PROJECT 2: Comparision of RED vs. DropTail Queuing

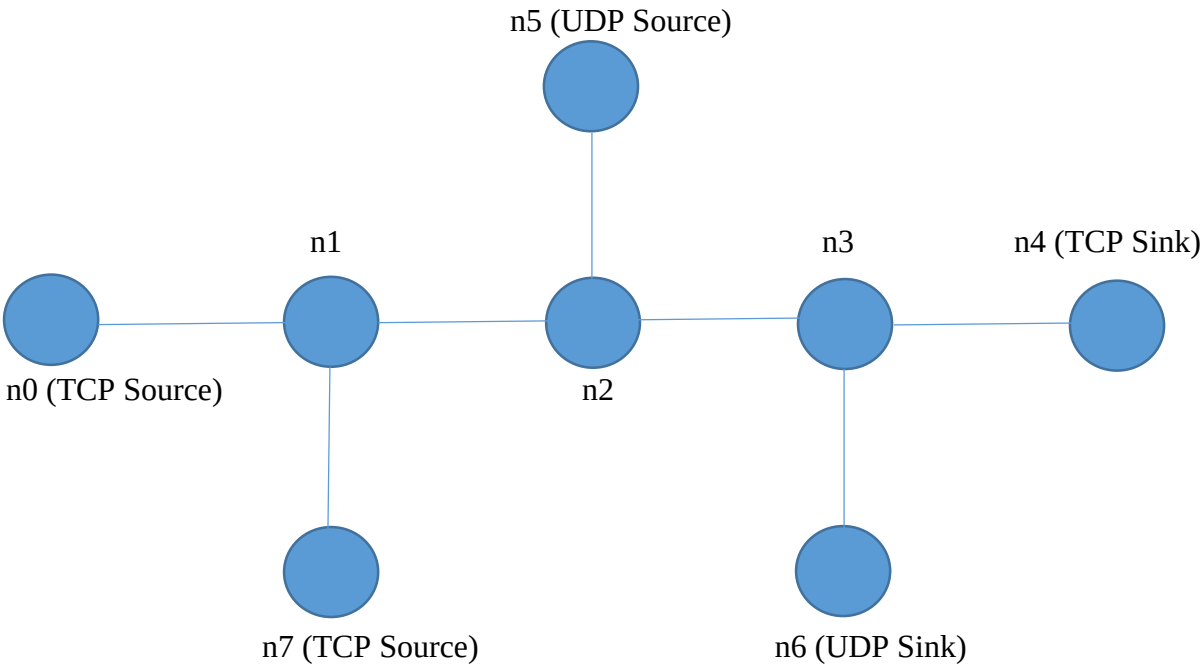
Name: Om Kale GTID: 903041445
Course: ECE-6110

Some background:
RED gateways can be used for congestion avoidance in packet-switched networks. For this, the gateways first compute the average queue size. If average queue size exceeds a certain preset threshold then the gateway drops or marks each arriving packet. RED gateways keep the average queue size low, accommodates transient bursts in traffic (by a temporary increase in the queue) and avoids global synchronization of many connections decreasing their window at the same time. In my experiments, these gateways are designed to accompany transport-layer congestion control protocol by TCP.
The RED congestion control mechanisms monitor the average queue size for each output queue and, using randomization, choose connections to notify of that congestion. They can mark a packet by dropping it at the gateway or by setting a bit in the packet header. When average queue size exceeds a max threshold, every arriving packet is marked at the RED gateway.

The RED gateways uses a weighted average of the total queue length to determine when to drop packets. When a packet arrives at the queue, if the weighted average queue length is less than a minimum threshold value, 'min_{th}', no drop action will be taken and the packet will simply be enqueued. If the average is greater than min_{th} but less than a maximum threshold, max_{th}, an early drop test will be performed as described below. An average queue length in the range between the thresholds indicates some congestion has begun and flows should be notified via packet drops. If the average is greater than the maximum threshold value, a forced drop operation will occur. An average queue length in this range indicates persistent congestion and packets must be dropped to avoid a persistently full queue. (The forced drop is also used when the queue is full but the average queue length is still below the maximum threshold.)
(Note: By using a weighted average, RED avoids overreaction to bursts and instead reacts to longer-term trends. Furthermore, because the thresholds are compared to the weighted average (with a typical weighting factor, *wq*, of 1/512), it is possible that no forced drops will take place even when the instantaneous queue length is quite large.)

