REDUCING INTERNET TRANSPORT LATENCY

The New AQM Kids on the Block: An Experimental Evaluation of CoDel and PIE

Naeem Khademi <naeemk@ifi.uio.no>
David Ros <dros@simula.no>
Michael Welzl <michawe@ifi.uio.no>

17th IEEE Global Internet Symposium – 2014
Toronto, ON, Canada

Outline

AQM

AQM mechanisms considered

- CoDel
- PIE
- ARED

Experimental Setup

A Basic Test

Parameter Sensitivity

Conclusions and Future Work

Q&A

Active Queue Management (AQM)

- **Problem:** Standard loss-based TCP's congestion control plus Large unmanaged buffers in Internet routers, switches, device drivers,... (a.k.a Bufferbloat)
- **Cause:** Latency issues for interactive/multimedia applications
- **Solution:** AQM tries to signal the onset of congestion by (randomly?) dropping/marking packets

AQM Goals

- Maintain low average queue/latency
- Allow occasional packet bursts
- Break synchronization among TCP flows

The New AQM Kids on the Block...

- Two very recent proposals:
  - (FQ_)CoDel (IETF 84)
  - PIE (IETF 85) mandatory in DOCSIS 3.1 CM
- Some older AQMs dating back to early 90's/00's (*RED, REM, BLUE, CHOKe,...)
  - Designed to be better than RED, just like CoDel and PIE
- Little academic literature available on CoDel and PIE

Literature *(bold = peer-reviewed)*

	CoDel	PIE	FQ_CoDel
Wired, sim	**[NJ12][GRT+13]**[WP12] [Whi13]	[Whi13]	[Whi13]
Wired, real-life	**[GRT+13]** ✓	✓	✓
Wireless (any)	✓	✓	-

NOTE: [WP12] and [Whi13] are on DOCSIS 3.0 while [GRT+13] has tests with Low-Priority congestion control.

The New AQM Kids on the Block (cont.)

AQM Deployment Status

- (W)RED is available on plenty of HW but mostly "turned off"

Mentioned Reasons for Lack of Deployment

- Bad implementation (?)
- Hard to tune RED params

- Sally Floyd's ARED (2001 technical report, available in Linux) adaptively tunes RED params aiming for a certain target queuing => with fixed BW maps to a "target delay"
- Target delay can be set in ARED, CoDel and PIE

The New AQM Kids on the Block (cont.)

CoDel – Controlling Delay

- Tries to detect the *standing queue* by measuring minimum *sojourn delay* ($delay_{min}$) over a fixed-duration *interval* (default 100 ms)
- Uses timestamping
- If $delay_{min} > target$ for at least one *interval*, enters *dropping mode* and a packet is dropped from the tail (deque)
- **Next dropping time:** Dropping interval decreases in inverse proportion to the square root of the number of drops since the dropping mode was entered
- Exits *dropping mode* if $delay_{min} \leq target$
- No drop when queue is less than 1 MTU

The New AQM Kids on the Block (cont.)

CoDel Assumptions

- 100 ms is nominal RTT assumed typical on the Internet paths
- *interval* $=$ 100 *ms*; assures protection of normal packet bursts
- A small *target* standing queue (5% of nominal RTT) is tolerable for achieving better link utilization

The New AQM Kids on the Block (cont.)

PIE – Proportional Integral controller Enhanced

- *Lightweight* as it uses delay estimation instead of timestamping
- Uses a *Proportional Integral (PI)* controller design
- Uses trend of latency (increasing or decreasing) over time to determine the congestion level
- $E[T]$ as current estimated queuing delay during every t_{update}, N as current queue length and μ is the draining rate

$$E[T] = N/\mu$$

- Randomly drops on enque based on probability p

$$p = p + \alpha * (E[T] - T_{target}) + \beta * (E[T] - E[T]_{old})$$

The New AQM Kids on the Block (cont.)

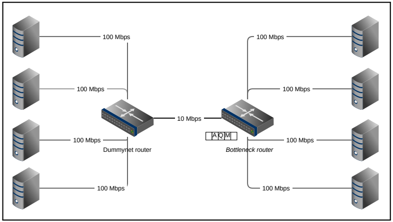
ARED – Adaptive RED

- Tries to solve RED's main problem of parameter tuning to keep the average queue length ($\bar{N}$) around a desired *target_queuing*
- $target\_queuing = (th\_max + th\_min)/2$
- Observes $\bar{N}$ to make RED more/less aggressive
- Updates RED's p_{max} adaptively (every 500 ms by default) using an AIMD policy
- Only useful in fixed-BW scenarios ($target\_delay = target\_queuing/BW$)

Experimental Setup

- **Traffic:** 60 sec (or 300 sec if RTT=500 ms) of TCP traffic by *iperf*, repeated for 10 runs
- **AQM iface:** GSO TSO off, BQL=1514, txqueuelen=1000
- **TCP:** Linux default with *reno*
- **Topology:** Dumbbell with 4 sender-receiver pairs

