NETWORKS LAB ASSIGNMENT - 5
Using NS3 - Network Simulator

Group Members

Aditya Kumar Jha (11010102)

Harshil Lodhi (11010121)

Shobhit Chaurasia (11010179)

PROBLEM - 1

Copy first.cc file in folder “4” of our solution folder into the scratch directory of your NS3 installation. Build the script using the following command
./waf

Execute the script using the following command
./waf --run scratch/first

To ensure that there is no conflict, we need to install servers on different ports.
The two server were installed on port no. 9000 and 9001
Flowmonitor throughput stats:

1. 2ms latency
   Flow 1 (10.1.1.1:49153 -> 10.1.1.2:9000)
   Throughput: 2.17723 Mbps
   Flow 2 (10.1.1.1:49154 -> 10.1.1.2:9001)
   Throughput: 1.49384 Mbps
2. 10ms latency
   Flow 1 (10.1.1.1:49153 -> 10.1.1.2:9000)
   Throughput: 0.686792 Mbps
   Flow 2 (10.1.1.1:49154 -> 10.1.1.2:9001)
   Throughput: 0.600183 Mbps
3. 20ms latency
   Flow 1 (10.1.1.1:49153 -> 10.1.1.2:9000)
   Throughput: 0.370099 Mbps
   Flow 2 (10.1.1.1:49154 -> 10.1.1.2:9001)
   Throughput: 0.343396 Mbps

Following are the pcap traces showing delivery of packets.

At node 0:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.1	10.1.1.2	IPv4	1054	Bogus IP header length (0, must be at least 20)
2	0.001686	10.1.1.1	10.1.1.2	UDP	1054	Source port: 49154 Destination port: etlservicemgr
3	0.203372	10.1.1.2	10.1.1.1	IPv4	1054	Bogus IP header length (0, must be at least 20)
4	0.205059	10.1.1.2	10.1.1.1	UDP	1054	Source port: etlservicemgr Destination port: 49154

At node 1:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.1	10.1.1.2	IPv4	1054	Bogus IP header length (0, must be at least 20)
2	0.000000	10.1.1.2	10.1.1.1	IPv4	1054	Bogus IP header length (0, must be at least 20)
3	0.001686	10.1.1.1	10.1.1.2	UDP	1054	Source port: 49154 Destination port: etlservicemgr
4	0.001686	10.1.1.2	10.1.1.1	UDP	1054	Source port: etlservicemgr Destination port: 49154

