QCN Stability Study

IEEE 802.1Qau
Stockholm Interim, Sept. 2007

M. Gusat, C. Minkenberg, R. Birke and R. Luijten
IBM Research GmbH, Zurich

Outline: QCN Stability Factors

- Case I: Derivative Gain "w"
  - impact of fixed w value
- Case II: Adaptive Sampling "P_s"
  - analysis of loop stability vs. delay
- Case III: Primal-Dual stability conditions
- Conclusions

Case I: Impact of Fixed Derivative Gain

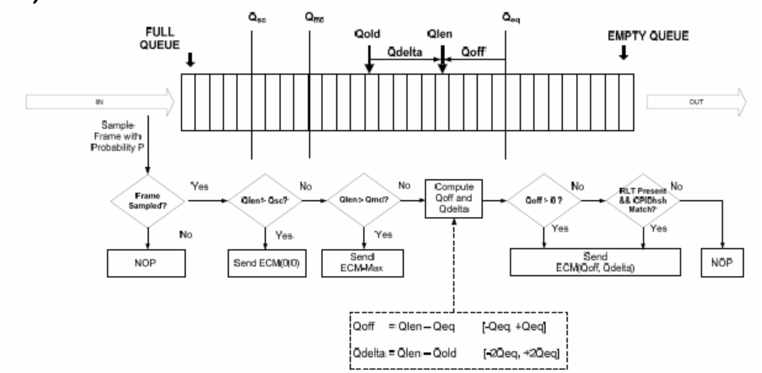
Delay effect through "w"

QCN stability with w=2.0 across a range of delays typical for small/medium datacenters

QCN Feedback

QCN feedback reduction: 2D → 1D, from *{q, q'}* to $F_{b-}(t)$

1. System's state variables (queue load sensor*)
   1. $q = Q_{off} = q(t) - Q_{eq}$, and,
   2. $q' = Q_{delta} = dq/dt$.



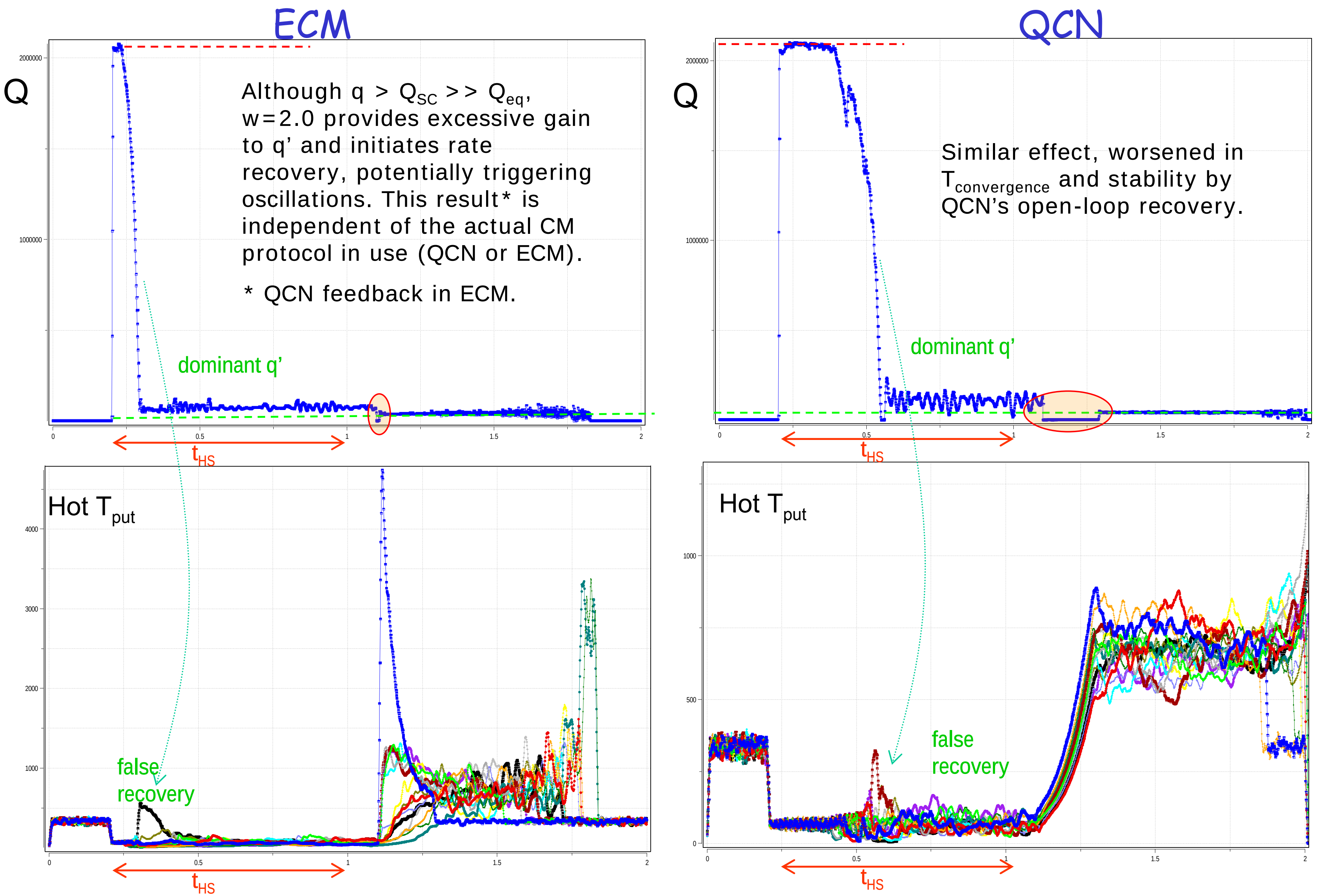
2. Negative-only F_b is signaled
   1. $F_{b-}(t) < 0$
   2. $F_{b-}(t) = (q(t) - Q_{eq}) + w*(dq/dt) = q + w*q'$, *w = derivative gain*
   3. Calculated in situ (per switch queue) and 6b *quantized as a single* state var F_b .