Model	Dell OptiPlex GX620
CPU	Intel(R) Pentium(R) 4 CPU 3.00 GHz
RAM	1 GB PC2-4200 (533 MHz)
Ethernet	Broadcom NetXtreme BCM5751 RTL-8139 (*AQM interface*) RTL8111/8168B (*Dummynet* router)
Ethernet driver	tg3 8139too (*AQM* interface) r8168 (*Dummynet* router)
OS kernel	Linux 3.8.2 (FC14) Linux 3.10.4 (*AQM* router) (FC16)



Experimental Setup (cont.)

- AQM parameters used *unless* otherwise noted.

CoDel

interval=100 ms
target=5 ms

PIE

parameters in [pie].

PIE Parameter	Default value
t_{update}	30 ms
T_{target}	20 ms
α	0.125
β	1.25

ARED

parameters in [FGS01].

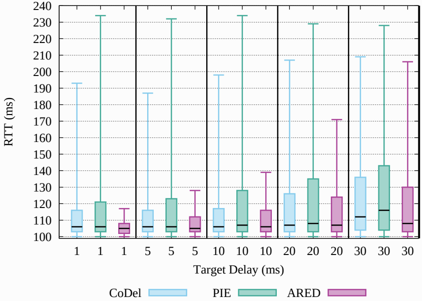
ARED Parameter	Default value
interval	500 ms
α	$min(0.01, p_{max}/4)$
β	0.9
th_min	$0.5 * target$
th_max	$1.5 * target$

Experimental Setup (cont.)

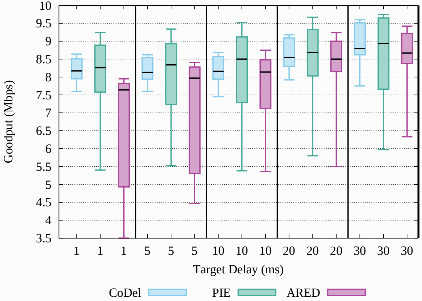
- RTT is measured on per-packet basis using *Synthetic Packet Pairs (SPP)* tool [spp]
  - Gives a very precise distribution of perceived RTT on the path
- Goodput is measured per 5-sec intervals
  - long-term throughput/goodput does not reflect AQM performance *over time* (e.g. bursts of packet drops are not desired)

A Basic Test

Single TCP Flow (RTT_{base}=100 ms)



(a) Per-packet RTT



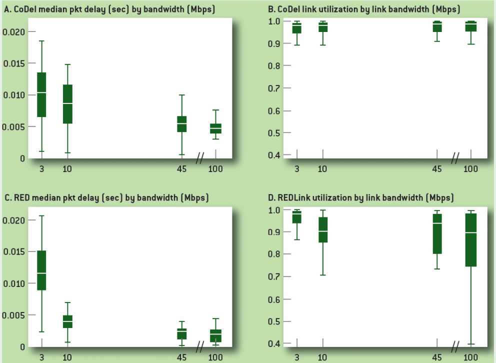
(b) Goodput

Per-packet RTT and goodput. Bottom and top of whisker-box plots show 10th and 90th percentiles respectively.

A Basic Test (cont.)

- A similar trend can be observed between CoDel and **RED** in a different test in [NJ12]

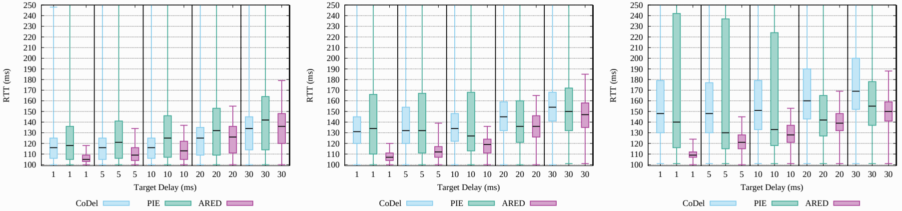
CoDel vs. RED from K. Nichols, “Controlling Queue Delay” [NJ12]



FTP traffic mix w/ and w/o web-browsing and CBR applications and RTTs from 10∼500 ms.

Parameter Sensitivity (cont.)

