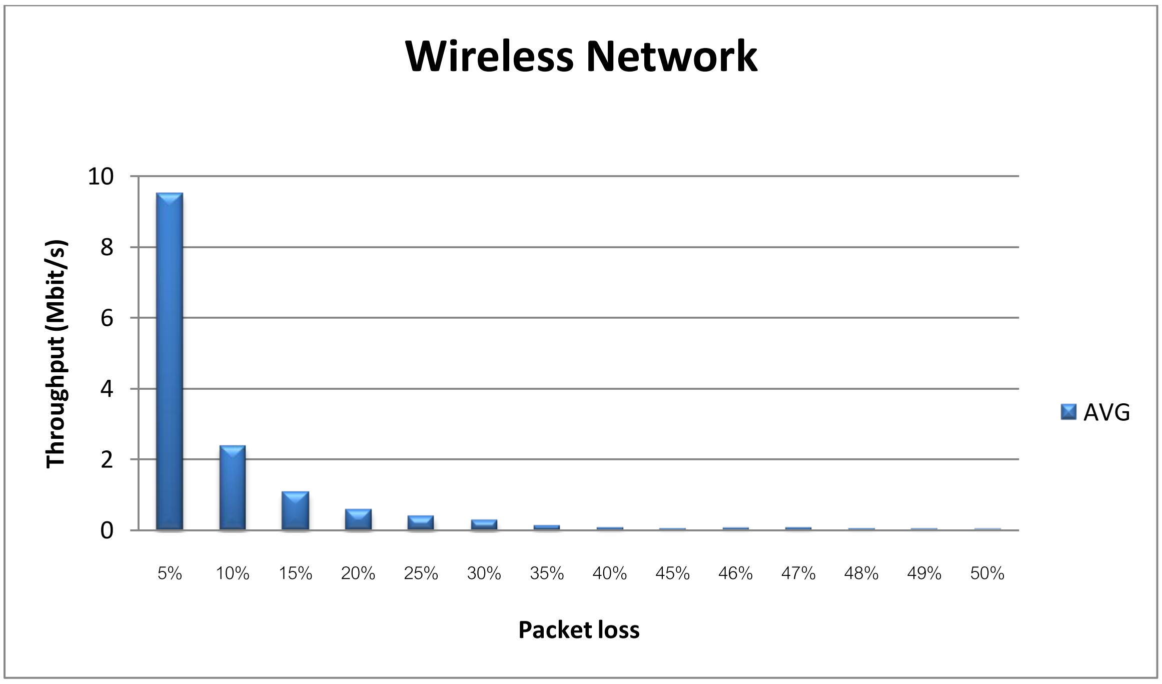


Figure : graph that show our result for use **iptables** to dropping packet with wireless network(average value)

Summary

In this experiment, **Iptables** is doing good like the last experiment but the throughput of 5% randomly dropping is less than the wired network it’s maybe because of TCP is design for the wired network so, the wireless net work cannot sent the packet in full rate.

Experiment 2.3 Use **tc** to drop packet on wire network

In this experiment we will dropping packet on wire network again but use different function now in this experiment we will use **tc** to drop packet, **tc** is a command to control the packet queuing mechanisms that it’s also can dropping packet as well

So, first let open the TCP server on computer A

```
Iperf -s
```

Next use **tc** on the computer B

```
sudo tc qdisc add dev eth0 root netem loss 5%
```

This code will tell that the packet on the kernel will drop 5% on interface eth0(wire)

Now let run Iperf on computer B

```
Iperf –c IP –t 60s
```

But if we want to change package loss percentage we have to delete the old one first before adding a new one

We also use this command to delete it

```
sudo tc qdisc del dev eth0 root netem loss 5%
```

And then perform every step again

But in our group bring shell script file to use in this experiment to safe the time and make it easy

```
m='%'

f=1

s=10

for j in {1..5}
```

```
do
            for i in 5 10 15 20 25 30 35 40 45 50 51 52 53 54 55
            do
            echo "================= Packageloss: "${i}${m}"==========" >> ass3-wired-
loss-${f}.txt


            sudo tc qdisc add dev eth0 root netem loss ${i}${m}
            sudo tc qdisc show dev eth0 >> ass3-wired-loss-${f}.txt
       iperf -c 192.168.1.248 -t 60 >> ass3-wired-loss-${f}.txt
       sleep 5s
            sudo tc qdisc del dev eth0 root netem loss ${i}${m}
            let  i=i+10
            done
            echo "File "${f}" successfully generated"
            let  f=f+1
done
```

And this is our result

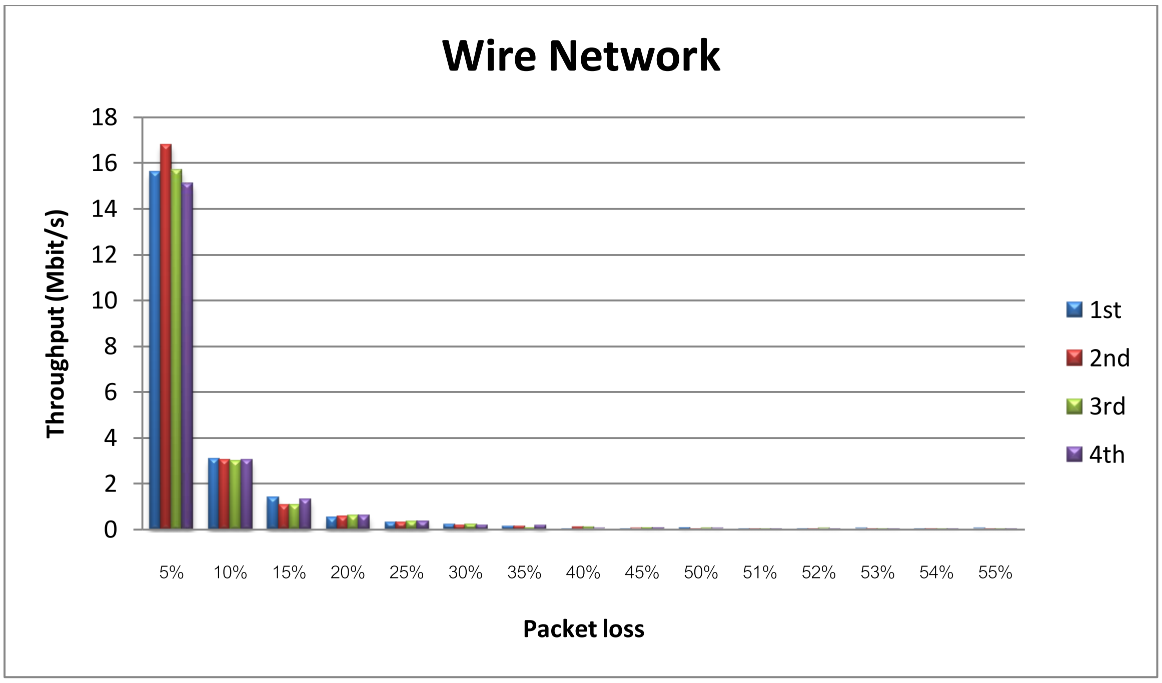


Figure : graph that show our result for use **tc** to dropping packet with wire network(4 times)

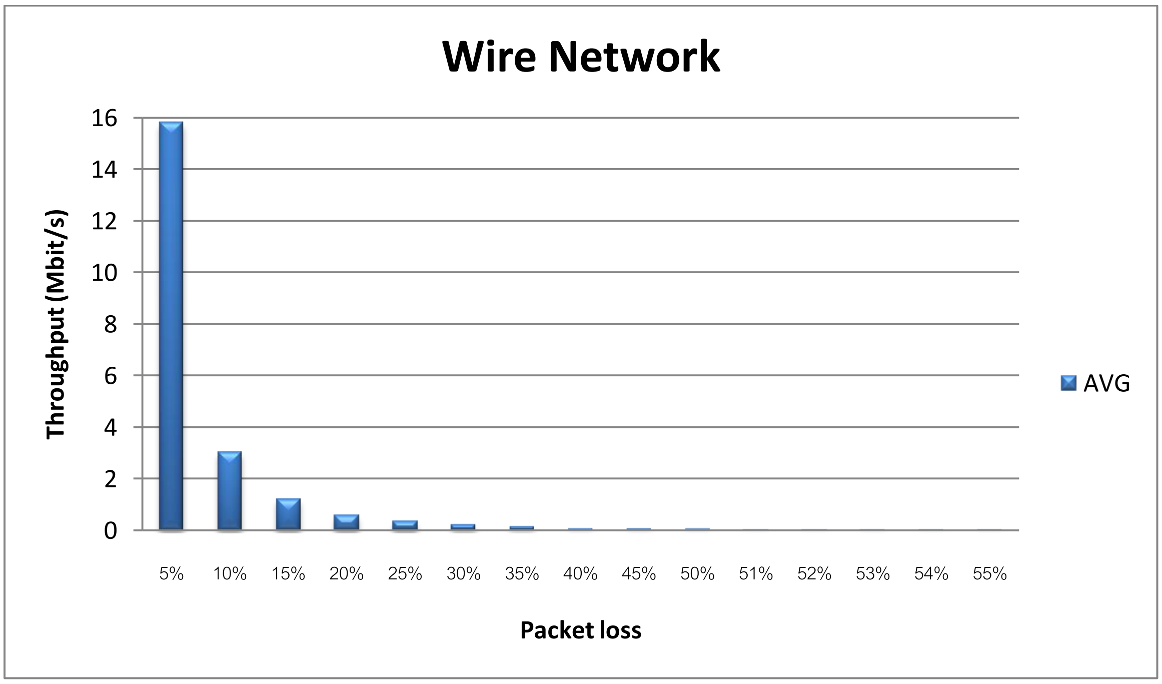


Figure : graph that show our result for use **tc** to dropping packet with wire network(Average value)

<u>Experiment 2.3</u> Use **tc** to drop packet on wireless network

In this experiment will perform every step like the last experiment but it have to change the interface from **eth0** to **wlan** because we want to test on the wireless network

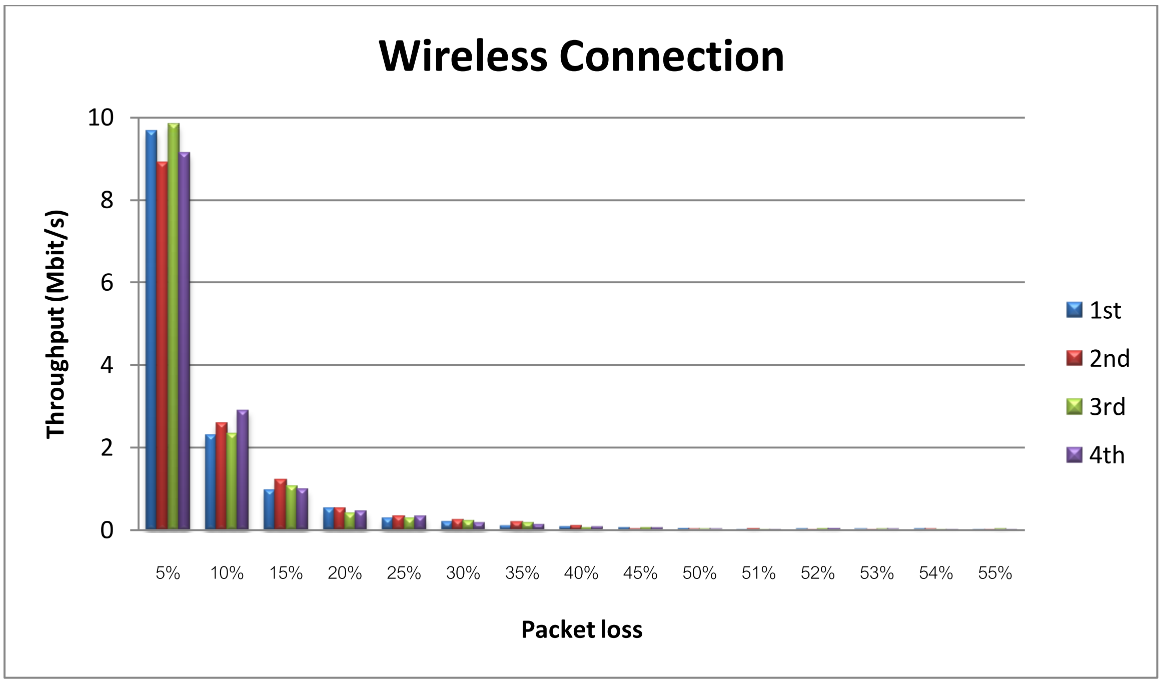


Figure : graph that show our result for use **tc** to dropping packet with wireless network(4 times)

