

# ECE 6110 – CAD-COMMUNICATION NETWORKS

## PROJECT 2 REPORT

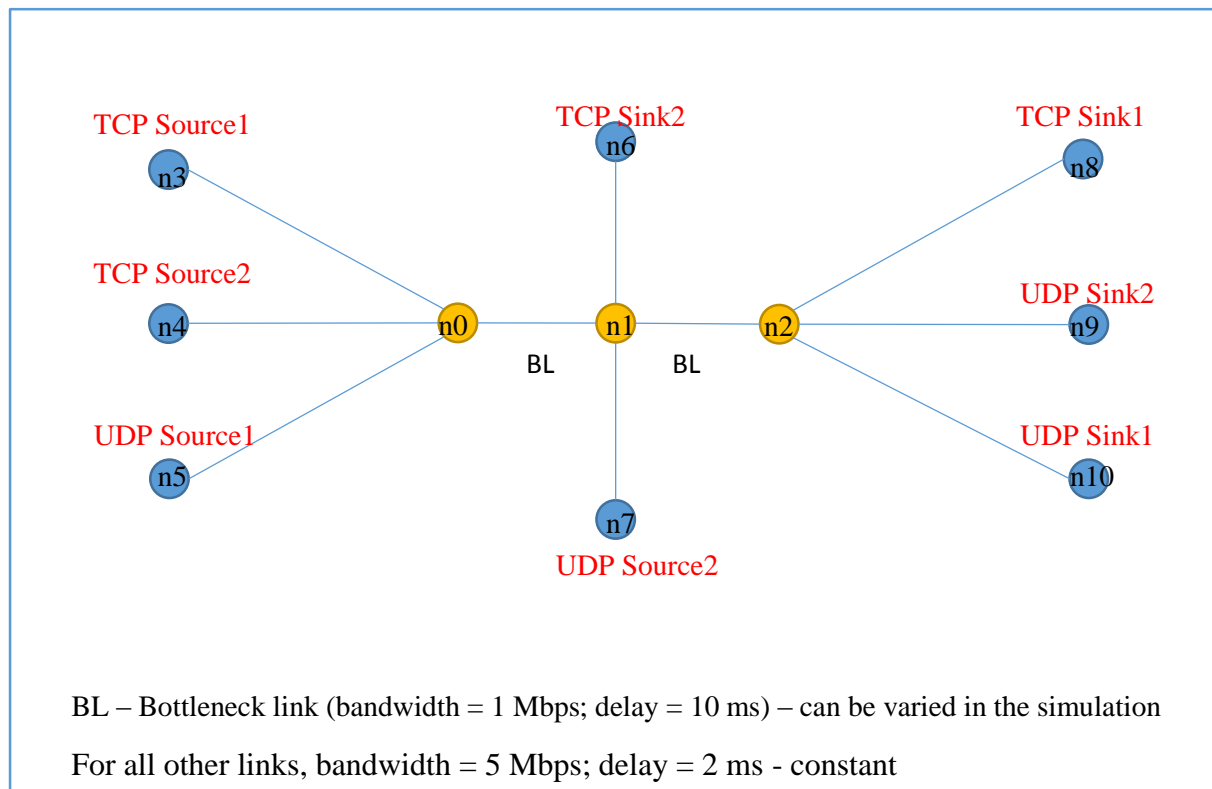
### Comparison of RED vs. DropTail Queueing

*Name: Uday Kiran Ravuri*

*T-Square username: uravuri3; GT ID: 903145978*

#### TOPOLOGY:

This project uses the ns3 simulator to compare the performance of RED Queue and DropTail Queue by varying several parameters as will be discussed below. A non-trivial network is set up and the variation in throughput is measured at the flow sinks. The topology used for this project is as shown below:



*Figure 1: Network Topology*

In the above figure, the yellow shaded nodes represent the routers. The blue nodes represent the sources and sinks. I initiated 4 flows (2 TCP and 2 UDP) in this network. This is in contrast to Jeffay's experiment wherein they used only HTTP (TCP) traffic for obtaining their results.

