PI2
AQM for classic and scalable congestion control

Koen De Schepper
Bob Briscoe
Olga Bondarenko
Inton Tsang

April 4th, 2016
PI recap

Every $T_{update}$ interval do:

$$\Delta p = \alpha^*(current\_queue - TARGET) + \beta^*(current\_queue - prev\_queue)$$

$$p = p + \Delta p$$
Enhancements are: rate estimation, queue delay and gain scaling
Choosing $\alpha$ and $\beta$ gains

- Higher $\alpha$ and $\beta$ values give faster response
- To be stable, the phase and gain margins must be $> 0$
- Gain margin evolves diagonally with $p$
- Reason is $\sqrt{\text{in}}$

$$ r_{reno} = \frac{1.22}{\sqrt{p \cdot \text{RTT}}} $$
PIE solution: $\alpha$ and $\beta$ scaling

- PIE in Linux has extra internal alpha and beta parameters:
  
  ```
  if (p<1%)
      at=a/8
      bt=b/8
  else if (p<10%)
      at=a/2
      bt=b/2
  else
      at=a
      bt=b
  ```

- Current PIE draft has more if’s
PI2 solution: remove the $\sqrt{\phantom{}}$

- Replaces gain scaling with a square: $P[d] = (p')^2 = p$
- PI2 controls $p'$ which is actually $\sqrt{p}$ so $r \approx 1/p'$
PI2 also supports scalable TCP

- Scalable TCP needs no scaling, nor squaring
- Can use the same parameters as PI2 for Reno or Cubic
PI2 needs no $\alpha$ and $\beta$ scaling

• By squaring at the end, Reno can be controlled like a Scalable TCP

• Models used for:
  
  – TCP Reno on PI:
    \[
    \frac{dW(t)}{dt} = \frac{1}{R(t)} - 0.5 \frac{W(t)W(t - R(t))}{R(t - R(t))} p(t - R(t))
    \]  
    \[1][2]
  
  – TCP Reno on PI2:
    \[
    \frac{dW(t)}{dt} = \frac{1}{R(t)} - 0.5 \frac{W(t)W(t - R(t))}{R(t - R(t))} (p'(t - R(t)))^2
    \]
  
  – Scalable TCP on PI2:
    \[
    \frac{dW(t)}{dt} = \frac{1}{R(t)} - 0.5 \frac{W(t - R(t))}{R(t - R(t))} p'(t - R(t))
    \]


PI2 needs no $\alpha$ and $\beta$ scaling

- By squaring at the end, Reno can be controlled like a Scalable TCP
- Margins stay above 0 independent from $p'$ getting smaller
- Gain margin evolves flat with $p'$
Single Q PI2 experiments

- Lunix implementation
- DualQ option not used here

ECN Classifier

ECT(1) → Mark → p' > R

ECT(0) → Mark^2 → p'/2 > R²

Non-ECT

Rate estimation

Queue delay

Target Delay

Queue length

p'

p^2

p'/2

α, β

Target Delay

Rate estimation
Evaluation with Cubic & DCTCP

Steady state rate has been modeled for existing CC-AQM schemes:

- Cubic in Reno mode: \( r_{\text{reno}} = \frac{1.22}{p^{1/2} \cdot \text{RTT}} \)
- Cubic in DCTCP mode: \( r_{\text{cubic}} = \frac{1.17}{p^{3/4} \cdot \text{RTT}^{1/4}} \)
- DCTCP: \( r_{\text{dc}} = \frac{2}{p^2 \cdot \text{RTT}} \)

Deviations are DCTCP on non-step threshold:

- Cubic in Reno mode: \( r_{\text{creno}} = \frac{1.68}{p^{1/2} \cdot \text{RTT}} \)
- Deviation coupling used: \( r_{\text{dc-p}} = \frac{2}{p \cdot \text{RTT}} \)

Experiments:
- RTT: 5, 10, 20, 50, 100 ms
- Rate: 4, 12, 40, 200, 400 Mbps

Probability coupling used:

\[
p_{\text{creno}} = \left( \frac{p_{\text{dc-p}}}{2} \right)^2
\]
Dynamic tests

Fig 8 in [3]: Note: plots show average queue delay over 1s intervals
0-50s: 10 TCP flows; 50-100s: 30 TCP flows; 100-150s: 50 TCP flows; 150-200s: 30 TCP flows;
200-250s: 10 TCP flows; RTT=100ms Link=20Mbps

Cubic

DCTCP

Throughput ratio
one flow each

PIE

PI2

Cubic ECT(0)
Cubic Non-ECT
Cubic ECT(0)
Cubic Non-ECT
DCTCP ECT(1)
Reno mode
Cubic mode

Rate balance [ratio]

RTT[ms]: 4 | 12 | 40 | 200 | 400
Link[mbps]: 4 | 12 | 40 | 200 | 400
Throughput ratio
wrong ECN marking; one flow each
Queuing delay
one flow each

Average Q delay:
- Cubic Non-ECT
- Cubic ECT(0)
- DCTCP ECT(1)

\[ \alpha = 0.125 \]
\[ \beta = 1.25 \]

PIE

PI2

=99^{th}

\[ \rightarrow \text{less responsive} \]

Higher gain possible for PI2

\[ \text{PI2 using basic } \alpha \text{ and } \beta \text{ PIE parameters} \]
DualQ is deployment goal

• L4S (smoothing in the end system) and DualQ with immediate network marking is outperforming any smoothed AQM (such as PIE and PI2)

• PI2 is useful to control Classic Q size, and signal can be applied on both Scalable (L4S) and Classic TCP

• Experiments here without DualQ
  → no ultra low latency
  → show impact of one change at a time
  → focus on throughput fairness
Conclusion

• Scalable (L4S) and Classic TCPs can get equal throughput (tcp-fairness)

• PI2 simplifies Classic queue control and supports Scalable queue / rate control

• PI2 supports higher $\alpha$ and $\beta$ values
Questions

koen.de_schepper@nokia.com