Project 2: RED vs. Drop Tail Queuing

The following topology was constructed in order to examine the effects of TCP and UDP traffic at a bottleneck, using a RED queue. The bottleneck links are installed between n1 and n2 and n2 and n3, and the common sink is n4.

```
// Network topology
//                       (UDP)                (UDP)
//                        n5                   n6  (enable/disable)
//                        |                    |
//                        |  2 Mbps            |  4 Mbps
//                        |  10 ms             |  10 ms
//                        |                    |
//       n0------------- n1(R)--------------n2(R)---------------n3(R)------------n4 (Sink)
//    (TCP)
//             5 Mbps           1 Mbps               1 Mbps           5 Mbps
//             10 ms            20 ms                20 ms             10 ms
//
// - Flow from n0 to n4 using BulkSendApplication.
// - Flow from n5 to n4 using OnOffApplication.
// - Flow from n6 to n4 using OnOffApplication.


// Use: ./waf --run "p2v4 --red_dt=DT/RED --queueSize=integer --windowSize=integer --minTh=integer --maxTh=integer
//                       --qw=integer --maxP=interger --RTT=integer --dRate=DataRate"
```

This report is structured in such a way that, the results of the DropTail vs. RED experiments are presented first and the effects of tuning the RED queue to find the best configuration are shown later. RED showed a slightly better performance than Drop Tail at high loads, and a comparable and sometimes better performance than Drop Tail at low loads, for this particular topology and with the variation of parameters that have been presented below, however further experiments need to be performed using different configurations and variation of parameters.

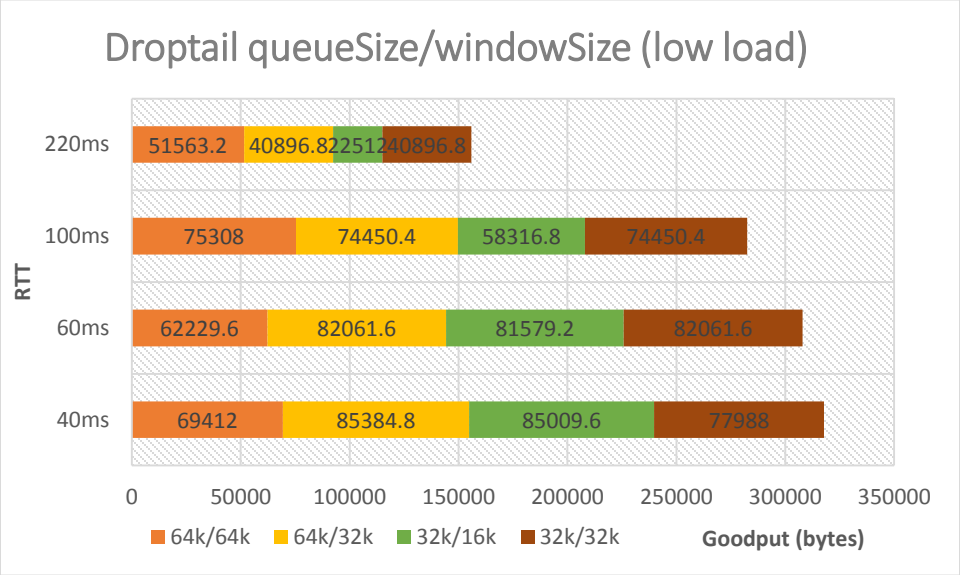
Run Script Format:

./waf --run "p2v4 --red_dt=DT/RED --queueSize=integer --windowSize=integer --minTh=integer --maxTh=integer --qw=integer --maxP=interger --RTT=integer --dRate=DataRate"

