COL334
Assignment 5

REVIEW OF
"CONTROLLING QUEUE DELAY" BY KATHLEEN NICHOLAS AND VAN JACOBSON

1 KEY POINTS HIGHLIGHTED

The key highlight of the paper is the comparison of different mechanisms to deal with the ”persistently full buffer” problem, a part of the bufferbloat[1] problem; causes and consequences of the same are discussed. This is followed by a discussion on how queuing[2] occurs in packet-switched networks. Then there is detailed study of the ”no-knobs” AQM (Active Queue Management) called CoDel (Controlled Delay) and the other AQM solutions for dealing with large-sized buffers and dynamically changing network characteristics. The different models are compared and pros and cons of usage of each highlighted. In the end, this establishes that overall, CoDel fairs better than the other different AQM solutions for resolving queuing delays in packet-switched networks in modern internet.

2 KEY STRENGTHS AND RELEVANCE OF THE RESEARCH

This research is very important to address the problems of queuing and buffering in modern networks which are very huge and complicated, with changing link rates, bandwidths etc. So a system like CoDel is the need of the hour to adapt to different network environments without any explicit configuration. The experimental findings in this paper strongly establish the best performance delivered by CoDel in terms of utilization, per-packet delay and rapid adaptation to dynamic changes in configuration. Also, it requires least parameters to be set for functioning.

Desirable Queue Management Scheme for Modern Internet :

1. Least parameters
2. Should treat good and bad queue differently
3. Controls delay for varying RTT’s, link rates etc.
4. Adapts to dynamically changing link rates, keeping utilization same
5. Simple, efficient and deployable for all types of routers

Unique Innovations in CoDel :

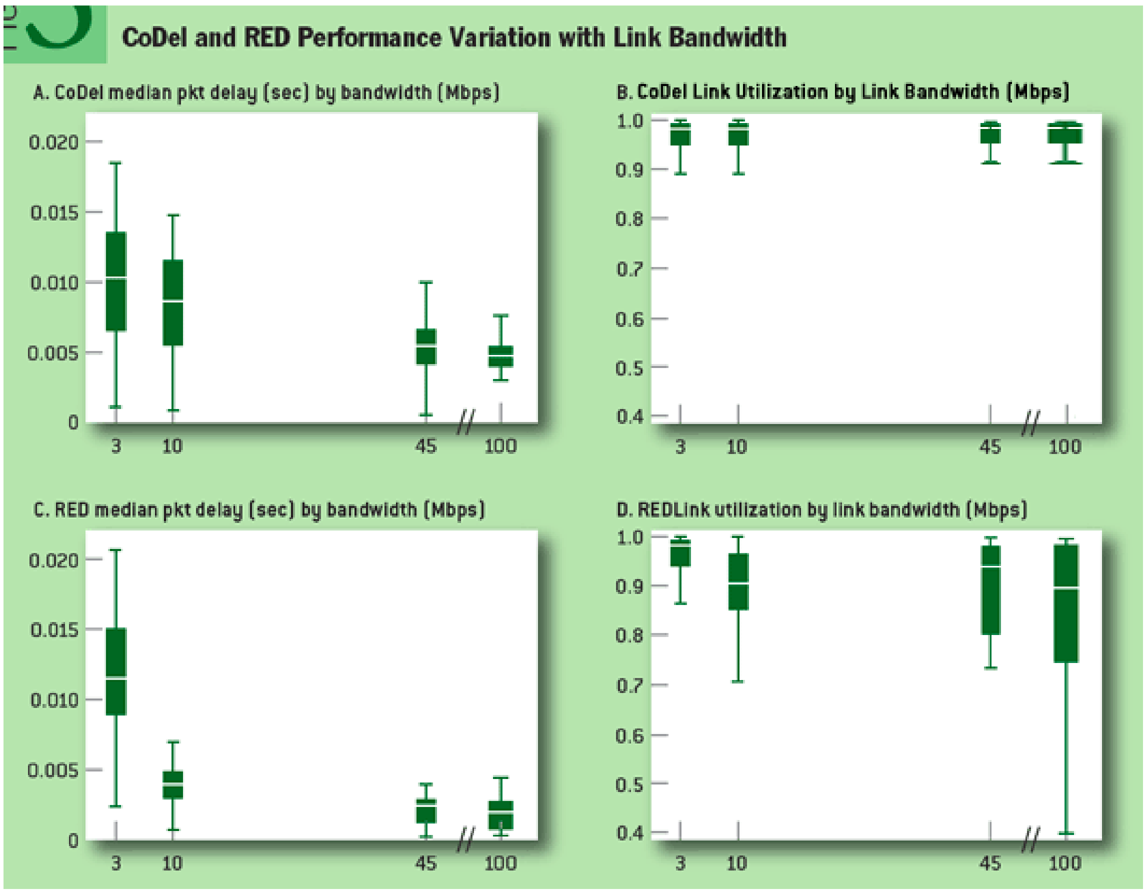
In the algorithm, “local minimum queue” is used as a measure of standing queue and used to calculate persistent delay; queue-size averages, thresholds, occupancy, link utilization, drop rate etc don’t matter.

