FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT

DEPARTMENT OF ELECTRICAL ENGINEERING

Mobile and Wireless Networks

EEE4121F

MODULE B: PROJECT

By

EVANS TJABADI

May 2019

PLAGIARISM DECLARATION

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.

3. This report is my own work.

4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

EVANS TJABADI

Tuesday 7th May, 2019

1 Introduction

The next generation network, 5G is an advanced wireless networking system promising to provide 10 to 1000 times better performance compared to the current network [1]. The 5G uses advanced technologies and software techniques for more efficient utilisation of network and radio (spectrum) resources. Some of these include, software defined network (separation of control and data plane through software), Massive MIMOs to connect more and dense group of devices, and Vitalisation of resources. As much as the bandwidth of the network affects performance, latency (Round-Trip Time) affects equally and this project explores the latency effect on the network.

2 Purpose of the Project

The RTT between any two communication entities is theoretically bound by the speed of light, or how far the two entities can be apart is limited by the speed of light. But, in addition to this propagation delay, the nodal processing delays at each hop in between these entities will become more significant.

The aim of this project is to minimise the nodal processing delays at each hop by varying the network switches and processing techniques and then to model a linear network with 30 to 50 hops and investigate how far apart the two communicating entities can be from each other to achieve a RTT of 0.5 ms.

3 Experiments

3.1 Determining the best Switch

To minimise the nodal processing delay, the most best performing switch will be chosen based to less processing time. The experiment investigates the performance of the User, OVSK and Linuxbridge switches. The two communicating hosts are connected to one switch (which is connected to a default controller) and the bandwidth of the links is set to be 10 Mbps. The figures below below show creation of the network with each switch and the RTT ping results for the different switches.
Fig.1 and Fig.2 show the results of the 'user' switch.

```
eee4121f@eee4121f-VirtualBox:~/Desktop/ProjectB/Code$ sudo mn --link tc,bw=10,delay=0ms --switch user
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 0ms delay) (10.00Mbit 0ms delay) (h1, s1) (10.00Mbit 0ms delay) (10.00Mbit 0ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```

Figure 1: The network with two hosts and one 'user' switch.

```
mininet> h1 ping -c10 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.141 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.490 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.274 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.163 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.165 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.303 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.108 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.270 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.270 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=1.92 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9196ms
rtt min/avg/max/mdev = 0.108/0.410/1.921/0.514 ms
```

Figure 2: The results network with two hosts and one 'user' switch.

Fig.3 and Fig.4 show the results of the 'ovsk' switch.

```
eee4121f@eee4121f-VirtualBox:~/Desktop/ProjectB/Code$ sudo mn --link tc,bw=10 --switch ovsk
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit) (10.00Mbit) (h1, s1) (10.00Mbit) (10.00Mbit) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit) (10.00Mbit)
*** Starting CLI:
```

Figure 3: The network with two hosts and one 'ovsk' switch.

```
mininet> h1 ping -c10 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.136 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.138 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.139 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.181 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.136 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.130 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.078 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9202ms
rtt min/avg/max/mdev = 0.058/0.118/0.181/0.035 ms
```

Figure 4: The results network with two hosts and one 'ovsk' switch.

Fig.5 and Fig.6 show the results of the Linuxbridge switch.

```
eee4121f@eee4121f-VirtualBox:~$ sudo mn --link tc,bw=10 --switch lxbr
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit) (10.00Mbit) (h1, s1) (10.00Mbit) (10.00Mbit) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```

Figure 5: The network with two hosts and one Linuxbridge switch.

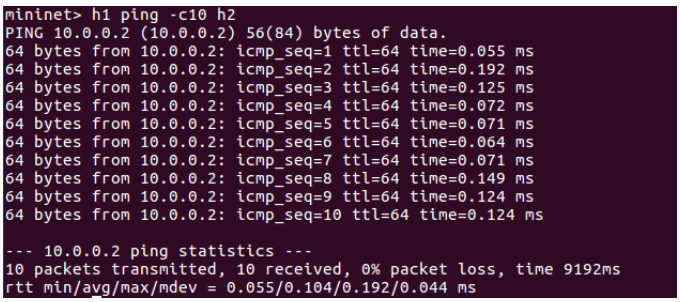

Figure 6: The results network with two hosts and one Linuxbridge switch.

From the results of the ping RTTs, it is observed that the Linuxbridge is the best performing switch in terms of the least nodal processing delay when all the other parameters are set to default.