TCP flow 1 → from n3 to n8

TCP flow 2 → from n4 to n6

UDP flow 1 → from n5 to n10

UDP flow 2 → from n7 to n9

## PROCEDURE/PARAMETERS/LOGISTICS OUTLINE:

All these connections flow via the routers, i.e. n0, n1 and n2 which are connected by bottleneck links. We can see that some of the flows use only one of the two bottleneck links while others use both of them. Both the bandwidth and delay for the bottlenecks can be varied by the user.

The TCP flows use ns3's BulkSendApplication at their sources. This application tries to send maximum amount of data in the shortest duration possible. I have set the 'maxBytes' parameter to a very high value and so, the application never stops sending data. Hence, this source can be a major cause of congestion in the network. It tries to squeeze in as many packets as it can but always follows the congestion control rules of the underlying 'TcpTahoe' implementation. For simplicity, other TCP variants have not been considered in this project as I had already discussed some of them in the previous project. The 'packet size' used by the application can be varied by the user.

The UDP flows use the OnOff Application in ns3. UDP is a connection less protocol and only concentrates on sending data without automatically controlling its rate depending on the network conditions. Hence, a high UDP rate can give us a good estimate of a congested network. Traffic will be sent at a constant rate defined by 'DataRate' (when the application is in ON state) and with a 50% duty cycle. The parameters that the user can vary for this application are 'data rate' and 'packet size'. This is the primary load control mechanism that I used for my experiments.

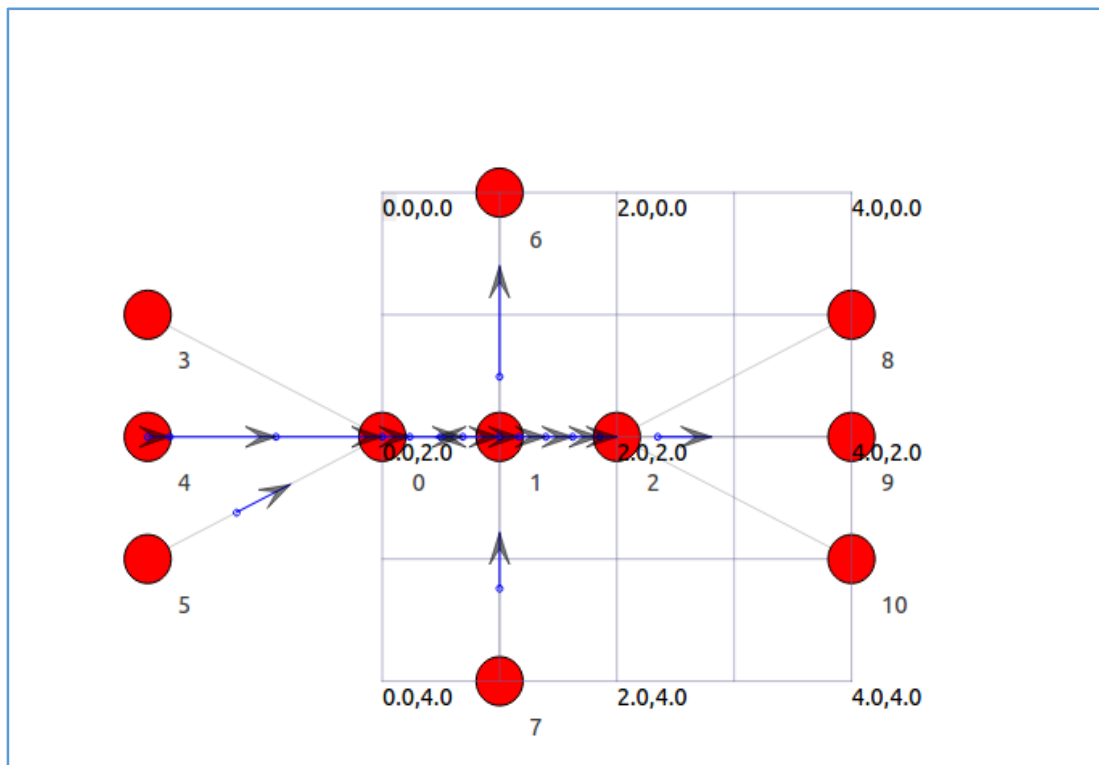
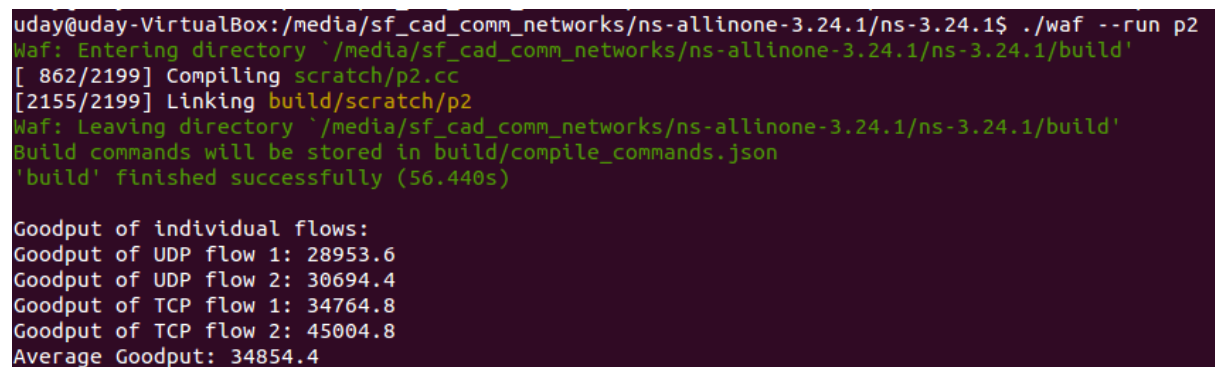