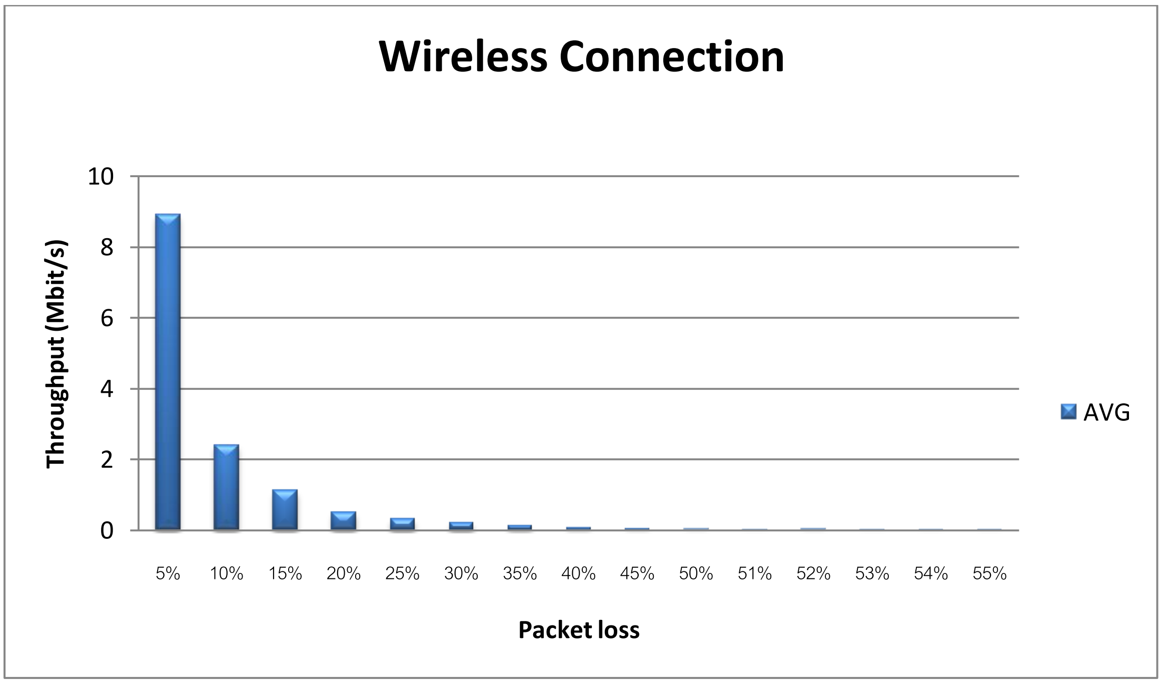


Figure : graph that show our result for use **tc** to dropping packet with wireless network(Average value)

Summary

For this experiment, that use **tc** to control the packet queuing mechanism with wired and wireless network **tc** can control dropping packet perfectly as you can see form the graph when the dropping rate was increased the throughput will increasingly drop down too

Conclusion

For all of the experiment that our group was perform in “Impact of packet drop on TCP performance” such as **tc** or **iptables** function for wire and wireless network we can conclude that the TCP have mechanism that can retransmit lost data and can slow sending rate when packet was lost so, for all of this mechanism can lead to the whole time sending rate longer that cause to the lower throughput of the network and also for both function **tc** and **iptables** can dropped perfectly but in **tc** can dropped more packet than **iptables** function it maybe cause of the mechanism of **tc** that can control the packet queuing perfectly

Experiment 3   Multiple TCP sessions

In this experiment, we will make the scenarios that we have many TCP sessions run to the server use **iperf** then study on the throughput that will happen

Experiment 3.1 Multiple TCP sessions by using hand manual to open terminal and run the test

In this experiment, we try to test the multiple TCP sessions by using hand manual to open terminal and run the test.

First on the computer A run the TCP server using:

```
iperf -s
```

And now on the computer B run a TCP test for 2 sessions (use 2 terminal to run) first :

```
iperf –c  IP
```

Then, run a 3 sessions and 4 sessions (use 3 terminal and 4 terminal) for 5 times each

From the experiment above we can get the throughput that will show in the result in graph below

