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

<table>
<thead>
<tr>
<th>Literature (bold = peer-reviewed)</th>
<th>CoDel</th>
<th>PIE</th>
<th>FQ_CoDel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wired, sim</td>
<td>[NJ12] [GRT+13] [WP12] [Whi13]</td>
<td>[Whi13]</td>
<td>[Whi13]</td>
</tr>
<tr>
<td>Wired, real-life</td>
<td>[GRT+13] ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wireless (any)</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
</tbody>
</table>

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

1. 100 ms is nominal RTT assumed typical on the Internet paths
2. $interval = 100 \text{ ms}$; assures protection of normal packet bursts
3. 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 $\mu$ is the draining rate

$$E[T] = \frac{N}{\mu}$$

- Randomly drops on enque based on probability $p$

$$p = p + \alpha \ast (E[T] - T_{target}) + \beta \ast (E[T] - E[T]_{old})$$
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_queueing
  
  - $target\_queueing = \frac{th\_max + th\_min}{2}$
  
- 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
  
  - $target\_delay = target\_queueing / BW$
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

<table>
<thead>
<tr>
<th>Model</th>
<th>Dell OptiPlex GX620</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel(R) Pentium(R) 4 CPU 3.00 GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>1 GB PC2-4200 (533 MHz)</td>
</tr>
<tr>
<td>Ethernet</td>
<td>Broadcom NetXtreme BCM5751\nRTL-8139 (AQM interface)\nRTL8111/8168B (Dummynet router)</td>
</tr>
<tr>
<td>Ethernet driver</td>
<td>tg3\n8139too (AQM interface)\nr8168 (Dummynet router)</td>
</tr>
<tr>
<td>OS kernel</td>
<td>Linux 3.8.2 (FC14)\nLinux 3.10.4 (AQM router) (FC16)</td>
</tr>
</tbody>
</table>
AQM parameters used *unless* otherwise noted.

### CoDel
- interval = 100 ms
- target = 5 ms

### PIE
- Parameters in [pie].

<table>
<thead>
<tr>
<th>PIE Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{update}}$</td>
<td>30 ms</td>
</tr>
<tr>
<td>$T_{\text{target}}$</td>
<td>20 ms</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.125</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1.25</td>
</tr>
</tbody>
</table>

### ARED
- Parameters in [FGS01].

<table>
<thead>
<tr>
<th>ARED Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>500 ms</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>min($0.01$, $p_{\text{max}}/4$)</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.9</td>
</tr>
<tr>
<td>$th_{\min}$</td>
<td>0.5 * target</td>
</tr>
<tr>
<td>$th_{\max}$</td>
<td>1.5 * target</td>
</tr>
</tbody>
</table>
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

Per-packet RTT and goodput. Bottom and top of whisker-box plots show 10th and 90th percentiles respectively.
A similar trend can be observed between CoDel and RED in a different test in [NJ12]

FTP traffic mix w/ and w/o web-browsing and CBR applications and RTTs from 10~500 ms.
Parameter Sensitivity (cont.)

Target Delay

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.

N. Khademi, D. Ros, M. Welzl

The New AQM Kids on the Block...
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

4 senders, $RTT_{\text{base}}=100$ ms.

(j) Goodput
Parameter Sensitivity (cont.)

PIE’s $t_{update}$ Interval – Target Delay=5 ms

(k) Per-packet RTT

4 senders, $RTT_{base}=100$ ms.

N. Khademi, D. Ros, M. Welzl

The New AQM Kids on the Block...
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