Figure 2: Snapshot of the network animator at simulation time  $t=0.52$  seconds

Queues are implemented only on the bottleneck links. Either 'DropTail' or 'RED' is implemented at a time for a simulation. RED is based on probability and is theoretically fairer than DropTail queue. For DropTail queue, the only parameter varied is 'queue size' (in bytes). For RED queue, the parameters varied are minimum threshold for triggering probabilistic drops (minTh), maximum threshold for forced packet drops (maxTh) and queue limit. The other RED parameters such as queue weight and maximum drop probability have not been varied in this experiment. In addition, the queue size of DropTail queue and the queue limit of RED queue are set to the same value for fair comparison.

Other TCP parameters whose effects are considered here are segment size (segSize) and receiver advertised window size (windowSize).

Each simulation is run for 10 seconds and the average goodput for all four flows is used for comparison. This measure is justified because the goodput for the individual flows follow the same trend with respect to each other for most of the simulations. For example, Tcp flow 2 will have better goodput than Tcp flow 1 as the latter is traversing two bottleneck links before reaching the destination. The network topology is asymmetric so the Tcp goodputs also depend on Udp data rates. However, we are not comparing the fairness among flows here and hence, I considered the average of the goodputs as a base quantity. A snapshot of the stdout is shown in Figure 3.



```
uday@uday-VirtualBox: /media/sf_cad_comm_networks/ns-allinone-3.24.1/ns-3.24.1$ ./waf --run p2
Waf: Entering directory `/media/sf_cad_comm_networks/ns-allinone-3.24.1/ns-3.24.1/build'
[ 862/2199] Compiling scratch/p2.cc
[2155/2199] Linking build/scratch/p2
Waf: Leaving directory `/media/sf_cad_comm_networks/ns-allinone-3.24.1/ns-3.24.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (56.440s)

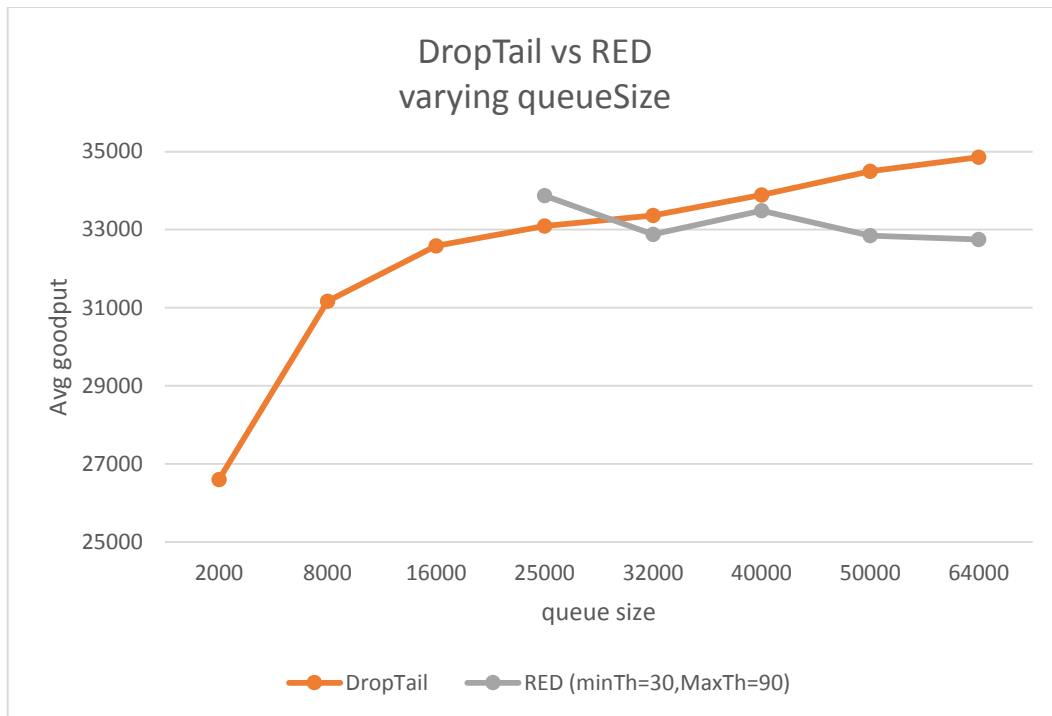
Goodput of individual flows:
Goodput of UDP flow 1: 28953.6
Goodput of UDP flow 2: 30694.4
Goodput of TCP flow 1: 34764.8
Goodput of TCP flow 2: 45004.8
Average Goodput: 34854.4
```

*Figure 3: Standard output for a simulation with default parameters*

## EXPERIMENTS AND OBSERVATIONS:

For the experiments, I ran the simulations to observe the results of DropTail queue by varying above mentioned parameters and then I followed a similar procedure for RED queue.

From the figure below, it is clear that goodput increases with increasing DropTail queue size. The reason is that more packets can be accommodated in a larger queue and leads to better performance. This trend saturates for higher queue sizes because the effects of other parameters kick in and dominate the effect of queue size. This was discussed in detail in Project 1.