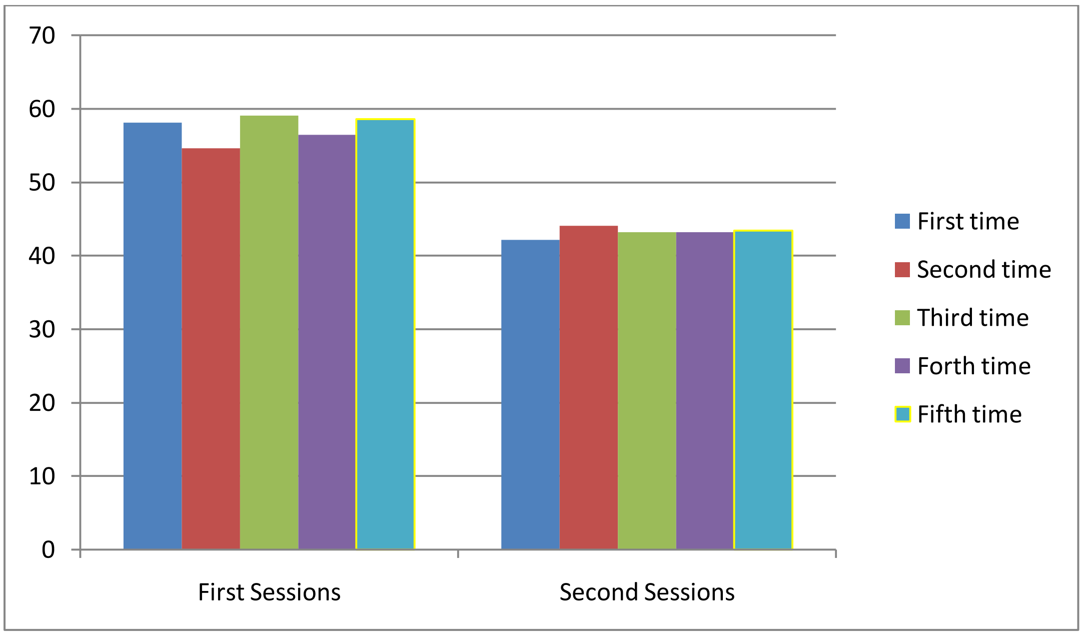


Figure: Graph of the throughput that run 2 TCP sessions on five time of test

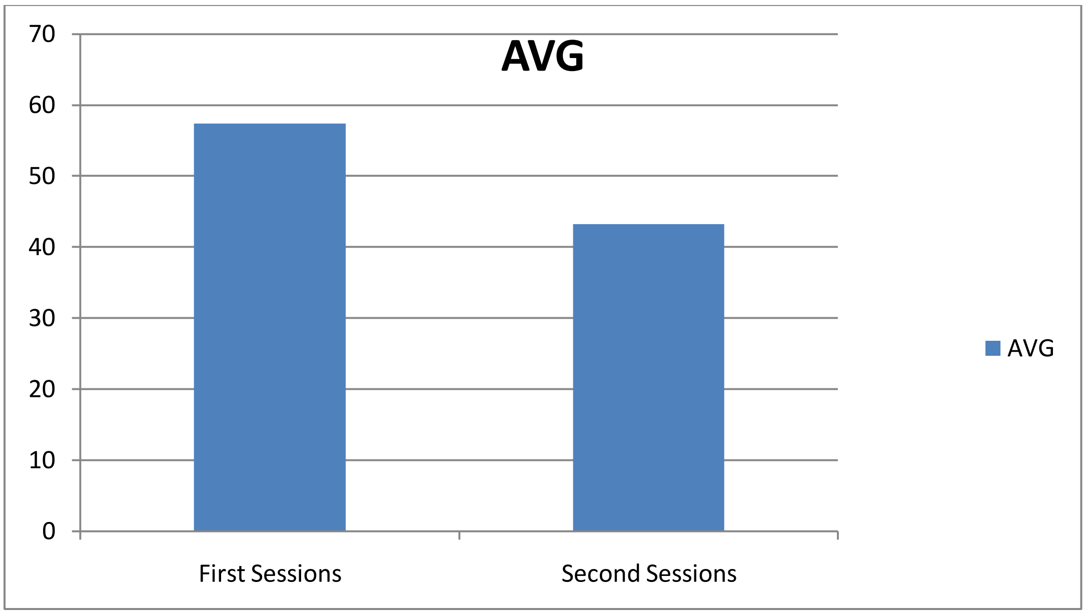


Figure: Average value of the 3 sessions throughput

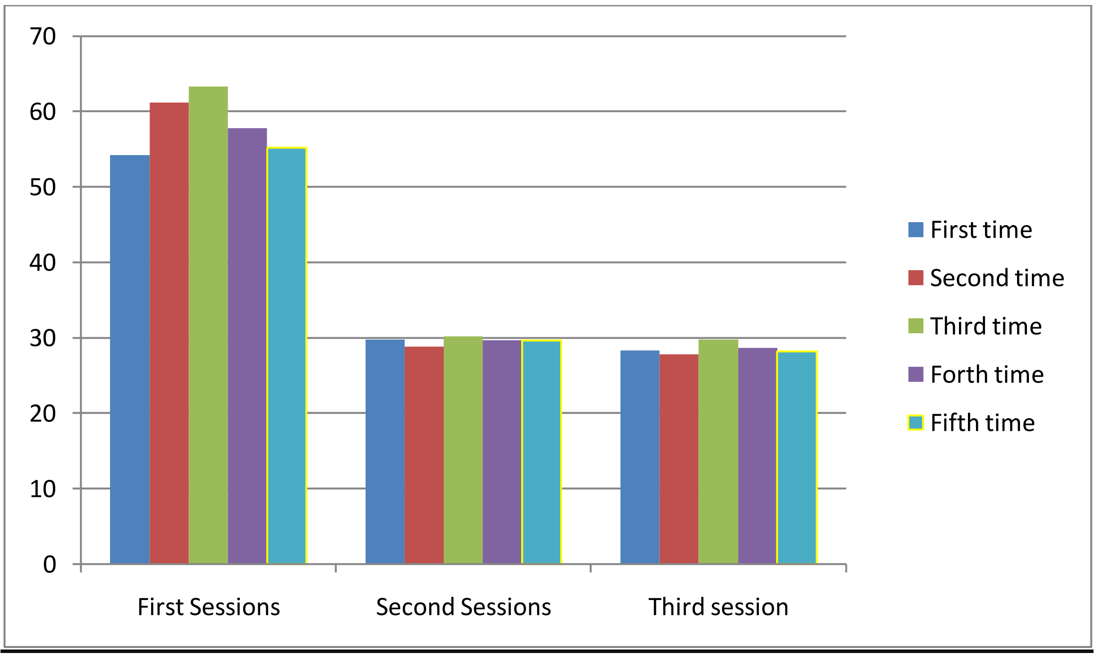


Figure: Graph of the throughput that run 3 TCP sessions on five time of test

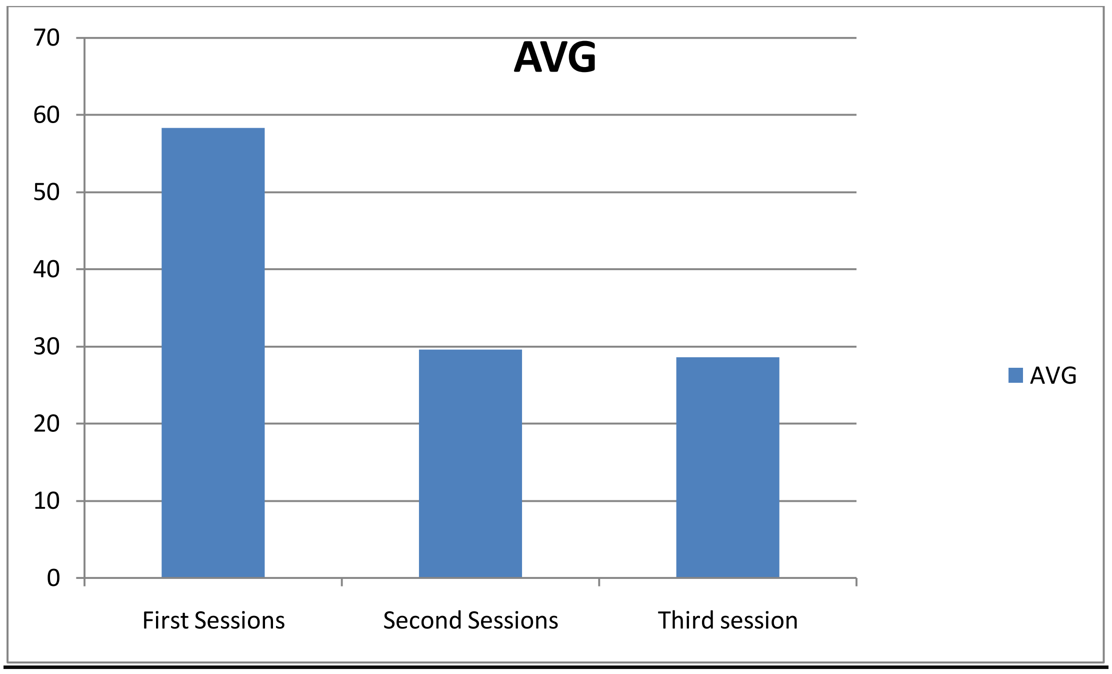


Figure: Average value of the 3 sessions throughput

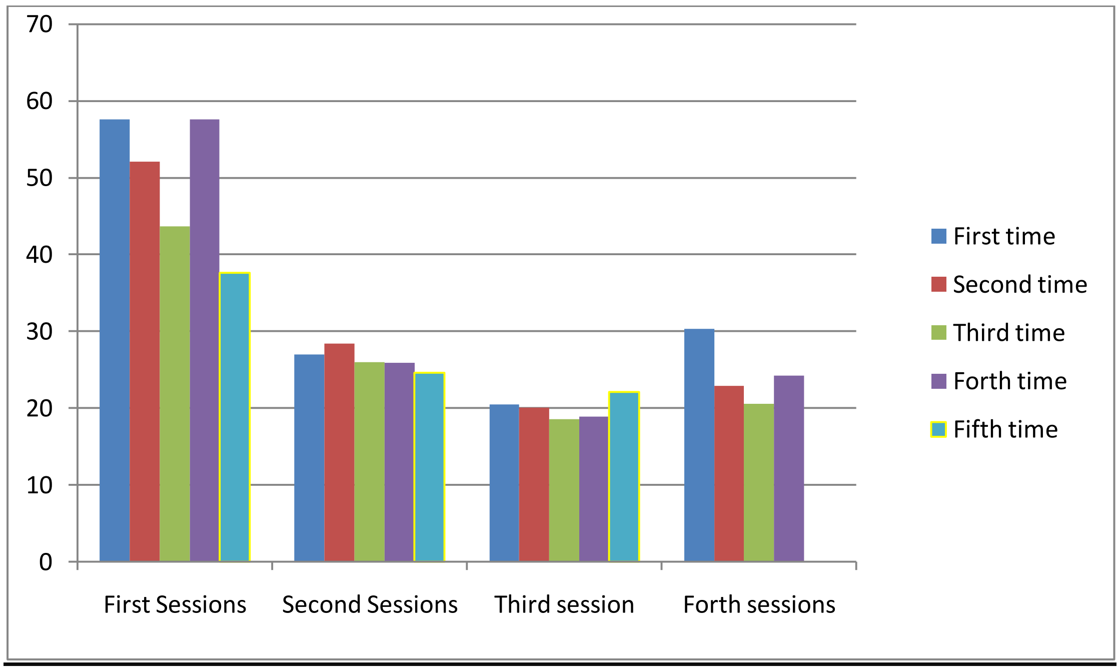


Figure: Graph of the throughput that run 4 TCP sessions on five time of test

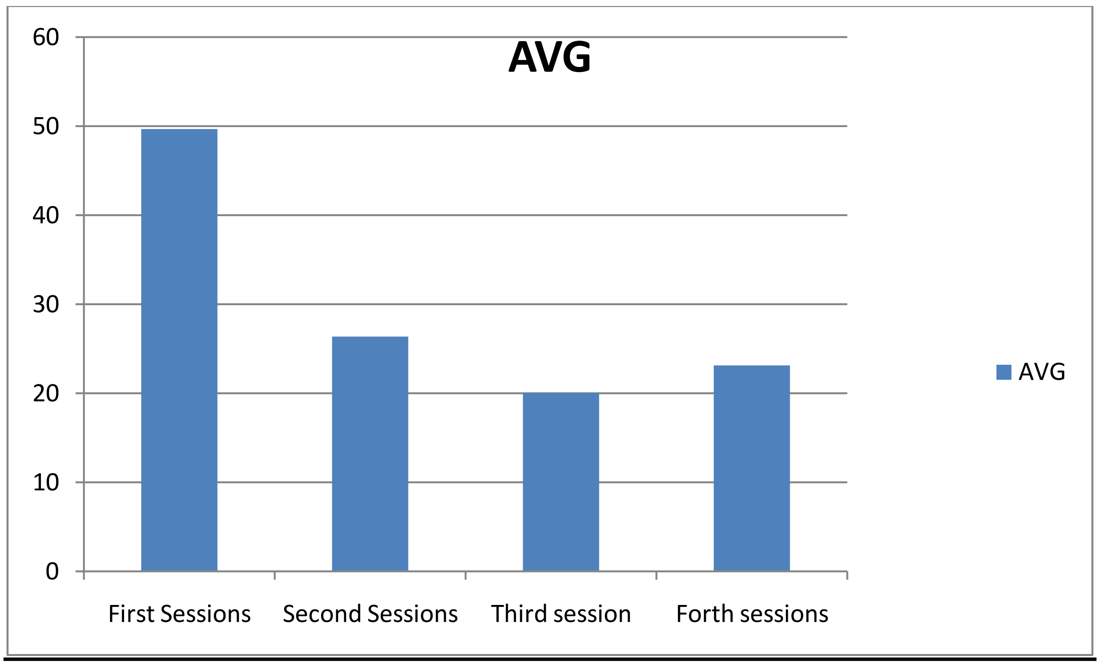


Figure: Average value of the 4 sessions throughput

Summary

In this experiment we use manually open and run the test on a single **iperf** server

And as you see from a graph as many sessions increased the more throughput drop down on the session that add up.

Experiment 3.1 Running multiple TCP sessions start at the same time

From the problem of the last experiment that the throughput will depend on the speed of the hand to run the test and it will not run at the same time. So, in this experiment we will use shell script file to run the test on the terminal that can help us to open and run the test at the same time

First , on the computer A doing the same run the TCP server using:

```
iperf -s
```

Now, use shell script file to run on the computer B to open many terminal use time = 60s to run the test:

```
gnome-terminal          --tab -e "iperf -c IP -t 60s"
```

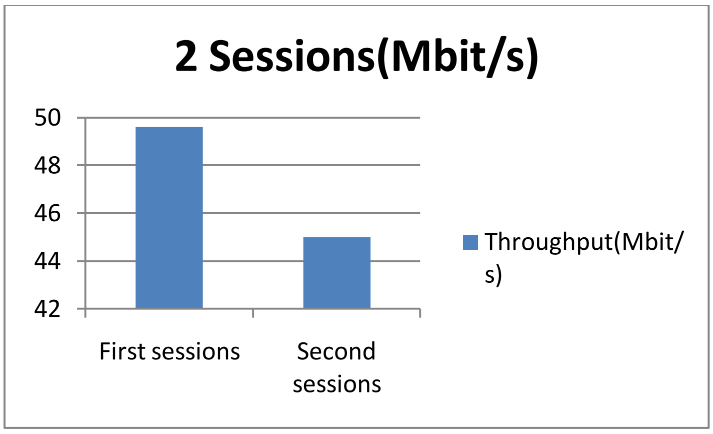
Then, write this command one more in script to add more sessions in the test:

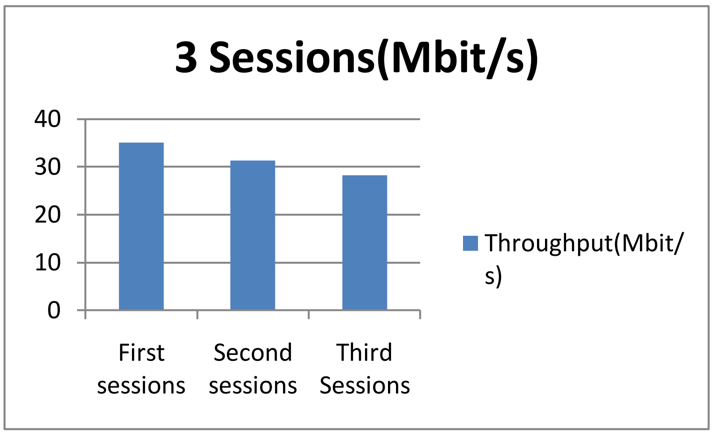
```
gnome-terminal          --tab -e "iperf -c IP -t 60s"

gnome-terminal          --tab -e "iperf -c IP -t 60s"
```

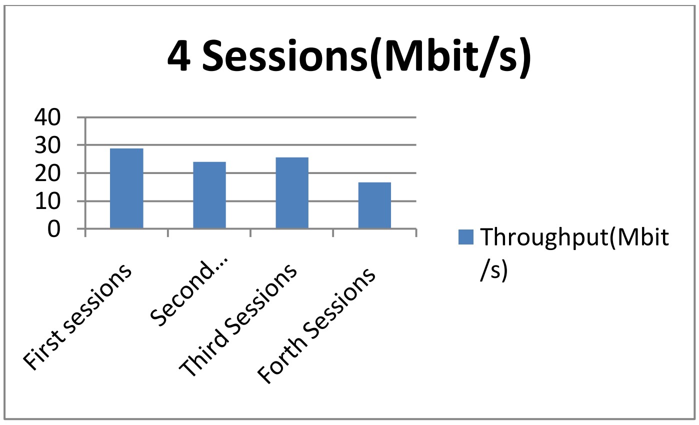
Save and then run it on terminal

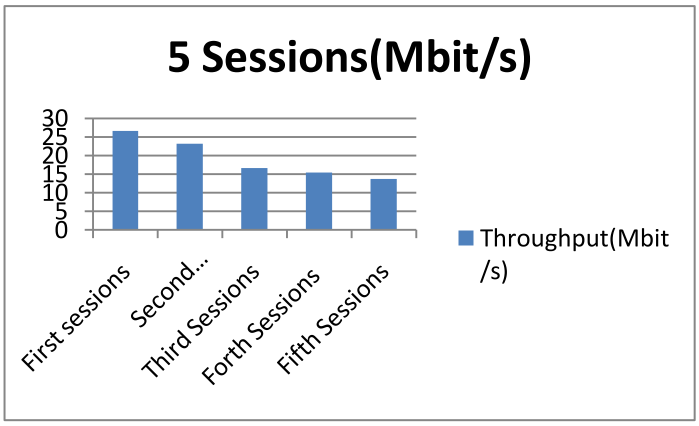
And also increase number of sessions by 1 until 20