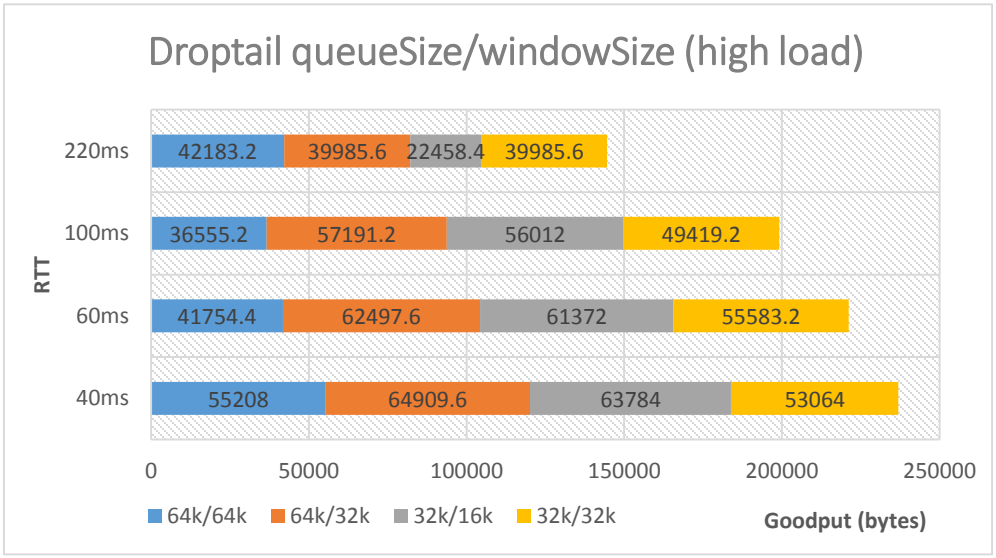
Note: The data rate specified in the --dRate variable, is for both the nodes n5 and n6 and is in the ns3 DataRate format. (ex: 0.1Mbps or 1kbps)

Section 1: Droptail vs RED

In this case, we can compare the performance of Drop Tail and RED queues. First, we identify the best drop tail queue configuration for this topology by varying queue size, window size and load in the network.



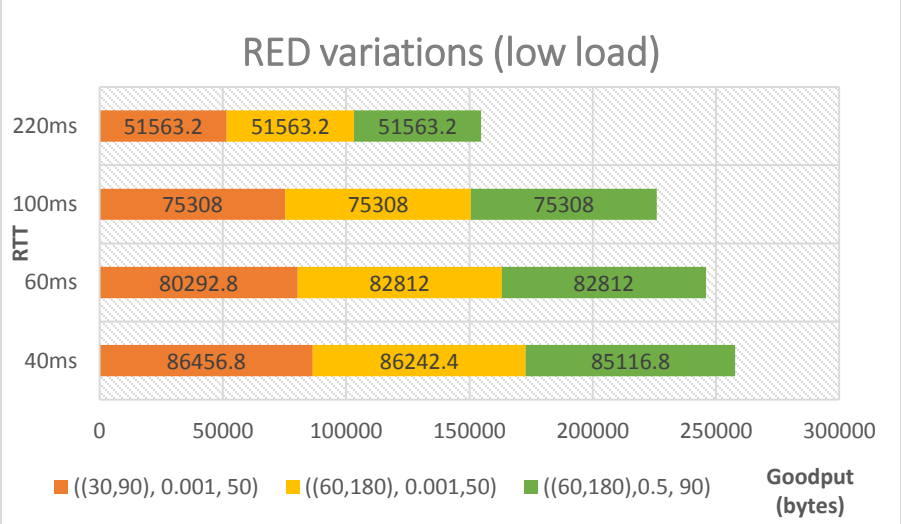
As we observe, in the above configuration with low load, the queueSize/windowSize combination that presents the best goodput on average among all RTTs is the 64k/32k configuration. This value can be used in order to compare Drop Tail and RED queues at low load conditions at various RTTs.