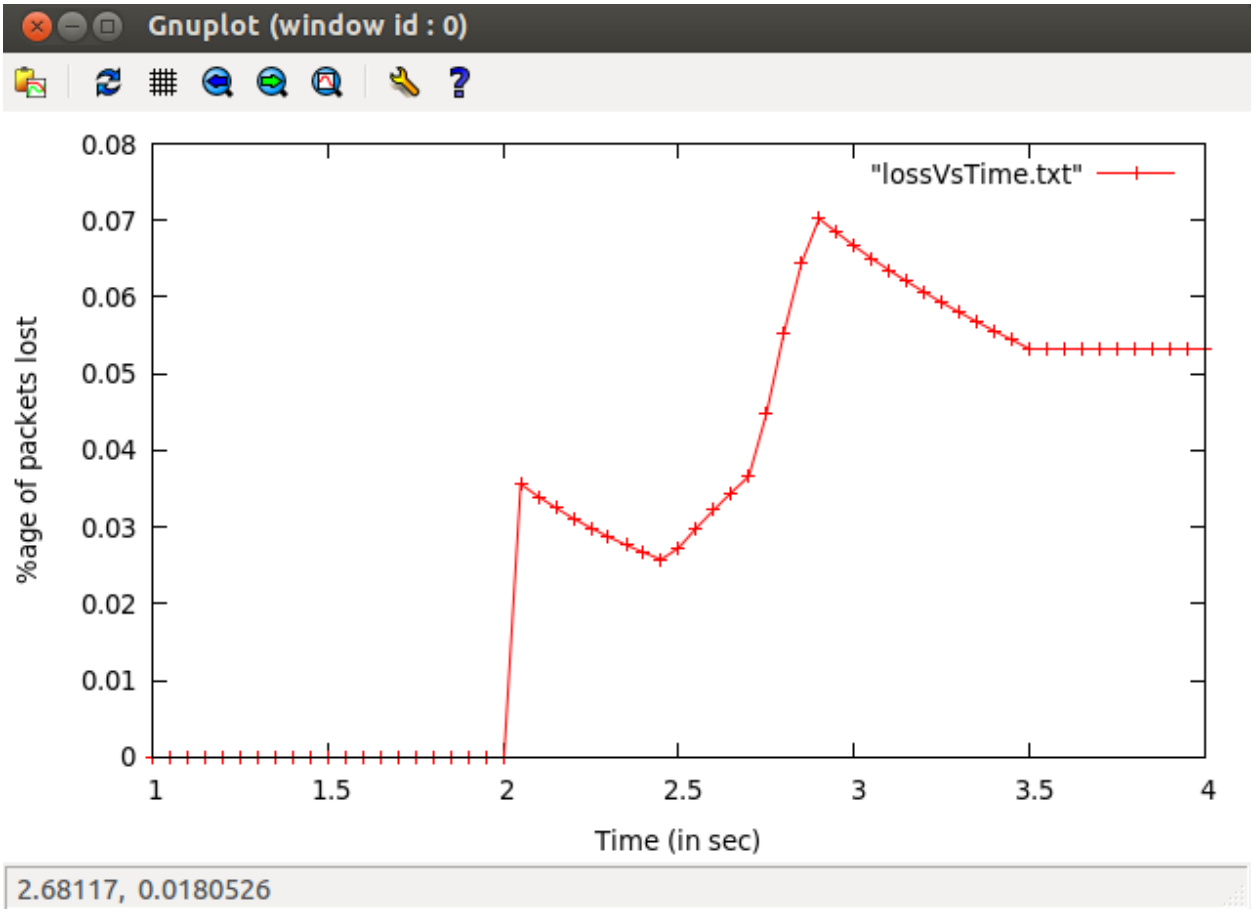
PROBLEM - 2

Copy second.cc file in folder “2” of our solution folder into the scratch directory of your NS3 installation. Build the script using the following command
./waf

Execute the script using the following command
./waf --run scratch/second

NOTE - We have calculated the no. of packets lost for the packets **that were destined to the node 2** from node 0 [refer problem statement figure].

This will generate lossVsTime.txt which contains percentage loss values as a function of time (in sec)
Following is the graph of No. of packets lost vs Time (in sec) :-



Total transmitted packets (destined to node2) = 1406
Total lost packets (destined to node2) = 75
Packets Lost Percentage (totalLost/totalTranmitted) [destined to node2]: 5%

Note that the plot is the %age of total packets lost right from starting as a function of time. So, the slope of graphs give us the rate increase/decrease in packet loss.
Initially, owing to the global routing protocol, all the routing tables are set up. Assuming hop count as a measure of link costs, the routing tables would have been such that the CBR traffic from node0 goes to node2 via node1. With routing tables already set up, the packet loss was zero. Now at T=2.0sec, the node0-node1 link went down. So, until the routing protocol modifies the table again, the packets will be lost. As a result the %age of packet loss spikes after T=2.0 sec, signifying a sudden increase in the no. of packets lost. With periodic and triggered updates, the routing protocol (like OSPF) sets up the table and figures out an alternative path (node0-node3-node4-node2) and the rate of increase of packet losses (slope of the curve) start dropping. However, since the traffic on link n0-n3 has increased a lot (due to the other already running n0-n3 flow), packet losses continue to occur. At 2.7 sec, the n0-n1 link becomes live again. The re-setting up of table takes time and there is a momentory spike in loss% curve near T=2.7. However, after sometime (once tables are re-setup), the n0-n2 flow starts using the n0-n1-n2 path. This frees the congestion on n0-n3 link and with routing tables set up, the packet losses start decreasing (negative slope means that lost/transmitted ratio is decreasing. But “transmitted” value increases. So, “lost” value must be decreasing at a faster rate). Eventually, at T=3.5 sec, with all the flows complete, the curve flattens out, signifying no more losses (in fact, no more transmissions as well).

PROBLEM - 3

Copy third.cc file in folder “3” of our solution folder into the scratch directory of your NS3 installation. Build the script using the following command
./waf

Execute the script using the following command
./waf --run scratch/third

Throughput Rates will be displayed on the terminal.
5 files “Recv0.dat” - “Recv4.dat” will be created for receiver rates.
5 files “Cwnd0.dat” - “Cwnd4.dat” will be created for congestion windows.\
A “queue.dat” file will be created for queue size. Format of this file is
<time><EQ/DQ/DR> <queue-size>
where EQ,DQ,DR stands for Enque, Deque and Drop respectively.

