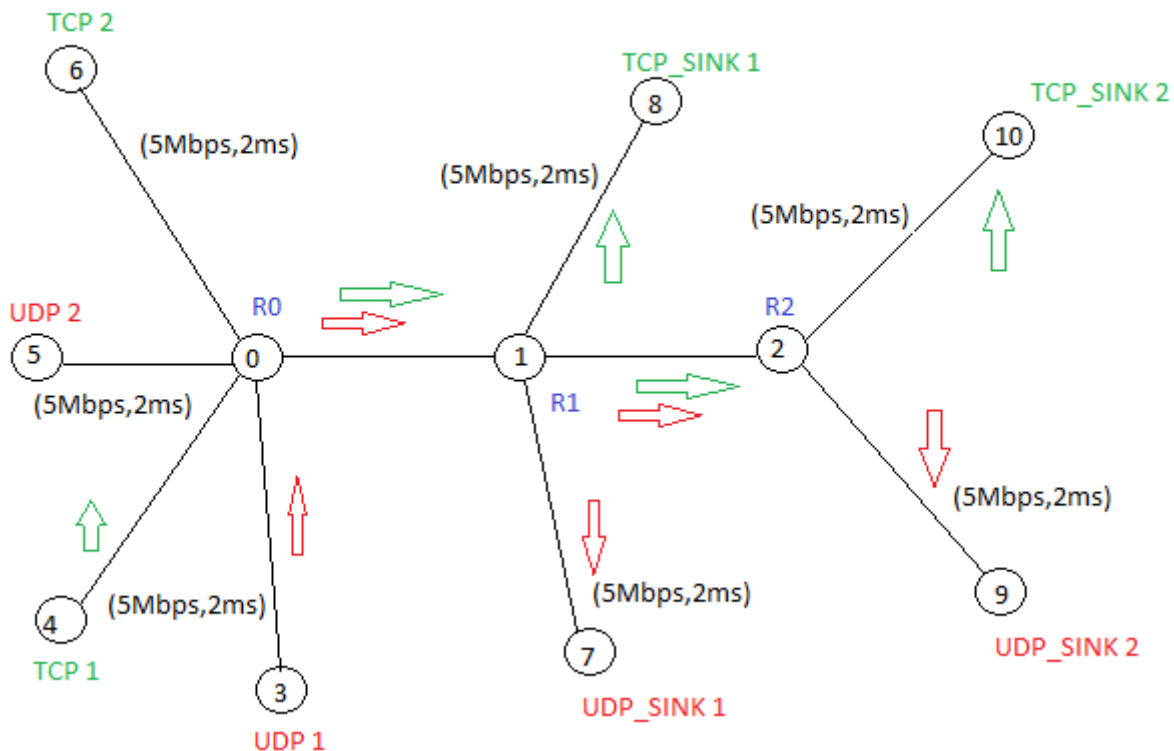


**Project 2: Comparison of RED vs. DropTail Queuing**

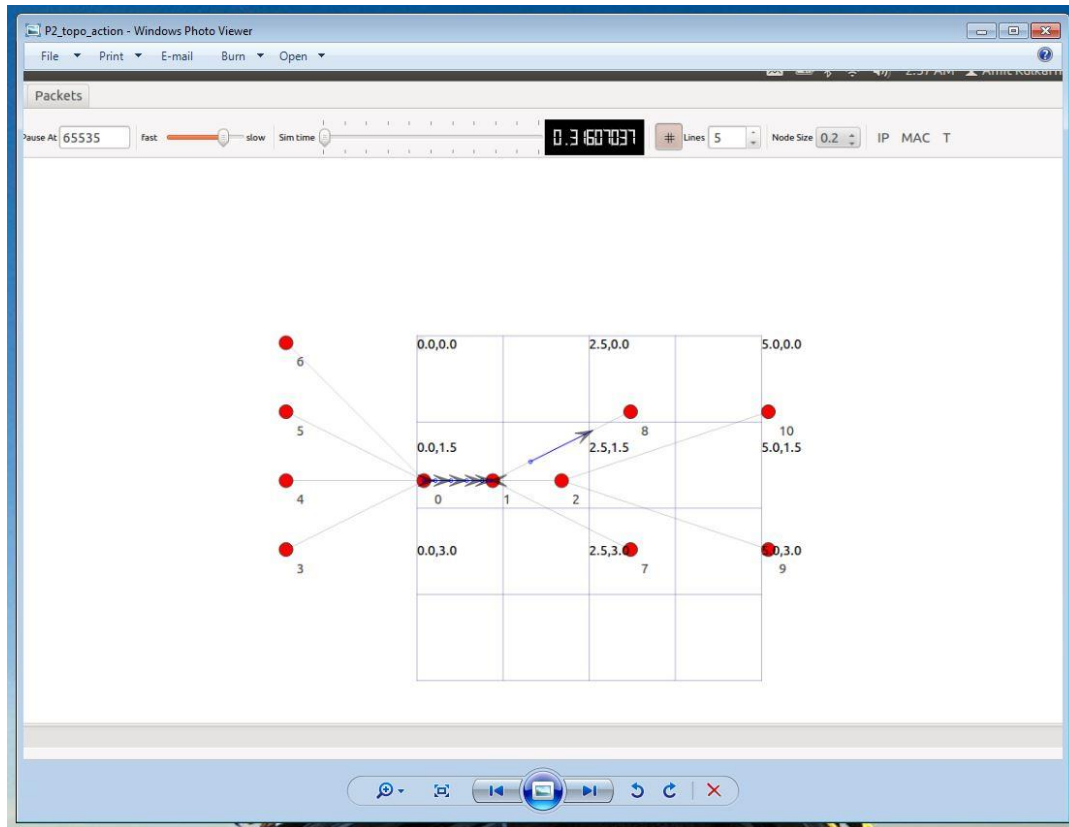
Majority of the internet traffic uses Transmission Control Protocol (TCP) as a transport layer protocol for end-to-end control of information transfer. Hence, it becomes necessary to measure, simulate and analyze the performance of TCP. In this project, we compare the performance of DropTail and Random Early Detection (RED) queueing methods for the topology shown below. Conclusions are made based on the results observed by changing the RED, DropTail and bottleneck link parameters.

**Topology:**



The topology shown above has four sources and four sinks. The traffic contains both TCP and UDP flows. The four sources are 2 UDP (UDP1 and UDP2) and 2 TCP sources (TCP1 and TCP2). Nodes 3 and 5 shown in the above topology have UDP sources, UDP1 and UDP2 respectively. Nodes 4 and 6 have TCP sources. Nodes 7 and 9 are respective UDP sinks and nodes 8 and 10 are respective TCP sources. Nodes 0, 1 and 2 are the router nodes. The sources and the sinks are arranged in such a way that some UDP and TCP flows experience more than one bottleneck link.