The RED control parameters that need to be carefully adjusted in these experiments include:
qlen: The maximum number of packets that can be in the queue
min_{th}: Queue length threshold for triggering probabilistic drops.
max_{th}: Queue length threshold for triggering forced drops. After this threshold all incoming packets are marked for dropping.
w_q: Weighting factor for the average queue length computation.
max_p: The maximum probability of performing an early drop.

My topology is as below:



In the above topology, n0 and n7 are TCP sources. n5 is the UDP Source and n6 is the UDP Sink, followed by n4 which is the TCP Sink. There are two bottleneck links in the above topology. n1-n2 and n2-n3. The TCP Sources at n0 and n7 experience bottlenecks at n1-n2 and n2-n3. The UDP source at n5 experiences bottleneck at n2-n3. The bandwidths between n0-n1 and n3-n4 are 5Mbps and the delay is 10ms. Between n1-n2 and n2-n3, the bandwidth and delays are 1Mbps and 20ms respectively. However I have later changed these values to evaluate performance.

Drop Tail Queue:

For the Drop-tail queue, I varied queue size and window size as follows with RTT=10ms.

I selected following values of (queuesize, windowsize)

(Qsize,WinSize)	TCP Goodput	UDP Goodput
32000,32000	30421.6	54912
64000,64000	34620.8	55436.8

As queuesize and window size both increase more packets can be held in the queue(because of large queue size and less packets are dropped. Also due to large advertised window the packets successfully transmitted from source to sink increases. Hence overall goodput increases.

Varying data rate: As data rate increases TCP goodput is decresing

Data Rate	Goodput TCP	Goodput UDP
0.1Mbps	47002.4	11187.2
0.33Mbps	40731.2	36441.6
0.5Mbps	29421.6	55436.8

Varying RTT: Data rate=0.5 Mbps, qsize,winsize=(64000,64000)

As RTT increases, delay is introduced and hence overall goodput decreases.

Round Trip Time(RTT)	Goodput TCP	Goodput UDP
10ms	29421.6	55436.8
25ms	25187.2	55416
40ms	25205.6	54732.8

RED Queue:

An exhaustive search for the best parameters is difficult. I have used an approach in which I first try to find the best value for (minth, maxth) for my topology. I use 4 combinations as specified in the tuning red paper for this. Then I use this best combination for further experimentation. Additionally it is important to note that data rate can only be varied for UDP sources and we cannot control the TCP sources. This data rate variation can be thought of as varying traffic load.

Varying $min_{th,}$ max_{th} keeping all the other parameters constant. (The paper suggests selecting maxth to be at least twice minth, I select maxth to be thrice of minth)

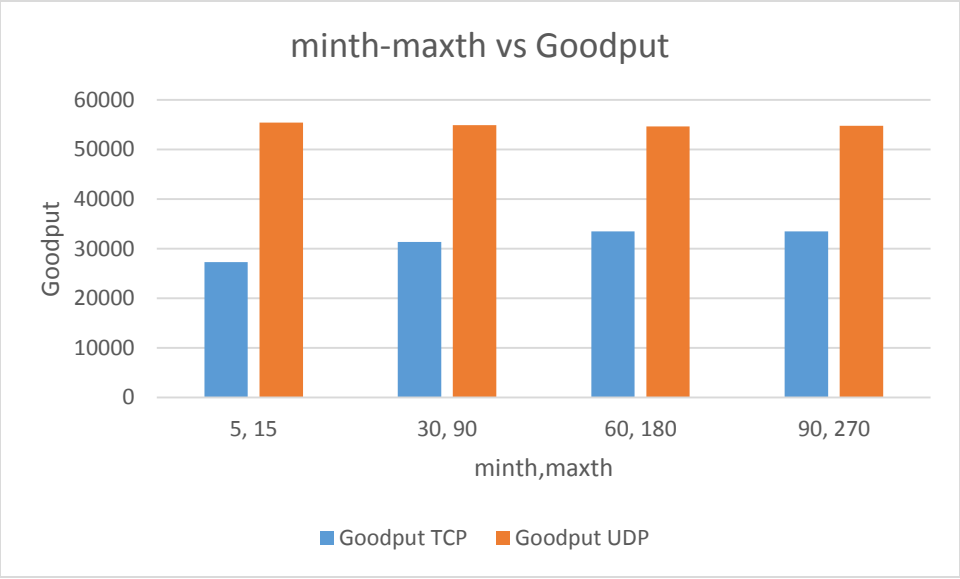
For RED:

RTT= 10ms, qlen=32000, maxP=50.

minth,maxth	Goodput TCP	Goodput UDP
5,15	27277.6	55424
30,90	31351.2	54860.8
60,180	32316	54604.8
90,270	33066.4	54118.4

RTT=10ms, qlen=64000, maxP=50, data rate=0.5Mbps

minth,maxth	Goodput TCP	Goodput UDP
5,15	27277.6	55424
30,90	31351.2	54860.8
60,180	33441.6	54643.2
90,270	33495.2	54732.8



The observation can be explained as below:

The variation of the Goodput is because initially, when (minth,maxth) is low (5,15), packets will be marked for dropping as soon as the average queue length reaches 5 packets. When the queue size reaches 15 packets, the limit has been reached and when the 16th packet arrives, it is dropped. Thus, it leads to an early drop of packets and hence the goodput is less initially. I observed that the values of (90,270) are the best. After this value, the increase is not that high. This is because if (minth, maxth) are increased to very high values, it is possible that the queue will start marking packets for dropping very late and this will decrease the goodput.

Comparison:

From the graphs above, I observe that minth, maxth for (90,270) gives the best goodput. Now comparing this, with Droptail queue with qsize,winsize of (64000,64000), with the same parameter settings as RED i.e. same qlen, datarate and RTT, I find that RED performs almost as well as Drop-Tail. Now using RED parameters and varying them as shown below, I find that RED can be made better than DropTail.

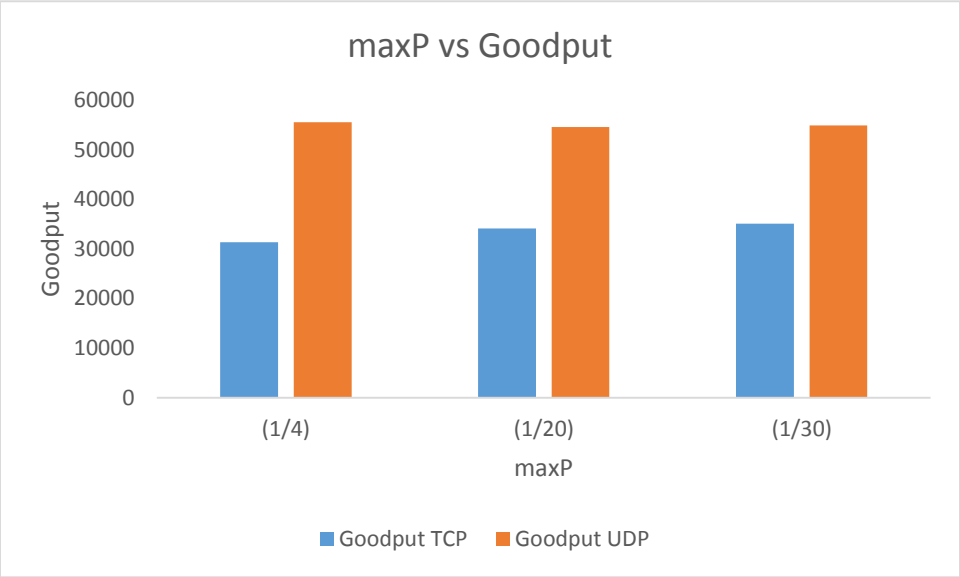
Maxp (Linterm in program)

For (minth,maxth) = (90,270) varying MaxP as 1/4, 1/20 and 1/30, RTT=10ms, qlen=64000bytes, data rate=0.5Mbps

maxP	Goodput TCP	Goodput UDP
1/4	31351.2	55526.4
1/20	34138.4	54566.4
1/30	35103.2	54886.4

RED is thus better.

The graph below shows the variation:



When maxP varies (with increasing value of the denominator), it means the flows tend to converge at some point. The once which have low goodput increase and once with high goodput decrease. The flows with high bandwidth decrease when maxP is varied from 1/4 to 1/30. Hence value of TCP goodput that we see at 1/30 is greater than that at 1/4. UDP Goodput observed at 1/4 reduces at 1/30. Thus proving that these flows will converge eventually. Additionally, I observed that the two flows converge more at a maxP value of 1/80.