Paramaters of simulation:
Droptail Queue - 10
TCP source data generation rate - 1.5 Mbps
Link Rate - 10Mbps
Link delay - 10 ms

Following are the flowmonitor stats of the five flows:

Flow 1 (172.16.24.1:49153 -> 172.16.24.2:9000)
Tx Bytes: 10126000
Rx Bytes: 10123840
Throughput: 1.54491 Mbps
Flow 2 (172.16.24.1:49154 -> 172.16.24.2:9001)
Tx Bytes: 8102536
Rx Bytes: 8102104
Throughput: 1.54412 Mbps
Flow 3 (172.16.24.1:49155 -> 172.16.24.2:9002)
Tx Bytes: 6076952
Rx Bytes: 6076088
Throughput: 1.54353 Mbps
Flow 4 (172.16.24.1:49156 -> 172.16.24.2:9003)
Tx Bytes: 4053848
Rx Bytes: 4052176

Throughput: 1.54333 Mbps

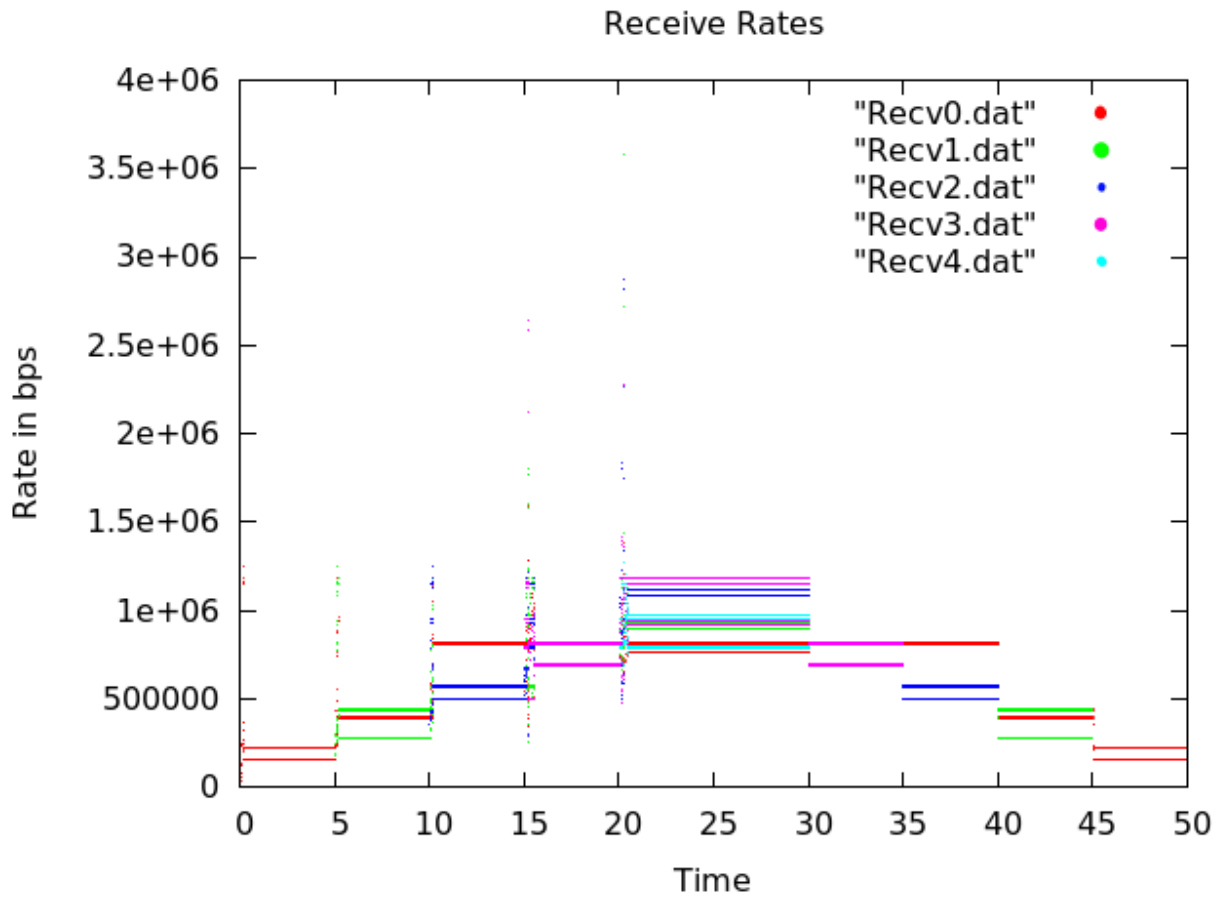
Flow 5 (172.16.24.1:49157 -> 172.16.24.2:9004)