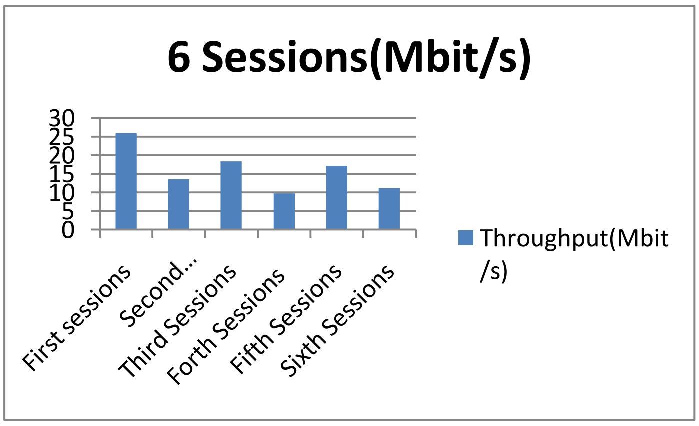
From the direction above we get result in the graph below

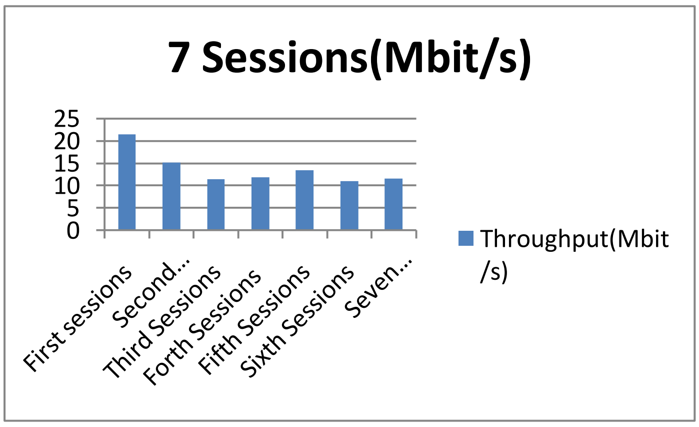


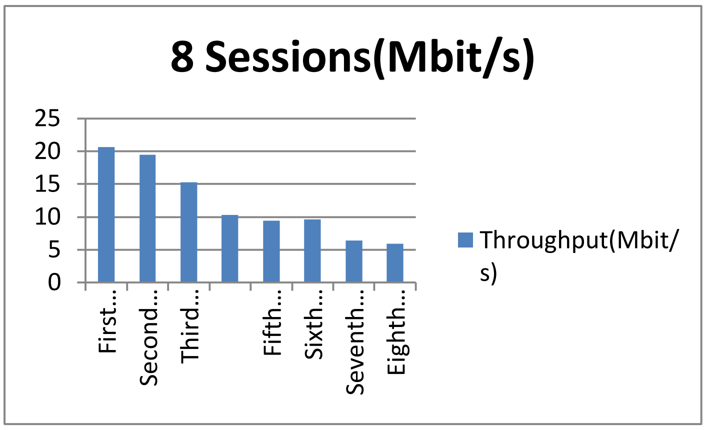


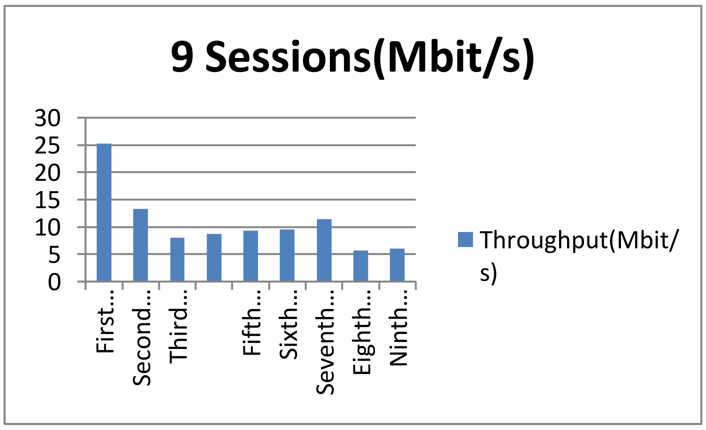

4 Sessions(Mbit/s)
40
30
20
10
0
First sessions
Second...
Third Sessions
Forth Sessions
Throughput(Mbit/s)

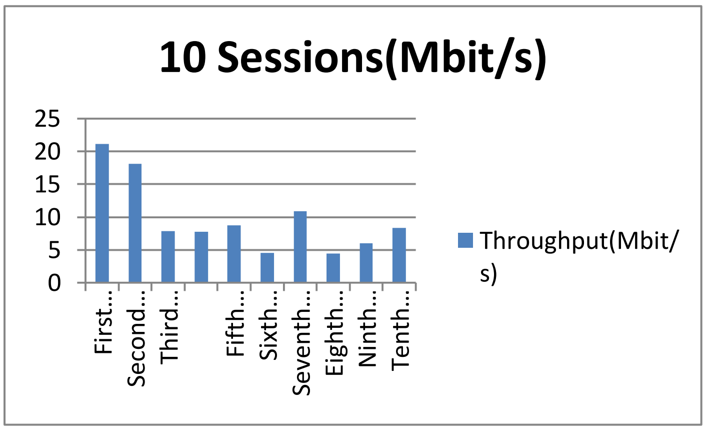

5 Sessions(Mbit/s)
30
25
20
15
10
5
0
First sessions
Second...
Third Sessions
Forth Sessions
Fifth Sessions
Throughput(Mbit/s)

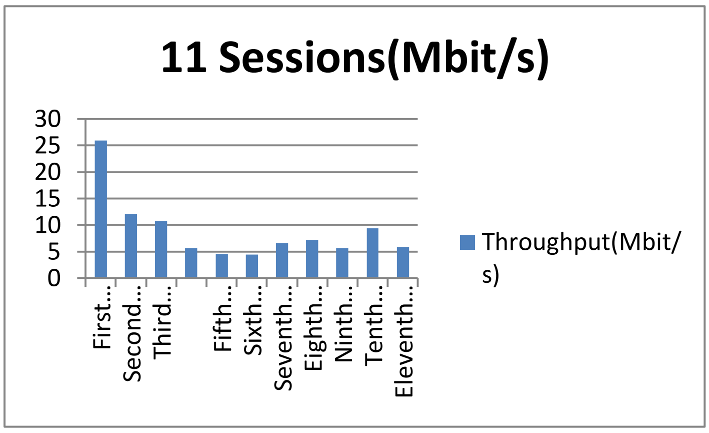

6 Sessions(Mbit/s)
30
25
20
15
10
5
0
First sessions
Second...
Third Sessions
Forth Sessions
Fifth Sessions
Sixth Sessions
Throughput(Mbit/s)

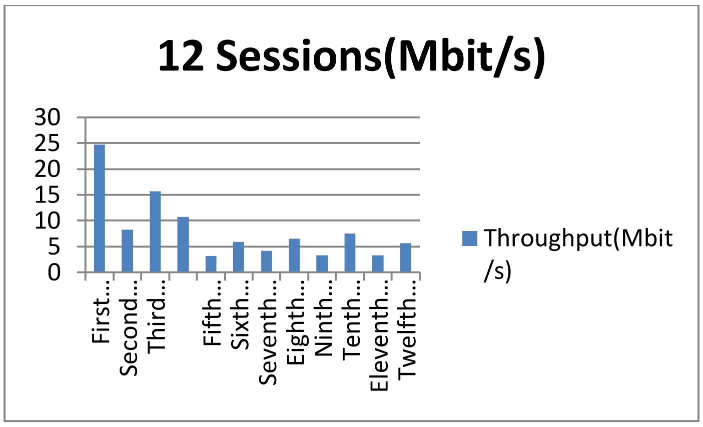

7 Sessions(Mbit/s)
25
20
15
10
5
0
First sessions
Second...
Third Sessions
Forth Sessions
Fifth Sessions
Sixth Sessions
Seven...
Throughput(Mbit/s)

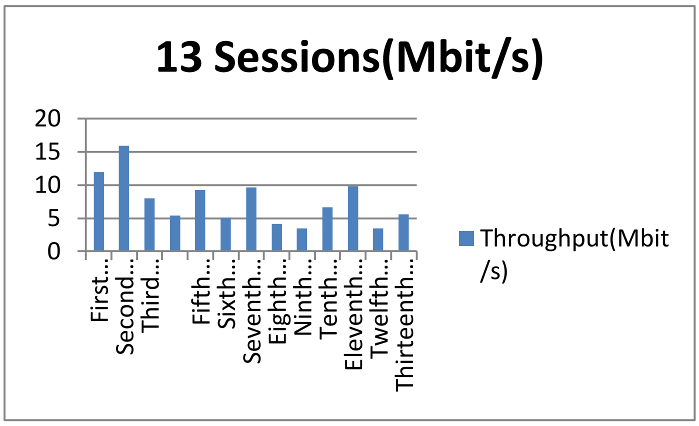

8 Sessions(Mbit/s)
25
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Throughput(Mbit/s)

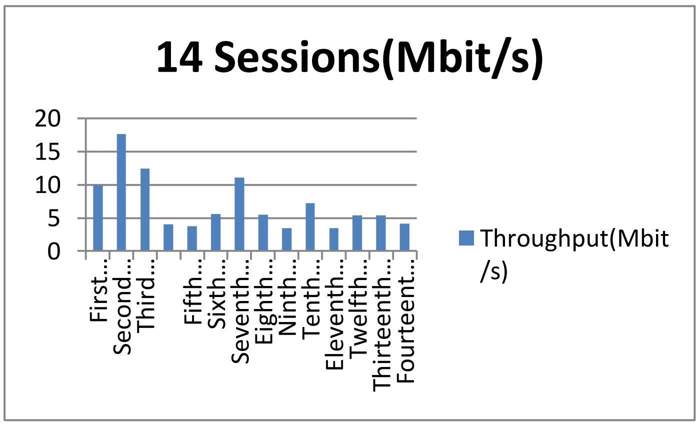

9 Sessions(Mbit/s)
30
25
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Ninth…
Throughput(Mbit/s)

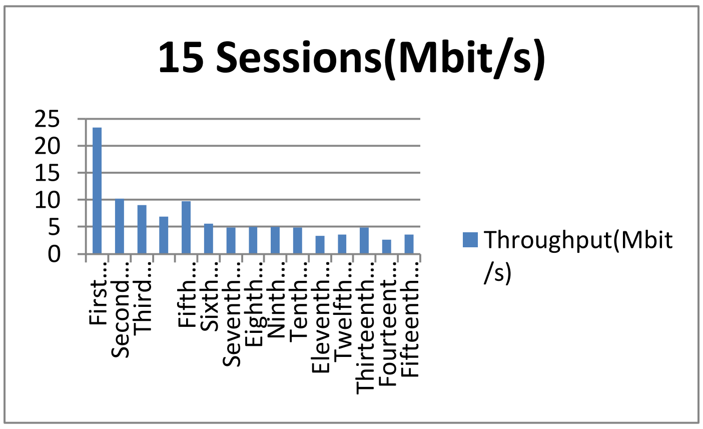

10 Sessions(Mbit/s)
25
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Ninth…
Tenth…
Throughput(Mbit/s)

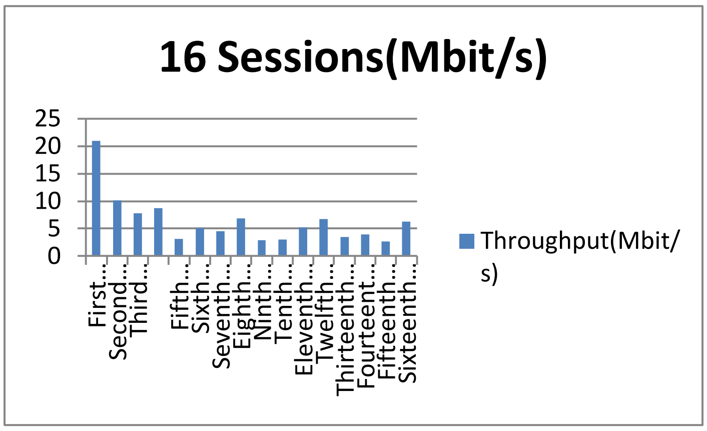

11 Sessions(Mbit/s)
30
25
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Ninth…
Tenth…
Eleventh…
Throughput(Mbit/s)