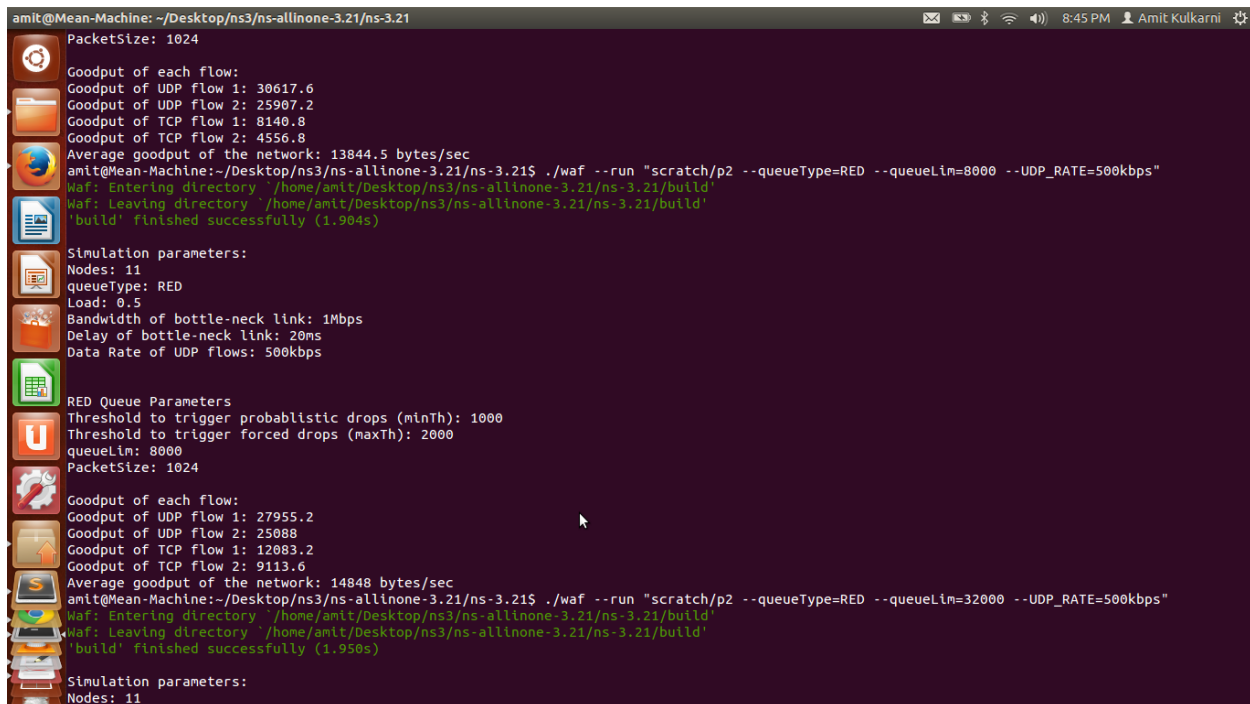
The link bandwidth and delay is kept constant at 5Mbps and 2ms. The bottleneck bandwidth and delay are varied to observe changes in the goodput. The topology in NetAnim looks as shown below.



All the sources share the network bandwidth. The natural behavior of the users is to utilize as much bandwidth as possible. However, in reality all networks are band limited. If a source tries to utilize as much bandwidth as possible, packets would be dropped and hence congestion occurs. TCP implements congestion avoidance algorithms but there is still a way in which TCP can be designed to “hog” the network. UDP, on the other hand, sends as much data as possible and drops packets in this process. Since, UDP is a non-reliable transport protocol, it keeps sending data. This leads to congestion in the network. Hence, to effectively control congestion, congestion avoidance/control techniques are applied at the router nodes. There are two methods of controlling congestion at the routers. Scheduling Algorithms which determine which sources will transmit its data and for what time period. Queueing Algorithms. These methods are implemented at router level by work by managing the flow at the buffers. This method is based on probability and is fairer than DropTail queue.