Also, queue size is measured not in units of bytes or packets but as packet-sojourn time because packet delay is a better measure of performance at the user end
During implementation, no locks are needed and scheduling takes place only when packets are dequeued from the buffer. Packets are not dropped when there is less than one MTU (Maximum Transmission Unit).

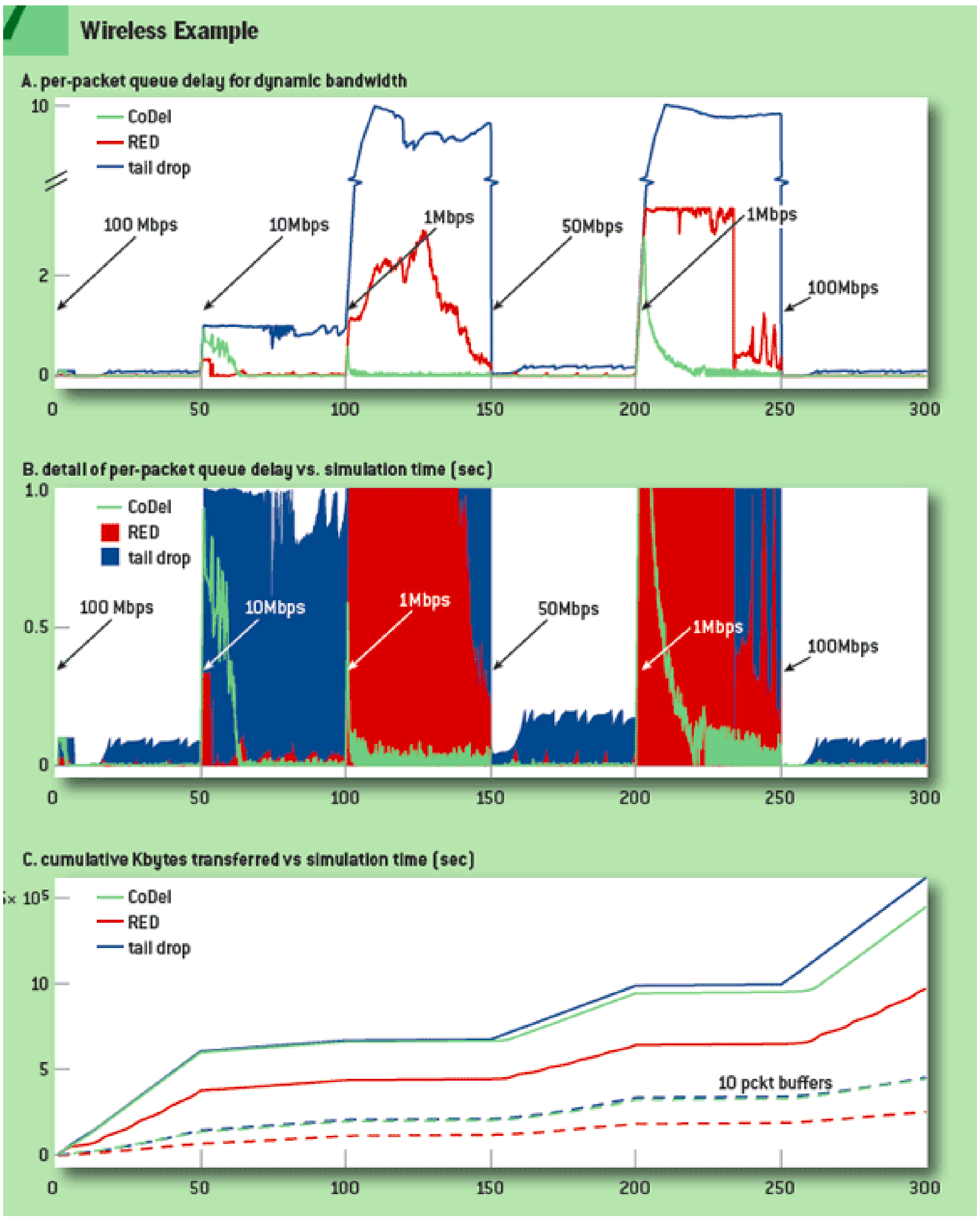
Overall Algorithm :
When queue delay exceeds least-interval target, packet is dropped. Since the dropping state is entered, next drop time is decreased in inverse proportion to square-root of number of drops (Equation : link)
Dropping is stopped when queue delay goes below target value.

