Networks Lab Report
Assignment 1

AIM of the Assignment : The main aim of this assignment is to get familiarized with mininet and become able to form and test custom topologies using python scripts. It also aims to differentiate between UDP and TCP transport protocols based on their measured throughputs while varying various parameters like delay, loss and bandwidth of the inner links.

Part 1

Objective

For the first part, we use two hosts are connected via a switch. Both the links from H1 to the switch and H2 to the switch have 1 Mbps bandwidth, 1 ms of propagation delay and no channel loss. Now first we form a TCP server - client with H2 as the server and H1 as the client, and check the H2's measured bandwidth as throughput.

Repeat the above for UDP making H2 as the UDP server and H1 as the UDP client and test the throughput for various bandwidths of the UDP client (H1) set bandwidths, namely the following :

I. 64 Kbps
II. 128 Kbps
III. 256 Kbps
IV. 512 Kbps
V. 1024 Kbps
VI. 2048 Kbps
VII. 4096 Kbps

Procedure

I. Assuming mininet is set up on the system in a virtualbox, first boot up mininet from the oracle vm virtual box, now type the following and note the ip for eth0.

```
>ifconfig
```

Now, ssh login to your virtual mininet using local terminal (to create xterms for h1 and h2), use ip you got from previous step from mininet terminal. And type the following for desired topology for part 1:

>sudo mn --link tc,bw=1,delay=1ms,loss=0

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(1.00Mbit 1ms delay 0% loss) (1.00Mbit 1ms delay 0% loss) (h1, s1) (1.00Mbit 1ms
 delay 0% loss) (1.00Mbit 1ms delay 0% loss) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(1.00Mbit 1ms delay 0% loss) (1.00Mbit 1ms delay 0% loss)
```

Setting bandwidth to 1Mbps, delay to 1ms and loss to 0 % for all links

Now, from your terminal use

>xterm h2

And enter the following command for creating TCP server in H2

>Iperf -s

```
root@mininet-vm:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 14] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 56484
[ ID] Interval       Transfer     Bandwidth
[ 14]  0.0-19.7 sec  2.25 MBytes   957 Kbits/sec
```

This is now our TCP server.

Now open the client by typing the following in your terminal

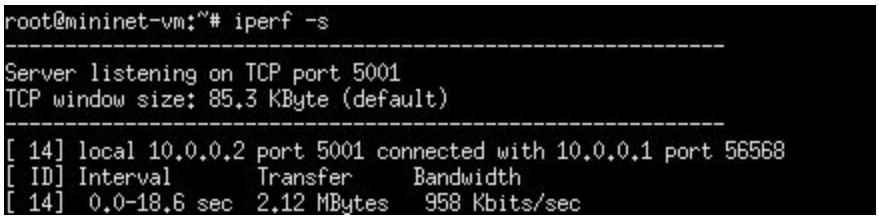
>xterm h1

And type the following for TCP throughput test

>iperf -c 10.0.0.2

```
root@mininet-vm:~# iperf -c 10.0.0.2
------------------------------------------------------------
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 13] local 10.0.0.1 port 56568 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-10.6 sec  2.12 MBytes  1.69 Mbits/sec
```

Now,we get the required TCP throughput reading from the bandwidth of the TCP server that is the xterm window of H2.



Alternatively,
For tcp iperf server in h2 with h1 as clien

mininet> h2 iperf -s &
mininet> h1 iperf -c h2

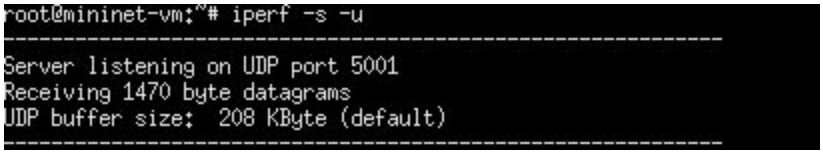
II. Follow the steps in I of part1 above with just the following modification :

a.For all iperf commands (in xterm of H1 and H2) add a '-u' flag and for iperf of client (H1's xterm window) add a '-b <specified bandwidth>' flag too. That is:
In xterm of H2 ,type:

>iperf -s -u

Instead of

>iperf -s



Now, H2 becomes our UDP server.

And in xterm of H1, type (for part 1.2.1):

>iperf -c 10.0.0.2 -u -b 64k

Instead of

>iperf -c 10.0.0.2

```
root@mininet-vm:~# iperf -c 10.0.0.2 -u -b 64k
------------------------------------------------------------
Client connecting to 10.0.0.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 13] local 10.0.0.1 port 40019 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-10.3 sec  80.4 KBytes  64.0 Kbits/sec
[ 13] Sent 56 datagrams
[ 13] Server Report:
[ 13]  0.0-10.3 sec  80.4 KBytes  64.0 Kbits/sec   0.140 ms    0/   56 (0%)
```

Now just like for TCP ,we get the required UDP throughput reading from the bandwidth of the UDP server that is the xterm window of H2.