### Code:

The code takes in an input as 'queueType'. If queueType = RED, the default parameters of RED queue are applied to the bottleneck links between routers R0, R1 and R2. Similarly, if the queueType entered is 'DropTail', DropTail parameters are applied to the link. Provisions have been made to change the different link parameters from command line. The goodput of the network is calculated as a whole. Packets received from all the sources are counted and average goodput is calculated. The screenshot below shows the output of the program.



```
amit@Mean-Machine: ~/Desktop/ns3/ns-allinone-3.21/ns-3.21
PacketSize: 1024
Goodput of each flow:
Goodput of UDP flow 1: 30617.6
Goodput of UDP flow 2: 25907.2
Goodput of TCP flow 1: 8140.8
Goodput of TCP flow 2: 4556.8
Average goodput of the network: 13844.5 bytes/sec
amit@Mean-Machine:~/Desktop/ns3/ns-allinone-3.21/ns-3.21$ ./waf --run "scratch/p2 --queueType=RED --queueLim=8000 --UDP_RATE=500kbps"
Waf: Entering directory '/home/amit/Desktop/ns3/ns-allinone-3.21/ns-3.21/build'
Waf: Leaving directory '/home/amit/Desktop/ns3/ns-allinone-3.21/ns-3.21/build'
'build' finished successfully (1.904s)

Simulation parameters:
Nodes: 11
queueType: RED
Load: 0.5
Bandwidth of bottle-neck link: 1Mbps
Delay of bottle-neck link: 20ms
Data Rate of UDP flows: 500kbps

RED Queue Parameters
Threshold to trigger probabilistic drops (minTh): 1000
Threshold to trigger forced drops (maxTh): 2000
queueLim: 8000
PacketSize: 1024

Goodput of each flow:
Goodput of UDP flow 1: 27955.2
Goodput of UDP flow 2: 25088
Goodput of TCP flow 1: 12083.2
Goodput of TCP flow 2: 9113.6
Average goodput of the network: 14848 bytes/sec
amit@Mean-Machine:~/Desktop/ns3/ns-allinone-3.21/ns-3.21$ ./waf --run "scratch/p2 --queueType=RED --queueLim=32000 --UDP_RATE=500kbps"
Waf: Entering directory '/home/amit/Desktop/ns3/ns-allinone-3.21/ns-3.21/build'
Waf: Leaving directory '/home/amit/Desktop/ns3/ns-allinone-3.21/ns-3.21/build'
'build' finished successfully (1.950s)

Simulation parameters:
Nodes: 11
```

### Experiments and observations:

For the experiments, I ran the simulation to observe the results of DropTail queue for specified values of DropTail parameters and followed a similar process for RED queue. The goodputs of all the sinks are added and the average is taken for consideration.

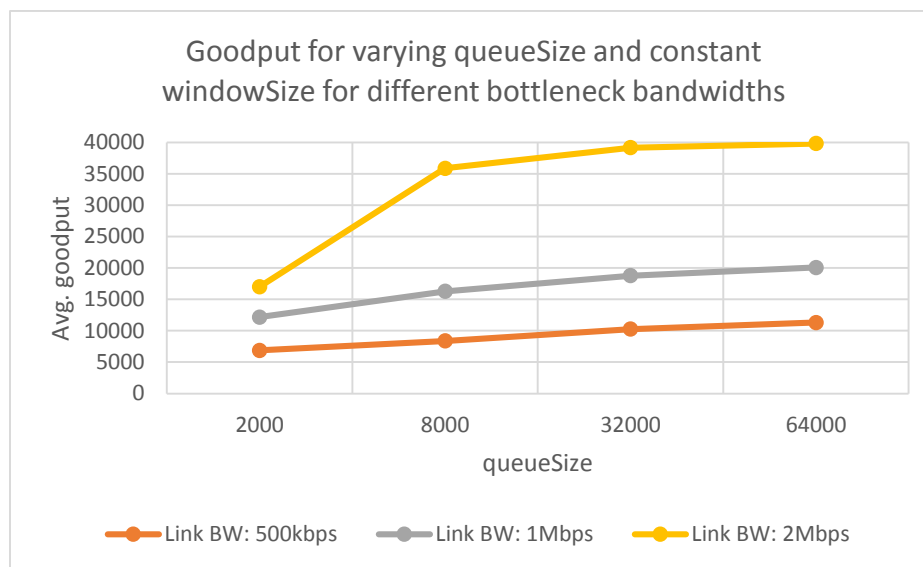
#### A] DropTail queue

DropTail queueing method is a simple active queue management (AQM) technique. This method is independent of the network traffic. It will drop the packets if the queue is full. Hence, the goodput of DropTail queue varies heavily with the increasing queueSize and bandwidth at the bottleneck links. For a given value of windowSize, the goodput increases with increasing value of

queueSize. The results obtained from this simulation are as shown below. It can be seen that as the bandwidth value increases the goodput increases. The goodput also increases for increasing value of queueSize. windowSize = 8000.

