Peter Vieira
2014/02/24

ECE 6110 – CAD For Communication Networks
Lab 2 – Comparison of Random Early Detection (RED) vs. Drop Tail Queuing

I. Introduction

A simple dumbbell style topology was constructed containing two bottlenecks using either Random Early Detection (RED) or Drop Tail style queues. The topology contains three source On/Off applications, one using UDP and the other two using TCP protocol. Figure 1 shows the network topology with the bandwidth and delay properties of each channel link. The sources all start sending data on a 15 Mbps / 5 ms delay channel, which then hits the first of two bottleneck links. The third router link is also a bottleneck with a bandwidth of 1 Mbps and a delay of 10 ms. Finally, three sinks, each dedicated to a separate source application receive all the data on a channel with 10 Mbps bandwidth and 1ms delay. The simulation was run for 10 seconds.

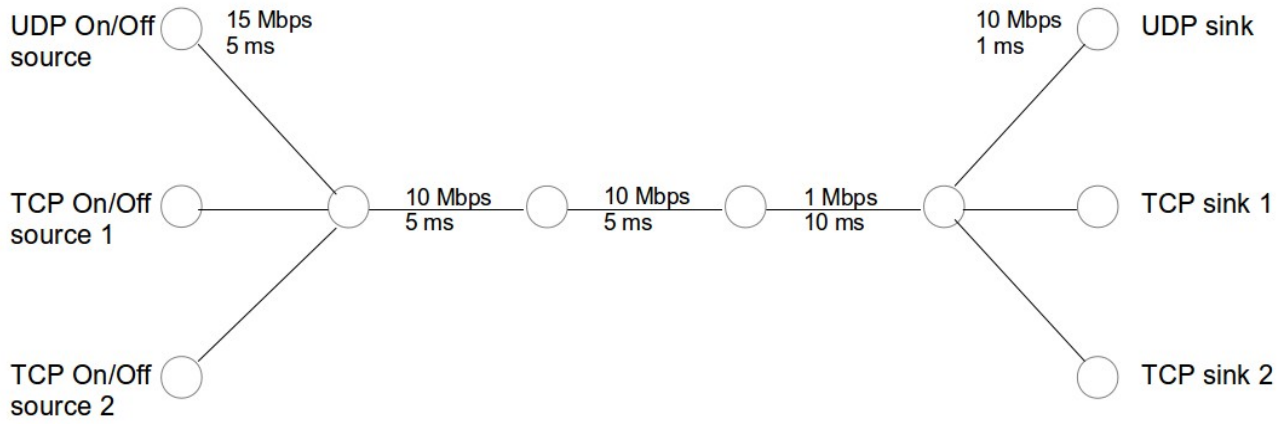

Fig. 1: Network Topolgy

RED queuing involves some more sophisticated techniques that attempt to overcome some of the flaws of Drop Tail queuing. RED queues start dropping packets when the queue gets sufficiently filled before congestion occurs with the assumption that congestion is going to occur soon. This attempts to reduce over packet loss by dropping packets probabilistically before queues overflow. This probabilistic dropping of packets helps prevent uneven flows from occurring. On the most basic level, RED queues start dropping packets probabilistically when the average queue size is greater than the minimum threshold (minTh) and less than the maximum threshold (maxTh). When the average queue size becomes greater than maxTh, forced packet drops begin occurring. This signifies persistent congestion. RED queuing also using a weight average to avoid overreaction to bursts and therefore reacts to long-term trends instead.

Drop Tail queuing involves a First In First Out (FIFO) queue, where lockout can occur where a small number of flows can take over the queue during congestion. This happens because slower flows reach queues when they are full and packets get dropped as compared to faster flows. Latency is increased for all flows when the queue is constantly full. Also, shorter queues decrease latency, but don't

accommodate brief bursts of data.

In this experiment, five RED parameters were varied: minimum threshold (minTh), maximum threshold (maxTh), maximum probability (maxP), weight (Wq) and queue length (qLen). For Drop Tail, the queue size was varied.

II. RED vs. Drop Tail Queuing Setup and Assumptions

Using Network Simulator 3 this topology was constructed and simulated. First, the network was simulated using Drop Tail queues at the two bottlenecks as baseline for comparison against the RED queue. During the simulation the throughput was measured as a function of the queue length of the Drop Tail queue. Four different network loads were simulated using the On/Off applications and setting there rates such that the simulated load would be a percentage of the bandwidth of the smallest bottleneck link. TCP-Tahoe protocol was used for the TCP On/Off applications. The assumptions made for the simulations conducted were:

1. The receiver windows are always large enough.
2. The On/Off applications are on half the time.
3. The On/Off applications send data out at a constant rate in bits/second.
4. The load stays constant on the network.
5. 10 seconds of simulation is enough to obtain general representative data.