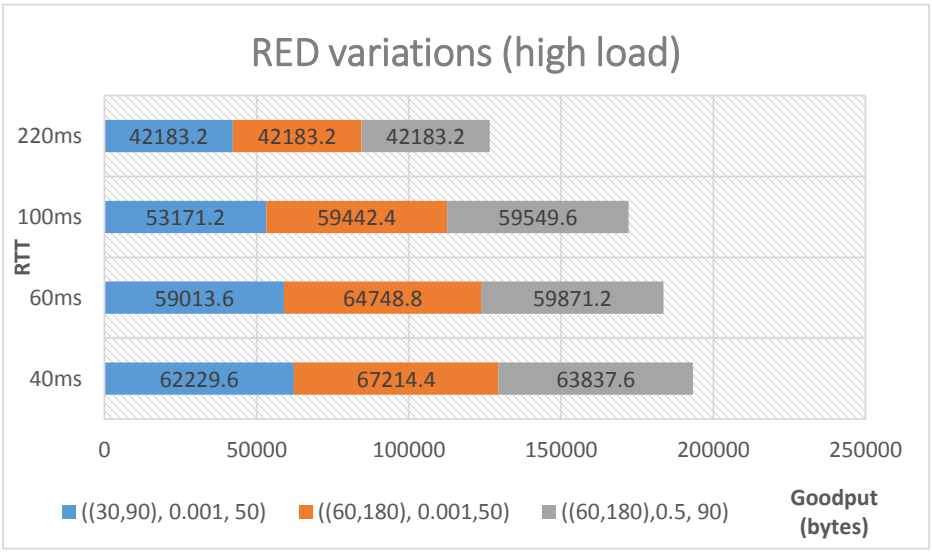
As we observe, in the above configuration with high load, the combination that presents the best goodput on average among all RTTs is still the 64k/32k configuration.

From the analysis above, we pick the best performing configurations for the RED queue and test them in both low and high load conditions for the same varying RTT values as shown with Drop Tail. Then by comparing the best performing RED and drop tail values at both low and high load conditions, we can estimate, although with not complete certainty, which queue management system shows a better performance for the given topology.

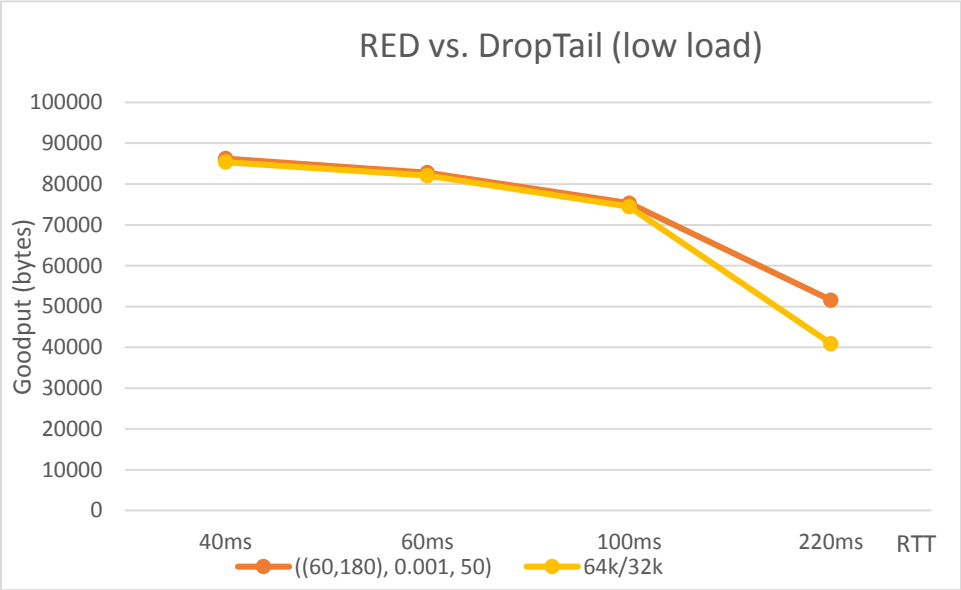
Now we compare the performance of RED at these load and RTT conditions with the following (min/max threshold, QW, maxP) **configurations, ((30,90), 0.001, 50); ((60,180), 0.001, 50); ((60,180), 0.5, 90); and why these are chosen is shown in section 2**.