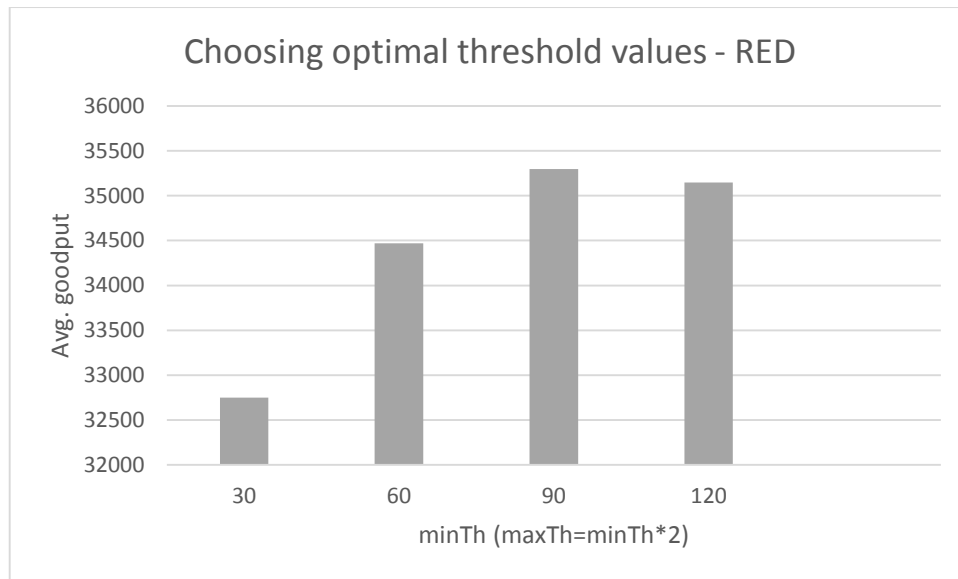


*Figure 4: Comparing DropTail and RED by varying queue size*

We can observe that there is not much improvement in case of RED queue (the goodput for queue sizes less than 25000 have not been plotted since maxTh parameter exceeds the queue limit). For the above simulation, I chose minTh as 30 packets and maxTh as 90 packets, with a fixed packet size of 256 bytes. The RED trend can be explained by the choice of the thresholds. As queue size increases, the optimal values (for best performance) for the thresholds also change accordingly. As we kept the same thresholds, performance actually decreases. It is for the same reason that the DropTail and RED lines cross each other. In other words, for a queue limit of 25000 bytes and the specified thresholds (maxTh \* packetSize is close to queue limit, in this particular case), the probability mechanisms used by RED reduces the packet drops by indirectly informing the Tcp sender to reduce his rate before the packets can actually be dropped. Because of this early detection and notification, the goodput is better for RED than DropTail. But when the queue size is increased above 25000, the thresholds do not seem to produce the said effect and hence DropTail overtakes RED.

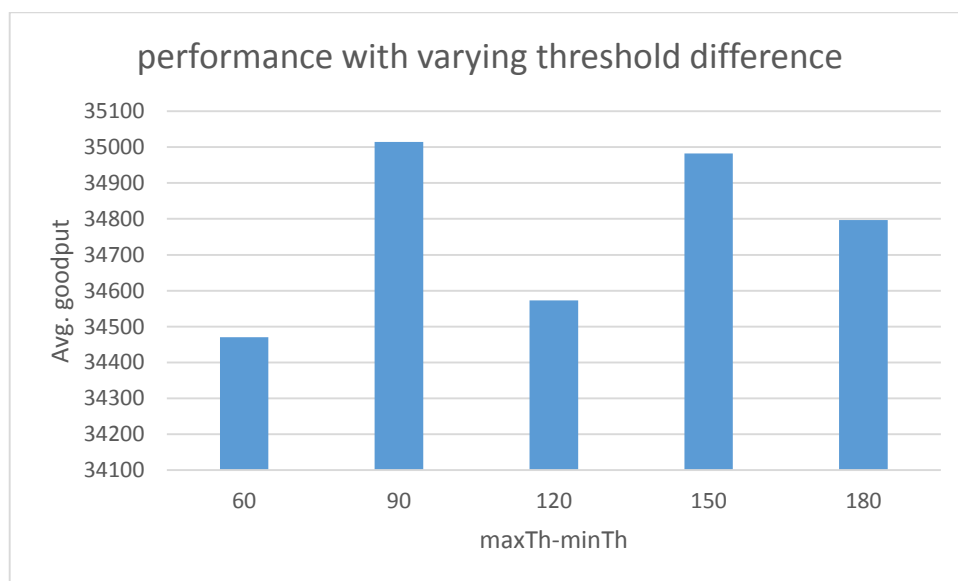
The paper by Van Jacobson and Sally Floyd suggested that maxTh should be atleast twice minTh. Keeping this in mind, for a fixed queue size of 64000 bytes, I ran simulations to identify the optimal range of minTh and maxTh. The resulting plot is shown below.