Tx Bytes: 2028144

Rx Bytes: 2025840

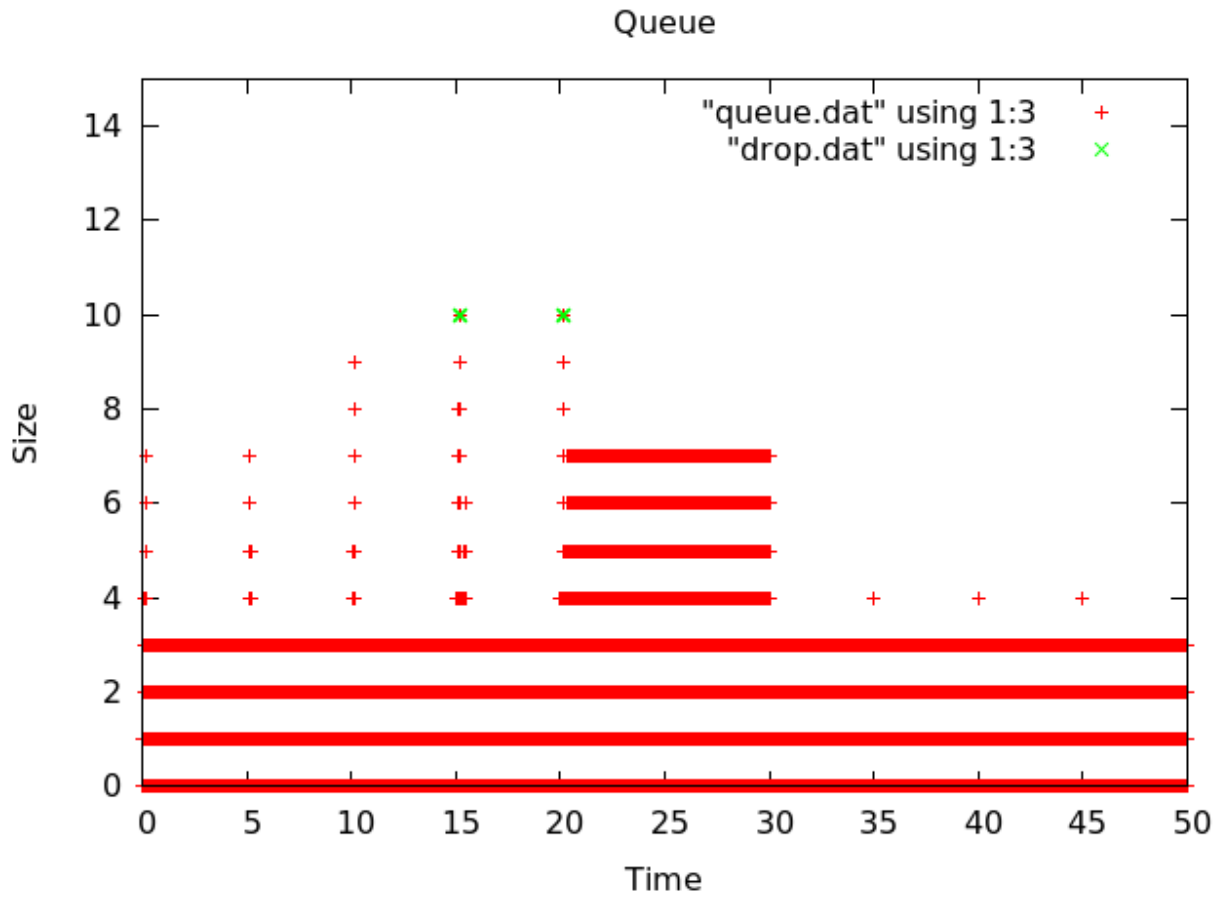
Throughput: 1.54049 Mbps

Receiver Rates Graph:



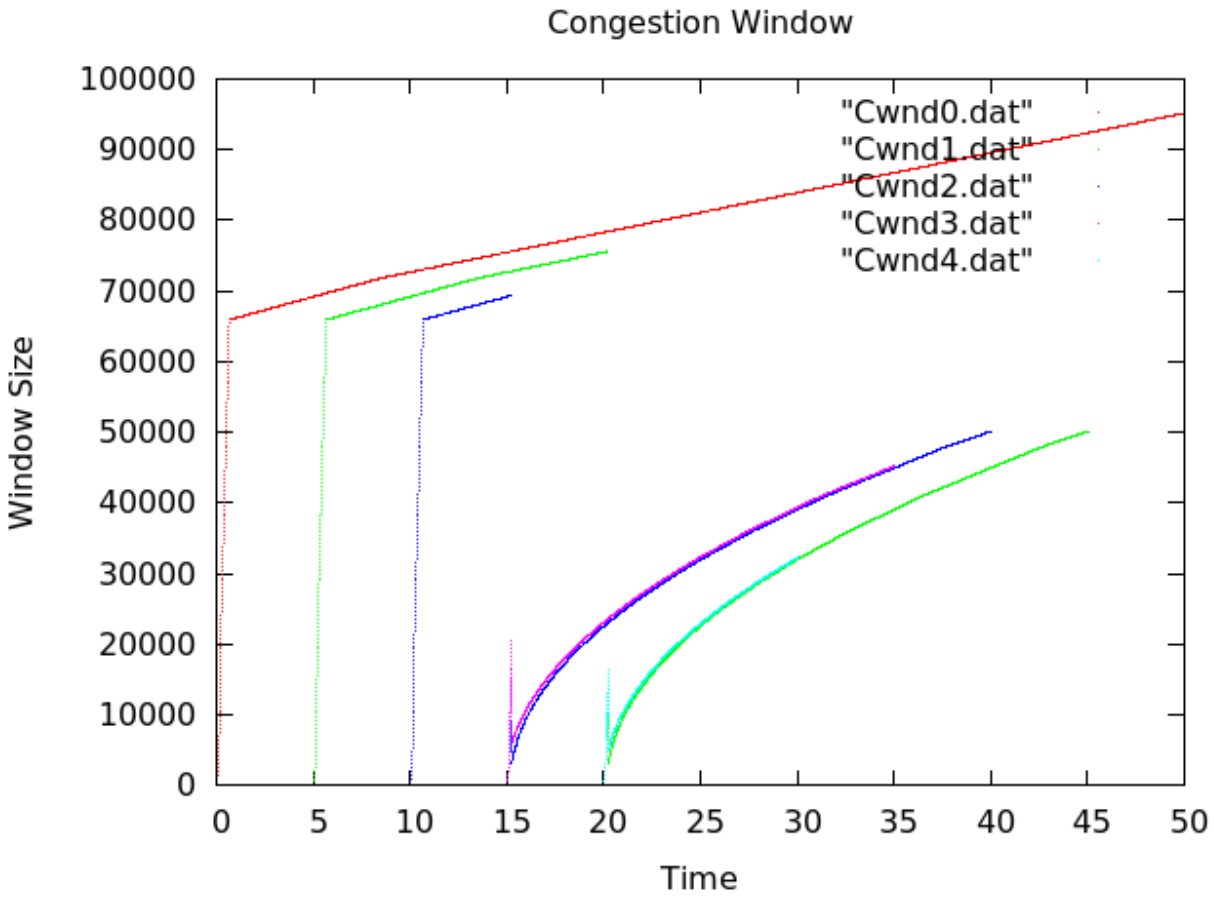
We can see how fairly tcp divides bandwidth between different connections through this graph. The points are plotted as dots in the graph. A line in graph is a collection of many points.

Queue Size Graph:



The green cross is the point where packet was dropped from the queue. The queue size is 10.

Congestion Window Graph:



One can clearly see the slow start, fast recovery and fast retrasmit phases of New Reno TCP protocol. It also displays the fairness how previous connections windows were reduced and new connections were given a chance.

What do you observe about fairness among the various TCP connections?

Ans - Yes Fairness was there among various TCP connections. It is clearly visible from the graph.

What do you observe about the queue size and packet drops?

Ans - Queue Size mostly remains constant at a particular level which can be seen by the horizontal lines. Due to congestions, sometime the queue size increase and when the queue becomes full, packet drop which is shown by green dots on the graph.

PS - gnuplot graph generation scripts are also attached (queue.gp && gnuplot congestion.gp && gnuplot recvrates.gp). Copy these to your ns3 folder and Use gnuplot <script-name>.gp to generate graph

PROBLEM - 4

First we'll compare the current throughput vs time elapsed for the two links for that follow the following steps ->

Copy fourth1.cc file in folder "4" of our solution folder into the scratch directory of your NS3 installation.

Execute the script using the following command
./waf --run scratch/fourth1

This will generate 2 plot files 'plot1.plt' and 'plot2.plt' in your current directory .

Now, to generate the corresponding PNG file you need to execute the following command ->
gnuplot plot1.plt

This will generate a file named 'plot1.png'.