```
root@mininet-vm:~# iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 13] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 40019
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[ 13]  0.0-10.3 sec  80.4 KBytes  64.0 Kbits/sec   0.140 ms    0/   56 (0%)
```

Alternatively,

For udp server in h2 with h1 as client

mininet> h2 iperf -s -u &

mininet> h1 iperf -c h2 -u -b <bandwidth>

Observation :

The TCP throughput measured using the iperf command is : 958 Kbps

Now, The UDP throughputs based on their various set bandwidth are :

VIII. 64 Kbps - 64 Kbps

IX. 128 Kbps - 128 Kbps

X. 256 Kbps - 256 Kbps

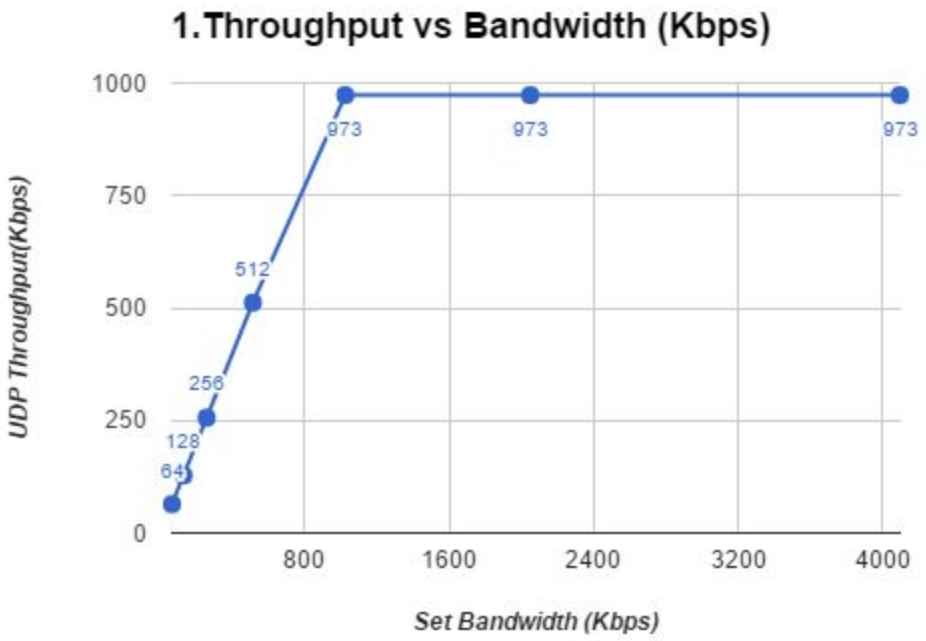
XI. 512 Kbps - 512 Kbps

XII. 1024 Kbps - 973 Kbps

XIII. 2048 Kbps - 973 Kbps

XIV. 4096 Kbps - 973 Kbps

The UDP bandwidth vs UDP throughput plot is as follows :



Conclusion

The TCP's throughput is measured to be 958 Kbps which is close to the network's minimum bandwidth.

From the plot, we conclude that throughput increases with client's bandwidth till the client's bandwidth reaches 1024 Kbps after which it becomes constant at about 973 Kbps which is trivial since the network's links have minimum bandwidth of 1Mbps and hence after 1 Mbps the client's bandwidth doesn't affects the measured throughput since maximum allowable bandwidth is 1Mbps.

Part 2

Objective

We need to form a custom topology with two hosts and two switches, linked as follows:
1.H1 connected to S1
2.S1 connected to S2
3.S2 connected to H2
For links 1 and 3, we set the parameters as 3 Mbps, 1ms and no loss.
The middle link that is link 2's parameter is variable and is changed after every part to note the effect of link's delay and bandwidth on the UDP and TCP throughputs
In this experiment,we need to check the effect of bandwidth and delay of the link between the switches S1 and S2 (written as S1-S2) over TCP and UDP throughput.

Procedure

We proceed as per part 1 modifying the first step that is forming the network topology. We need to create custom topology for which we can use a python script.
In the python script, we create our desired topology using the mininet API and setting the bandwidth, loss and delay for the mentioned links. For the middle link that is the one between the switches, the configuration of bandwidth and delay is variable and hence will be changed to take throughput for other parts.

Now, copy the script in the home folder of mininet after ssh login and use the following to create the desired topology using the python script

```
>sudo mn --custom ~/mininet/topo-2sw-2host.py --topo mytopo --link tc
```

The following output shows up in the terminal for part a:

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(3.00Mbit 1ms delay 0% loss) (3.00Mbit 1ms delay 0% loss) (h1, s3) (0.50Mbit 1ms delay) (0.50Mbit 1ms delay) (s3, s4) (3.00Mbit 1ms delay 0% loss) (3.00Mbit 1ms delay 0% loss) (s4, h2)
```

***** Configuring hosts**
h1 h2
***** Starting controller**
c0
***** Starting 2 switches**
s3 s4 ...(3.00Mbit 1ms delay 0% loss) (0.50Mbit 1ms delay) (0.50Mbit 1ms delay) (3.00Mbit 1ms delay 0% loss)
***** Starting CLI:**

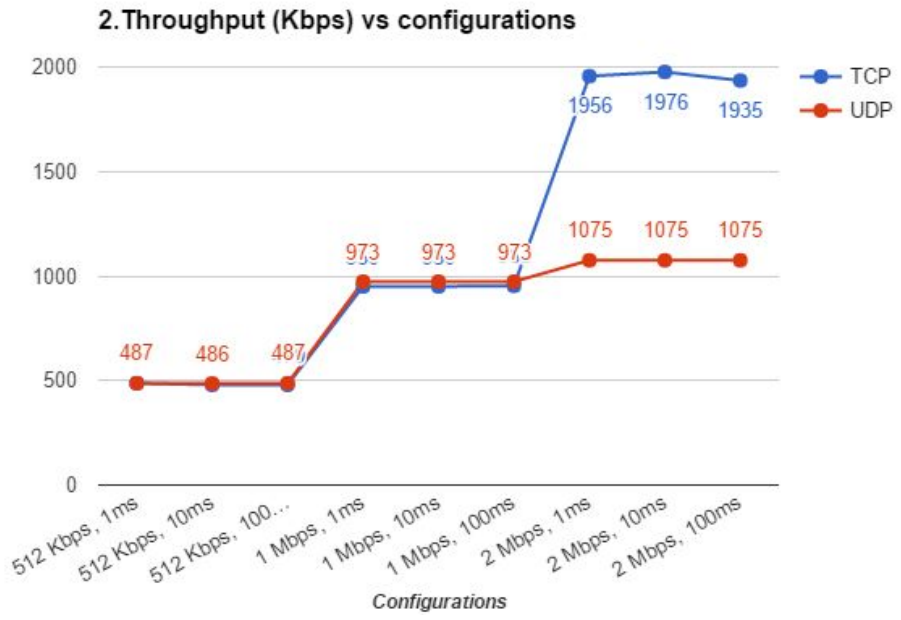
Now, we repeat part 1 for every subpart of part 2 that is for part a of part2, we first check the TCP throughput and then repeat what we did in part 1 for various bandwidths of UDP and record the throughputs.

