QCN: Notes on a Stable Improvement of Transient Response: Part 2

Abdul Kabbani, Rong Pan,
Balaji Prabhakar and Mick Seaman

Outline

- 2-QCN
  - A unified, simplified redefinition
  - Settling to lower rate quickly (e.g. severe bottleneck and PAUSE)
- Inferring available bandwidth in multipath scenarios
  - Method 1: Probing “congested paths”
  - Method 2: “Path-based” congestion notices

2-QCN: A redefinition

- A convenient way of viewing QCN is using
  - Current Rate (CR): Current transmission rate of the RL.
  - Target Rate (TR): Where CR wants to get to.
    - TR always greater than CR
    - TR may exceed 10 Gbps, CR can never exceed 10 Gbps

- Rules for changing CR and TR
  - When Fb<0 signal arrives
    - During FR1 (first cycle of FR)
      -- CR goes down with every Fb<0 signal, TR remains unchanged
    - During FR2 or higher
      -- RL into FR1; TR <--- CR just before ding; CR <--- CR(1-G_d|F_b|)
  - At the end of each FR cycle
    - CR <--- (CR+TR)/2; TR does not change
  - At the end of each cycle of AI or HAI
    - TR <--- TR + 12 Mbps for AI, or TR <--- TR + 12*cycle_cnt Mbps for HAI
    - CR <--- (CR+TR)/2

Settling to lower rate quickly

- It is important to settle RLs quickly to a ***lower*** rate
  - E.g. when a severe bottleneck appears, or when PAUSE is asserted and a saturation tree begins to form
- The addition to the algorithm is as follows
  - At the end of the FR1,
    - If TR > 10*CR, then TR <--- TR/8; CR <--- (TR+CR)/8
  - By reducing the ***transience*** time
    - Packet drops or bad effects occurring during congestion episodes are highly reduced
    - The effect is most noticeable when the RTT is large, because bursty dings are quite likely in this case, and the RLs take a long time to get into steady-state
    - Sims in Atlanta

Grabbing bandwidth: The multipath problem

- The SONAR idea presented last week had good recovery times while leaving stability completely unaffected
  - However, in the presence of multipathing, SONAR pings may not explore all the available paths

- We discuss two methods
  - Method 1: “Ping congested paths” is an extension of SONAR
  - Method 2: “Path-based congestion notice”

- Method 1
  - Insert a flowid into each packet
  - A CP sends the flowid back to the RL with an Fb<0 signal
  - RL stores the flowid from the last ding
  - When It wants to send a ping, it sends out the ping on a packet whose flowid equals the one stored
  - This makes it more likely that the “last congested path” gets pinged, similar to pinging a CP using CPID

Discussion of Method 1

- It is not exact
  - No guarantee that there will be a packet going through the last congested path
  - No guarantee that that path is the only bottleneck
  - No guarantee that the flowid we come up with is adequate
- Switch may receive a lot of back-to-back pings
  - Because SONAR pings are like pre-sampled packets, even though each RL only sends one ping every 10ms, it is possible for a switch to get back-to-back pings from many RLs
  - Better if the switch did the sampling
- These and other considerations lead us to Method 2

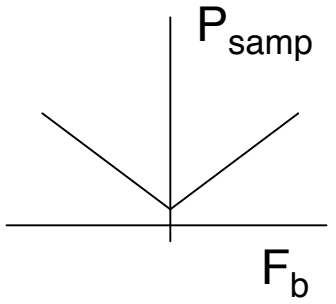
Method 2: Path-based congestion notices

- The key idea is simple to state
  - RLs will try to increase rate using a timer, not just a byte-counter
  - Therefore, switches which have no bandwidth available need to pro-actively push back
  - This means, ***multipathing or not***, every congested path will continually push back
  - **Main issue:** Choosing the timer value at the RL
    - Too small means aggressive source behavior, too large means longer bandwidth recovery times; but this is just a trade-off, the method is fundamentally correct

- Method 2: The details
  - A switch is either in "bandwidth available mode" or in "bandwidth NOT available" mode
    - Recall: bandwidth available means queue size is close to zero for a while
  - Therefore there are two congestion sensors at each switch at any time
    - Fb: which is a multibit signal
    - BA: a binary "bandwidth available" signal; BA = 0 means bandwidth NOT available
    - **Note:** Fb < 0 implies BA = 0, but not the other way around