During simulation of the RED queuing for the two bottleneck links, the same settings were used for the On/Off applications as in the Drop Tail setup.

III. Results

Figure 2 shows the results for simulating the network with Drop Tail queues. For smaller network loads the TCP applications results in much higher throughput than the UDP application. This makes sense because the smaller load means less build-up of the queues and less dropped packets. Generally, the TCP throughputs increased as queue length increased, with the exception of the 90% and 100% load tests. This is possibly due to the fact that the receiver window was set to 64000 bytes and with the larger loads, the receiver buffer was filled most of the time when the queue length was 64000 bytes. The throughput of the UDP application was constant within each load setting. This is probably because of UDP's connectionless nature and lack of acknowledgements, which cause it to just send data regardless of lost packets.

Figure 3 shows the results of simulating the network with RED queues. The throughput of the UDP application stayed constant with varying parameters and constant load. This seems to be a bug in the code that has not been found yet. Figure 4 shows how the goodput of the applications faired for different loads. As the load increased so did the throughput of the UDP application.
The trend for the TCP applications was a decrease with increased load. This signifies that smaller loads resulted in less filling of queues. Oddly, the UDP application completely outperformed the TCP applications by an order of magnitude.

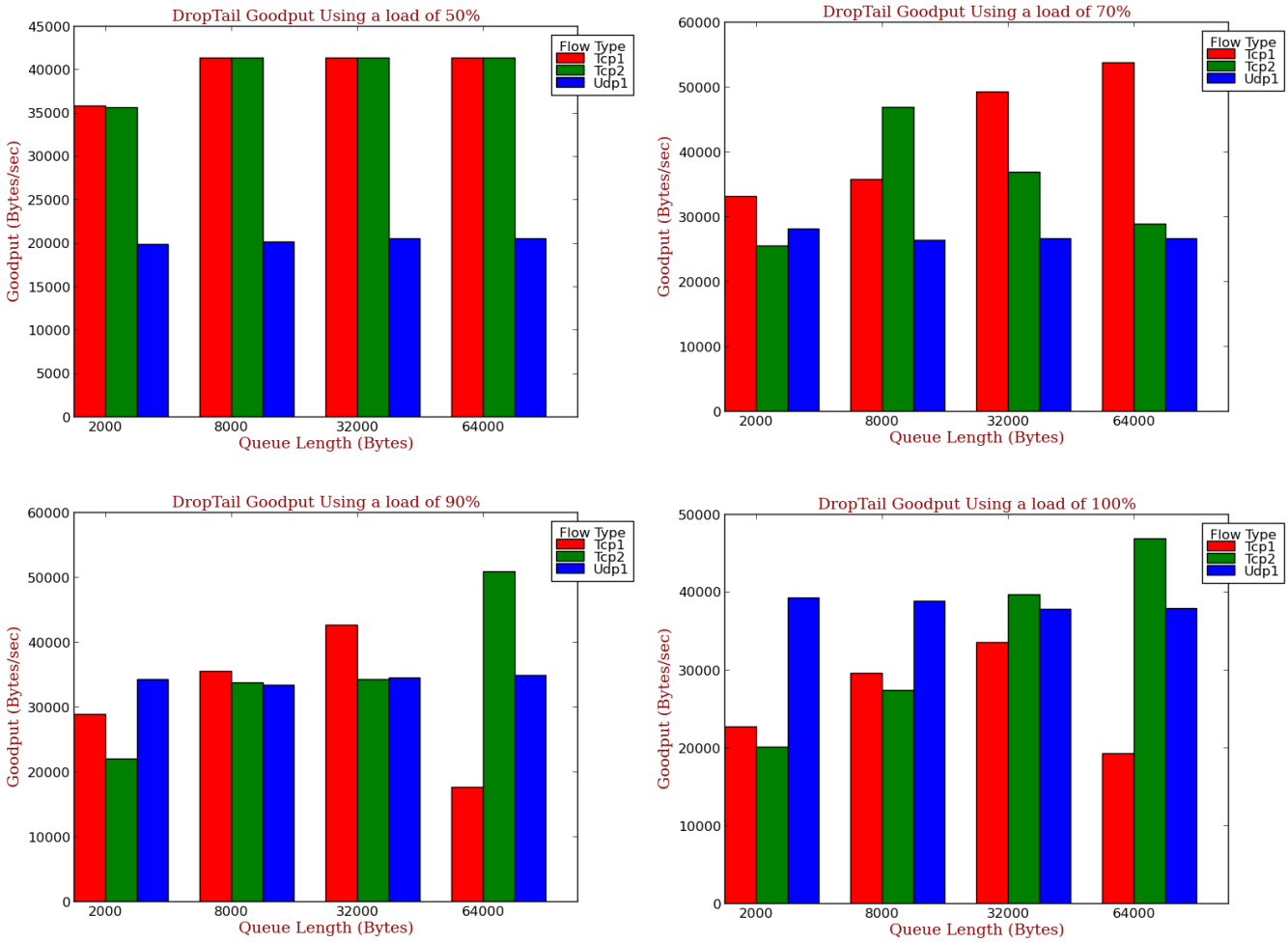

Fig. 2: Goodputs Using Drop Tail Queue with Varying Loads

Comparing the results from RED queuing versus Drop Tail queuing the results for the UDP application throughput were staggeringly similar. The throughput of the TCP applications for RED queuing were an order of magnitude smaller than for Drop Tail queuing. This is very surprising because RED queuing was designed to handle non-ideal situations better than Drop Tail. Either reality does not reflect the algorithm designs or the simulation is flawed in some way.

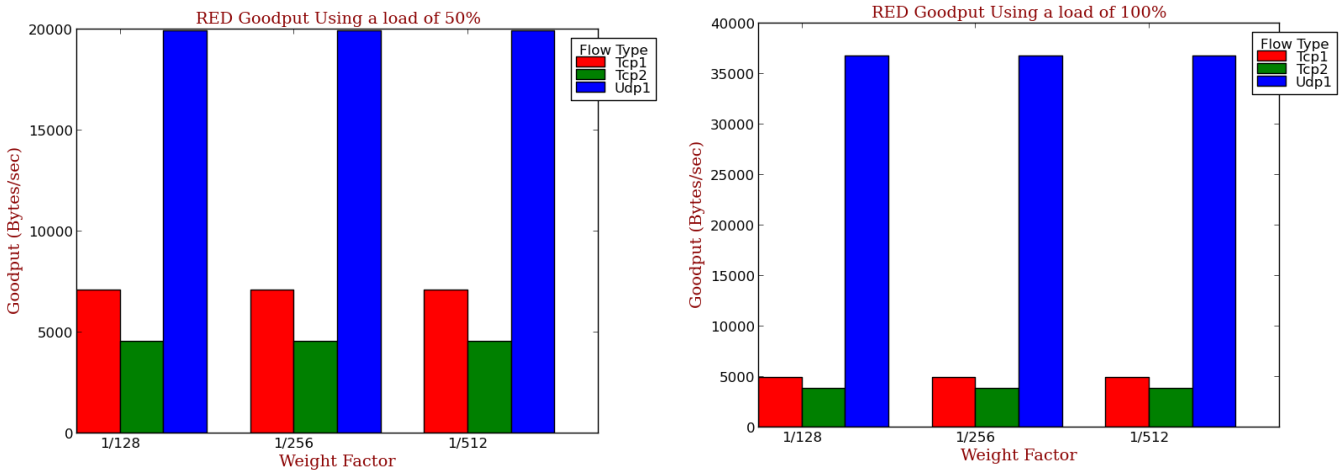

Fig. 3: Goodputs Using RED Queue with Varying Weighting Factor

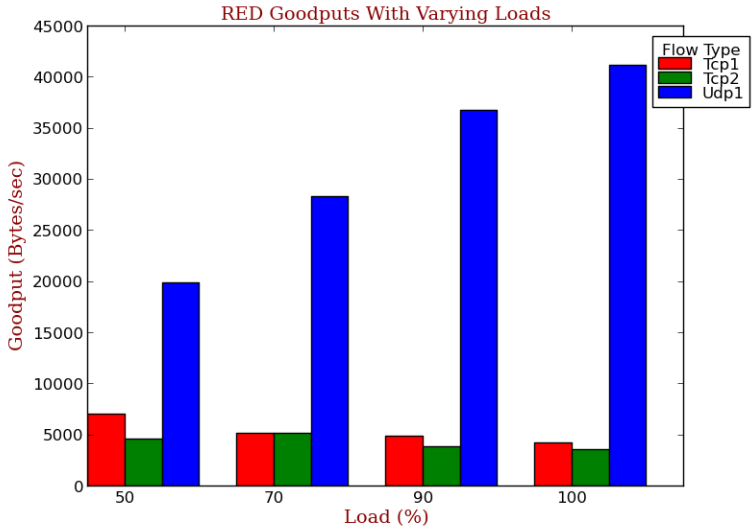

Fig. 4: RED Queues with Varying Loads. minTh=5, maxTh=90, maxP=1/20, Wq=1/128, qlen=480

IV. Conclusion

Overall, the performance of the simulations were very surprising and unexpected in many ways. Primarily, the TCP applications in the RED simulations performed much worse than those in the Drop Tail simulations. This is likely due to errors in the simulation, as discrepancies between Drop Tail and RED queuing should never be this large. The UDP application, at times, beat out the TCP applications. This is most likely due to the fact that the load was split between the three, but also because UDP protocol lacks acknowledgements and connections and therefore can gain throughput where TCP has more overhead.