Now, after every part out of nine, we will update our middle link parameters in the python script kept in home directory of mininet, and form the topology again using the following two commands in succession :

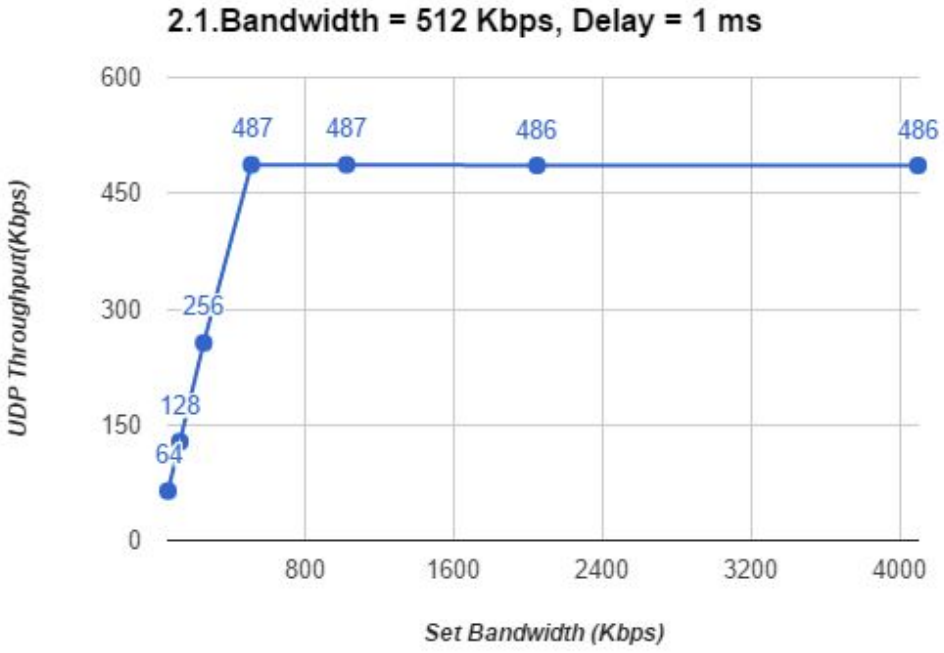
>sudo mn -c
>sudo mn --custom ~/mininet/topo-2sw-2host.py --topo mytopo --link tc

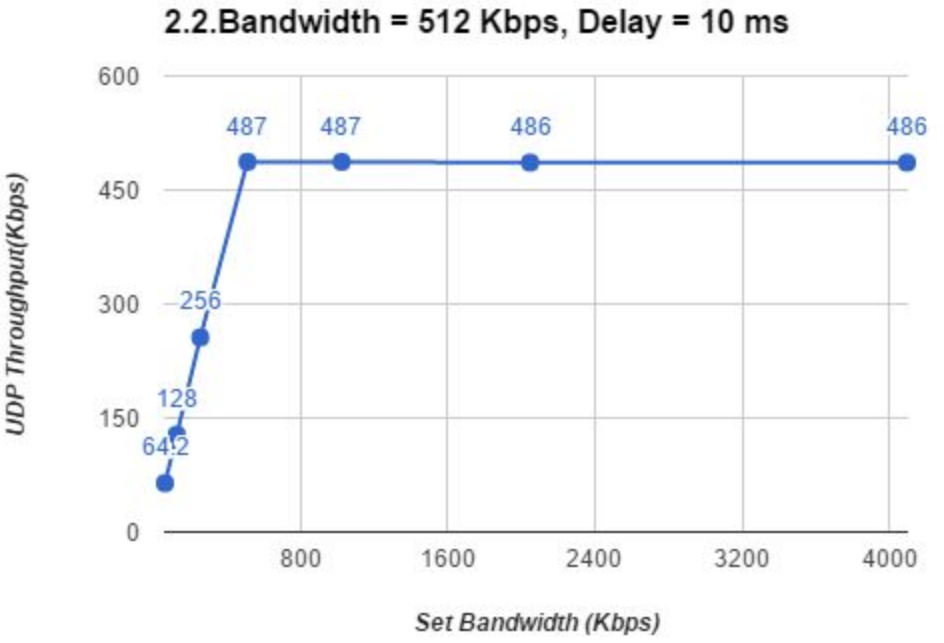
Observation

With default bandwidths for UDP and TCP, the throughput vs various configurations vary as follows :