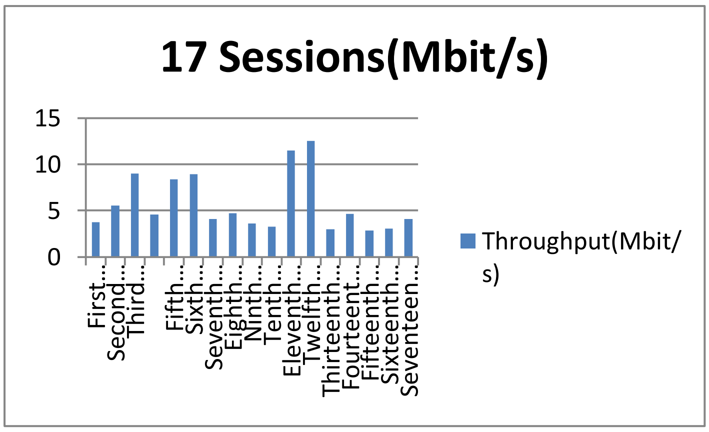

12 Sessions(Mbit/s)
30
25
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Ninth…
Tenth…
Eleventh…
Twelfth…
Throughput(Mbit/s)

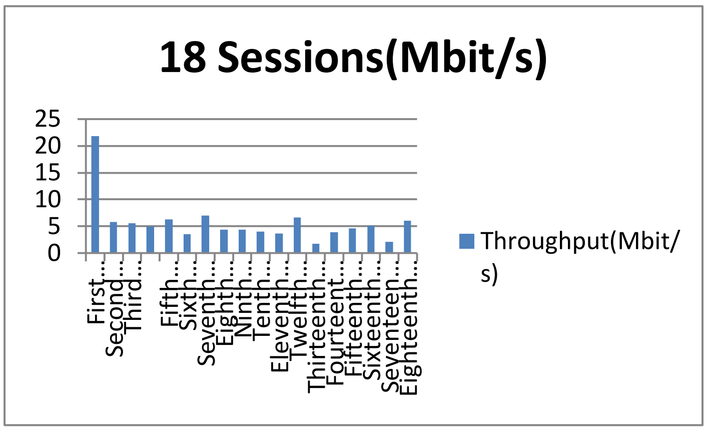

13 Sessions(Mbit/s)
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Ninth…
Tenth…
Eleventh…
Twelfth…
Thirteenth…
Throughput(Mbit/s)

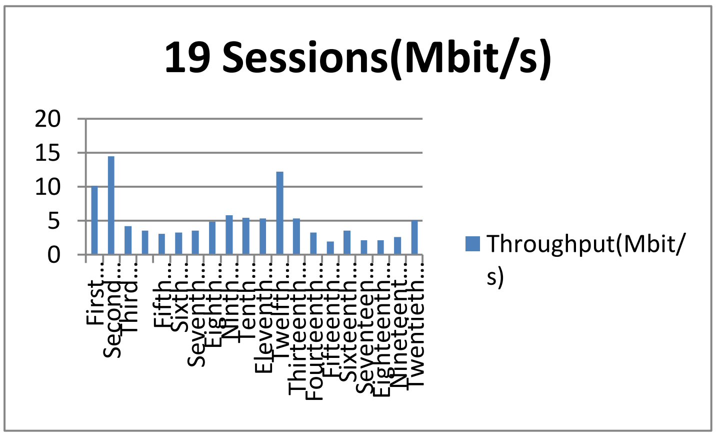

14 Sessions(Mbit/s)
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Ninth…
Tenth…
Eleventh…
Twelfth…
Thirteenth…
Fourteent…
Throughput(Mbit/s)


15 Sessions(Mbit/s)
25
20
15
10
5
0
First…
Second…
Third…
Fifth…
Sixth…
Seventh…
Eighth…
Ninth…
Tenth…
Eleventh…
Twelfth…
Thirteenth…
Fourteent…
Fifteenth…
Throughput(Mbit/s)


16 Sessions(Mbit/s)
25
20
15
10
5
0
First...
Second...
Third...
Fifth...
Sixth...
Seventh...
Eighth...
Ninth...
Tenth...
Eleventh...
Twelfth...
Thirteenth...
Fourteent...
Fifteenth...
Sixteenth...
Throughput(Mbit/s)


17 Sessions(Mbit/s)
15
10
5
0
First...
Second...
Third...
Fifth...
Sixth...
Seventh...
Eighth...
Ninth...
Tenth...
Eleventh...
Twelfth...
Thirteenth...
Fourteent...
Fifteenth...
Sixteenth...
Seventeen...
Throughput(Mbit/s)


18 Sessions(Mbit/s)
25
20
15
10
5
0
First...
Second...
Third...
Fifth...
Sixth...
Seventh...
Eighth...
Ninth...
Tenth...
Eleventh...
Twelfth...
Thirteenth...
Fourteent...
Fifteenth...
Sixteenth...
Seventeen...
Eighteenth...
Throughput(Mbit/s)


19 Sessions(Mbit/s)
20
15
10
5
0
First...
Second...
Third...
Fifth...
Sixth...
Seventh...
Eighth...
Ninth...
Tenth...
Eleventh...
Twelfth...
Thirteenth...
Fourteenth...
Fifteenth...
Sixteenth...
Seventeen...
Eighteenth...
Nineteent...
Twentieth...
Throughput(Mbit/s)

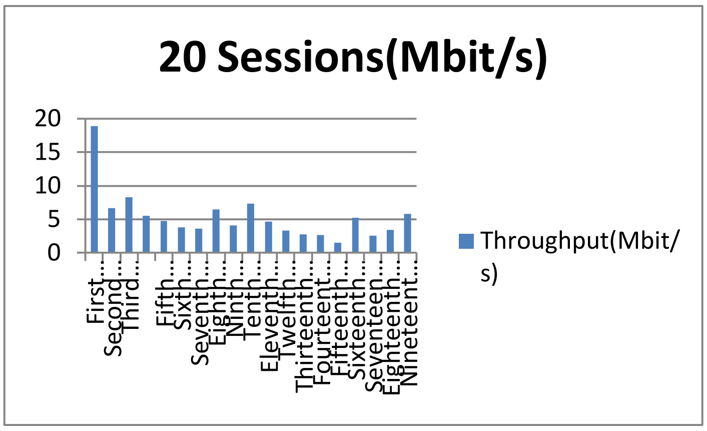


Figure: Graph that show the throughput of each number of TCP sessions

Summary

In this experiment we run multiple TCP sessions for 2 sessions - 20 sessions in the same time running and it’s show that in TCP mechanism will try to divide each sessions for fairness but it’s not fair for all 100%

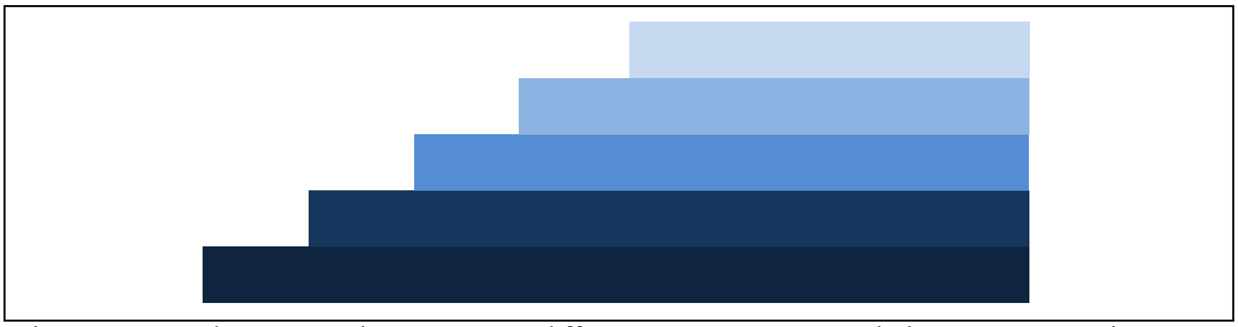
<u>Experiment 3.3</u> Multiple TCP sessions running on the different starting time

In this experiment, we design this experiment to make the different from the last experiment to get the different value of throughput. So, first we will do the same as every experiment

Computer A run TCP server:

```
iperf -s
```

Now, on computer B run the shell script file follow on this picture



That is we need to run each sessions on different starting time and also increase each sessions by 1

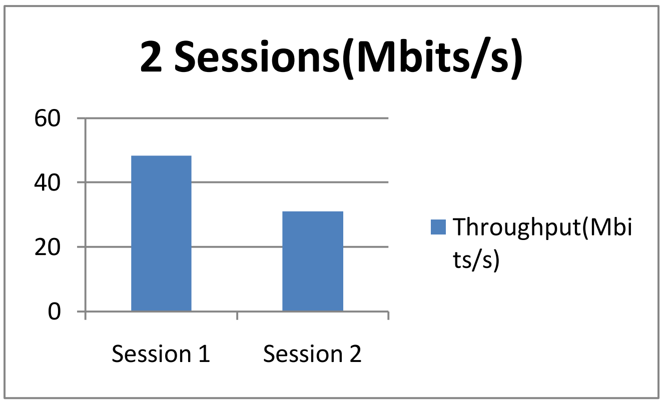
For the first time we start at a sessions set the first sessions to start immediately for 60 second and for the second sessions will start 10 second after the beginning of the first sessions for 50 second

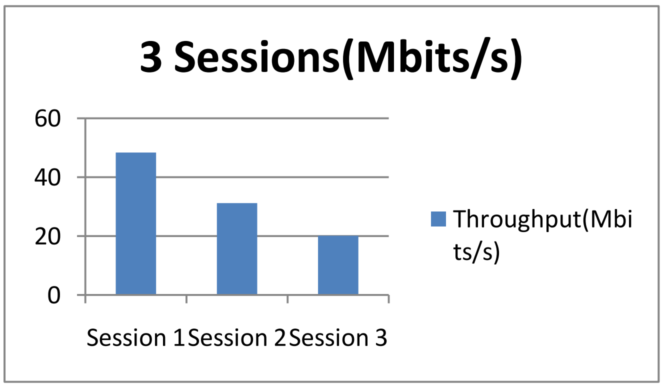
```
gnome-terminal          --tab -e "iperf -c IP -t 60s"

sleep 10s

gnome-terminal          --tab -e "iperf -c IP -t 50s"
```