3. According to pole-zero analysis the derivative gain **w** provides a "leading zero" predictor => <u>should</u> compensate the *variable* lag/delay.

4. Not possible in QCN: w=2.0 is (i) *fixed*; (ii) the F_b value is *quantized* @ **CP** in a *single* var, (iii) then passed to RP after *<u>variable</u> lag*.

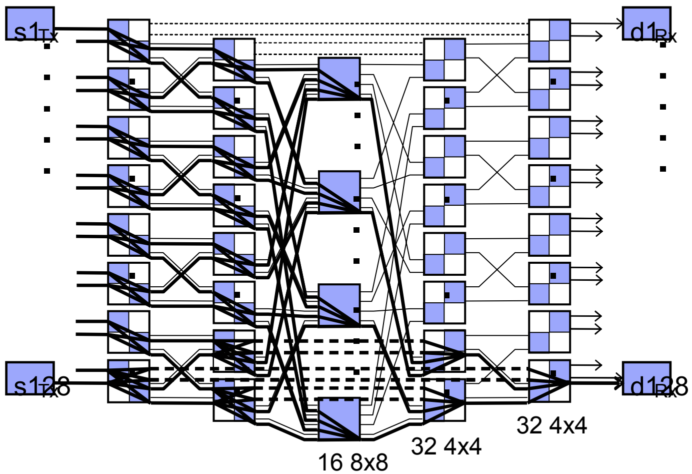
   - Control theory tells us to adjust w per feedback loop. (which...?)