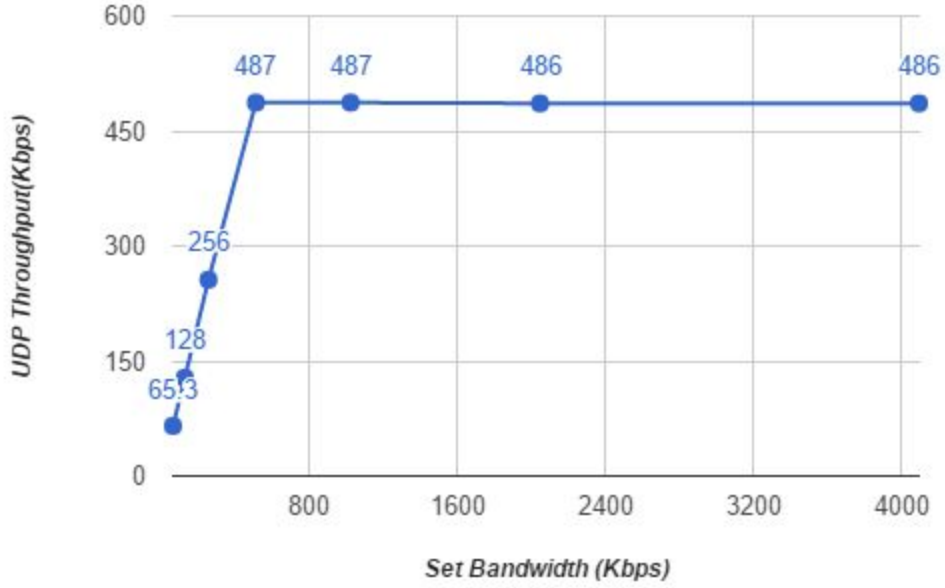


Now ,for different bandwidths of UDP and varying configurations of the middle link we have the following plots :

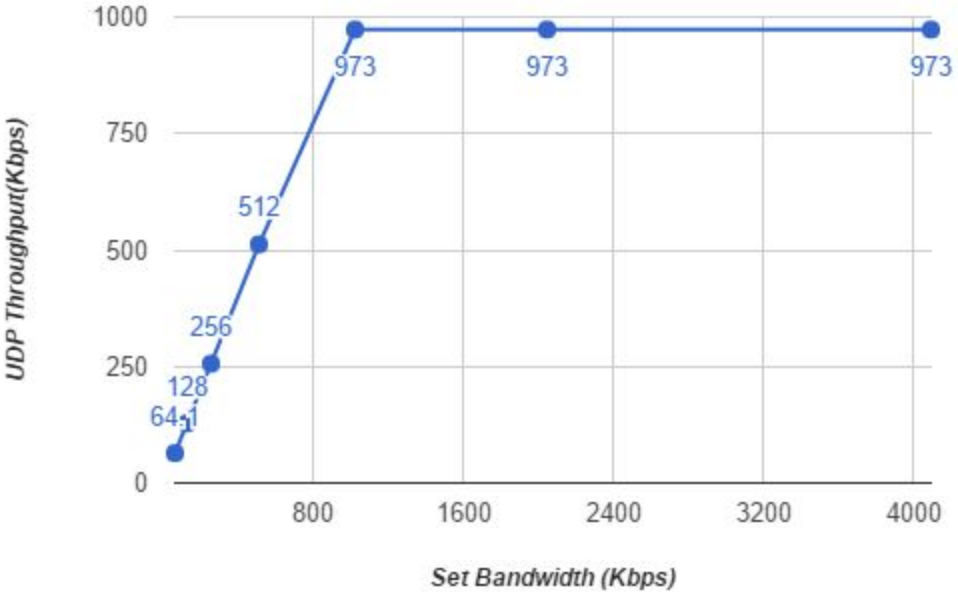




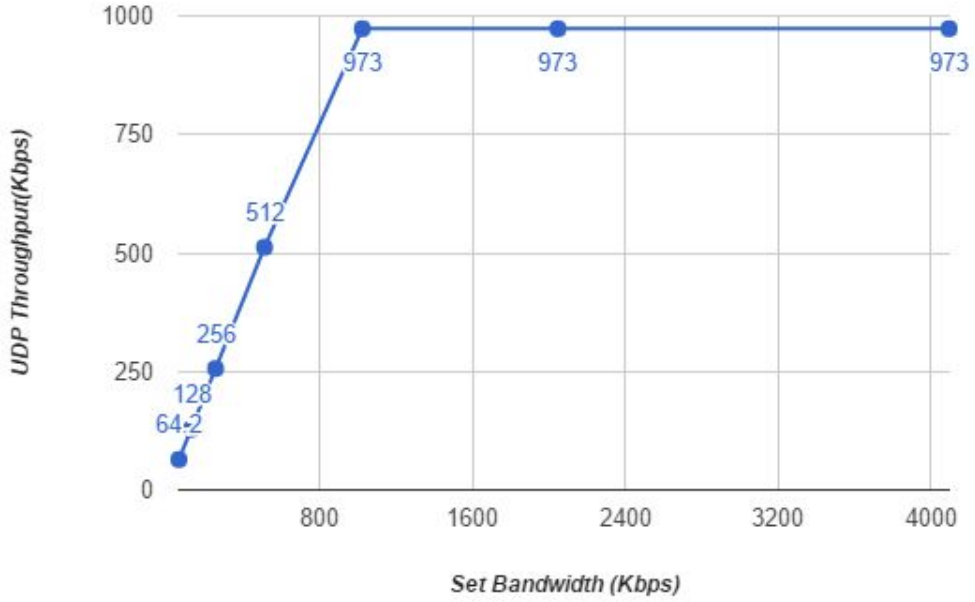
2.3.Bandwidth = 512 Kbps, Delay = 100 ms