From figure 5, we can infer that the optimal values of minTh and maxTh are 90 and 180 respectively. The queue limit in this case has been fixed to 64000 bytes. For a packet size of 256 bytes, a maximum of 250 packets could be accommodated. When the maxTh is 240 packets (last case), it is very close to the queue limit. The queue triggers forced drops when the average queue size reaches this value. But, when the average is this high, the instantaneous value could easily be matching the queue limit. Hence, this threshold setting is not keeping the average queue size in appropriate bounds. So, the goodput is affected.



*Figure 5: Goodput variation for different sets of RED thresholds*

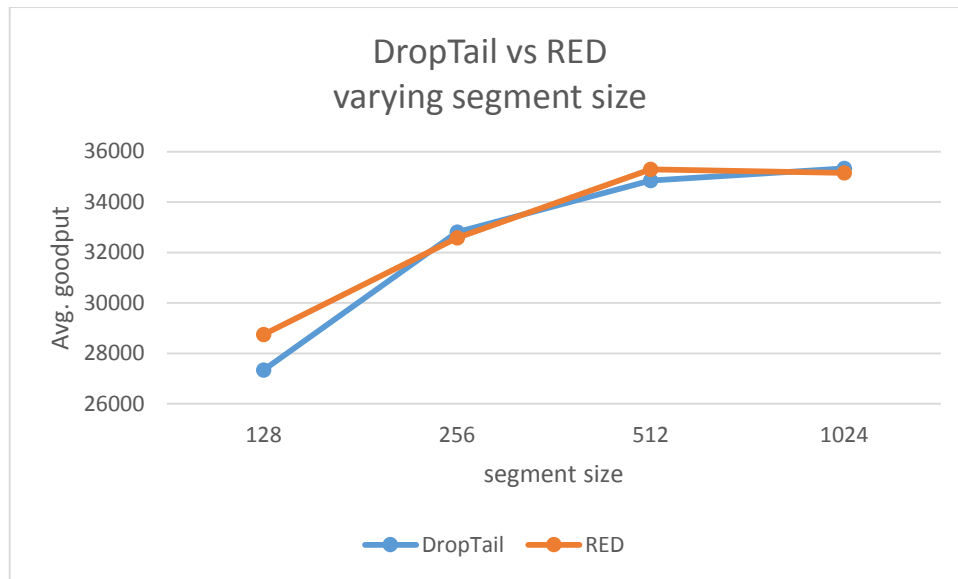
Now, let's vary the difference between the two threshold quantities. The obtained results are plotted in Figure 6.



*Figure 6: Goodput variation with increasing (maxTh-minTh) values*

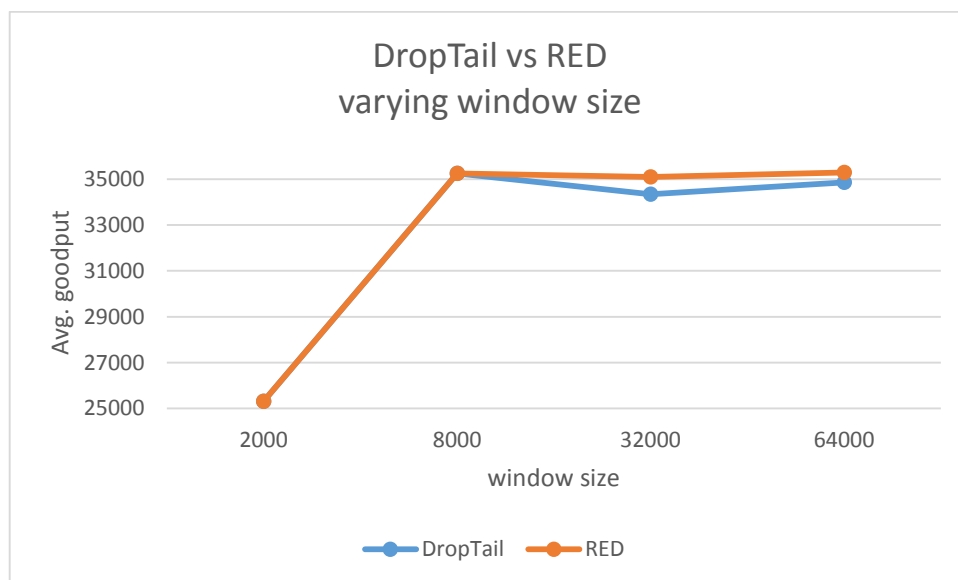
From this figure, we can conclude that for a queue size of 64000 bytes and a packet size of 256 bytes and a minTh of 60 packets, maxTh of 150 packets seems to be the best choice for performance. That is, it is better to choose a maxTh value which is close to twice the minTh value for optimal goodput. This concedes with the argument presented in the Floyd paper.