QueueSize	Goodput for BW = 500kbps	Goodput for BW = 1Mbps	Goodput for BW = 2Mbps.
2000	6881.28	12175.4	17008.6
8000	8366.08	16260.9	35860.5
32000	10270.7	18769.9	39157.8
64000	11305	20070.4	39823.4

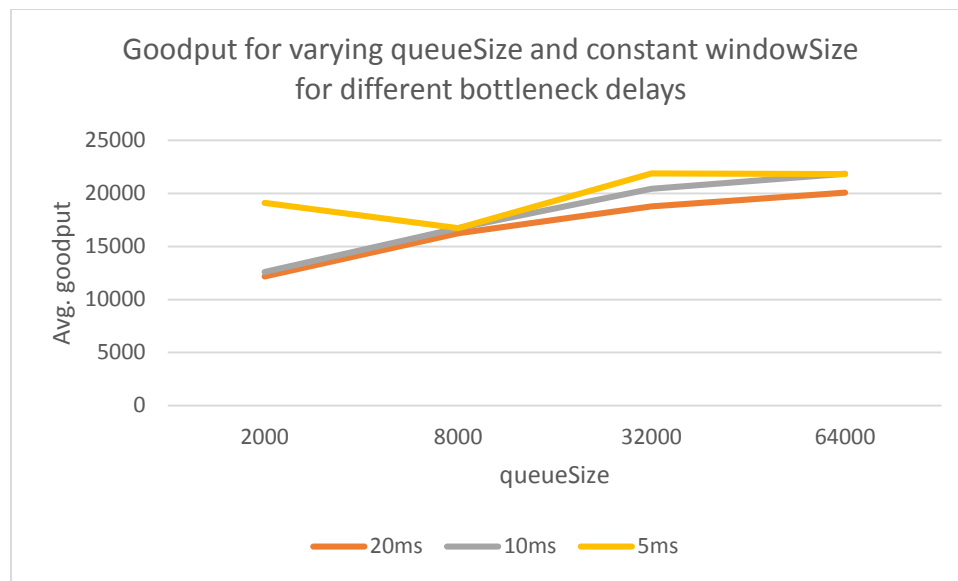
The graph below pictorially represents the values shown.



Similar experiment was done by changing the value of bottleneck delay. The goodput should increase for decreasing value of bottleneck delays and increasing values of queue sizes. The values given below confirm this result. The table shows the goodput for different queuesizes and bottleneck delays or the Round Trip Time (RTT).

QueueSize	2000	8000	32000	64000
20ms	12175.4	16250.9	18769.9	20070.4
10ms	12585	16732.2	20449.3	21852.2
5ms	19118.1	16732.2	21882.9	21841.9

The graph below pictorially represents the values shown.



It can be seen at the goodput is higher for smaller bottleneck link delay.

## **B] RED queue**

Random Early Detection or Random Early Discard is a more complicated queueing method compared to DropTail queue. It was observed in DropTail queue that, if the buffer was full, the packets were dropped. This is independent of the transport protocol used. Because of this, the network remains under-utilized. RED was designed to address this issue.

### Operation:

RED makes use of statistical probabilities to drop packets. This makes this method fairer as compared to DropTail. The operation is fairly simple to understand, if the router buffer is empty or almost empty. The probability of dropping the packers is low and almost all incoming packets are accepted. As the traffic increases and the buffer space decreases, the probability of dropping packets increases. When this probability reaches 1, all the packets are dropped and congestion takes place. The main parameters of RED are queueLimit, maxTh (Max queue threshold) and minTh( minimum threshold). maxTh parameter triggers forced drops and minTh parameter is a threshold for probabilistic drops. The goodput also depends on the load or the data rate of the sources.

For the experiments, I varied the value of  $\Delta$ , say,  $\Delta = (\text{maxTh} - \text{minTh})$ . For a given value of  $\Delta$ , the goodputs were observed for increasing values of data rate. Since, the data rate is directly proportional to the load, I vary the data rate in the code.