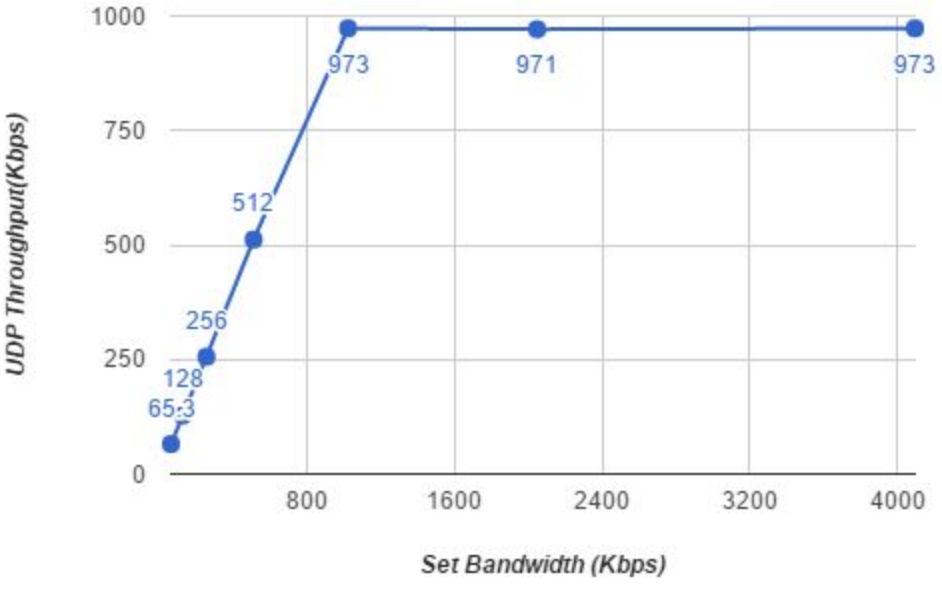
2.4.Bandwidth = 1 Mbps, Delay = 1 ms



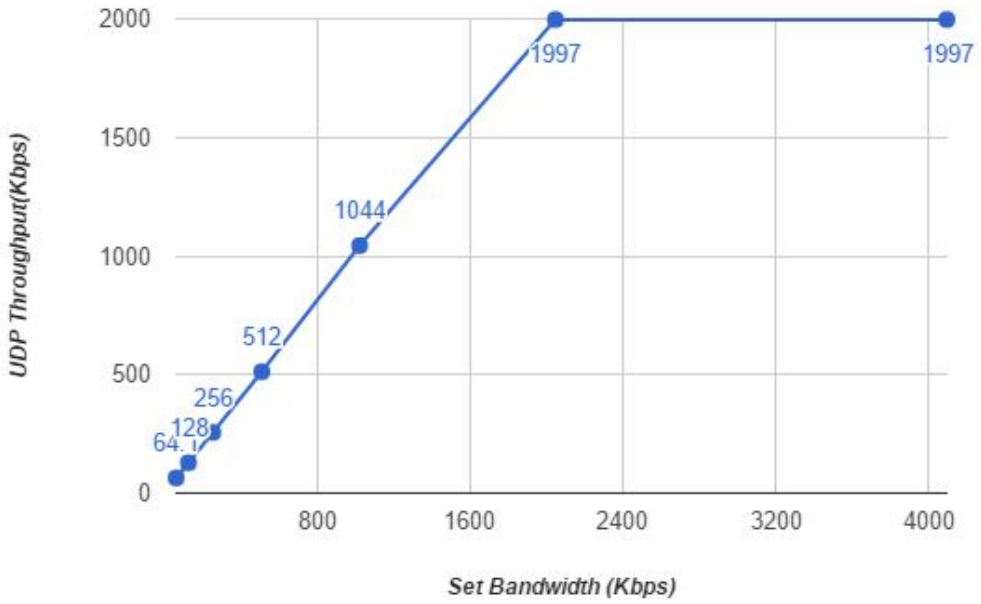
2.5.Bandwidth = 1 Mbps, Delay = 10 ms



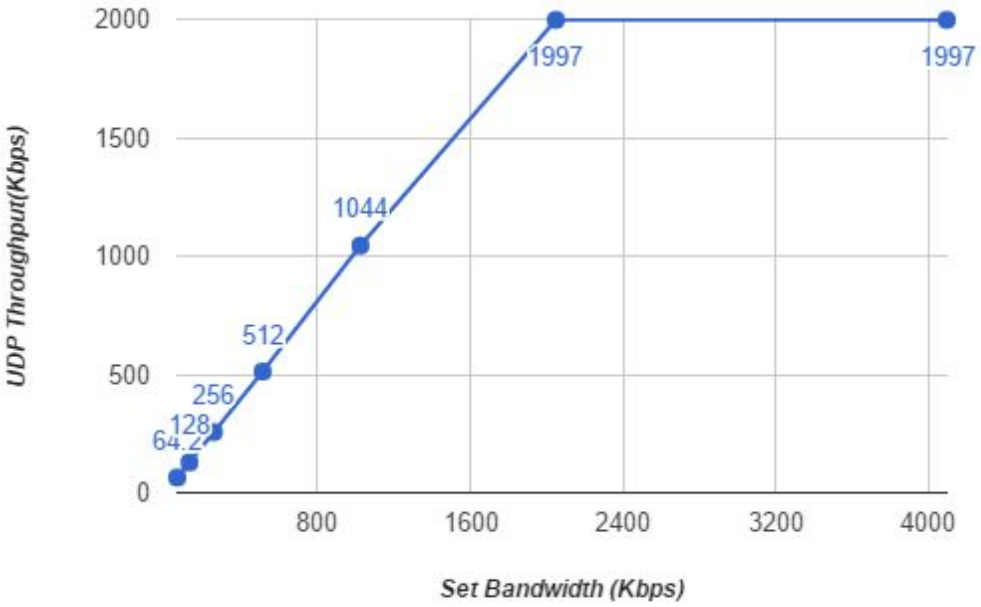
2.6.Bandwidth = 1 Mbps, Delay = 100 ms

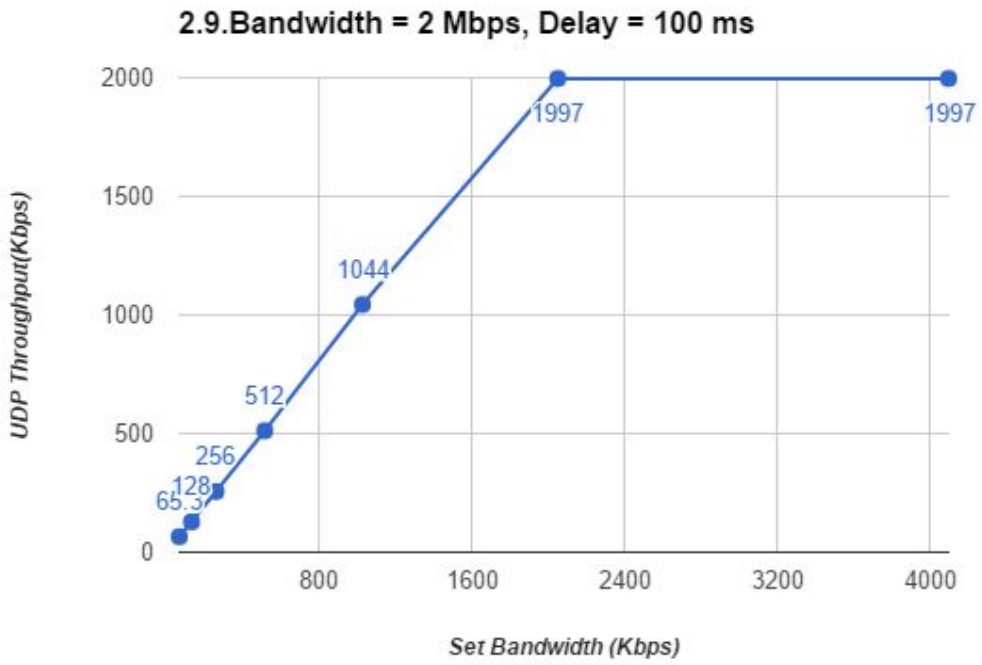


2.7.Bandwidth = 2 Mbps, Delay = 1 ms



2.8.Bandwidth = 2 Mbps, Delay = 10 ms





Conclusion

For plot 2, that is throughputs of UDP and TCP vs the nine configurations with default set bandwidths of the clients, we observe that TCP's throughput adapts according to the bandwidth of the middle link (since other two link's bandwidth is always greater than the middle link's bandwidth). However, as soon as the bandwidth of middle link is set to 2 Mbps, the TCP's throughput is seen to vary according to the delay too. As the delay increases, TCP's throughput first increases and then decreases for 2 Mbps case only (last three cases). This can be explained by the fact that TCP actually waits for the client's acknowledgment before sending further packets of data, this wait time is dependent on the delay, so as the delay increase, the sender has to wait for longer time and hence the amount of data sent in a particular time interval is decreased as shown by the throughputs readings.

The UDP's throughput doesn't depend on the delay as it uses different mechanism and doesn't wait for the client's acknowledgement before sending further data packets, so more or less UDP's throughput remains constant for different delays.

The UDP's throughput though depends on the middle link's bandwidth as it is the minimum bandwidth of the network which means it dictates the maximum data rate through the network irrespective of the client's bandwidth. So, till client's bandwidth is less than or equal to the middle link's bandwidth, the throughput is almost equal to