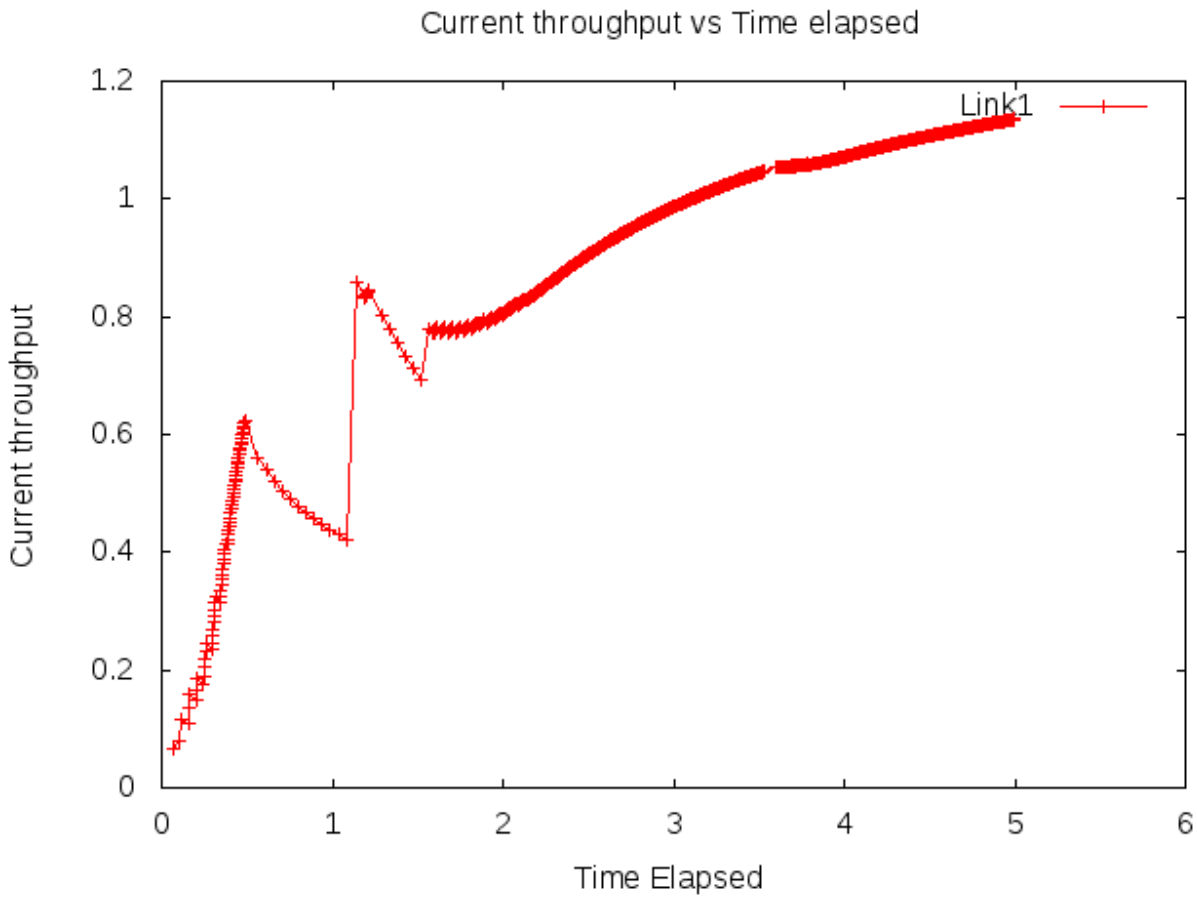


Image plot1.png

Similarly, ‘plot2.png’ can be generated.