Experimental Results :
Several simulation experiments were performed (using ns-2 simulator) on CoDel and results were compared with other AQM's.

1. Performance against a range of **static link rates** was observed for CoDel and RED using median packet delay and link utilization values vs the bandwidth. The delay was less in CoDel for higher bandwidths but at the smaller values, additional packets cause significant delay compared to the target. But this behaviour is desirable in modern networks. Lower utilization values for RED indicate its over-controlling nature.



2. For testing performance on **dynamic links**, link rates were changed at 50 ms intervals increasing and dropping alternately. Per-packet-queue-delay and cumulative Kbytes transferred vs the simulation time was noted for RED, TailDrop and CoDel. TailDrop keeps the buffer full and delay induced is more compared to RED which shows a poorer response to changing link rates compared to CoDel (it computes a new control point within 100 ms). CoDel transfers almost same Kbytes as TailDrop with identical throughput.



3. CoDel has better **drop-share fairness** than RED, hence more effective at dropping the right packets.

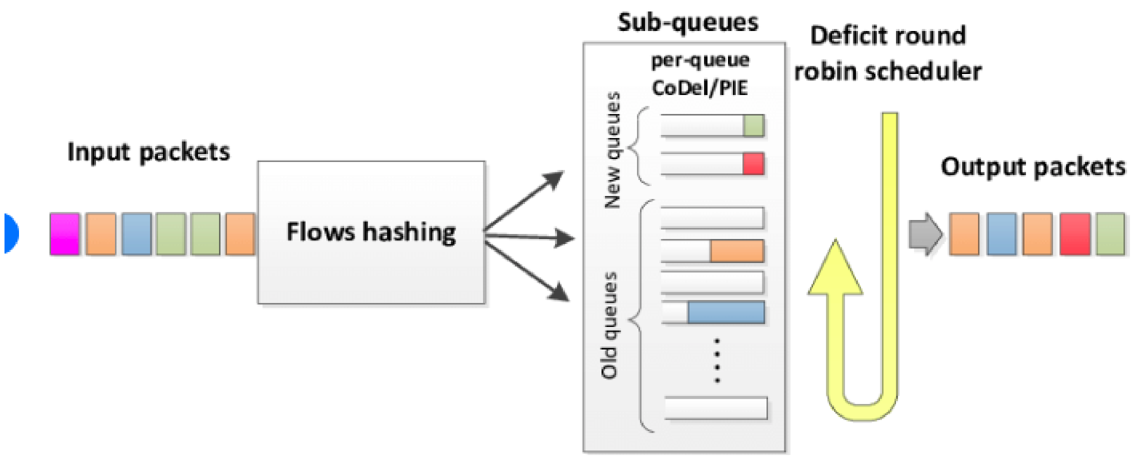
Also, CoDel's algorithm can be efficiently implemented in **Silicon** and it is the most efficient implementation in Linux-based routers.

3 POTENTIAL IMPROVEMENTS

While the paper lists the forthcomings of CoDel compared to other AQM's, it does not significantly highlight the shortcomings of the same except for deployment issues. CoDel can be implemented through CeroWrt-enabled edge router but the challenge lies in DiffServ queuing, the suggested approaches require ethernet configuration.
Also, the experimentation has been limited to ns-2 simulation with Linux TCP suite but since the code has been made available (link), the results after running it on a real LAN setup could have given a better insight.

4 APPLICATION AND SCOPE IN CURRENT INTERNET PROBLEMS

AQM is only a part of solution to bufferbloat. This paper was published in 2012 when research was centered around AQM's and CoDel is also an improvisation of the same. However recently the SQM (Smart Queue Management) schemes have been developed for routers and FQ-CoDel is widely used. But FQ-CoDel is a brainchild of this research work only. So the simulations and metrics recorded in this paper have further scope of use and research in addressing current internet problems.