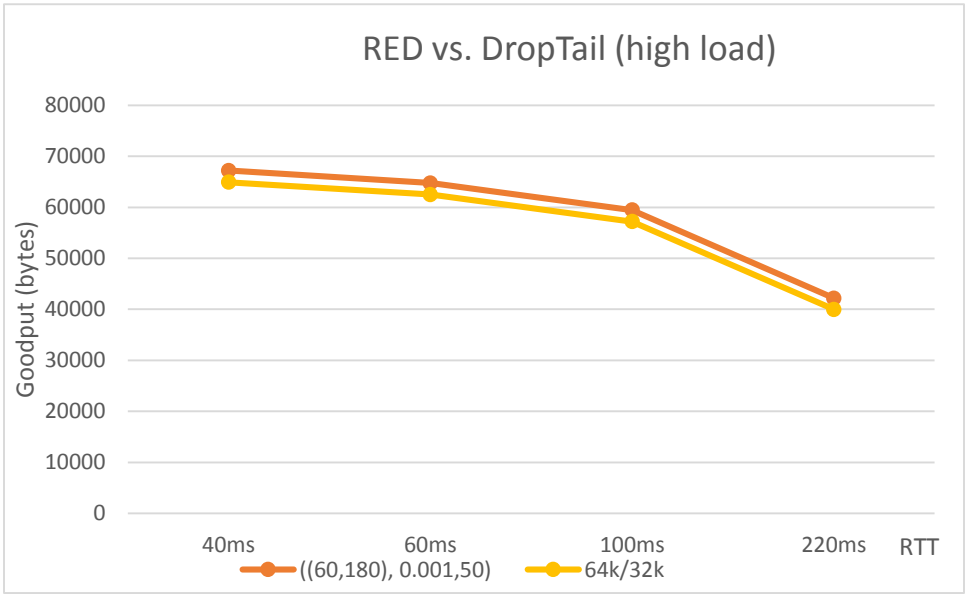
As we can see, the changes in goodput in low load and high load are uniform and are not affected by any particular configuration, and in both cases, the ((60,180), 0.001, 50) configuration yielded the best results. Hence, we can compare the performance of this configuration at both low and high loads, at various RTTs.



Conclusion:



At low loads, the "best configurations" (this is the most probable best configuration after section 2 and as suggested by the Tuning RED paper), we see that for low loads, RED is only slightly better for low RTTs, while for a high RTT RED is clearly better than Drop Tail.



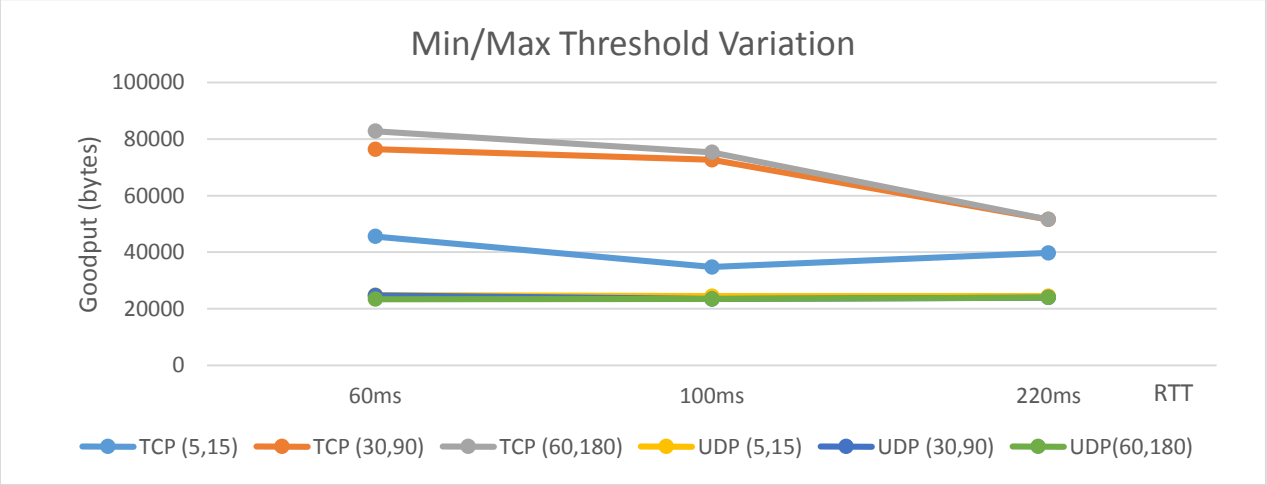
At high loads, the RED queue performs slightly better than the Drop Tail queue at almost all RTTs, and **RED offers a slightly better Goodput compared to DropTail**.