Now, we compare goodput for DropTail and RED queues by varying TCP parameters such as segment size and window size. For this observation, queueSize = 64000 and we choose the optimal values of thresholds obtained from our previous results, i.e. minTh = 90 packets, maxTh = 180 packets. The results are shown in the figures below.



*Figure 7: Comparing DropTail and RED by varying TCP segment size*

In the above figure, we observe that both the queues are performing almost the same with varying TCP segment sizes. Segment sizes affect the bandwidth shared among the flows (fairness) more than they affect queue performance. For low segment sizes however, RED dominates. This is due to the fact that more number of packets will need to be sent across the network. Hence, an algorithm which can manage queues efficiently (theoretically) will show best performance. As RED probabilistically detects congestion before it ever occurs, we can expect it to handle congestion effectively when more packets are involved in the network. Hence, goodput is better for RED for low segment size.

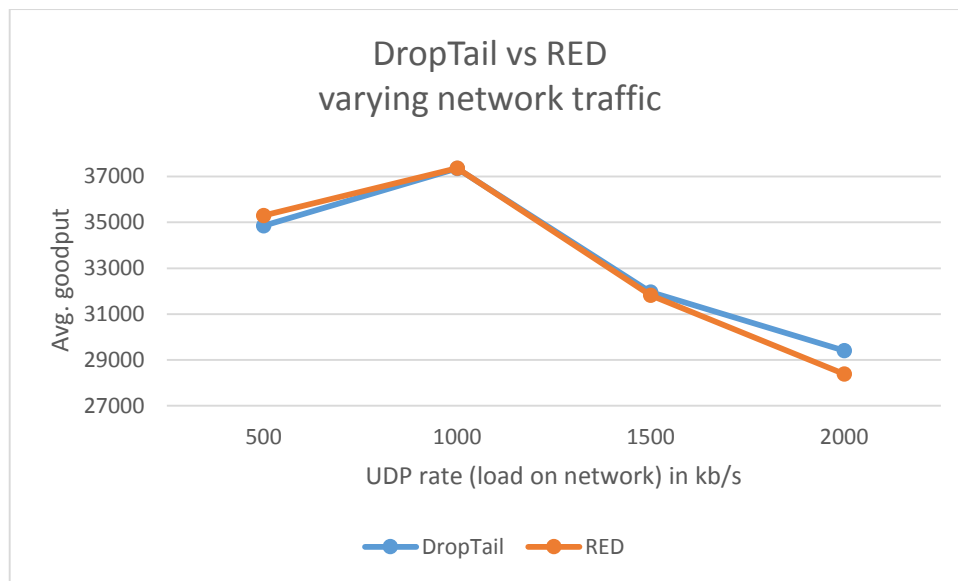


*Figure 8: Comparing DropTail and RED by varying receiver advertised window size*

We see that a window size of 2000 and 8000 bytes produce the exact same performance for both queues given this network topology. This signifies the importance of the receiver window. If the receiver does not have a sufficient sized buffer, then no matter how the

queueing is controlled, the goodput will be low because the sender will choose the minimum between CWND and receiver advertised window size. For higher advertised windows, RED algorithm proves to be slightly better than just dropping packets arriving at the end of queue although the goodput does not differ by much.

Now, we modify the data rates of the 2 UDP flows which share the bottleneck links with the TCP flows. As 'maxBytes' is set to infinity, the flow don't stop sending packets until the end of the simulation. So, a data rate means more traffic on the bottlenecks per unit time. This way, I am simulating load variation on network. By increasing the data rate, we expect to increase congestion in the network and hence produce more stress on the queue handling algorithms.