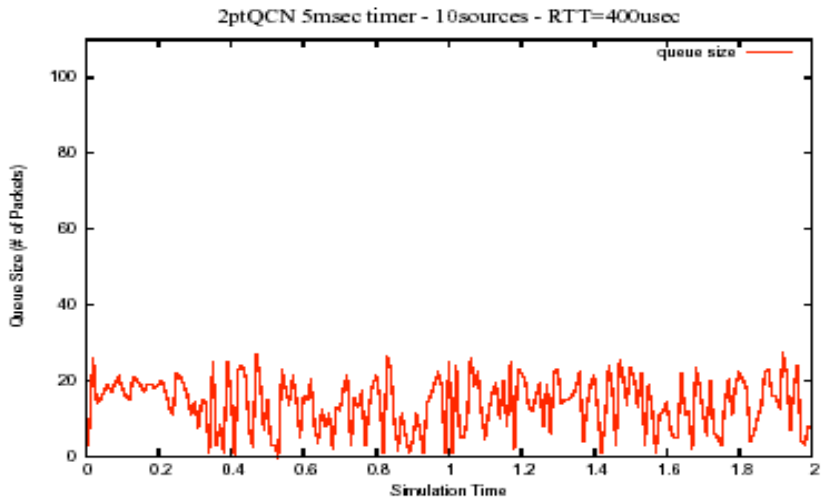
Method 2: Path-based congestion notices

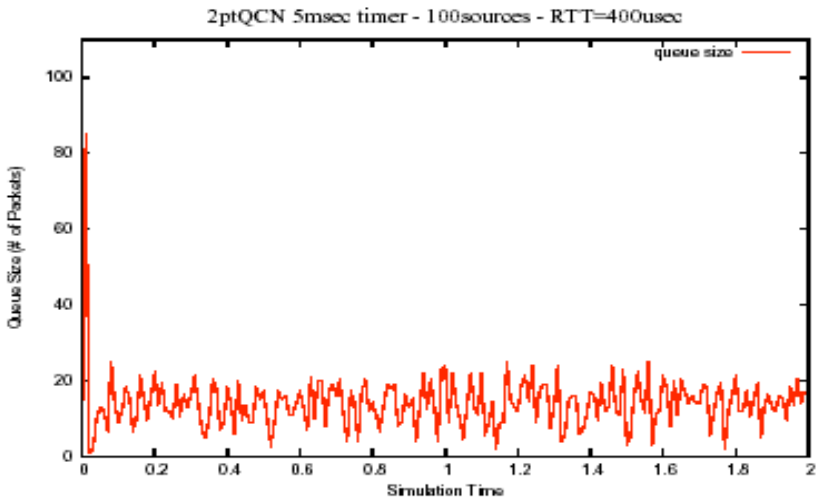
- At the switch
  - Sample packets with a probability which increases with Fb, ***both*** positive and negative
  - If Fb<0 for sampled packet, send to source
  - If Fb>=0
    - If BA=0, send "push back" message (Fb99) to source
    - If BA=1, do nothing