client's bandwidth. However, as soon as client's bandwidth exceeds middle link's bandwidth, the throughput is almost equal to the middle link's bandwidth which is trivial since it dictates maximum data rate which throughput measures. So, the throughput increases and then becomes more or less constant.

Part 3

Objective

In this part, we observed the effect of channel loss or link loss on the performance of TCP and UDP. We constructed a topology similar to part 2, the only difference being in this case we varied the loss rate associated with the link between switch S1 and S2. The bandwidth for all the three links is 1 Mbps along with 1ms of propagation delay for all the three links. Both the links between the hosts and the switches have a loss rate of 1%.

We will run iperf server at H2 and iperf client at H1 and will execute part1 for varied loss rate of the link between switches. The loss rate being :
(a) loss rate = 1%
(b) loss rate = 3%
(c) loss rate = 5%
(d) loss rate = 10%
(e) loss rate = 15%

Procedure

We proceed as per part 1 modifying the first step that is forming the network topology. We need to create custom topology for which we can use a python script.
In the python script, we create our desired topology using the mininet API and setting the bandwidth, loss and delay for the mentioned links. For the middle link that is the one between the switches, the configuration of loss is variable and hence will be changed to take throughput for other parts.

Now, copy the script in the home folder of mininet after ssh login and use the following to create the desired topology using the python script

```
>sudo mn --custom ~/mininet/topo-2sw-2host.py --topo mytopo --link tc
```