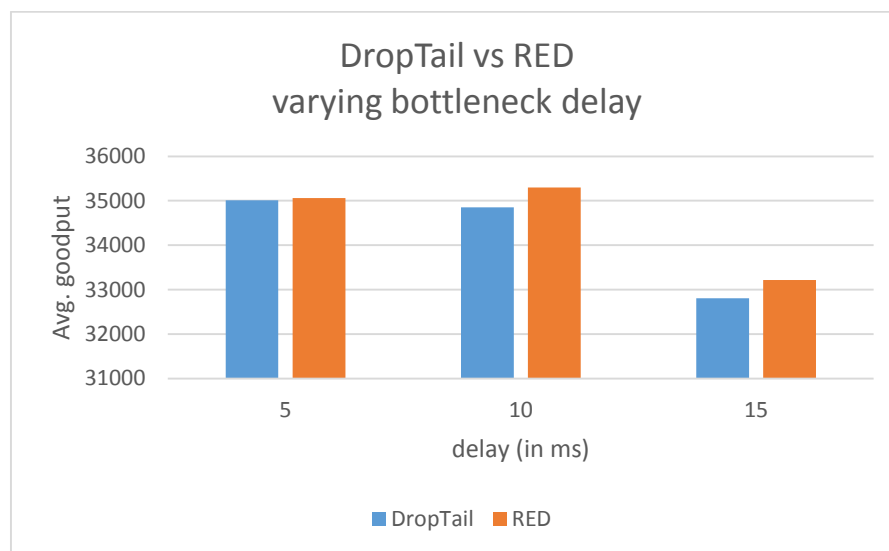
*Figure 9: Comparing DropTail and RED by varying amount of traffic in network (bottlenecks included)*

We have seen from the Floyd paper that RED queue is unbiased towards any flow including flows with bursty traffic. The TCP BulkSendApplication is sending packets at a fairly high rate. If the UDP rate is also high, then RED rejects packets from both flows according to their data rates. In addition, the average queue size in such a situation would almost always be higher than maxTh to trigger forced drops. So, RED queue tends to drop packets unnecessarily to try and avoid congestion whereas in DropTail case, the packets from one flow can still proceed while the other Tcp flow is in slow-start phase because the queue buffer space will be available after a little duration of having a completely full queue. In other words, as the RED queue fills up, the probability with which packets are marked or dropped increases and hence more drops occur for a given flow in case of a high data rate. This seriously affects the increase in CWND of a Tcp flow, as it has to start from SS phase multiple times in a particular period of time (we are using Tcp Tahoe in our experiments).

When I experimented with different bottleneck capacities, DropTail and RED queues showed almost the same goodput as shown in the table below.

Bottleneck BW	DropTail	RED
0.5Mbps	18048	18227.2
0.8Mbps	27494.4	27782.4
1Mbps	34854.4	35296
2Mbps	58643.2	58393.6
3Mbps	83654.4	83648

The figure below plots the goodput achieved by a DropTail queue and RED queue implementation with varying delays on the links connecting the routers. The values considered are 5, 10 and 15 ms. It shows that RED performs better than DropTail for higher delays (which also affect the round trip times, especially for TCP flows). So, given this network topology, we should choose RED for higher RTTs. As RED maintains a lesser average queue size, in general, the queueing delay and hence RTT is less than for a DropTail queue.



*Figure 10: Comparing DropTail and RED by varying bottleneck link delays*

## CONCLUSION:

For the chosen network topology, RED is found to perform better under the following situations: Less router bandwidth, high RTT, low segment size and low queue size.

For other cases, DropTail shows a slightly higher goodput. The results also depend highly on the threshold values chosen for a RED queue. I have experimented with the optimal values. Using other values could have reduced its performance. Varying the TCP parameters did not show much change in the performance of these queues. Major variations were seen for queue parameter and bottleneck link parameter changes. A network administrator should keep all these factors in mind, apart from the topology, to choose the queue which gives the required benefits.