1]  $\Delta = (\max Th - \min Th) = (2000 - 1800) = 200$

QueueLimit	2000	8000	32000	64000
Data Rate (load)				
500kbps	13844.5	14008.3	15493.1	15493.1
1Mbps	13803.5	13905.9	13926.4	13926.4
2Mbps	13506.6	14745.6	14653.4	14653.4
2.5Mbps	13465.6	14571.5	14663.7	14663.7

2]  $\Delta = (\max Th - \min Th) = (2000 - 1500) = 500$

QueueLimit	2000	8000	32000	64000
Data Rate (load)				
500kbps	13844.5	14172.2	15902.7	15902.7
1Mbps	13803.5	13957.1	14592	14592
2Mbps	13506.6	14592	14725.1	14725.1
2.5Mbps	13465.6	14684.2	14735.4	14735.4

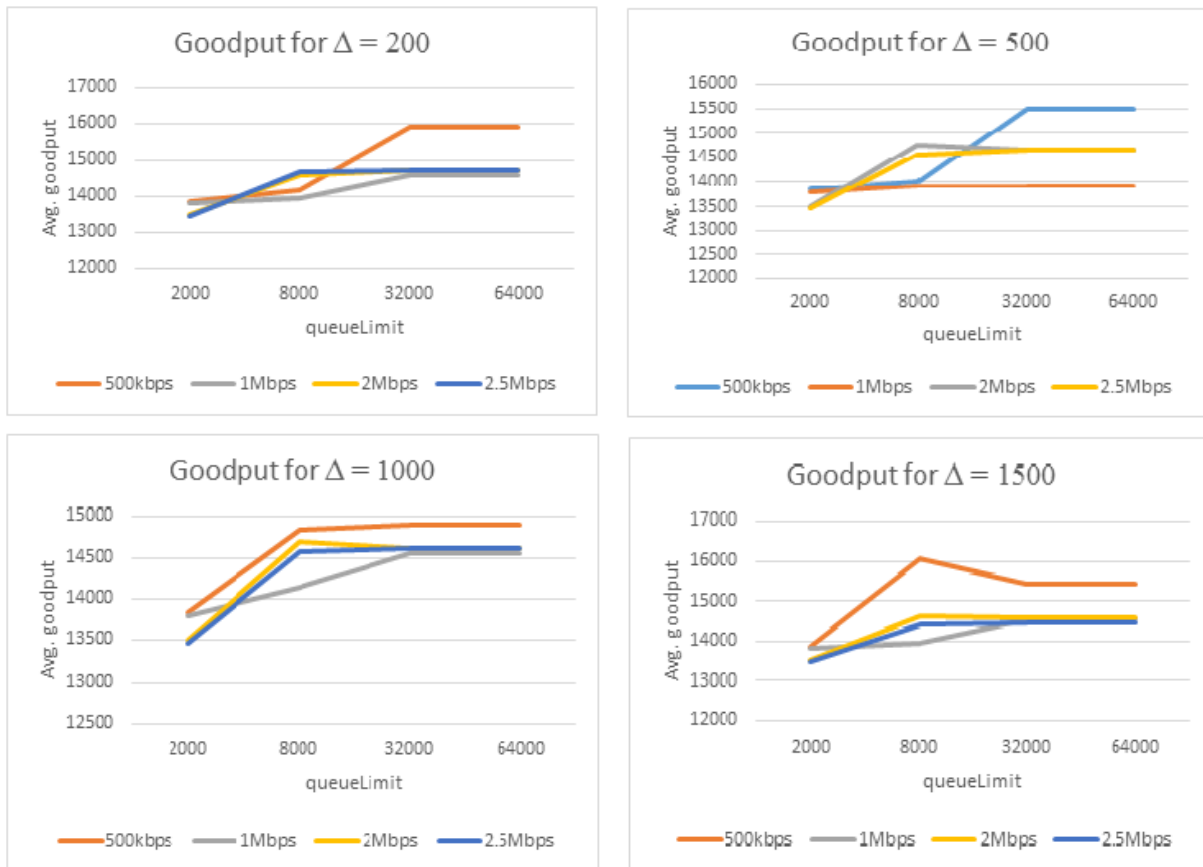
3]  $\Delta = (\max Th - \min Th) = (2000 - 1000) = 1000$