Section 2: Varying RED parameters

The Queue Limit parameter is set at 480 packets.

Effect of Min_{th} and Max_{th}:

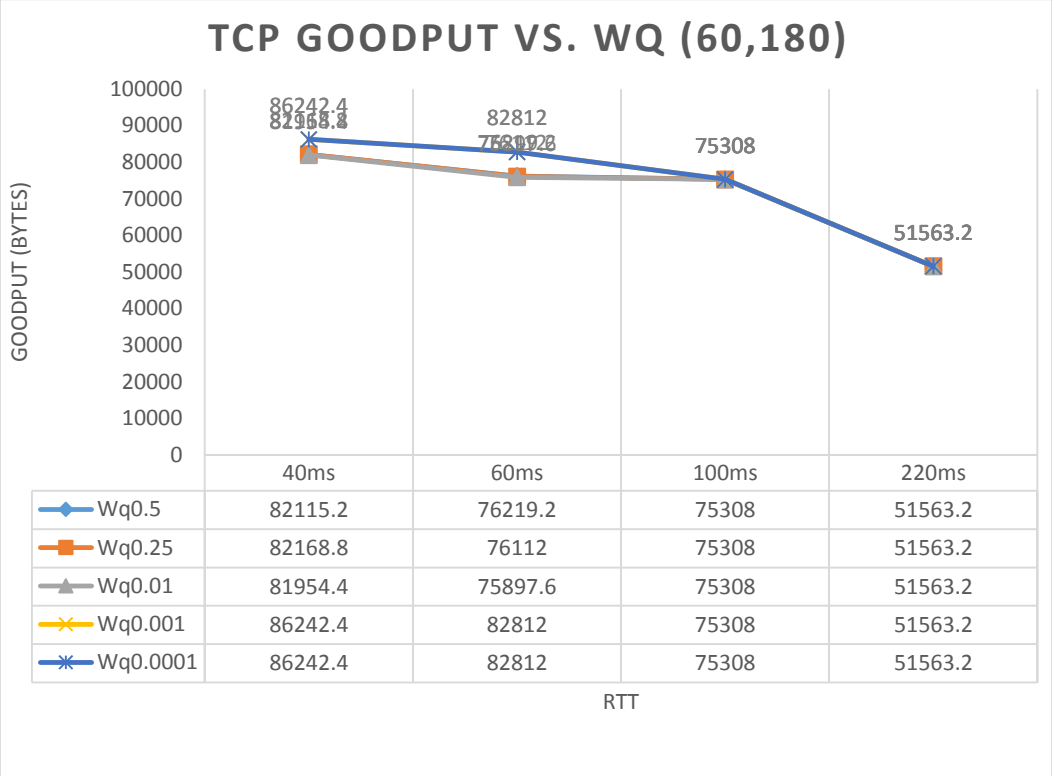
First, we keep the parameters for the UDP source constant, DataRate transmitting at 0.1 Mbps with the second UDP source switched off (initially), therefore only two flows pass through the bottleneck link. Wq, which is the sampling rate for the exponential moving average estimator is set to 1/packetSize (1/1024), as suggested by the "Tuning RED" paper. Now, in order to test the goodput for Min_{th}=5 and Max_{th}=15, Min_{th}=30 and Max_{th}=90, Min_{th}=60 and Max_{th}=180. It was suggested that the (30,90) would achieve the best overall performance and the (60,180) value gives a better goodput in the case of longer response times, we shall now test these values using different round trip times:



Observations:

- The (60,180) setting performed slightly better than the (30,90) setting at lower RTTs (60ms), however at higher RTTs (120ms) both the settings delivered a considerably good performance.
- The (5,15) setting performed poorly for the TCP flow, however one could argue that it was fair as it tried to equalize the goodput between the two flows.
- The UDP flow's goodput was rarely affected by the change in the parameters and it showed slightly better performance with the (5,15) setting at larger RTTs, than smaller ones.

Turn on Source 2 (UDP): Goodput of (60,180) still greater than the (30,90) case, even after increasing the load on the network by turning on source 2 at 0.1Mbps, although TCP goodput decreases proportionately.

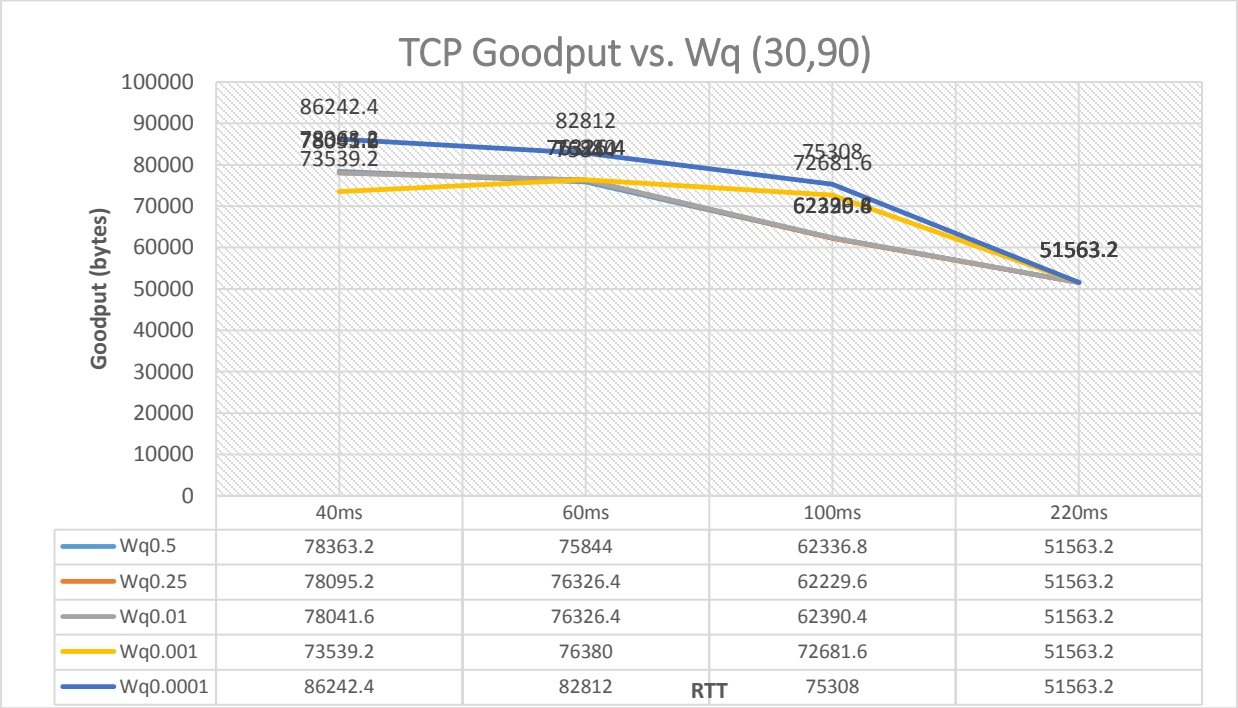


	40ms	60ms	100ms	220ms
Wq0.5	82115.2	76219.2	75308	51563.2
Wq0.25	82168.8	76112	75308	51563.2
Wq0.01	81954.4	75897.6	75308	51563.2
Wq0.001	86242.4	82812	75308	51563.2
Wq0.0001	86242.4	82812	75308	51563.2

Overall, the value (60,180) seemed to show a better goodput, in most cases, and even when the **congestion over the link was increased by turning on the second UDP source (3 flows)**, the performance of **(60,180) seemed to perform better** than all the other settings.