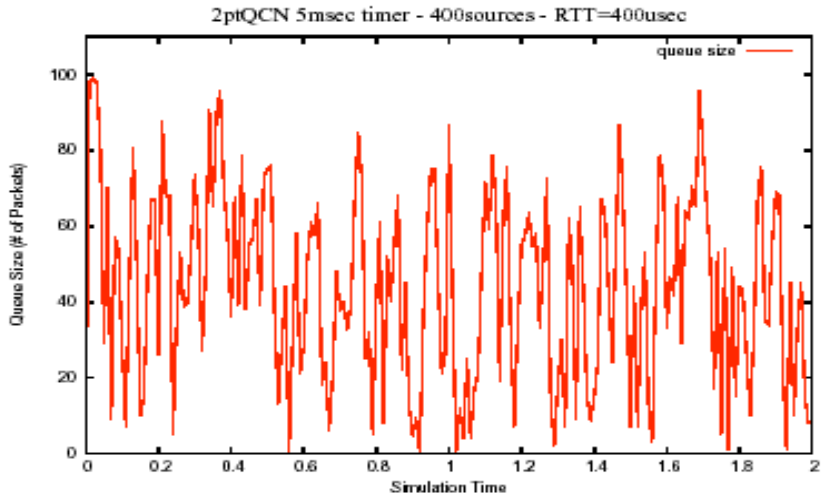


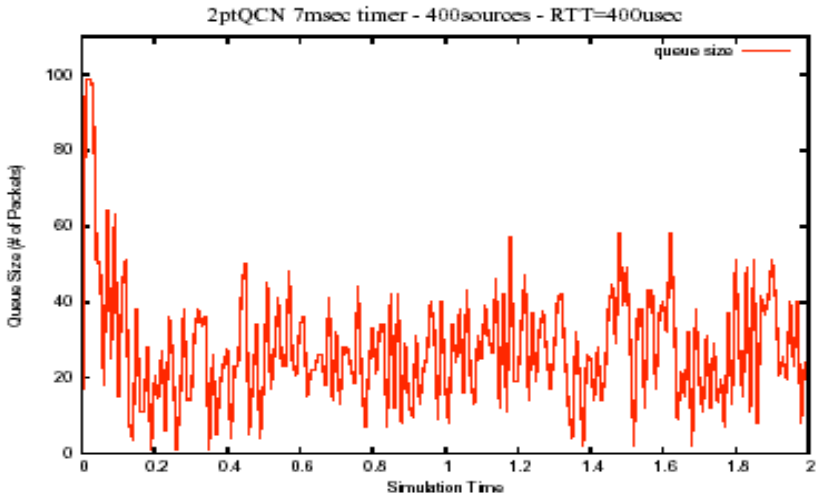
- At the RL
  - There is a timer which runs for T msecs
    - Timer is reset every time an Fb<0 or Fb99 message is received
  - When Fb<0 signal is received, same actions as before
  - When Fb99 signal is received
    - TR and CR remain unchanged
    - Increase the length of current cycle by 100 packets
  - When timer or byte-counter expires
    - Go to next cycle, update TR and CR as before

