QCN: Notes on a Stable Improvement of Transient Response

Abdul Kabbani, Rong Pan,
Balaji Prabhakar and Mick Seaman

Outline

- This is a presentation about two items
  1. Discussion of positive feedback, using probes or as in QECM
     - Increases gain in the feedback loop, causes loss of stability
     - Moreover, it does not probe the path
  2. A method for discovering available bandwidth in a 2-QCN world
     - Does not increase gain
     - Probes the path
     - Very quickly recovers bandwidth
     - Simplifies 2-QCN

- This is a first presentation; we have more data and further work which we will discuss next week
  – Welcome your feedback

Positive Feedback

- Using positive feedback; i.e. Fb > 0 values
  - Improves transient response
  - But adversely affects stability in large latency scenarios (from Guenter's presentation in Stockholm)

- We will see
  - This is due to an increase in the gain of the feedback loop
  - Large latencies aggravate the problem
  - Cannot aggressively increase transmission rate based on instantaneous value of Fb>0
  - Need a more "stable" indication of "available bandwidth"

Stability

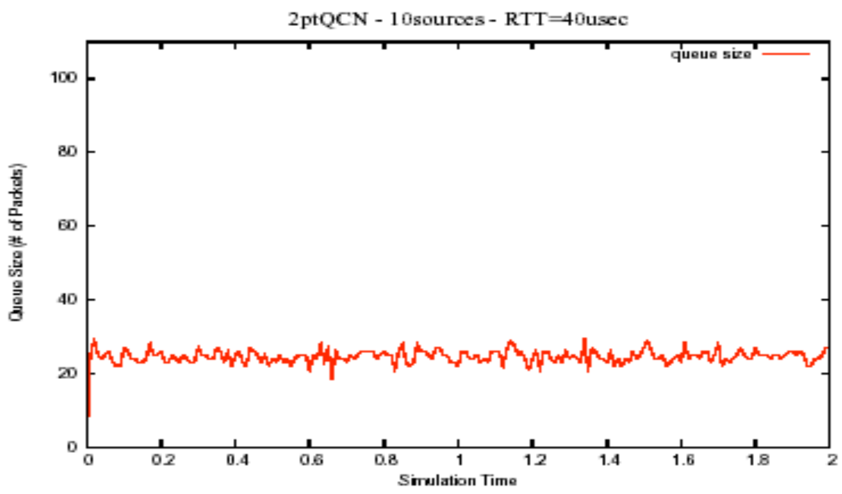
- Depends on RTT ***and*** the number of sources (N)
- Fb goes negative ***and*** positive as RTT and N increase
- So, we cannot infer that bandwidth is available at a switch just because the current value of Fb happens to be positive
  - This is the key point
  - The next few slides will show this

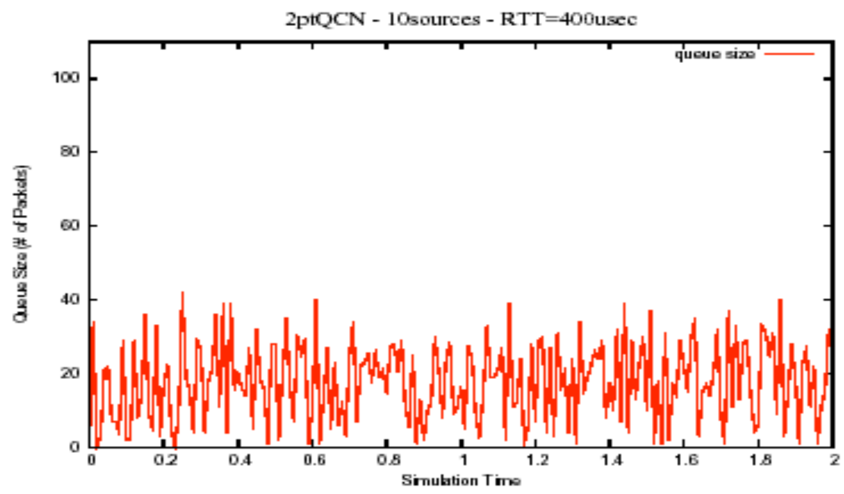
Stability

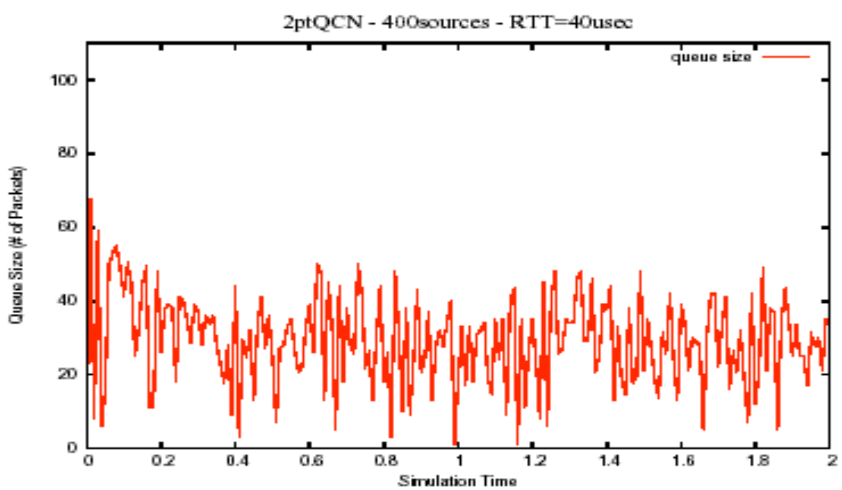
- Depends on RTT ***and*** the number of sources (N)
- Fb goes negative ***and*** positive as RTT and N increase
- So, we cannot infer that bandwidth is available at a switch just because the current value of Fb happens to be positive
  - This is the key point
  - The next few slides will show this