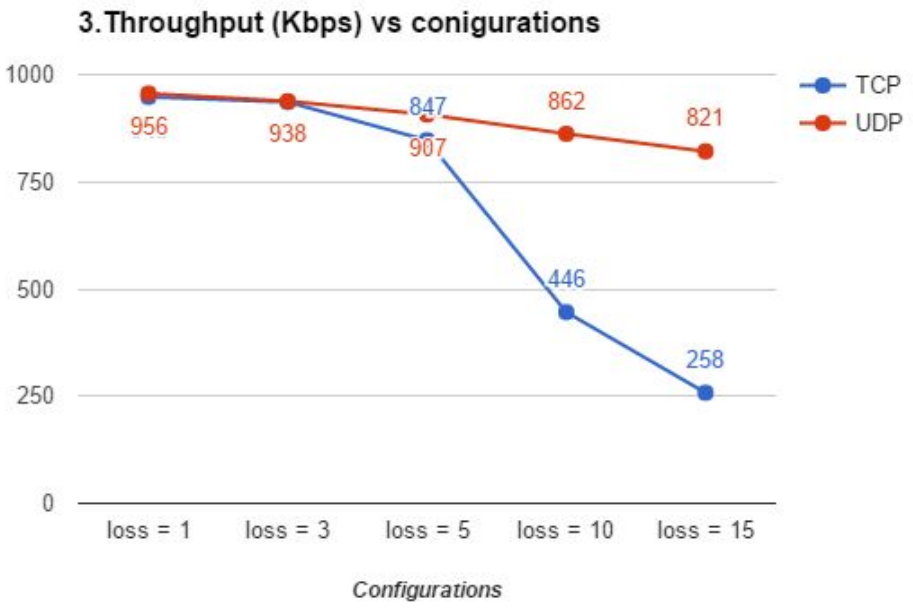
Now, we repeat part 1 for every subpart of part 3that is for part a of part 3, we first check the TCP throughput and then repeat what we did in part 1 for various bandwidths of UDP and record the throughputs.

Now, after every part out of nine, we will update our middle link parameters in the python script kept in home directory of mininet, and form the topology again using the following two commands in succession :

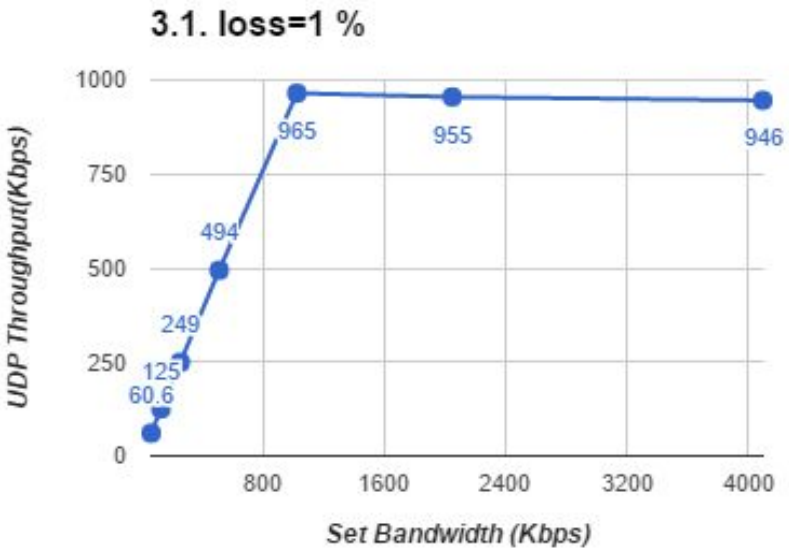
```
>sudo mn -c
>sudo mn --custom ~/mininet/topo-2sw-2host.py --topo mytopo --link tc
```

Observation

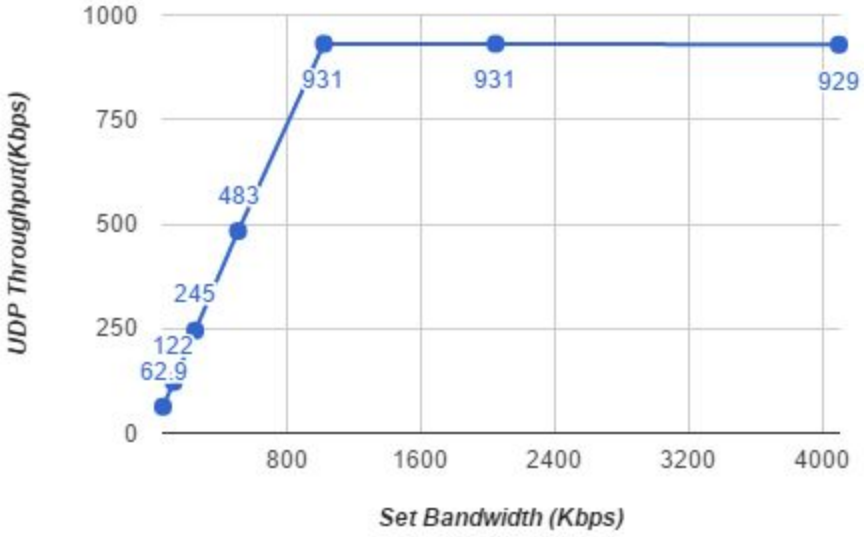
With default bandwidths for UDP and TCP, the throughput vs various configurations vary as follows :