A. Negligible RTT => Overcompensation => False Recovery... Must **reduce w**



B. Non-negligible RTT => Undercompensation => Must increase w

- Study effect of queuing delays in FT



$$RTT_{e2e} = \Sigma t_{queuing} + \Sigma t_{transport}$$

a) $\Sigma t_t >> \Sigma t_q$ → formal stability conditions

aka Type 1 stability in primal-dual

b) $\Sigma t_q >> \Sigma t_t$ → 'stochastic' stability

aka Type 2 stability, less formalized

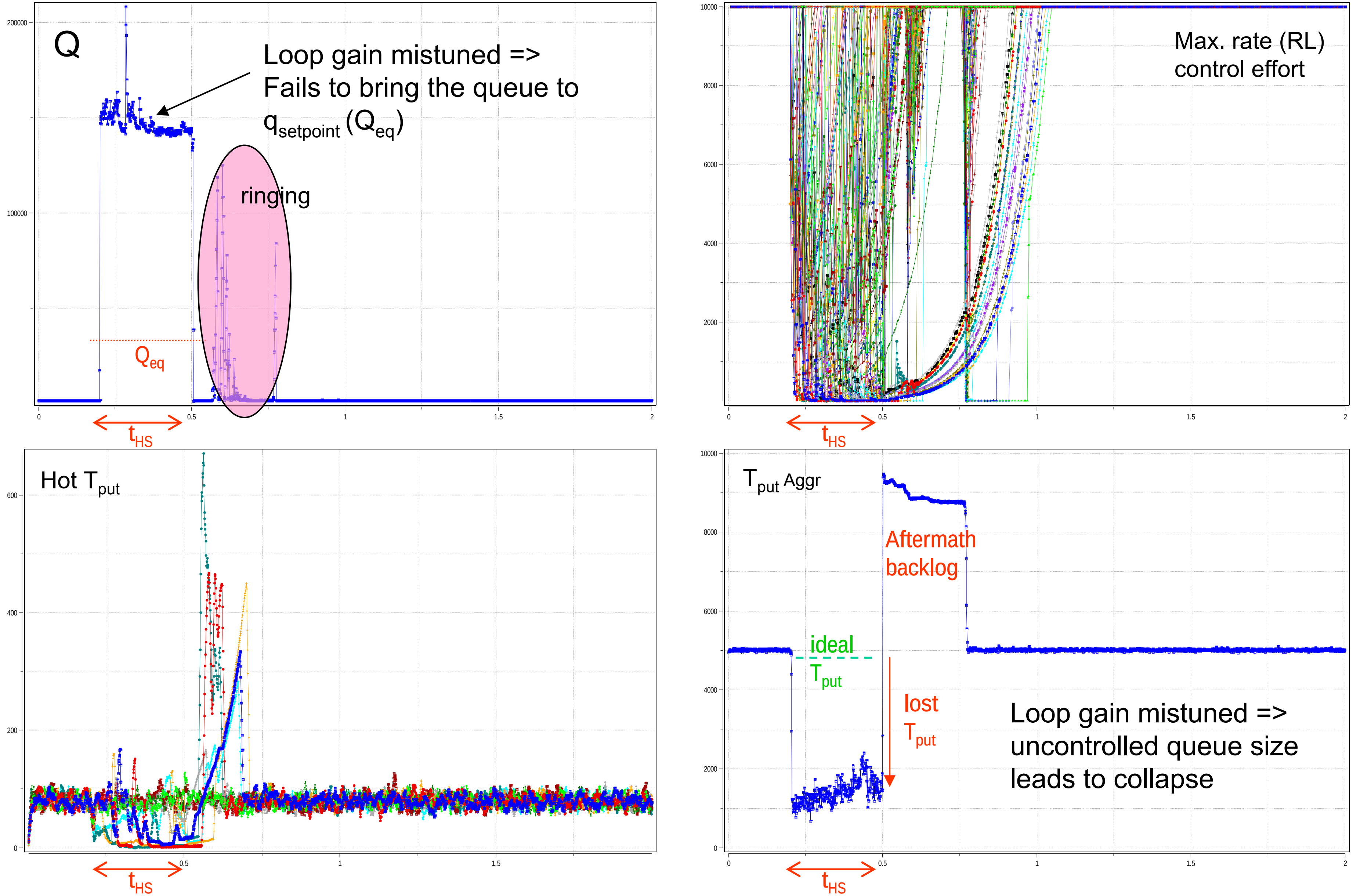
(a) well established in theoretical TCP and primal-dual CM studies