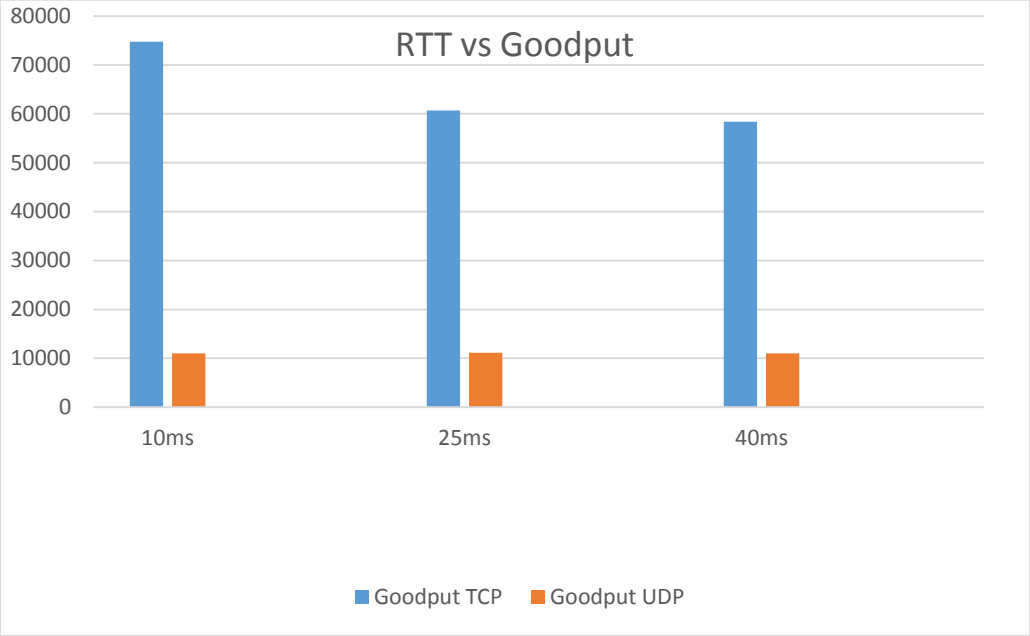
Now varying data rate with constant RTT=10ms, maxP=30 as observed from above.

Data Rate	Goodput TCP	Goodput UDP
0.1Mbps	74767.2	10982.4
0.33Mbps	46841.6	36300.8
0.5Mbps	35103.2	54886.4

This can be explained easily. As Data rate increases (Note that this is the data rate of UDP Source that we are controlling), the UDP Goodput goes on increasing and TCP Goodput Decreases. TCP goodput decreases because seeing the burst in traffic, it does congestion control and thus reduces its data rate.

Now varying RTT, keeping maxP=30 and data rate=0.1Mbps, qlen=64000

Round Trip Time(RTT)	Goodput TCP	Goodput UDP
10ms	74767.2	10982.4
25ms	60670.4	11161.6
40ms	58419.2	11020.8



Explanation of the above graph:

As RTT increases the overall goodput is decreasing. The reason for this is the larger RTT introduces delay in the network. It also increases transit time through the network. Also we see that this variation in RTT affects TCP goodput. This is because TCP is a connection oriented protocol and is used for congestion avoidance. It has to receive acknowledgements and the increased RTT delays them, thus reducing goodput. UDP just keeps on sending and doesn't care about the ACK's and hence its goodput is not affected. RTT only changes delay between source and sink.

Conclusion:

From all the above simulations and results I found that the RED Queue does perform better in certain scenarios especially if there is an exhaustive testing for the best RED parameters. Also, I observed that sometimes, droptail queue performs better than RED. However that is when we only compare RED(only qlen, data rate, RTT) to Droptail (Queue size, winsize, RTT and data rate). Further, the RED parameters (maxP, minth-maxth) can be changed and made to give better throughput as proved in the simulation results.

Thus, in conclusion I did not find any major improvement by using RED as it may give better performance only after careful variation of parameters which involves lot of complexity to be incorporated in the network. Droptail Queues, on the other hand are very easy to implement and can give comparable performance.