Target Delay



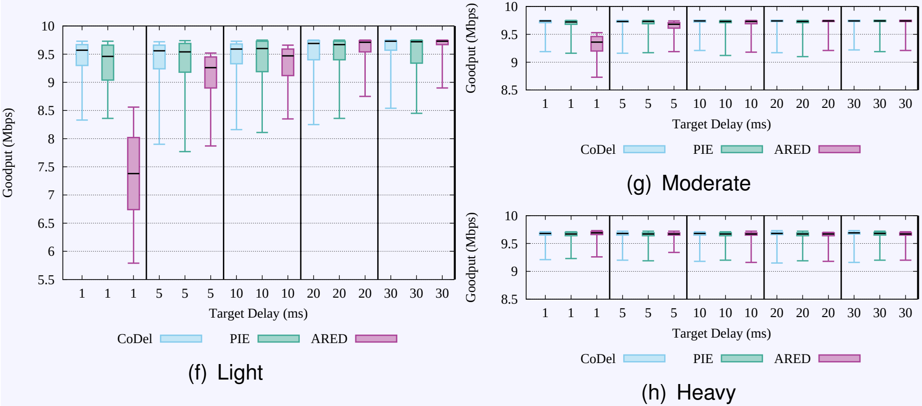
(c) Light (d) Moderate (e) Heavy

Per-packet RTT. Light, moderate and heavy congestion scenarios (4 senders and RTT_{base}=100 ms).

Light, *moderate* and *heavy* congestion correspond to 4, 16 and 64 concurrent TCP flows respectively.

Parameter Sensitivity (cont.)

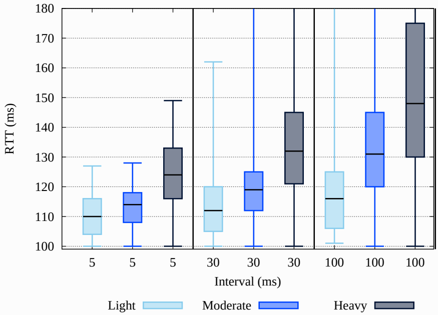
Target Delay



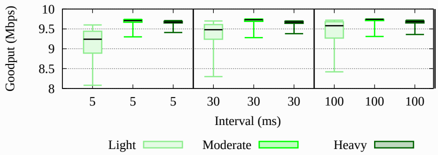
Goodput. Light, Moderate and Heavy congestion scenarios (4 senders and RTT_{base}=100 ms).

Parameter Sensitivity (cont.)

CoDel's Dropping Mode Interval – Target Delay=5 ms



(i) Per-packet RTT

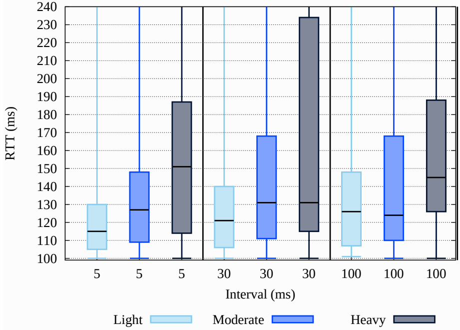


(j) Goodput

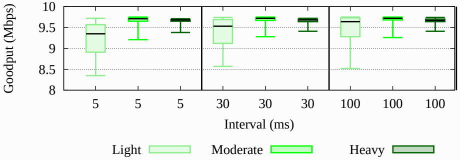
4 senders, RTT_{base}=100 ms.

Parameter Sensitivity (cont.)

PIE's *t_update* Interval – Target Delay=5 ms



(k) Per-packet RTT



(l) Goodput

4 senders, RTT_{base}=100 ms.

Conclusions

- **ARED:** *Only* performed worse than CoDel or PIE with small number of flows
- **CoDel:** Dropping mode interval can be reduced to lower the delay without degrading the goodput much
- **PIE (as implemented in Linux):** Long distribution tail for low target delays

Future Work

- More realistic traffic types (here, only bulk TCP traffic) including bursty traffic
- *Simulations* for environment parameters that cannot be produced with our testbed

Bibliography

[FGS01] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker.
Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management.
Technical report, 2001.

[GRT+13] Y. Gong, D. Rossi, C. Testa, S. Valenti, and D. Taht.
Fighting the Bufferbloat: On the Coexistence of AQM and Low Priority Congestion Control.
2013.

[NJ12] Kathleen Nichols and Van Jacobson.
Controlling Queue Delay.
Queue, 10(5):20:20–20:34, May 2012.

[pie] PIE Linux code (Cisco).
`ftp://ftpeng.cisco.com/pie/linux_code/`.

[spp] Synthetic Packet Pairs.
`http://caia.swin.edu.au/tools/spp/`.

[Whi13] Greg White.
A Simulation Study of CoDel, SFQ-CoDel and PIE in DOCSIS 3.0 Networks.
Technical Report, CableLabs, April 2013.

[WP12] Greg White and Joey Padden.
Preliminary Study of CoDel AQM in a DOCSIS Network.
Technical Report, CableLabs, November 2012.

Q&A

More on experimental results: N. Khademi, D. Ros, and M. Welzl, *"The New AQM Kids on the Block: Much Ado About Nothing?"*, Technical Report 434, Department of Informatics, University of Oslo, 23 October 2013, available at http://urn.nb.no/URN:NBN:no-38868