QueueLimit	2000	8000	32000	64000
Data Rate (load)				
500kbps	13844.5	14848	14889	14889
1Mbps	13803.5	14141.4	14551	14551
2Mbps	13506.6	14704.6	14612.5	14612.5
2.5Mbps	13465.6	14581.8	14612.5	14612.5

4]  $\Delta = (\max Th - \min Th) = (2000 - 500) = 1500$

QueueLimit	2000	8000	32000	64000
Data Rate (load)				
500kbps	13844.5	16046.1	15421.4	15421.4
1Mbps	13803.5	13926.4	14530.6	14530.6
2Mbps	13506.6	14602.2	14592	14592
2.5Mbps	13465.6	14428.2	14438.4	14438.4

If we observe the goodput values for every  $\Delta$ , for the queueLimit of 8000 and 32000. The goodput values increase as the value of  $\Delta$  increases. By increasing the difference between maxTh and minTh, the probability of dropping packets decreases and hence throughput increases. The graphs for the above results are shown below.

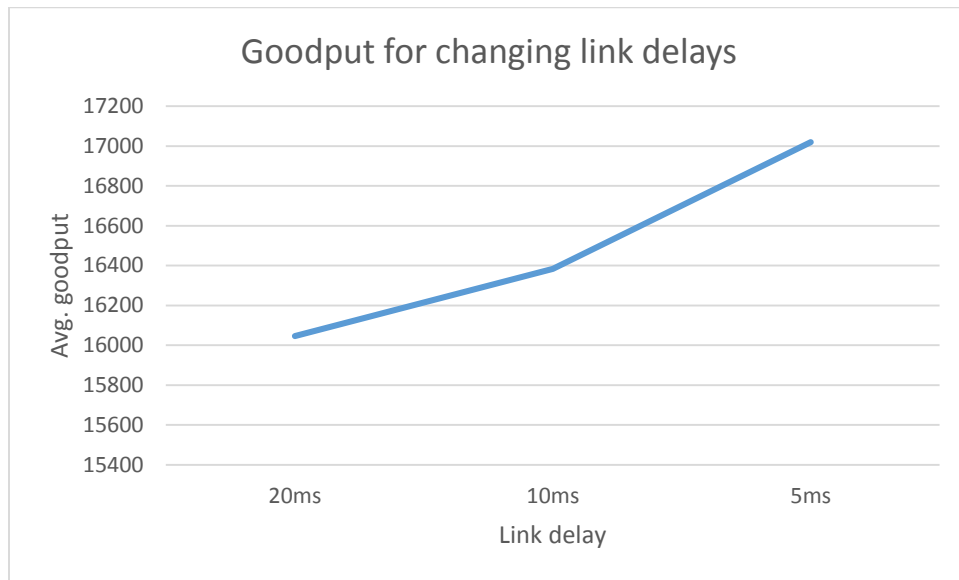


The above graphs show the goodput for different values of queueLimit and data rate.

For the final experiment, I kept the values of queueLimit = 8000, Bottleneck BW = 1Mbps, and observed the goodput for a constant data rate of 500kbps and varying values of bottleneck delay. The results observed are as follows:

Delay	Goodput
20ms	16046.1
10ms	16384
5ms	17018.9

It can be seen that as the delay decreased the goodput increased for the network.



The above graph shows increasing value of goodput for decreasing link delays.

### Conclusion:

After completing this project and observing the results, it can be seen that DropTail essentially avoids congestion by dropping packets from the buffer. If the queueSize is increased the, goodput of the network increases. The goodput of DropTail queue also depends on segment size and windowSize which we observed in project 1. For RED queue, the results comply with the theory. For increasing value of  $\Delta$  which is the difference between maxTh and minTh, the goodput of the network increases.

### References:

1. Floyd, S. and Jacobson, V. 1993. Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking.
2. Mikkel Christiansen, Kevin Jeffay, David Ott, F. Donelson Smith. 2001. Tuning RED for Web Traffic.
3. Wikipedia.