Effect of varying Wq (QW):

In this case, we vary the parameters for RTT and keep the UDP source DataRate transmitting at 0.1 Mbps with the second UDP source switched off, therefore only two flows pass through the bottleneck link. Wq, which is the sampling rate for the exponential moving average estimator is set to 1/packetSize (1/1024) by default, but now we vary this parameter to 0.25 (highly reactive setting, the changes in goodput reflect immediately in the queue's behavior) and less reactive values like 0.002, 0.001, 0.0001, and we checked this for a min/maxth setting of (30,90) and (60,180) :



Observations:

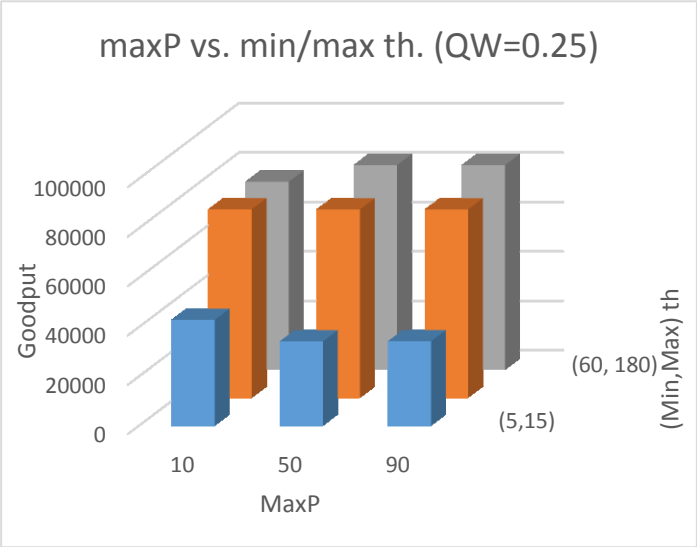
- Variations in the value of QW only when the RTT is low, and over a larger RTT the goodput saturates to the same value in both cases of min/max thresholds (30,90) and (60,180).
- The larger values of QW are preferred in the case of short RTTs, as there is a slight improvement in goodput at these values in both the cases.
- There is an exception for the value of 0.0001, in both cases where the goodput increased to a great extent in both cases, and this result might be neglecting the Wq parameter altogether.
- The min/max threshold setting of (60,180) consistently gives a better goodput in most cases considered compared to the (30,90) case.

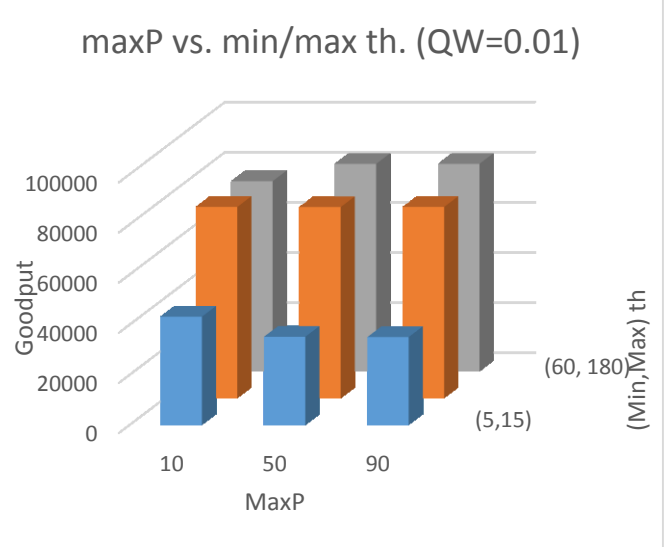
Turn on Source 2 (UDP): The results are consistent even after turning on source 2 and setting it to 0.1Mbps although TCP goodput decreases proportionately.