3.2 Determining the best protocol TCP or UDP

The user datagram protocol (UDP) is a fast and unreliable methods of datagram transmission. It is not connection orientation and hence, requires no handshakes and connection setup in the beginning [2]. UDP doesn't not guarantee in order packet arrival at the receiver. Even though UDP performs error check, it doesn't do error corrections.

On the other hand, TCP (Transmission Control Protocol is a more reliable way of data transmission which offers in-order packets transmission and error correction for bit errors that happen during transmission [2]. The TCP is connected orientated and requires handshakes in the beginning. Because it is reliable, TCP is more used in the network or communications than UDP. Some firewalls block UDP traffic because there is no reasonably defined firewall policy of dealing with the UDP traffic.

3.3 Determining the best TCP congestion Control method

The best switch has been selected. The experiment now uses TCP transmission protocol to select the best network congestion control algorithm. The TCP BBR (Bottleneck Bandwidth and Round-Trip propagation time) and TCP RENO congestion control algorithms have been investigated. The Fig.7 below shows the results of the ping tests on the network with two hosts and one switch when deploying the RENO and BBR algorithms. The ping tests were run 100 times for each algorithm and the averages are presented in the Fig.7.

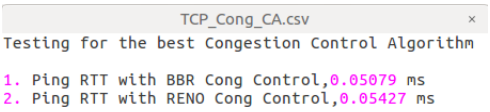

Figure 7: The network results with two hosts and one Linuxbridge switch and BBR and RENO congestion control algorithms.

From the Fig.7 above, it is observed that the BBR congestion control algorithm outperforms the RENO algorithm in terms of the nodal processing delay when implemented with the Linuxbridge switch.

3.4 Determining the best queuing algorithm

The best TCP congestion control algorithm has been selected above. The experiment now investigates different queue management algorithms with the Linuxbridge switch and all the other parameters set to default. The fair queuing (FQ), first-in-first-out (FIFO), and stochastic fair queuing (SFQ) algorithms for queue management in the network buffers have been investigated. The Fig.8 below shows results of the experiment. The ping test was run 100 times on each algorithm and the average RTT was taken.

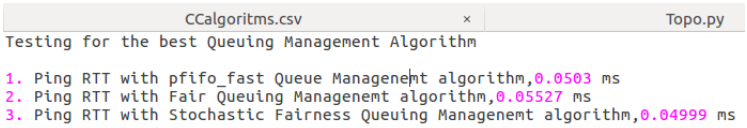

Figure 8: The network results with two hosts and one Linuxbridge switch, BBR congestion control algorithms and different queuing management algorithms.

From the Fig.8, it is observed that the SFQ is the best queue management algorithm in terms of the packet processing delays.

3.5 Total nodal processing delay

The experiment has identified the best switch, congestion control and the best queue management algorithm in terms of nodal processing delay. The Linuxbridge switch, TCP BBR congestion control algorithm and the stochastic fair queuing algorithm have been deployed in the network with host1 connected to host2 over the switches or hops, all connected in a linear topology. The Fig.9 below shows the results of creating the network.

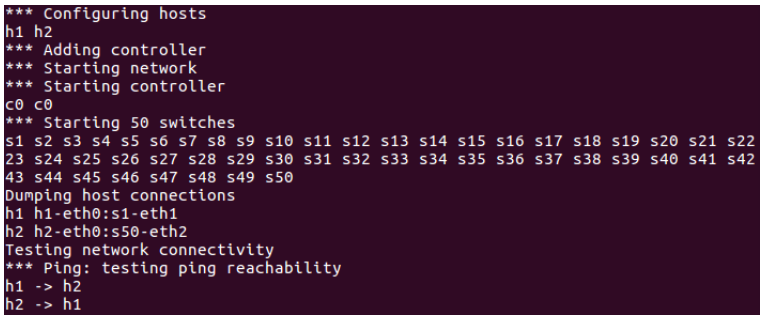

Figure 9: The result of creating the network.

The Fig.10 below shows the end to end round trip time of a ping test from host 1 to host2. The test was run 100 times and the average was taken.

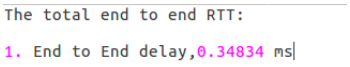

Figure 10: The end to end round trip time with 50 hops.

4 The theoretical maximum distance with nodal processing delays

From the Fig.10 above, $RTT = 0.34834\ ms$.
Therefore End to End delay, $d_{proc} = RTT/2 = 0.17417\ ms$
d_{proc} is the total nodal processing delay in the network since the links in the network have no link delays.