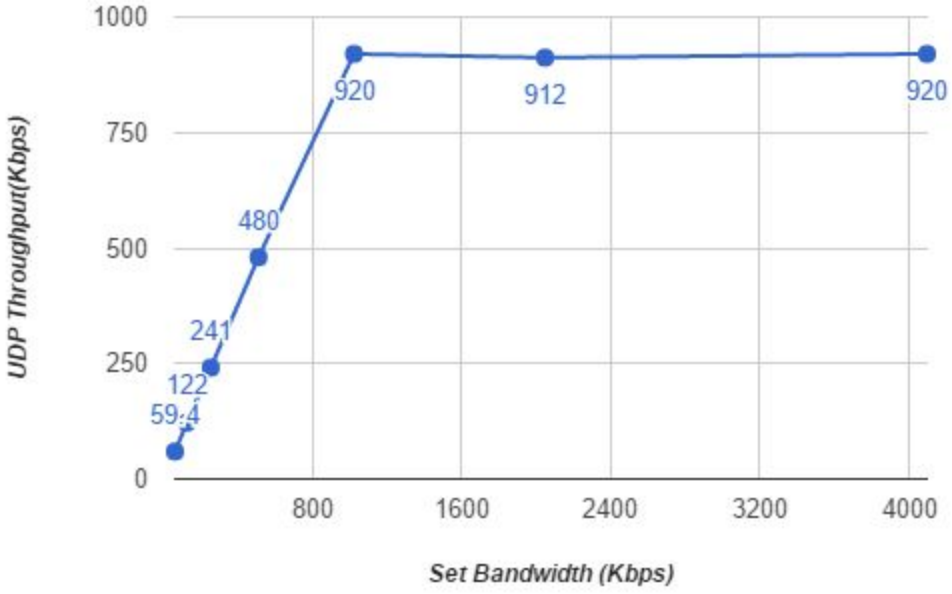
Now ,for different bandwidths of UDP and varying configurations of the middle link we have the following plots :



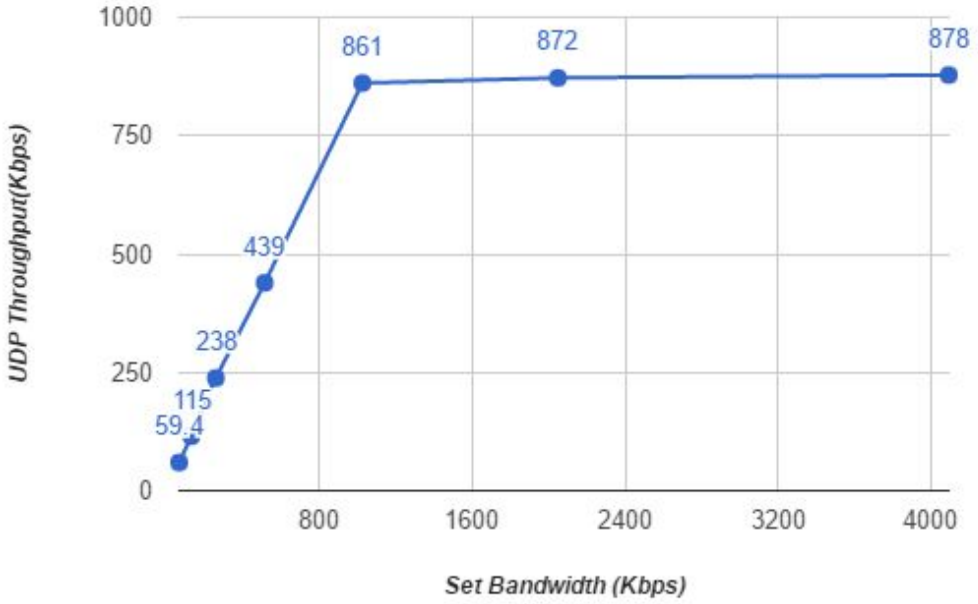
3.2. loss=3 %



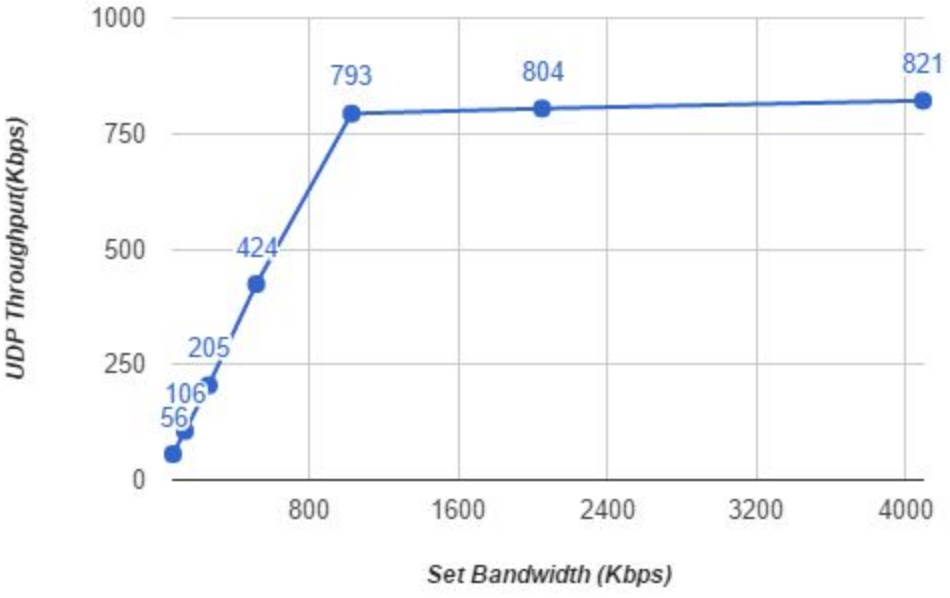
3.3. loss=5 %



3.4. loss=10 %



3.5. loss=15 %



Conclusion

Observing the UDP and TCP bandwidth at various losses we can conclude that the peak bandwidth will always be less than 1024 Kbps (trivial as mentioned above as the bandwidth of the link between the hosts and switches is 1024 Kbps) but on on increasing the link loss between two switches, the peak bandwidth decreases which should decrease because packet loss will increase latency due to additional time needed for retransmission.

Comparing TCP and UDP bandwidth , at higher loss rates TCP will just crawl in case of bandwidth because at higher loss rate TCP will run out of buffer space and will wait until retransmitted lost packet have been received. While UDP has different mechanism for handling packet loss. It just drops the lost packets and provides no recovery for them.

Authored by **:** **14CS10006 (Apurv Kumar)**
14CS30034 (Shubham Sharma)