- Impact of long link delay previously shown in .1au

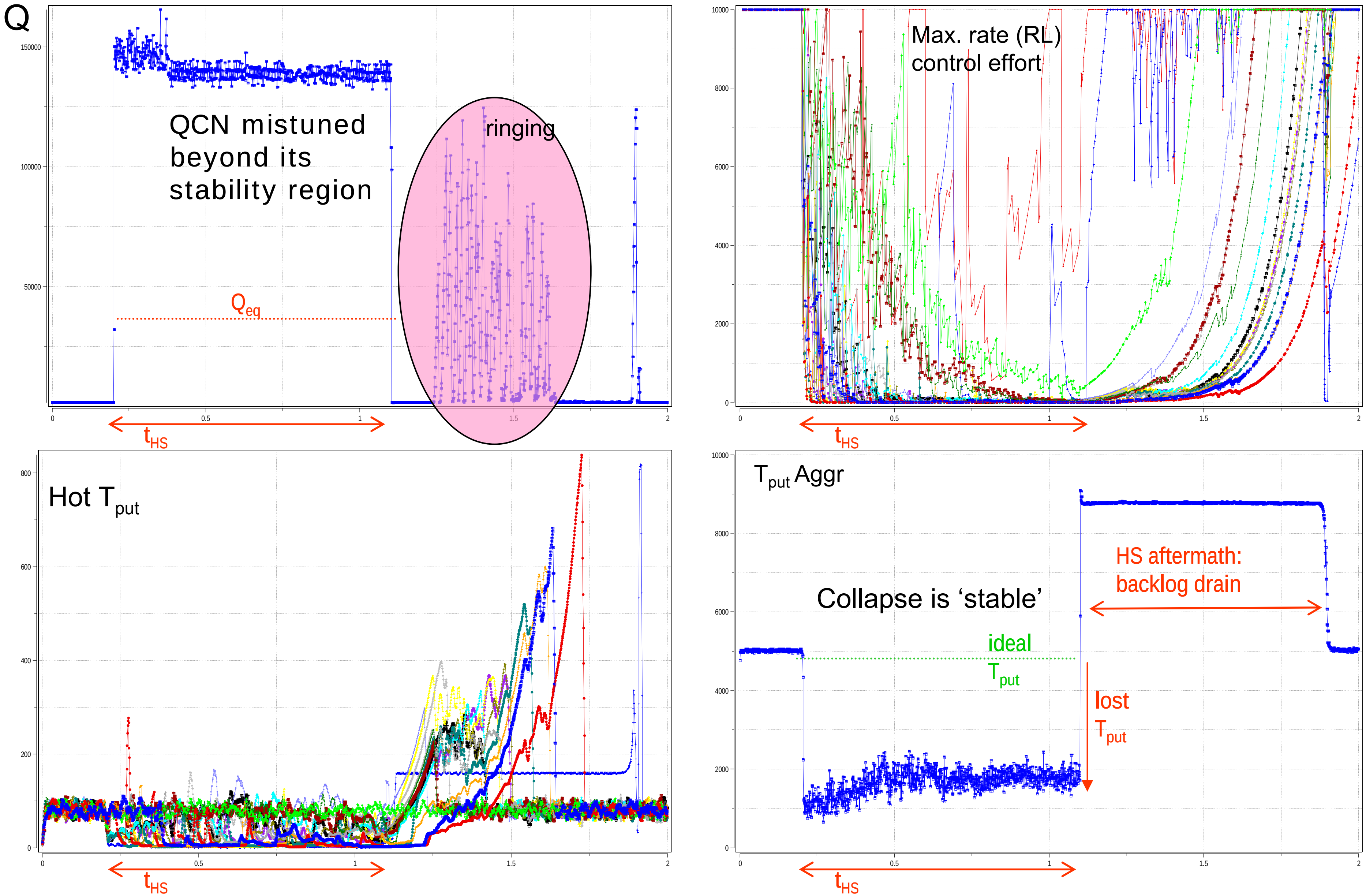
(b) We study a 5-level fat-tree w/ negligible RTT_{link}

- $RTT_{e2eMax} = \Sigma t_q = 5 * 100pkt \sim 0.6ms$

B. 5-level fat tree: QCN w/ w=2.0
May take longer to stabilize? See next...



Longer HS duration, still no stable operation...



Case I Recommendation: Adjust W w/ RTT

Fixed derivative gain ➔ Instability & Collapse (consistent in fat-trees). Hence...

- Make "lead zero" compensation possible in QCN (enable D from PID)
  - CP's role
    - ✓ the F_b value should *NOT be calculated and quantized* @ CP in a *single* var
      - send ***q and q' independently quantized*** to RP
  - RP's role
    - ✓ calculates the F_b value per flow (or group thereof) based on *q*, *q'* and *w*
    - ✓ w=2.0 is (i) a default param value, not *fixed – it differs per flow*;
    - ✓ reconsider the RP table: How to plug w?
      - *w* = O(RTT) => **RP sends RTT probe** to reflection point (CPID or destination)

        e.g. $w = \lg(rtt(t) / RTT_{ref})$),

        retain first 2 terms of Taylor series approximation → $\ln(1+x) = x - x^2/2$

- Add: Delay probing
  - ideally RP → CP → RP (requires CPID)
  - e2e RTT probing: TBD.

Q: Is an adaptive *w sufficient* for stability?