Simulations: Stability with Method 2

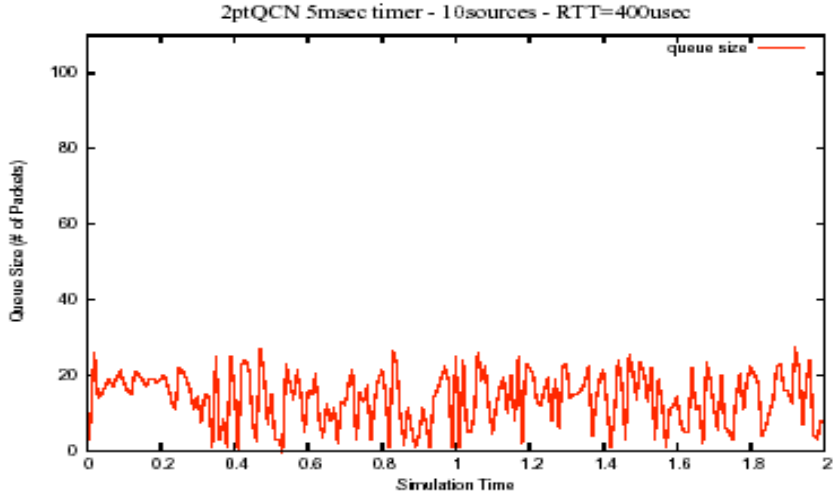


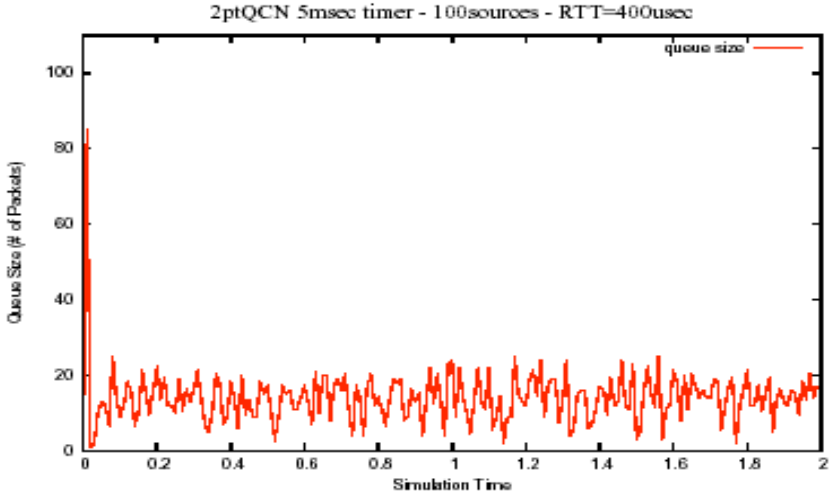


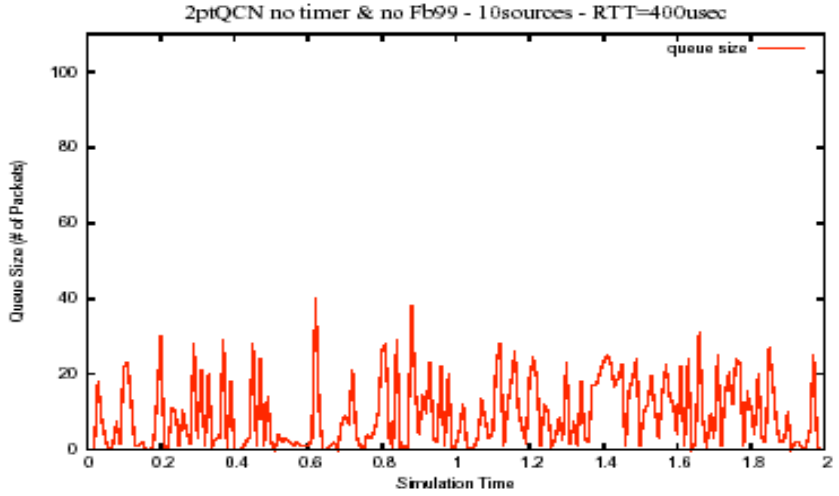


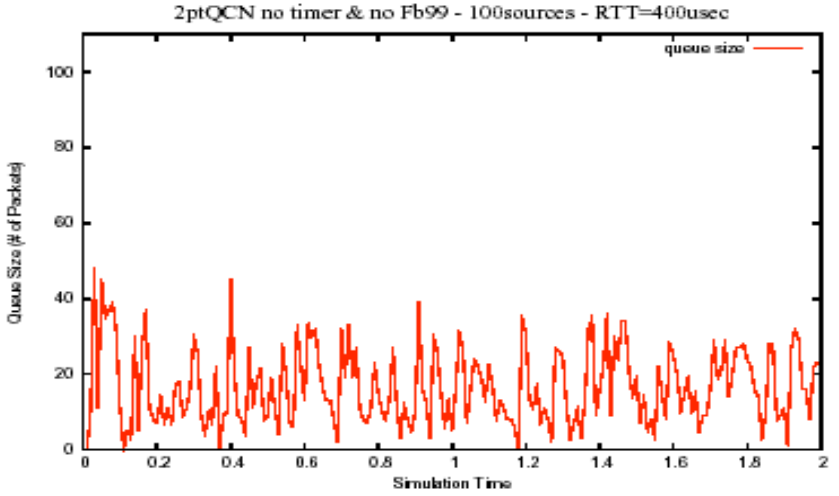


Stability improves due to cycle-stretching when Fb99 is received



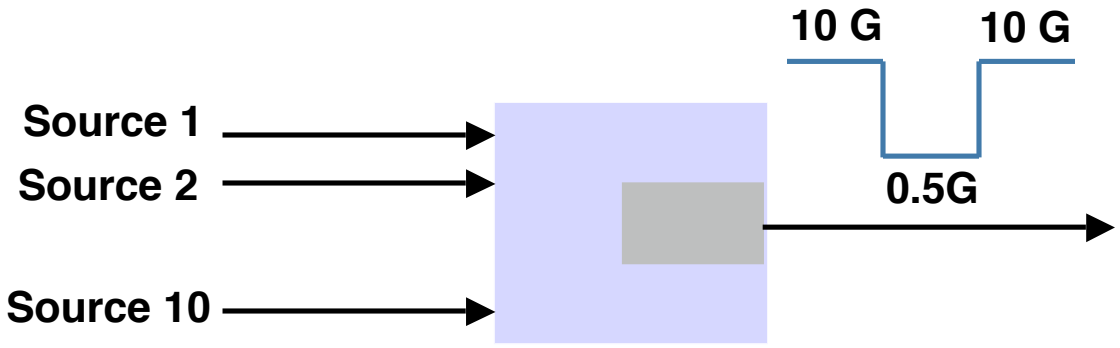




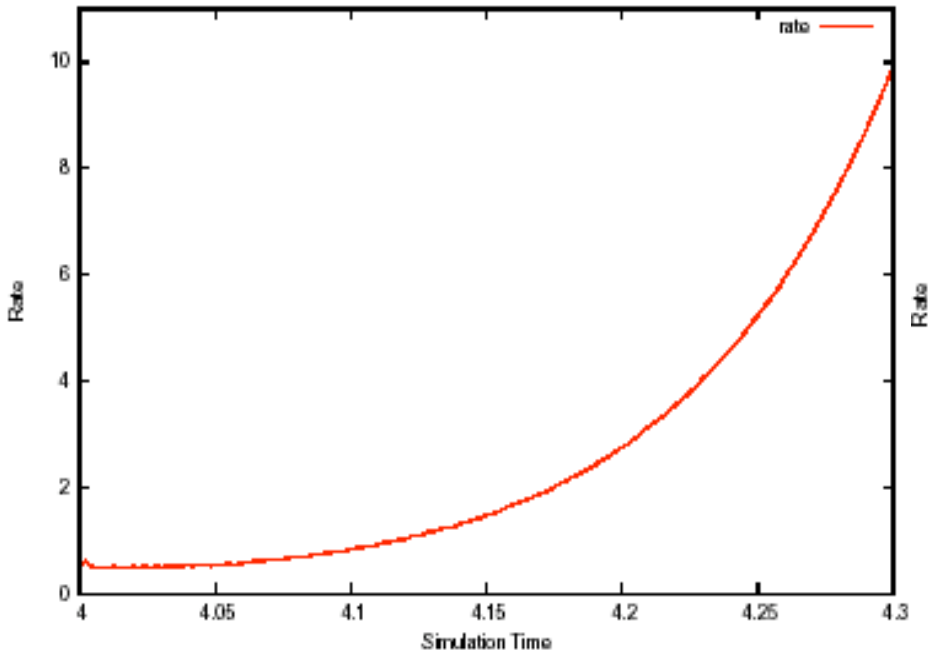


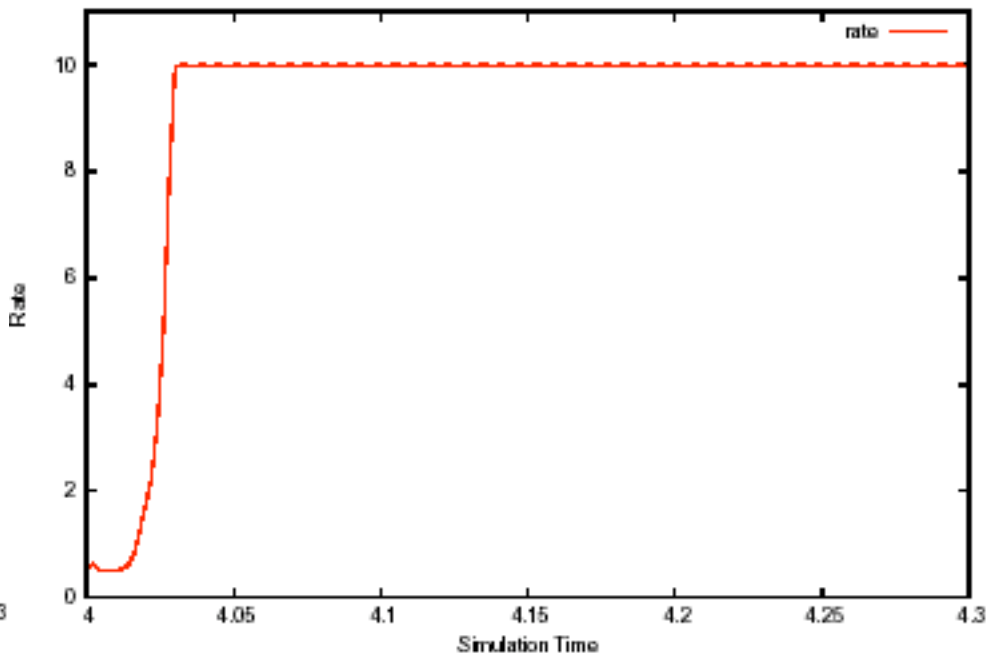
Recovery time: OG Hotspot

- Parameters
  - 10 sources share a 10 G link, whose capacity drops to 0.5G during 2-4 secs
  - Max offered rate per source: 1.05G
  - RTT = 40 usec
  - Buffer size = 100 pkts; Qeq = 22
  - Bandwidth recovery timer: 5 msecs
  - Drift timer disabled



Bdwdth Recovery





Time improvement

-- 300+ msecs to 28 msecs

Conclusions

- 2-QCN
  - Simplified, unified by the TR--CR formalism
  - Included a method that improves “downward transience;” when severe bottleneck appears or saturation trees forms
  - Two methods discussed for dealing with “upward transience”
    - Method 1 builds on SONAR
    - Method 2 more correct, but needs a liberal choice of timer value
- More sims and complements in Atlanta