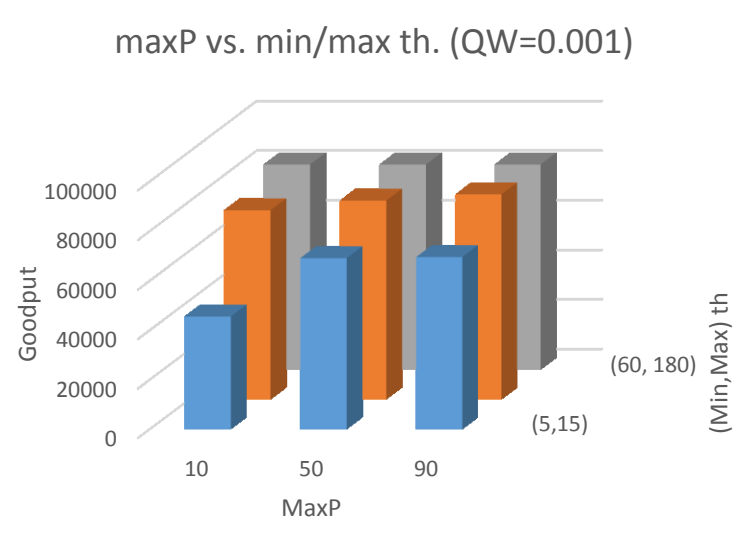
Overall, setting the Wq (QW) parameter to 1/packetSize yields the best results as pointed out in the Tuning RED paper, and the results do show this.

Effect of Maximum drop Probability:

In this case, we keep the parameters for RTT constant (60ms) and keep the UDP source DataRate transmitting at 0.1 Mbps with the second UDP source switched off, therefore only two flows pass through the bottleneck link. Wq, is set to 1/packetSize (1/1024), default and we want to vary maxP with the following values 1/10, 1/50, 1/90. We also consider the effect maxP has on max/min thresholds and QW variations:







Observations:

- A larger value of maxP affects the goodput of the larger flows and improves the goodput of starved flows, hence a certain degree of **fairness can be observed as we increase maxP**.
- The default value of maxP being 1/50 is not always good, in most cases, setting it to a value of 1/90 was found to yield a slight improvement in TCP goodput compared to the other cases.
- When the QW value (0.25) was increased (more reactive to changes in goodput); then the (5,15) tuple reacted to the changes in maxP more than the other values of min/max threshold. The larger threshold values were relatively immune to this variation.
- When the QW value (0.001) was reduced(less reactive), the goodput generally improved in all cases and the (5,15) case showed a greater degree of improvement, and maxP variation caused great changes in goodput. In the case of (30,90) the goodput showed a slight improvement when

it was set to 1/90, in the final case and we can see the evident increase in goodput in the final case.

It can be seen that even though in a few cases a high value of maxP is beneficial, it doesn't give maximum average goodput, and hence a low value of maximum dropping probability is preferred, and it can be seen that maxP can be set in **the range 1/90 to 1/50 for optimal performance**.

Now that we have found some good working configurations of RED for the specified topology, and the comparison with the performances of the RED queue with that of the DropTail queue have been presented above. Further experiments using various configurations and various variations in parameters need to be performed order to present a more through conclusion.

References

[1] Christiansen, M., Jeffay, K., Ott, D., & Smith, F. D. (2000, August). Tuning RED for web traffic. In *ACM SIGCOMM Computer Communication Review* (Vol. 30, No. 4, pp. 139-150). ACM.

[2] Floyd, S., and Jacobson, V., Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413. Available via http://wwwnrg.ee.lbl.gov/nrg/