QCN: Improving Transient Response

Abdul Kabbani, Rong Pan
Balaji Prabhakar, Mick Seaman
Outline of presentation

- Statement of problem
- Algorithm for improving transient response of 2-QCN
- A principle underpinning the algorithm
- Trade-offs: Milking the principle further
Transient response of 2-QCN

• When bandwidth is available
  – 2-QCN takes longer to grab it
  – Qs: Is 2-QCN fundamentally handicapped by a lack of positive feedback? Or, can a source detect and grab available bandwidth in a simple manner?

• A key issue
  – Any attempt at improving transient response should not harm steady-state stability
Algorithm

• Estimate congestion at the source
  – Maintain an estimate of Fb, say Fb-hat, at each RL
  – Fb-hat is counted using a 5- or 6-bit saturation counter
  – Fb-hat is thought of as a source’s estimate of congestion

• Updating Fb-hat
  – For every Fb recd by RL: Fb-hat <-- Fb-hat + Fb
  – For every 50 pkts transmitted: Fb-hat <-- Fb-hat/2 (just right shift)

• Using Fb-hat: cycle-shrinking
  – Every time we begin a cycle of FR or AI…
    • If Fb-hat is small (e.g. 0 or 1): reduce length of cycle to 50 pkts from 100 pkts

• Idea: small Fb-hat implies no dings for a while, hence it is likely there is no congestion; so a source can quickly get to AI and grab extra bdwdth
  – Note: in equilibrium, Fb-hat will be more than 1, hence no cycle-shrinking will occur, hence stability is preserved
Simulation Comparison

• Parameters
  – 10 sources sharing a link; RTT = 40 microseconds
  – Buffer size = 100 pkts; Qeq = 22
  – Link BW: 10G during 0--2 sec and 4--6 sec; 0.5G during 2--4 sec
  – Fb-hat saturated at 31
  – FR cycle-shrinking: 50 pkts if Fb-hat is 0 or 1, 100 pkts otherwise
  – AI: also 50 or 100 pkts depending on Fb-hat as above
  – AI amount: 25 Mbps

• Note on choice of cycle-shrinking
  – We have chosen non-aggressive parameters above for cycle-shrinking
    • E.g. we have also tried shrinking cycle lengths to 25, 50 or 100 pkts depending on Fb-hat
    • We have also used gentle rate increases during AI
  – More aggressive choices certainly improve recovery time a lot, but we need to keep the basic trade-offs in mind
    • Complexity vs performance
    • Responsiveness vs stability margin
  – But, there is good potential for exploiting Fb-hat better (more later)
0.5G Bottleneck: Rate

Old 2-QCN: 202 msec
New 2-QCN: 136 msec
3-QCN: 96 msec

Note: We have seen recovery times as low as 113 msecs, the number above is on the high side; the average was around 125 msecs
0.5G Bottleneck: Queue
Most sources recover around 100 msecs, one source takes 200 msec
0.5G Bottleneck, Max source rate: 0.9G
Effect of straggler (this is random)
0.5G Bottleneck: Rate
Bernoulli sources, max offered rate 0.85G

Without Fb-hat: 180 msec
With Fb-hat: 110 msec
Sanity check: 6 sources sharing 10G link
Queue size with cycle-shrinking
Sanity check: 6 sources sharing 10G link
Rate with cycle-shrinking

![Graph showing rate over simulation time](image-url)
Sanity check: 6 sources sharing 10G link
Individual rates
Introducing Fb-hat symmetrizes the source and switch behavior

- **Switch**
  - Has input: Packets or source rates
  - Observes: Qoff, Qdelta
  - Goal: Drive Q to Qeq and Qdelta to zero
  - Action taken to achieve goal: Send Fb signals to sources

- **RL**
  - Has input: Fb signals from network
  - Observes: Fb-hat
  - Goal: Drive Fb-hat close to zero (i.e. just above 1, the threshold)
  - Action taken: Change transmission rate

This is like a primal-dual algorithm for congestion management

- Primal variable, source rate: Input to switch but output from RL
- Dual variable, Fb: Input to RL but output from switch
• A principle: The switch and source (or RL) pass just the right signals to each other so as to solve the global bandwidth partitioning problem in a distributed fashion.

• Clearly, other algorithms can be obtained from this principle; e.g. we have tried:
  1. Cycle lengths of 25, 50 and 100 pkts depending on Fb-hat values
  2. Stretching cycle lengths to 150 pkts if Fb-hat is large
  3. Letting Fb-hat go negative; this lets source know with more certainty that bandwidth is available

• As mentioned, these improvements reduce the transient response further (e.g. we had roughly 85 msec recovery time using option 1)
  – But they introduce slightly more work at the source and may affect the stability margin

• Overall, the approach promises to lessen the impact of not receiving explicit positive feedback at the source
• Check stability with large number of sources (100, 200, 500…)  
  – Want to ensure that cycle shrinking is essentially inactive in equilibrium; hence it is primarily useful only in transience  
  – We have found this is indeed the case (sims next week)

• In conclusion, the principle of primal-dual control yielded a simple Fb-hat based algorithm for improving transient response  
  – Refinements may improve the performance even more  
  – However, it is good to be cautious and draw the line somewhere in trade-off space