Case II: Adaptive Sampling Rate "P_s"

Delay effect through "P_s"

QCN stability with adaptive sampling under increasing delays

Primal side of loop: QCN-hat Rate Increase

Rate Increase Control (RIC) in 3 concurrent/sequential phases

RIC(t) =

1) (Discounted?) ***Extra/Fast recovery***: Reclaim the previous R_d by binary increase

$$t_{Fb-} < t \leq t_{FR} \Rightarrow r_{new} + \sum_{i=0}^{5} \sum_{j=0}^{25..100T_f} f(R_0, R_d, t_{Fb\_}) \rightarrow r_j(t)dt + \frac{R_d}{2^i} \simeq 1 - e^{-kt}$$

* Double integrator w/ (a) initial condition Rd; (b) enable t_{Fb-} ; (c) reset. Executes only once after enable. Byte-based counters, possibly enhanced w/ timer (switch condition?).

2) ***Active (AI) or hyperactive increase*** (MI): Probing for the previous equil.

$$t_{FR} \leq t < t_{AI/MI} \Rightarrow r_{new} \approx e^{xt}$$

* the choice of AI vs. MI depends on traffic and CM target

3) ***Drift***: MI to claim excess C (newly available Bw)

$t_{AI/MI} \leq t$ => *multiplicative increase*

4) ***F_{b-}hat*** = F_{b-}hat + $F_{b\ i}$, for i < k*50 else F_{b-}hat = F_{b-}hat/2

+ integrator to grab newly available Bw

- introduces an additional pole

Dual side: QCN as a Control Loop w/ Lag (T) and Delay (τ) What happens when delay exceeds the dominant lag?

- Delay fundamentally affects closed loop control. Critical when $T > \tau$
- QCN[1]: load sensor model reduced to 1st order system w/ dominant lag (sampling time constant T) and non-negligible delay ($\tau = RTT_{e2e}$)

1. $$QCN^1(s) = \frac{e^{-\tau s}}{1 + sT}$$

   RTT delay

   Sampling/Marking lag

   Note: QCN's control loop is a higher order system

2. $\tau = \tau_{queue} + \tau_{transport} \approx \tau_{queue}$, Conservative assumption in datacenters

3. $T = 1/f_s = 1/(P_s * \lambda_{aggr}) = 1/(P_s * n * \lambda(t))$, n = no. flows @ CP, $\lambda(t)$ = rate
   1. $0 < P_s \leq 1$,
   2. QCN: P_s = 1..10%

4. (1 + 3) => $$QCN^1(s) = \frac{n \cdot \lambda \cdot e^{-\tau s}}{n \cdot \lambda + \frac{s}{P_s}}$$

   non-linear, but of different rates

Analyse the effect of "$P_s : \tau$"-ratio during HS

1. $P_s \uparrow \Rightarrow T \downarrow$ (improved observability)
2. $\tau = RTT_{e2e} \uparrow$ (2-5 orders of magnitude)

Why QCN's Adaptive Sampling Depends on RTT probing?

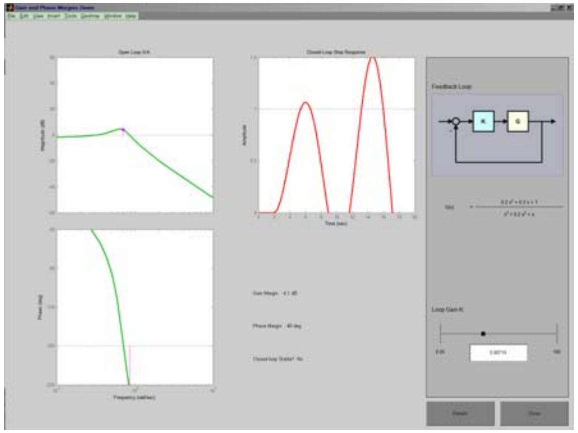
- Observations
  1. Whenever delay exceeds sampling lag the loop becomes unstable
     1. Hence the intrinsic conflict between increasing P_s and delay stability
     2. No clear trade-off is possible w/ RTT knowledge
  2. Sampling is aggregate @ CP, while F_b is per flow @ RP
  3. CP does not know RTT, nor "n" (# flows)
  4. Flooding RPs w/ bursts of outdated feedback requires adaptivity
     1. near RP's benefit directly from an increased P_s
     2. remote RP's don't... (must filter - decimation, Kalman)

- RTT probing is a good candidate

Matlab Demo: Delay Impact on Closed Loop Stability

- QCN control loop
  - primal: switched increase/decrease controllers => ~ PID
  - dual: 1^{st} order load sensor

- Stability depends on P_s, RTT, w, Gain and additional poles introduced by switching (lumped in T_i)
  - fragile stability: open loop recovery adds to lag ➔ reduced phase margin



- Take home
  - Stability vanishes proportional to T_s/τ
  - High sensitivity to tuning

Case III: Primal-Dual Stability Condition

From R. Johari and D.K.H. Tan. End-to-end congestion control for the Internet: delays and stability. IEEE/ACM Transactions on Networking, 9(6):818–832, 2001.