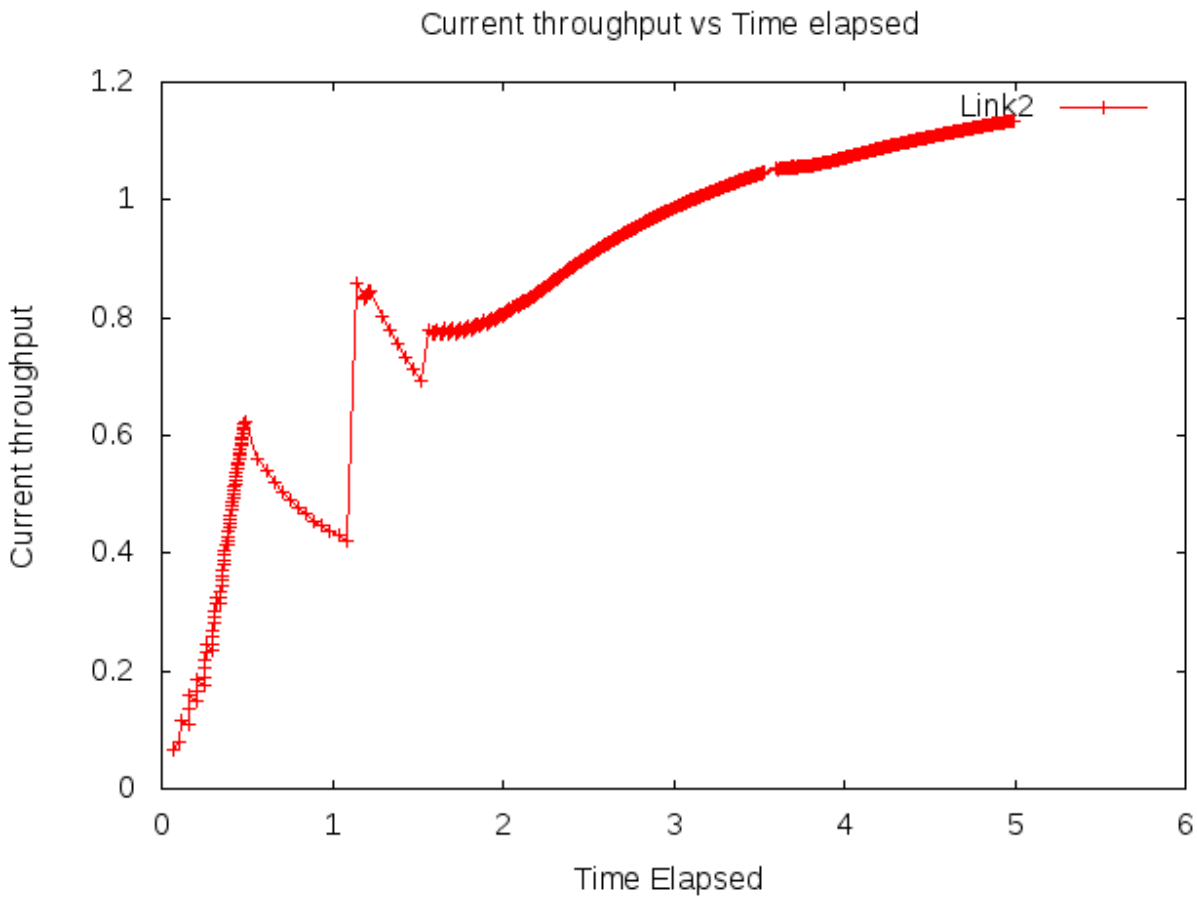


Image plot2.png

As you can see, the two curves are almost identical because the links(1 and 2) have the same rtt. So, they get an equal share of the bandwidth on the link from n0 to n1.

P.S. -> The graphs look identical but they are not exactly, because if you see the files plot1.plt and plot2.plt then you will recognise that the data points differ on the 3rd decimal place onwards. Thus, its hard to distinguish them on a graph.

Now, we are going to vary the rtt of the link 2 from 10ms to 100ms in units of 10ms and measure the final throughput achieved on both the links. The steps to generate the plots are as follows ->

Execute the script using the following command
./waf --run scratch/fourth2

This will generate 2 plot files ‘plot3.plt’ and ‘plot4.plt’ in your current directory .

Now, to generate the corresponding PNG file you need to execute the following command -> gnuplot plot3.plt

This will generate a file named ‘plot3.png’.

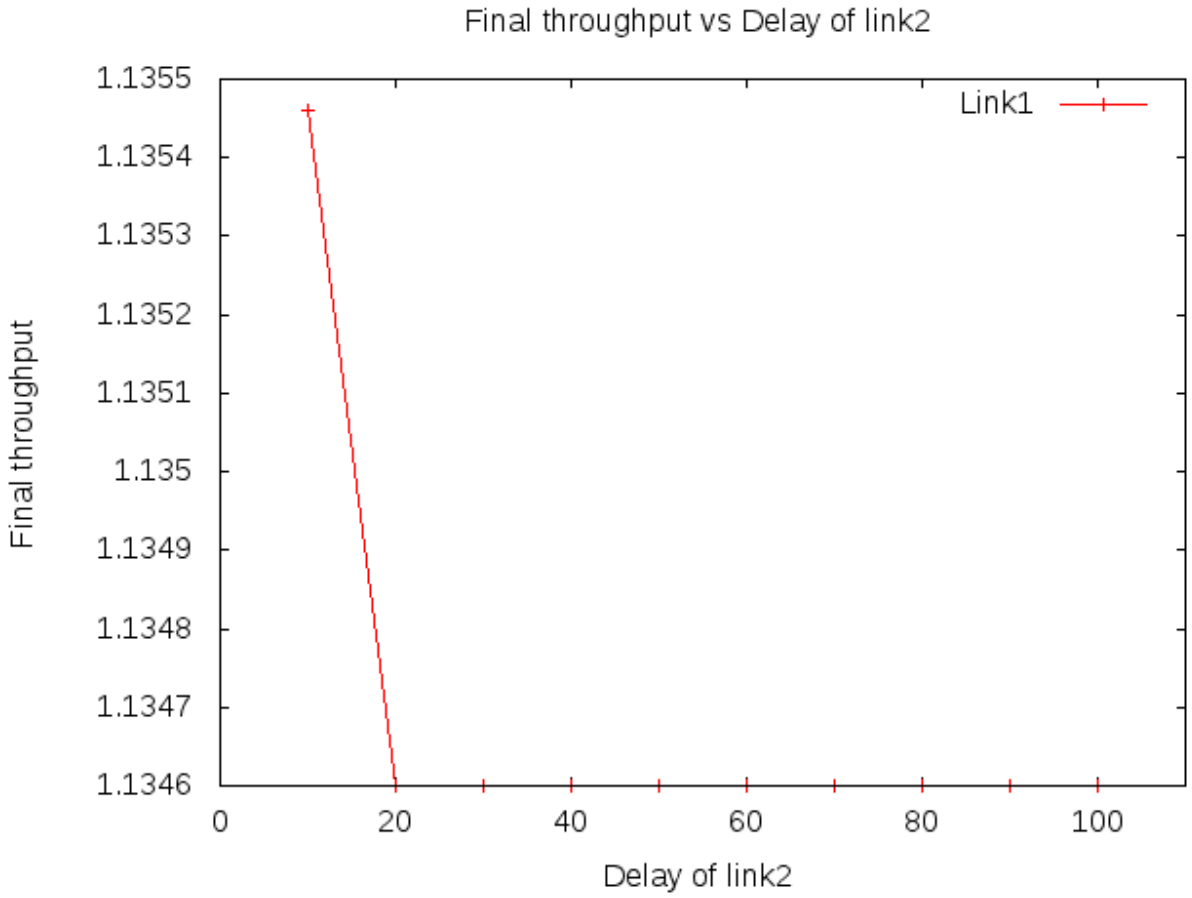


Image plot3.png

Similarly, ‘plot4.png’ can be generated.

