Feedback Request Strawman: Using .1Qau for Load Monitoring in DC Networks

Mitch Gusat
IBM ZRL, Aug. 28, 2008

What is Fb_Rq? On demand status info

Problem: Monitoring, app-level (L4+) performance profiling, runtime load balancing, adaptive routing...

Solution: Build on the investment in .1Qau-compliant switches

=> Deliver the full/available feedback to sources !

(before congestion arises)

Benefits

1. Speed: L2 feedback
2. Accuracy: Q info is already known to CP for QCN Fb. Ship it to RP!
3. Communicate Fb up the L3-7 stack. Use Flow-/RP-ID (?).

Fb_Rq Basics

- Monitoring options
  1. Proactive: RP-initiated => RP autonomosly issues Fb_Rq
  2. [Reactive: CP-initiated => RP begins to ping after QCN CNM]
  3. Single CP (reflect) vs. path (reflect reply & fwd request) Fb
  4. Stateless (anon.) vs. statefull CP (pings counted per FlowID)

Fb_Rq Strawman's Steps

1. RP: injects Fb_Rq pkt w/ L2 flag and Seq./Flow/RP-ID
2. CP: receives Fb_Rq
    1. sets Psample=1 (or disregards if busy or in "silent" mode)
    2. dumps queue status info (see next)
    3. sends Fb_Rp (CNM-like) back to originating SRC
    4. optionally also forwards Fb_Rq if DST != local CP
        => path profiling (multi-pathing issues)

Extended Queue Status (EQS)

1. *Prio, Qsize, Qeq, Qoff, Qdelta* + options

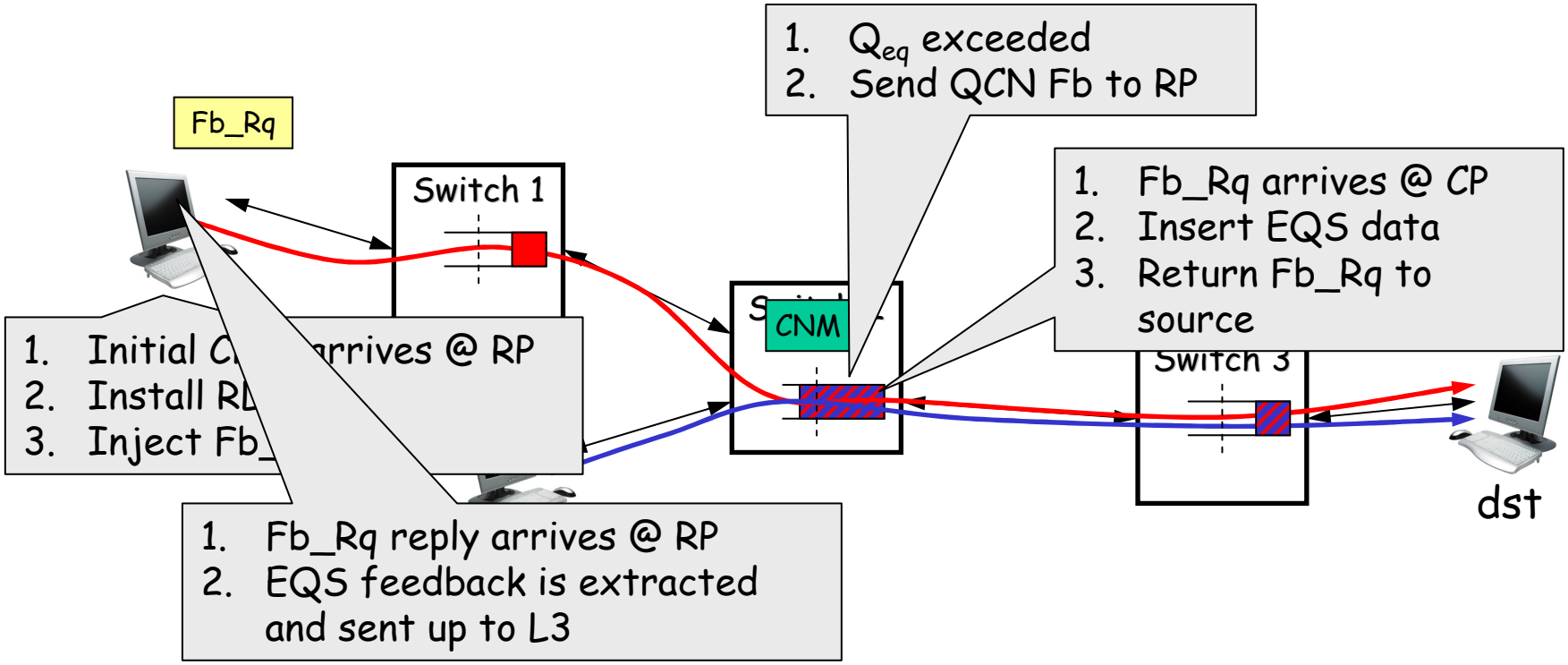
2. **PingCnt**: # of pings (from any FlowID) RX-ed since the last change of q' sign
   - ➔ marks one monotonic episode (of Q qrowth or drain)
   - ➔ If aggregate per CP, it can provide HSD/no-sharers (if RP maintains its own PingCnt)

3. **TXCnt**: # of pkts forwarded since the last change of q' sign
   - → as proxy hint for avg. service rate

4. ...

5. [additional info, e.g. pointer to a complete CP "brain dump"]

Reactive Fb_Rq Operation (animated)



- Reactive probing is triggered by QCN frames
  - hence only rate-limited flows are probed
  - Insert one Fb_Rq ping every n KB of data sent per flow, e.g. n = 750 KB
  - Single CP probing: CPID of probes = destination MAC
- Pro-active probing needs no CNM, but n should be based on actual load and delay

Conclusion

Q: What is being enabled?

A: Anticipate overload "see it coming"

- Potential for ***early*** custom response to congestion thru application specific logic:
  - e.g. app-driven adaptive routing,
  - task migration,
  - collectives (MPI mcast, combining ops, locks),
  - LB-ing engines,
  - scheduling hints: "optimize for latency" or "optimize for throughput",
  - control of new session admittance (postpone a bkup after the trading rush),
  - redundant data placement,
  - …

BKUP

Why bother about Q'?

- Delay: queuing delay-dominated RTT destabilizes CM in large DC's
  - additionally the RP delay further reduces RTT budget (see 21st Aug. call)
- Oscillations
  - with quick On/Off congestion episodes false recoveries are frequent (presented in 2007)
  - when RTT > 0.5-1ms Qoff is (much) less significant than Qdelta
- Q' provides additional info

- Luckily Q' (aka Qdelta) is available @ CP

- Q' potential usage
  1. Q' marks monotonical periods: queue backlogging / draining
  2. can provide HSD (N sharers) Fb
  3. extends the state space {q,q'} for tighter RL control @ RP
     1. enables RTT compensation

Pkt. Format

- TBD…
  - Enhanced CNM: add EQS to CNM's Fb (QCN)