Theorem 5: Suppose $D_r = D$ for all $r \in R$. The system (4)–(5) is locally stable if the following condition is satisfied for all $r \in R$:

$$\kappa_r \left(\sum_{j \in r} p_j + \sum_{j \in r} p'_j \sum_{s: j \in s} x_s \right) < 2 \sin \left(\frac{\pi}{2(2D+1)} \right). \quad (16)$$

- Stability* condition under non-negligible delays
  - D= RTT delay; p_j= marking; x_s= rate; k_r= gain.
- Seen as a theoretical optimization problem, the primal-dual QCN algorithm must have a locally stable* solution depending on delay.
  - * predominance of queuing vs. transport delays in datacenters enforce stochastic stability conditions
- A 3rd argument for delay probing.

Conclusions and Recommendations

- We have analysed the delay impact on
    1. Fixed derivative gain w/ feedback degree reduction and calculation in switch
    2. Adaptive sampling

✓ Conclusions

1. Lack of ***delay adaption*** fundamentally impacts stability
2. No trade-off is apparent w/o actual delay knowledge

Recommendations

1. See pp. 9, 13 and 15

2. Adopt RTT probing.
    1. Proposed subpath probing: RP→CP→RP, using CPID
    2. If the above is not desirable (CPID issue), resort to e2e RTT probing (impact TBD).

Backup and Appendix

Simulation Parameters (see also fat tree specs for details)

- Traffic
  - I.i.d. Bernoulli arrivals
  - Uniform destination distribution (to all nodes except self)
  - Fixed frame size = 1500 B
- Switch
  - VOQ with 2.4MB shared mem
  - Partitioned memory per input, shared among all outputs
  - No limit on per-output memory usage
  - PAUSE enabled
    - ✓ Applied on a per input basis based on local high/low watermarks
    - ✓ $watermark_{high}$ = 141.5 KB
    - ✓ $watermark_{low}$ = 131.5 KB
- Adapter
  - RLT: VOQ and single; RR service
  - One rate limiter per destination
  - Egress buffer size = 1500 KB,
  - Ingress buffer size = Unlimited
  - PAUSE enabled
    - ✓ $watermark_{high}$ = 150 – rtt*bw KB
  - $watermark_{low}$ = $watermark_{high}$ - 10 KB
- QCN and ECM base
  - W = 2.0
  - Q_{eq} = 37.5 KB
  - $G_d = 0.5 / ((2*W+1)*Q_{eq})$
  - $G_{i0} = (R_{link} / R_{unit}) * ((2*W+1)*Q_{eq})$
  - $G_i = 0.1 * G_{i0}$
  - P_{sample} = 2% (on average 1 sample every 75 KB
  - $R_{unit} = R_{min}$ = 1 Mb/s
  - BCN_MAX enabled, thshld = 150 KB
  - BCN(0,0) dis/enabled, thshld =300KB
- QCN
  - Drift Factor = 1.005
  - Timer Period Drift = 0.0005 s
  - Extra Fast Recovery enabled
  - EFR MAX disabled.
  - A = 3 Mbps
  - Fast Recovery Threshold = 5
  - Hyper Active Increase disabled
  - No F_b-Hat

Non-negligible RTT => Undercompensation => Instability and Collapse... Must re-tune QCN and increase W

1. Shown 5L fat tree, Output Generated HS
   1. Tested from 3 to 7 levels: 16 to 256 nodes
   2. Traffic: *OG* of small to medium severity; shown 100->10% reduction
   3. t_{HS}
      1. short: 100-500ms
      2. long: 200-1100ms

2. N^2 flows: e.g. for 256 nodes => ~ 64K flows
   1. Distribs: uniform traffic without self-traffic. Bernoulli departure times and uniform across destinations. Only the flows going to the HS are recorded (256 nodes --> 255 flows) and the global T_{put}.

3. *OG*
   1. 0.5 background traffic.
   2. HS host reduces service rate to 10%
   3. HSV = 5