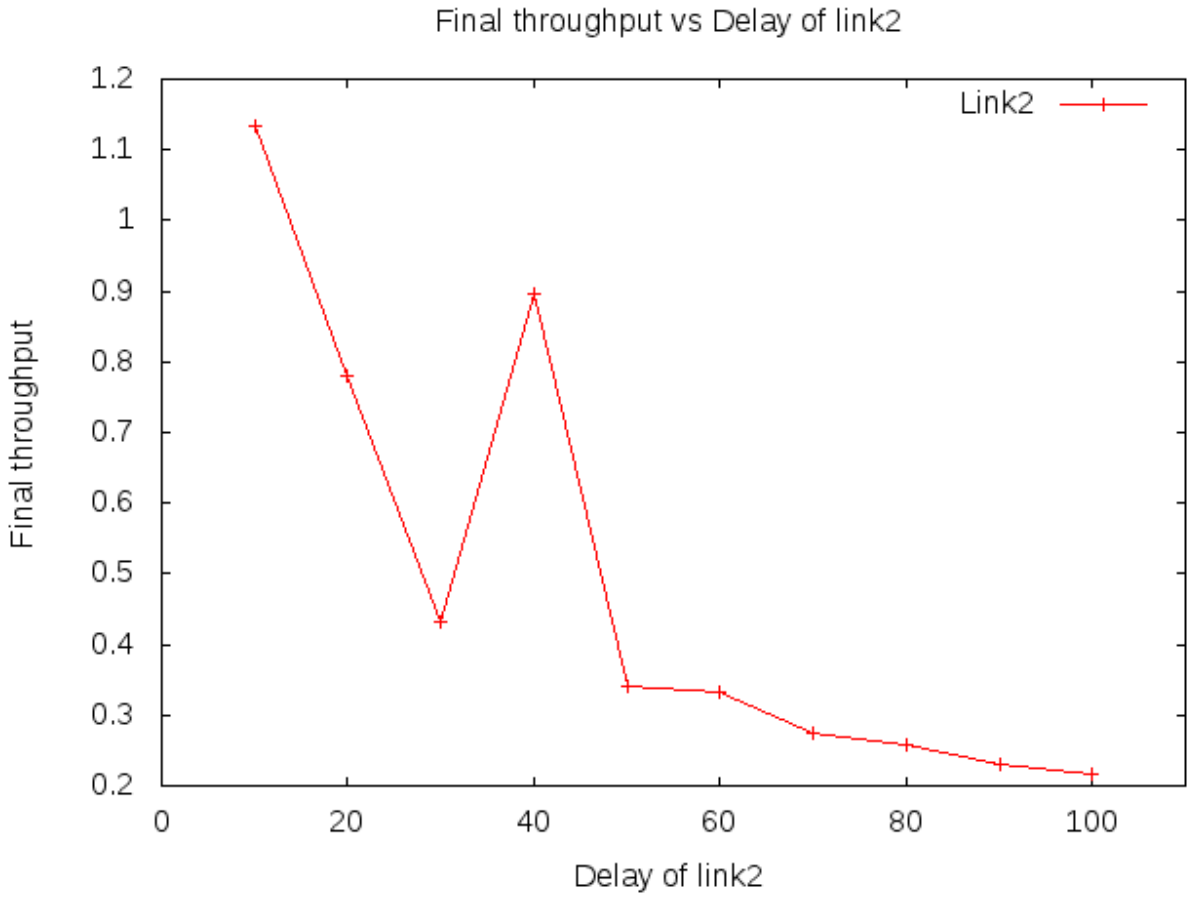


Image plot4.png

Assuming that FIFO is being used for queueing.

As we can see from the plots the final throughput of link1 is almost constant but the final throughput of link2 is getting reduced as the rtt is increasing. This has to be the case as the bandwidth of n0n1 link(10 Mbps) is larger than the sum of bandwidth on the n2n0 and n3n0(both being 1.5 Mbps thus the total being 10Mbps). So, no scheduling needs to be done on the n0 node. And, as it takes more time for the packets of node 3 to reach node 0 as compared to those of node 2(as link1 has lesser rtt). So, the throughput of n3 is getting reduced.