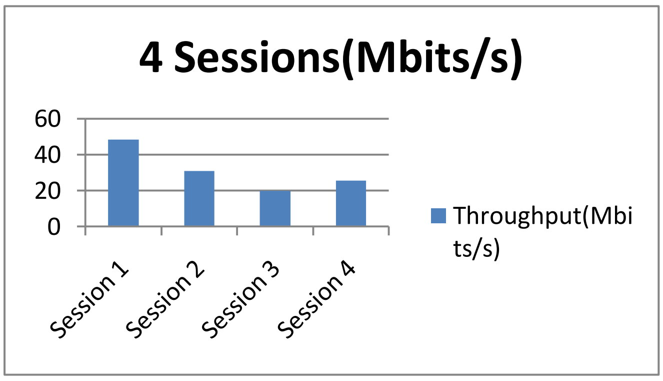
Then we need to increase each session by session and also increase the sleep time between each session and decrease the time for running on that sessions

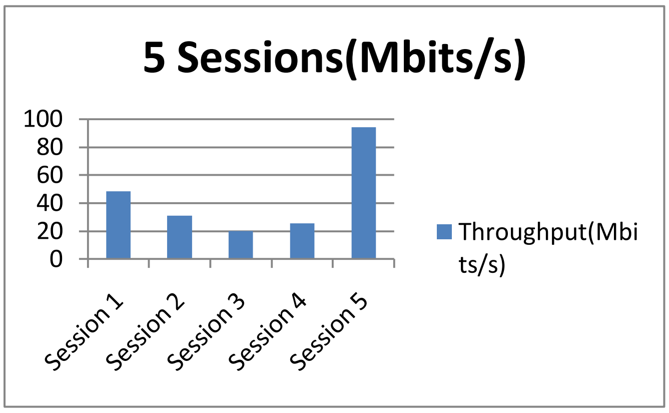
```
gnome-terminal          --tab -e "iperf -c IP -t 60s"

sleep 10s

gnome-terminal          --tab -e "iperf -c IP -t 50s"

sleep 20s

gnome-terminal          --tab -e "iperf -c IP -t 40s"
```

After experiment above we can get the result in the graph below









Summary

From this experiment we design a new experiment and we can get the result from the graph that it will have the higher throughput on the sessions as we compare from the last experiment so, it will cause of the sessions didn't run at the same time and it's will have some period that some sessions will have a single running or maybe the less sessions run in the same interval that lead to the higher throughput in some sessions.

Experiment 4 TCP and UDP sessions

In this experiment we investigate the bandwidth of TCP and UDP that run parallel to the server **Iperf** but have different parameter e.g. start time and end time in each experiment.

Experiment 4.1 Run TCP and UDP at the same time

In this experiment we try to run TCP sessions and UDP sessions together at the same time starting. So first we going to do is open TCP and UDP server with specific port on computer A

Starting TCP server on computer A with port 5001

```
Iperf -s  -p 5001
```

Also, starting UDP server on computer A with port 6001

```
Iperf -s -u -p 6001
```

So, what we going to do next is to run TCP session and UDP session on the same time so we use shell script file to solve this problem

We starting with one TCP and one UDP first:

```
gnome-terminal          --tab -e "iperf -c IP -p 5001 -t 60s"

gnome-terminal          --tab -e "iperf -c IP -u -p 6001 -b 100M -t 60s"
```

Save and run it on terminal

So, what we do next is just increased number of UDP one by one to the server

```
gnome-terminal          --tab -e "iperf -c IP -p 5001 -t 60s"

gnome-terminal          --tab -e "iperf -c IP -u -p 6001 -b 100M -t 60s"

gnome-terminal          --tab -e "iperf -c IP -u -p 6001 -b 100M -t 60s"
```

And graph below will show our result of this experiment

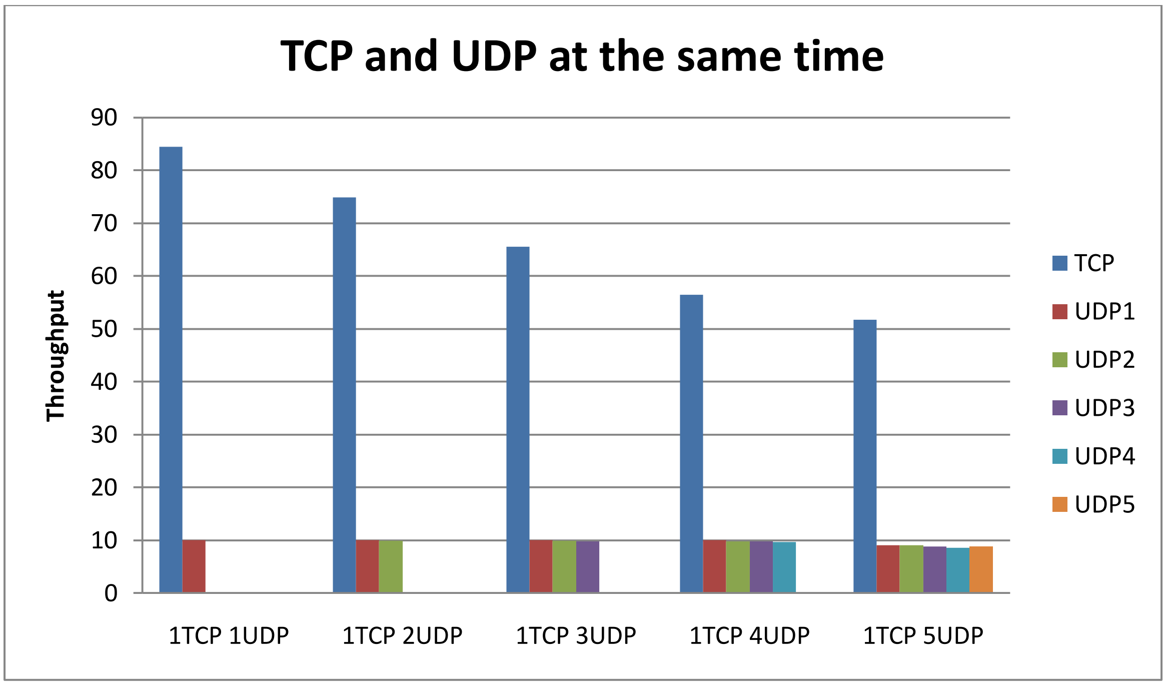


Figure : Graph that show the result of TCP sessions and UDP sessions sending at same time

<u>Summary</u>

From the graph when the clients in the UDP increase cause the linear decrease in bandwidth of TCP. The cause of decrease in bandwidth of TCP is because the clients in UDP have come in and have to share the sending rate to UDP.

<u>Experiment 4.2</u> Run TCP and UDP at different run time.

In this experiment we will run TCP sessions and UDP sessions together at the same time starting but different in time running. So first we going to do again is open TCP and UDP server with specific port on computer A

Starting TCP server on computer A with port 5001

```
Iperf -s  -p 5001
```

Starting UDP server on computer A with port 6001

```
Iperf -s -u -p 6001
```

So, next is to run TCP session and UDP session on the in different running time

We starting with one TCP and one UDP first:

```
gnome-terminal          --tab -e "iperf -c IP -p 5001 -t 60s"
gnome-terminal          --tab -e "iperf -c IP -u -p 6001 -b 100M -t 30s"
```

Save and run it on terminal

So, increased number of UDP one by one to the server like the first experiment

```
gnome-terminal          --tab -e "iperf -c IP -p 5001 -t 60s"
gnome-terminal          --tab -e "iperf -c IP -u -p 6001 -b 100M -t 30s"
gnome-terminal          --tab -e "iperf -c IP -u -p 6001 -b 100M -t 30s"
```

And graph below will show our experiment result

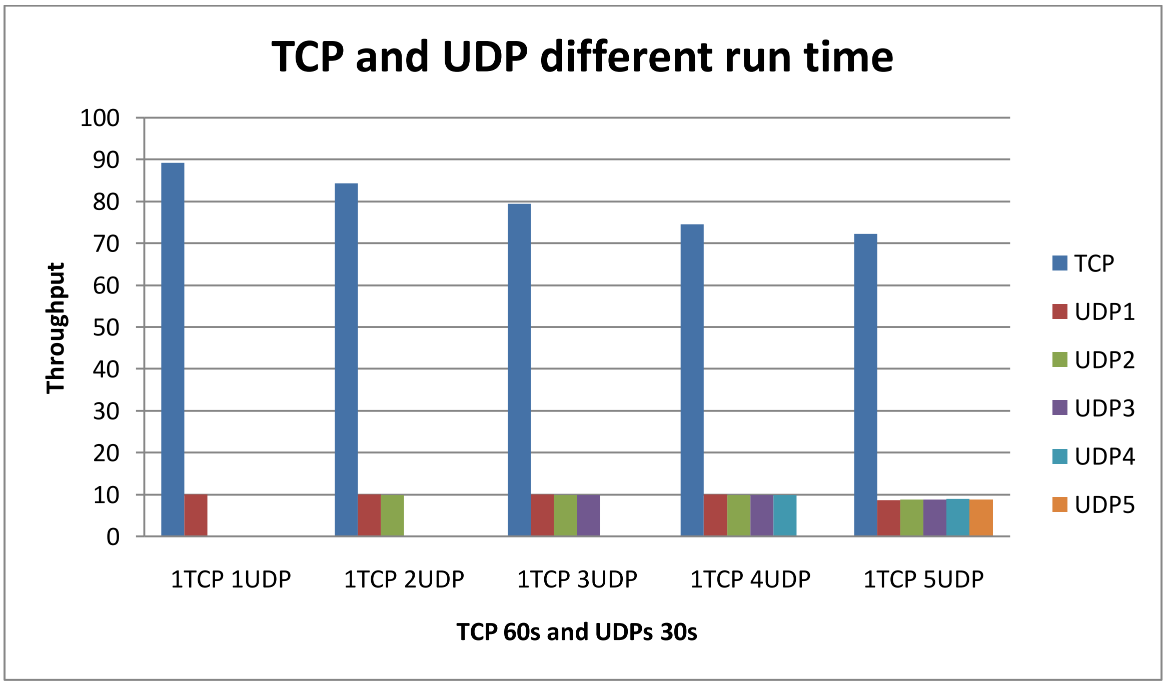


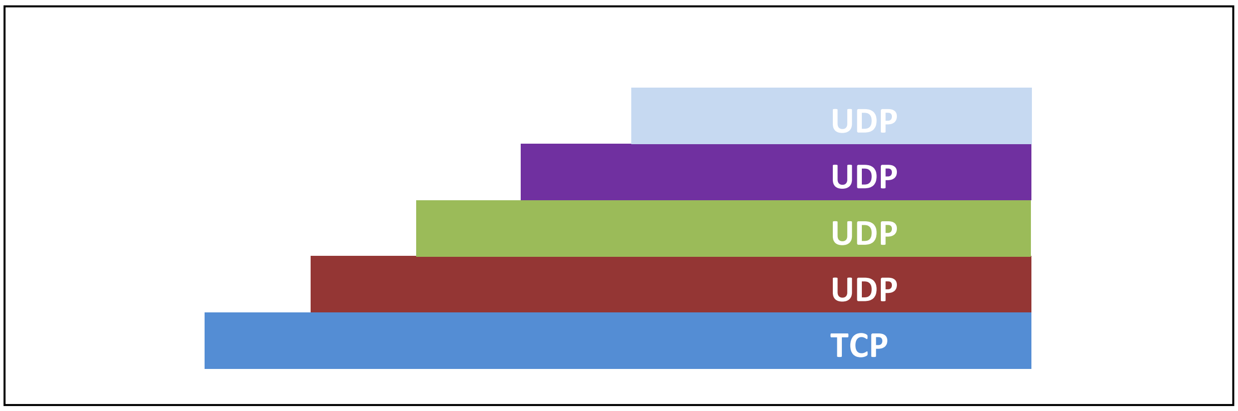
Figure : Graph that show the result of TCP sessions and UDP sessions sending at same time starting but different time running

Summary

From the graph, bandwidth of TCP is higher than first experiment because UDP client only run for 30 second but still cause linear decrease in bandwidth for TCP. TCP have high bandwidth than the previous experiment because after 30 second of UDP client the transfer has stop and TCP can run in full data rate for another 30 sec remain.