Simplified FQ-CoDel/FQ-PIE AQMs

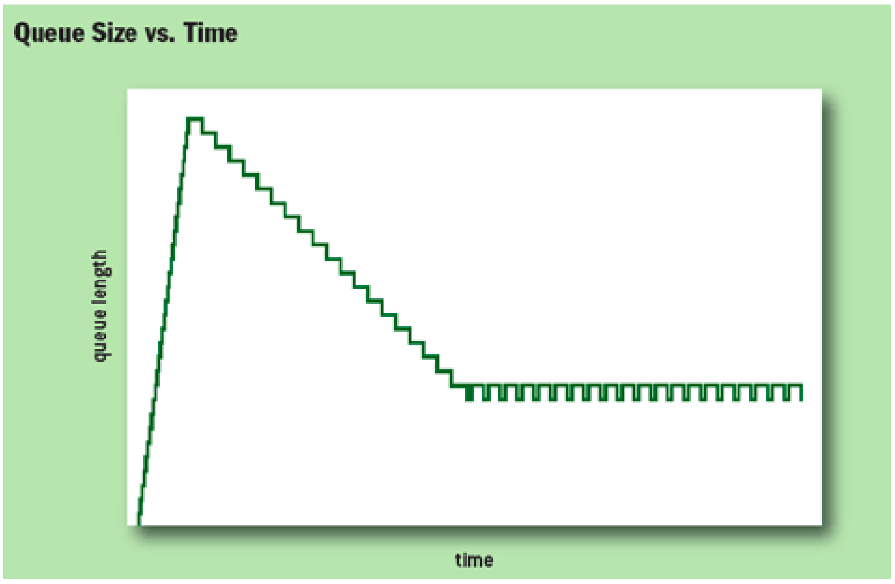
5 ADDITIONAL EXPLANATIONS

1. About BUFFERBLOAT :
Bufferbloat is cause of high latency in packet-switched networks due to excess buffering. The

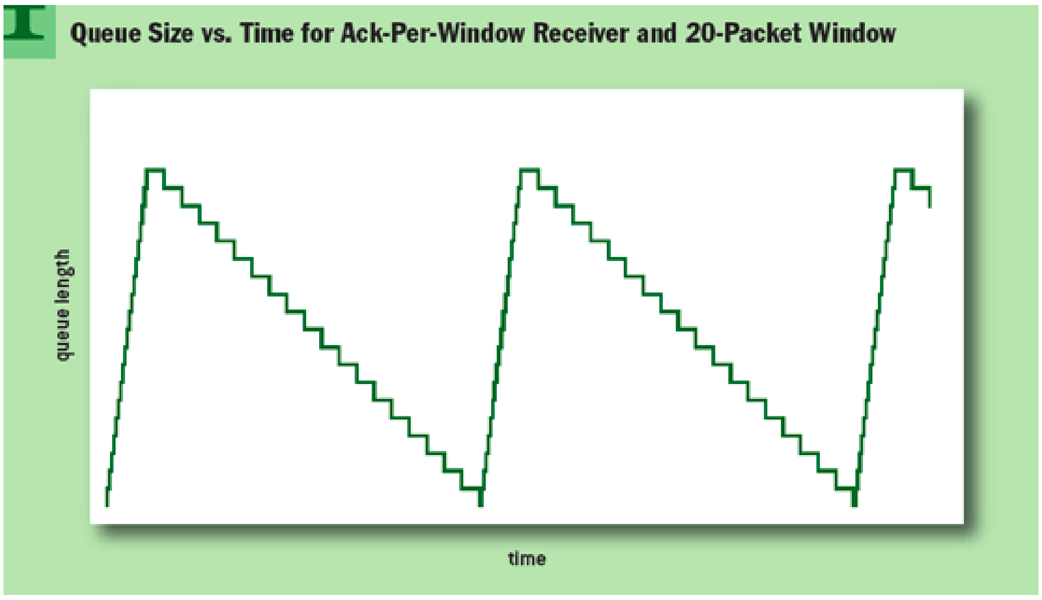
two major problems emerging in present day are proliferation of buffers and difficulty in adjusting the buffer to dynamic changes in network characteristics. The first proposal was usage of RED (Random Early Detection) AQM. It was simple and effective at reducing persistent buffers but the configuration was a major problem. The acute challenge with RED AQM was correct buffer sizing. While under-sizing caused problems listed in Guillaume Vu-Brugier et al, over-sizing caused an increase in delay and reduced link rates. Also, links vary in bandwidths and connections vary in RTT, so RED AQM has to be separately configured for different environments.

2. About QUEUES :

They are formed when there's mismatch in packet arrival and departure rates which in turn is a result of upstream resource contention, transport conversation startup transients or changes in the number of conversations sharing a link. Packet-buffers are formed due to bottle-neck links which induce waiting. Then as a result of continuous message exchanges between the receiver and the sender (including ack messages for each packet), the queue goes into a steady state after a point of time when all packets come at spaced-out intervals of time.



This standing queue, resulting from a mismatch between the window and pipe size, is the essence of bufferbloat causing large (excess) delays but not throughput improvements. So the way to deal with the standing queue is to somehow make the window size match the pipe size. But this is impossible to accomplish since window size is chosen by the sender and queues are formed at bottleneck gateways. Also, bandwidth at the bottleneck and RTT constantly change as a result of re-routing, physical environments etc.



In a variation if acks are sent for each window of packets instead of for each packet, then the queue size fluctuates uniformly between 0 and the max number of packets in the window. In this scenario, there is no excess queue, any attempt to tailor this queue will compromise on full-time occupancy of the bottleneck link.

Submitted by :
Manupriya Gupta
2018CS10355
(All images are taken from the original publication except for FQ-CoDel diagram taken from here)