Therefore to achieve an end to end delay of 0.5 ms, the propagation delay must be:

$d_{prop} = 0.5\ ms - 0.17417\ ms = 0.32583\ ms$.
With speed of light as the limit in the link delays, $c = 3 \times 10^8\ m/s$
Therefore, $max\ distance = c \times d_{prop} = (3 \times 10^8\ m/s)(0.32583\ ms) = 98\ km$

From the above calculation, with the best optimised nodal processing outlined in the experiment above, the two communicating entities can be only 98 km apart to achieve the end to end delay of 0.5 ms.

4.1 The model of the network with the distant factor

To model the network with the distance taken into account, it is assumed that all the hops are equally distant apart in the network. The propagation delay is split into the number of links (51 in this case), and each little split propagation delay is added to each link when the network is created.

Each link delay is given by $d_{prop}/51 = 0.32583\ ms = 0.00639\ ms$.

The Fig.11 shows the creation of the network and the the delay can be observed in the links.

```
eee4121f@eee4121f-VirtualBox:~/Desktop/ProjectB/Code$ sudo python Topo.py
[sudo] password for eee4121f:
net.ipv4.tcp_congestion_control = bbr
net.core.default_qdisc = sfq
*** Adding hosts
*** Adding switch
*** Creating links
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 s22
23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36 s37 s38 s39 s40 s41 s42
43 s44 s45 s46 s47 s48 s49 s50
*** Adding links:
Bandwidth limit 5000 is outside supported range 0..1000 - ignoring
(5000.00Mbit 0.00639 delay) Bandwidth limit 5000 is outside supported range 0..
000 - ignoring
(5000.00Mbit 0.00639 delay) (h1, s1) Bandwidth limit 5000 is outside supported
```

Figure 11: The network with the delay.

4.2 The results of the network

Unfortunately, upon creation of the network with the delays, the Mininet software requires the links to the TClink which change the delay characteristics of the links. This factor was taken into account during the selection of the best switch and processing algorithms for the network. The 0.5 ms was not achieved when a ping test was run with this network and TClinks. The Fig.12 show the RTT of the ping test with the completed network.

```
The total end to end RTT:

1. End to End delay,26.63667 ms
2. End to End delay,30.97773 ms
3. End to End delay,24.51531 ms
```

Figure 12: The results of the network with the delay.

5 Conclusions

The experiments in this project have investigated the performance of the switches available in mininet emulation software. Of the three investigated, namely, user, ovsk and the linuxbridge, it was found that the linuxbridge is the best performing in terms of the least nodal processing delay.

The project then investigated the performance of the TCP congestion control algorithms. Only BBR and RENO algorithms were investigated with the default queuing management algorithm. It was found that the BBR was a better congestion control algorithm in terms of nodal processing delay.

The project then investigated the performance of the queuing management algorithms and with all the other parameters set to default, it was found that stochastic fair queuing outperformed fair queuing and first-in-first out queuing algorithms.

The aim of the project was to determined how far apart the two entities can be from each other, and with the determined nodal and propagation delays, it was mathematically determined that the two entities can be only 98 km away from each other. The distance is found to be reasonable given the performance of the hops and the goal of 0.5 ms. Unfortunately, the link delays cannot be implemented in the mininet software due different link classes.

6 Usefulness and Expansion of the Work

The work presented in this project experiments on optimisation of the nodal processing delays and the using the speed of light limitation to determine the distance the communicating entities can be apart. The work is useful in that, given the distance between any nodes and the optimised nodal processing network, the network designer can determine the end to end communication time.

The work can be expanded to experiment on more network components and different algorithms. More switches, like the ovs can be added, and more congestion algorithms, like TCP cubic can also be added. The performance of TCP can be compared with that of UDP. Some more queuing management algorithms can also be added.

From the added network parameters, each parameters can be tested against all the other parameters in the other fields (rather than setting them to defaults), for example, test the congestion control algorithms with all the switches and all the queue management algorithms. This will create a huge performance matrix and the best combination can be deployed.

And lastly, investigate the link class than will allow the easy manipulation of the link delay to put the divided propagation delay and test the end to end of the distance factor modelled network.

References

[1] Huawei Technologies Co.Ltd. 5g: A technology vision. Vision report, January 2014. Available at https://huawei.eu/sites/default/files/huawei_5g_white_paper_en_20140129.pdf. Accessed: 05/05/2019.

[2] Saad Mneimneh. Computer Networks: UDP and TCP. Technical report, Hunter College of CUNY, New York.