Experiment 4.3 TCP and UDP start with different time.

In this experiment we also do everything same as last two experiment but in this experiment we want to run TCP sessions and UDP sessions at different starting time. So, we design this experiment to be like this picture



But we just increase UDP sessions one by one session. So, after we open TCP server and UDP server like last experiment we will use this shell script file to run

```
gnome-terminal            --tab -e "iperf -c  IP  -p 5001 -t 60s"
sleep 10s
gnome-terminal            --tab -e "iperf -c  IP  -u -p 6001 -b 100M -t 50s"
```

save and run it , and also increase UDP sessions one by one

```
gnome-terminal            --tab -e "iperf -c 192.168.1.248 -p 5001 -t 60s"
sleep 10s
gnome-terminal            --tab -e "iperf -c 192.168.1.248 -u -p 6001 -b 100M -t 50s"
sleep 20s
gnome-terminal            --tab -e "iperf -c 192.168.1.248 -u -p 6001 -b 100M -t 40s"
```

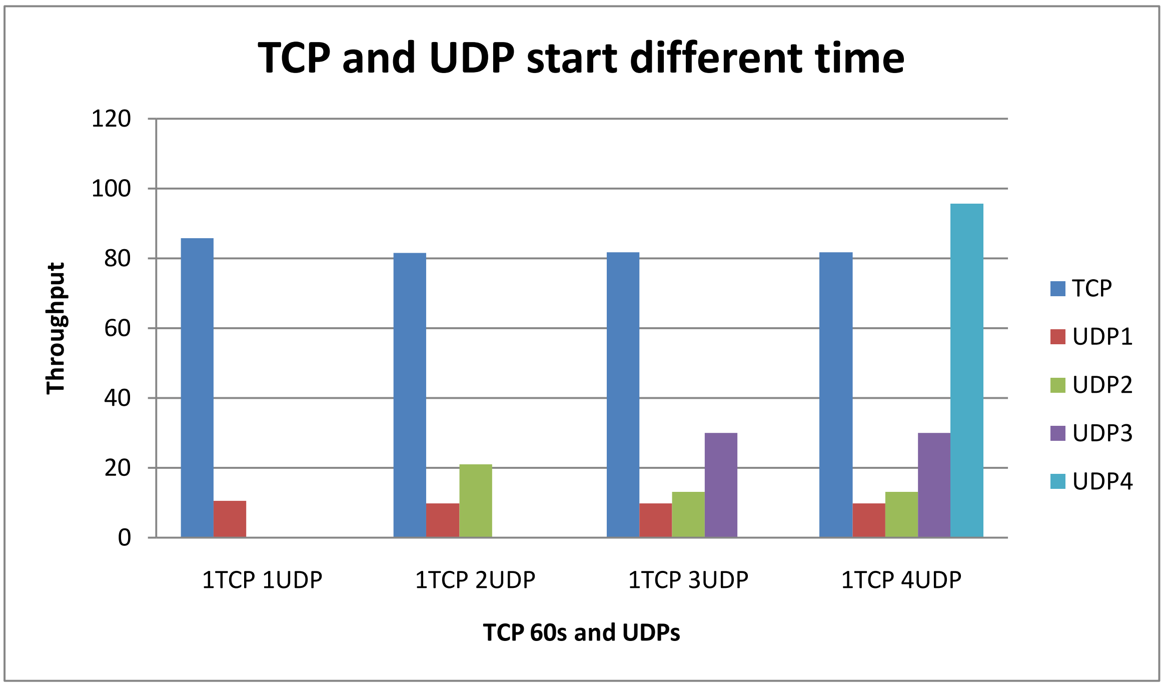


Figure : Graph that show the result of TCP sessions and UDP sessions sending at different time

<u>Summary</u>

From the graph show that the throughput of TCP was increase it maybe cause of the interval that we have to share for UDP was decrease and TCP can send at the full rate on that interval so that lead to the higher Throughput on TCP

<u>Conclusion</u>

For all of the experiment that our group have perform in “TCP vs UDP sessions” we can conclude that if we sent TCP sessions and UDP sessions at the same time TCP will share with UDP for fare it maybe cause of TCP will guarantee the correctness of arriving data but UDP didn’t guarantee the arriving data just sending packet

APPENDIX

TCP vs UDP

ex 1.1

UDP	AVG	1st	2nd	3rd	4th
100ms	23.875	21.6	26.9	24	23
200ms	22.575	20.6	21.9	24.1	23.7
300ms	23.275	20.6	24.3	24.3	23.9
400ms	21.2	20.8	22.9	20	21.1
500ms	23.55	24.3	24	22.9	23
600ms	22.575	23.2	23	20.7	23.4
700ms	19.5	20.2	18.7	20.1	19
800ms	17.625	17.6	17.6	17.4	17.9
900ms	15.55	14.7	15.8	15.8	15.9
1000ms	14.125	14	14	14	14.5
1100ms	13.025	12.9	13.1	13	13.1
1200ms	11.975	11.7	12.2	11.8	12.2
1300ms	10.975	10.3	11.2	11.2	11.2
1400ms	10.4775	10.4	10.5	10.71	10.3
1500ms	9.8175	10.1	9.81	9.57	9.79
1600ms	9.32	9.4	9.35	9.13	9.4
1700ms	8.5125	8.41	8.41	8.7	8.53
1800ms	7.845	7.7	8.06	7.7	7.92
1900ms	8.195	8.11	8.25	8.1	8.32
2000ms	7.175	7.01	7.01	7.48	7.2

TCP	AVG	1st	2nd	3rd	4th
100ms	12.375	13.4	11.7	12	12.4
200ms	10.3625	13.1	11.1	7.74	9.51
300ms	8.165	9.43	9.29	7.18	6.76
400ms	5.7975	6.67	8.08	2.16	6.28
500ms	4.005	4.97	3.91	2.4	4.74
600ms	2.98	3.91	2.86	1.45	3.7
700ms	2.4975	2.3	2.48	2.2	3.01
800ms	3.335	1.9	2.06	1.88	7.5
900ms	1.5225	1.52	1.54	1.56	1.47
1000ms	1.0365	0.757	0.919	1.24	1.23
1100ms	0.91	0.946	0.907	1.04	0.747
1200ms	0.8805	0.855	0.86	0.949	0.858
1300ms	0.669	0.559	0.712	0.696	0.709
1400ms	0.6595	0.656	0.657	0.663	0.662
1500ms	0.6065	0.609	0.608	0.604	0.605

1600ms	0.5475	0.486	0.568	0.569	0.567
1700ms	0.52475	0.526	0.526	0.521	0.526
1800ms	0.4965	0.496	0.497	0.497	0.496
1900ms	0.46975	0.471	0.472	0.465	0.471
2000ms	0.43425	0.435	0.435	0.435	0.432

ex1.2

TCP	AVG	1st	2nd	3rd	4th
100ms	10.6675	12.8	9.2	10.8	9.87
200ms	8.35	13.6	6.6	6.74	6.46
300ms	6.0275	12.6	4.18	2.78	4.55
400ms	5.1625	12.6	3.04	2.55	2.46
500ms	4.2275	11.8	2.03	1.1	1.98
600ms	3.805	11.2	1.42	1.25	1.35
700ms	3.5775	10.8	1.08	1.25	1.18
800ms	4.022	13.5	0.919	0.741	0.928
900ms	3.63175	12.2	0.797	0.674	0.856
1000ms	2.64175	8.68	0.569	0.755	0.563
1100ms	1.7615	5.24	0.599	0.602	0.605
1200ms	1.421	4.05	0.606	0.553	0.475
1300ms	0.95875	2.43	0.402	0.563	0.44
1400ms	0.70875	1.64	0.443	0.339	0.413
1500ms	0.6965	1.53	0.436	0.433	0.387
1600ms	0.479025	0.866	0.363	0.3231	0.364
1700ms	0.47075	0.842	0.307	0.35	0.384
1800ms	0.3565	0.62	0.258	0.258	0.29
1900ms	0.35075	0.593	0.22	0.245	0.345
2000ms	0.3465	0.602	0.261	0.262	0.261

UDP	AVG	1st	2nd	3rd	4th
100ms	21.95	18.9	24.3	20	24.6
200ms	21.85	15.3	23.1	23.3	25.7
300ms	22.525	22.2	24.4	20.7	22.8
400ms	22.05	18.3	24.5	21.1	24.3
500ms	22.8	22.1	22.2	20.5	26.4
600ms	21.8	19.5	21.4	23.2	23.1
700ms	19.975	19.8	20.2	19.7	20.2
800ms	15.925	13.7	17.8	17.7	14.5
900ms	15.5	14.6	15.8	15.8	15.8
1000ms	14.025	14	14	14.1	14
1100ms	12.975	12.7	13	13.1	13.1
1200ms	10.88	9.82	11.4	10.5	11.8
1300ms	10.8025	11.2	9.81	11	11.2
1400ms	9.825	8.3	10.4	10.3	10.3
1500ms	9.24	7.73	9.61	9.81	9.81
1600ms	8.975	9.22	8.49	9.48	8.71
1700ms	8.4825	8.41	8.41	8.61	8.5
1800ms	7.8325	8.07	7.38	8	7.88
1900ms	7.5975	8.81	5.61	7.63	8.34
2000ms	7.0275	7.02	7.04	7.03	7.02

Impact of packet drops on TCP performance

ex 2.1

	1st	2nd	3rd	4th	5th
5%	17.5	13.5	19.5	14.2	16.4
10%	3.7	3.06	3.89	2.48	2.91
15%	0.991	1.03	1.36	1.54	1.08
20%	0.584	0.457	0.304	0.678	0.7
25%	0.398	0.269	0.393	0.195	0.266
30%	0.214	0.27	0.316	0.162	0.157
35%	0.18	0.101	0.163	0.0933	0.157
40%	0.0316	0.0803	0.0636	0.0796	0.0684
45%	0.0262	0.0295	0.0766	0.0963	0.0406
46%	0.0326	0.0165	0.0241	0.0381	0.0296
47%	0.0417	0.0535	0.0417	0.0571	0.0364
48%	0.00973	0.0464	0.0344	0.031	0.0361
49%	0.019	0.0348	0.00829	0.0402	0.0454
50%	0.0511	0.0723	0.0338	0.0198	0.0168

ex 2.2

	1st	2nd	3rd	4th	5th
5%	10.3	11.4	9.95	8.8	7.04
10%	2.54	2.44	2.16	2.77	1.93
15%	1.33	1.14	0.891	0.668	1.33
20%	0.427	0.687	0.521	0.576	0.633
25%	0.34	0.429	0.39	0.374	0.351
30%	0.28	0.272	0.177	0.309	0.265
35%	0.109	0.128	0.122	0.167	0.0947
40%	0.0803	0.0758	0.0439	0.0688	0.0554
45%	0.0332	0.0457	0.0326	0.0407	0.0495
46%	0.0291	0.065	0.0557	0.0152	0.0581
47%	0.0727	0.0939	0.0346	0.047	0.045
48%	0.0233	0.0482	0.0551	0.0181	0.0453
49%	0.0409	0.0287	0.0356	0.0149	0.0399
50%	0.0172	0.038	0.0214	0.0154	0.0143

ex 2.3

	1st	2nd	3rd	4th
5%	15.6	16.8	15.7	15.1
10%	3.07	3.01	2.98	3.04
15%	1.38	1.04	1.07	1.31
20%	0.524	0.573	0.6	0.582
25%	0.313	0.314	0.352	0.365
30%	0.208	0.191	0.233	0.194
35%	0.138	0.128	0.0665	0.168
40%	0.0134	0.0924	0.0753	0.0377
45%	0.0275	0.0386	0.0636	0.0512
50%	0.0628	0.0081	0.0408	0.035
51%	0.0159	0.00478	0.0308	0.0317
52%	0.0299	0.024	0.0356	0.009
53%	0.0378	0.0284	0.0307	0.0197
54%	0.0252	0.0284	0.00663	0.0221
55%	0.0349	0.014	0.00368	0.0152

ex 2.4

	1st	2nd	3rd	4th
5%	9.68	8.89	9.83	9.14
10%	2.29	2.6	2.33	2.9
15%	0.97	1.22	1.06	0.984

20%	0.521	0.526	0.39	0.451
25%	0.286	0.341	0.28	0.329
30%	0.189	0.245	0.21	0.157
35%	0.0865	0.183	0.17	0.125
40%	0.068	0.102	0.0528	0.0649
45%	0.0495	0.0235	0.0484	0.0422
50%	0.038	0.0188	0.0214	0.0213
51%	0.0128	0.0312	0.00594	0.00448
52%	0.0223	0.0115	0.0331	0.0353
53%	0.0182	0.00795	0.0164	0.0171
54%	0.0243	0.0165	0.0133	0.00967
55%	0.00586	0.00278	0.0253	0.0145

Multiple TCP sessions

ex 3.1

	First sessions(Mbits/sec)	Second sessions(Mbits/sec)
First time	58.1	42.2
Second time	54.6	44.1
Third time	59.1	43.2
Forth time	56.5	43.2
Fifth time	58.6	43.4

	First sessions(Mbits/sec)	Second sessions(Mbits/sec)	Third sessions(Mbits/sec)
First time	54.2	29.8	28.3
Second time	61.2	28.8	27.8
Third time	63.3	30.2	29.8
Forth time	57.8	29.7	28.7
Fifth time	55.2	29.6	28.2

	First sessions(Mbits/sec)	Second sessions(Mbits/sec)	Third sessions(Mbits/sec)	Forth sessions(Mbits/sec)
First time	57.6	27.0	20.5	30.3

Second time	52.1	28.4	20.1	22.9
Third time	43.7	26.0	18.6	20.6
Forth time	57.6	25.9	18.9	24.2
Fifth time	37.6	24.6	22.1	17.5

ex 3.2

	Throughput(Mbit/s)
First sessions	49.6
Second sessions	45.0

	Throughput(Mbit/s)
First sessions	35.1
Second sessions	31.3
Third Sessions	28.2

	Throughput(Mbit/s)
First sessions	28.9
Second sessions	24
Third Sessions	25.7
Forth Sessions	16.7

	Throughput(Mbit/s)
First sessions	26.7
Second sessions	23.2
Third Sessions	16.6
Forth Sessions	15.4
Fifth Sessions	13.8

	Throughput(Mbit/s)
First sessions	25.9
Second sessions	13.5
Third Sessions	18.4
Forth Sessions	9.68
Fifth Sessions	17.1

Sixth Sessions	11.2

	Throughput(Mbit/s)
First sessions	21.4
Second sessions	15.1
Third Sessions	11.5
Forth Sessions	11.8
Fifth Sessions	13.4
Sixth Sessions	11
Seven Sessions	11.6

	Throughput(Mbit/s)
First sessions	20.6
Second sessions	19.5
Third Sessions	15.3
Forth Sessions	10.3
Fifth Sessions	9.46
Sixth Sessions	9.63
Seventh Sessions	6.38
Eighth Sessions	5.95

	Throughput(Mbit/s)
First sessions	25.2
Second sessions	13.3
Third Sessions	8.02
Forth Sessions	8.69
Fifth Sessions	9.35
Sixth Sessions	9.54
Seventh Sessions	11.4
Eighth Sessions	5.63
Ninth Sessions	5.98

	Throughput(Mbit/s)
First sessions	21.1
Second sessions	18.1
Third Sessions	7.9
Forth Sessions	7.8
Fifth Sessions	8.72
Sixth Sessions	4.49
Seventh Sessions	10.9
Eighth Sessions	4.43
Ninth Sessions	6.02

Tenth Sessions	8.33

	Throughput(Mbit/s)
First sessions	25.9
Second sessions	12.0
Third Sessions	10.7
Forth Sessions	5.62
Fifth Sessions	4.55
Sixth Sessions	4.44
Seventh Sessions	6.59
Eighth Sessions	7.22
Ninth Sessions	5.63
Tenth Sessions	9.43
Eleventh Sessions	5.86

	Throughput(Mbit/s)
First sessions	24.7
Second sessions	8.26
Third Sessions	15.7
Forth Sessions	10.7
Fifth Sessions	3.15
Sixth Sessions	5.92
Seventh Sessions	4.22
Eighth Sessions	6.56
Ninth Sessions	3.26
Tenth Sessions	7.54
Eleventh Sessions	3.23
Twelfth Sessions	5.68

	Throughput(Mbit/s)
First sessions	11.9
Second sessions	15.9
Third Sessions	7.96
Forth Sessions	5.4
Fifth Sessions	9.25
Sixth Sessions	5.07
Seventh Sessions	9.65
Eighth Sessions	4.09
Ninth Sessions	3.44
Tenth Sessions	6.65
Eleventh Sessions	9.78
Twelfth Sessions	3.47
Thirteenth Sessions	5.55

	Throughput(Mbit/s)
First sessions	9.88
Second sessions	17.6
Third Sessions	12.4
Forth Sessions	4.08
Fifth Sessions	3.79
Sixth Sessions	5.55
Seventh Sessions	11.1
Eighth Sessions	5.52
Ninth Sessions	3.49
Tenth Sessions	7.24
Eleventh Sessions	3.48
Twelfth Sessions	5.38
Thirteenth Sessions	5.42
Fourteenth Sessions	4.12

	Throughput(Mbit/s)
First sessions	23.4
Second sessions	10.2
Third Sessions	9.06
Forth Sessions	6.87
Fifth Sessions	9.74
Sixth Sessions	5.53
Seventh Sessions	4.89
Eighth Sessions	5
Ninth Sessions	4.99
Tenth Sessions	4.87
Eleventh Sessions	3.27
Twelfth Sessions	3.57
Thirteenth Sessions	4.92
Fourteenth Sessions	2.57
Fifteenth Sessions	3.51

	Throughput(Mbit/s)
First sessions	21
Second sessions	10.1
Third Sessions	7.79
Forth Sessions	8.71
Fifth Sessions	3.13
Sixth Sessions	5.15

Seventh Sessions	4.47
Eighth Sessions	6.81
Ninth Sessions	2.89
Tenth Sessions	2.97
Eleventh Sessions	5.19
Twelfth Sessions	6.73
Thirteenth Sessions	3.53
Fourteenth Sessions	3.97
Fifteenth Sessions	2.68
Sixteenth Sessions	6.26

	Throughput(Mbit/s)
First sessions	3.78
Second sessions	5.57
Third Sessions	9.03
Forth Sessions	4.57
Fifth Sessions	8.37
Sixth Sessions	8.91
Seventh Sessions	4.12
Eighth Sessions	4.75
Ninth Sessions	3.63
Tenth Sessions	3.26
Eleventh Sessions	11.5
Twelfth Sessions	12.5
Thirteenth Sessions	2.97
Fourteenth Sessions	4.65
Fifteenth Sessions	2.83
Sixteenth Sessions	3.09
Seventeenth Sessions	4.08

	Throughput(Mbit/s)
First sessions	21.8
Second sessions	5.81
Third Sessions	5.59
Forth Sessions	4.97
Fifth Sessions	6.26
Sixth Sessions	3.56
Seventh Sessions	7.05
Eighth Sessions	4.35
Ninth Sessions	4.33
Tenth Sessions	4.05
Eleventh Sessions	3.64
Twelfth Sessions	6.67
Thirteenth Sessions	1.7

Fourteenth Sessions	3.89
Fifteenth Sessions	4.57
Sixteenth Sessions	5.08
Seventeenth Sessions	2.17
Eighteenth Sessions	6.05

	Throughput(Mbit/s)
First sessions	18.9
Second sessions	6.66
Third Sessions	8.32
Forth Sessions	5.52
Fifth Sessions	4.75
Sixth Sessions	3.79
Seventh Sessions	3.59
Eighth Sessions	6.43
Ninth Sessions	4.07
Tenth Sessions	7.32
Eleventh Sessions	4.68
Twelfth Sessions	3.32
Thirteenth Sessions	2.75
Fourteenth Sessions	2.65
Fifteenth Sessions	1.5
Sixteenth Sessions	5.22
Seventeenth Sessions	2.59
Eighteenth Sessions	3.44
Nineteenth Sessions	5.81

	Throughput(Mbit/s)
First sessions	10.1
Second sessions	14.5
Third Sessions	4.14
Forth Sessions	3.56
Fifth Sessions	3.06
Sixth Sessions	3.24
Seventh Sessions	3.51
Eighth Sessions	4.88
Ninth Sessions	5.79
Tenth Sessions	5.37
Eleventh Sessions	5.29
Twelfth Sessions	12.2
Thirteenth Sessions	5.31
Fourteenth Sessions	3.19
Fifteenth Sessions	1.93
Sixteenth Sessions	3.51

Seventeenth Sessions	2.08
Eighteenth Sessions	2.06
Nineteenth Sessions	2.55
Twentieth Sessions	5.07

ex 3.3

	Throughput(Mbit/s)
First sessions	55.8
Second sessions	38.5

	Throughput(Mbit/s)
First sessions	48.4
Second sessions	31.1
Third Sessions	26.2

	Throughput(Mbit/s)
First sessions	48.4
Second sessions	31.1
Third Sessions	20.1
Forth Sessions	25.6

	Throughput(Mbit/s)
First sessions	48.4
Second sessions	31.1
Third Sessions	20
Forth Sessions	25.7
Fifth Sessions	94.1

TCP sessions vs UDP sessions

ex 4.1

	TCP	UDP1	UDP2	UDP3	UDP4	UDP5
1TCP 1UDP	84.5	10				
1TCP 2UDP	74.9	10	9.9			
1TCP 3UDP	65.6	10	9.91	9.81		
1TCP 4UDP	56.5	10	9.86	9.76	9.64	
1TCP 5UDP	51.7	9.03	9.08	8.85	8.65	8.84

ex 4.2

	TCP	UDP1	UDP2	UDP3	UDP4	UDP5
1TCP 1UDP	89.2	9.99				
1TCP 2UDP	84.3	9.98	9.95			
1TCP 3UDP	79.4	9.98	9.94	9.91		
1TCP 4UDP	74.5	9.98	9.94	9.9	9.88	
1TCP 5UDP	72.3	8.66	8.85	8.74	8.98	8.87

ex 4.3

	TCP	UDP1	UDP2	UDP3	UDP4
1TCP 1UDP	85.7	10.5			
1TCP 2UDP	81.6	9.85	20.9		
1TCP 3UDP	81.7	9.8	13	29.9	
1TCP 4UDP	81.7	9.79	13	29.9	95.7