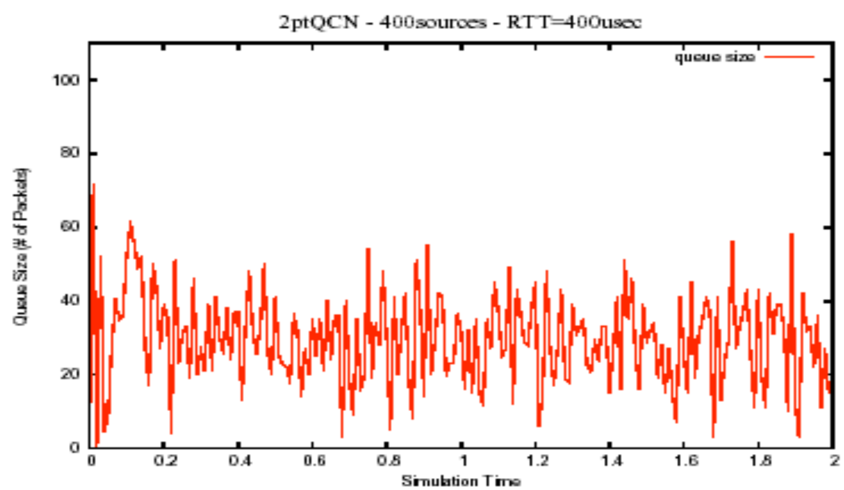
- Simulation scenario
  - Single link, 10G, 100 pkt buffer, q_{eq} = 22, AI increment = 12 Mbps
  - Drift timer ***disabled***
  - Total starting rate = 10G/N
  - We will compare QCN-P and QCN in the following scenarios
    - N = 10, RTT = 40 usecs
    - N = 10, RTT = 400 usecs
    - N=100, RTT = 40 usecs
    - N=100, RTT = 400 usecs
  - Interested in queue size ***and*** Fb values

QCN: queue sizes

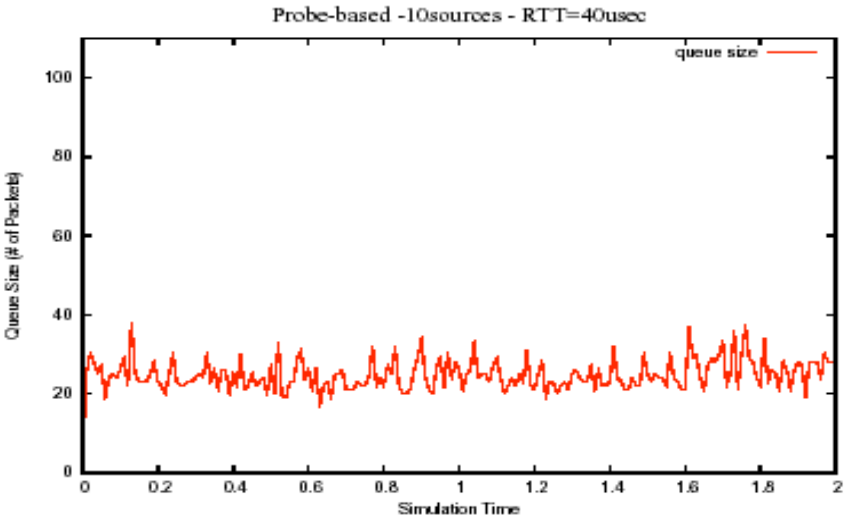


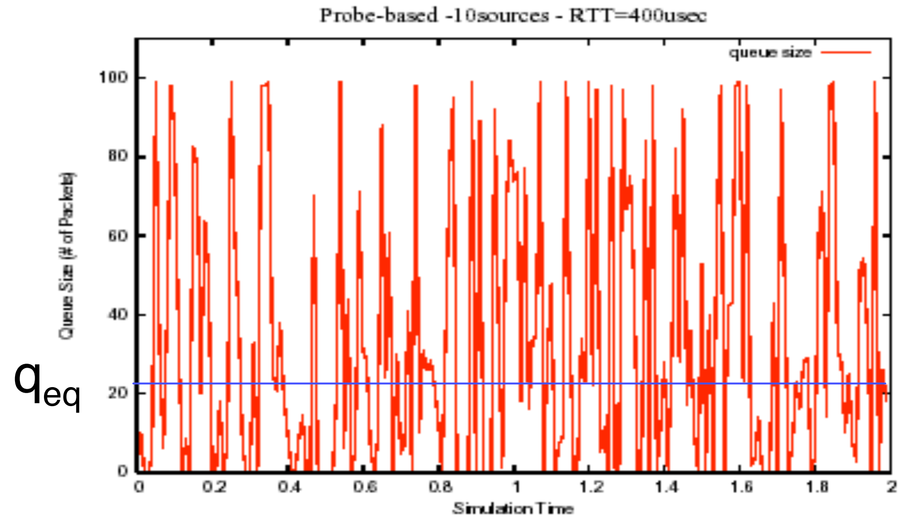


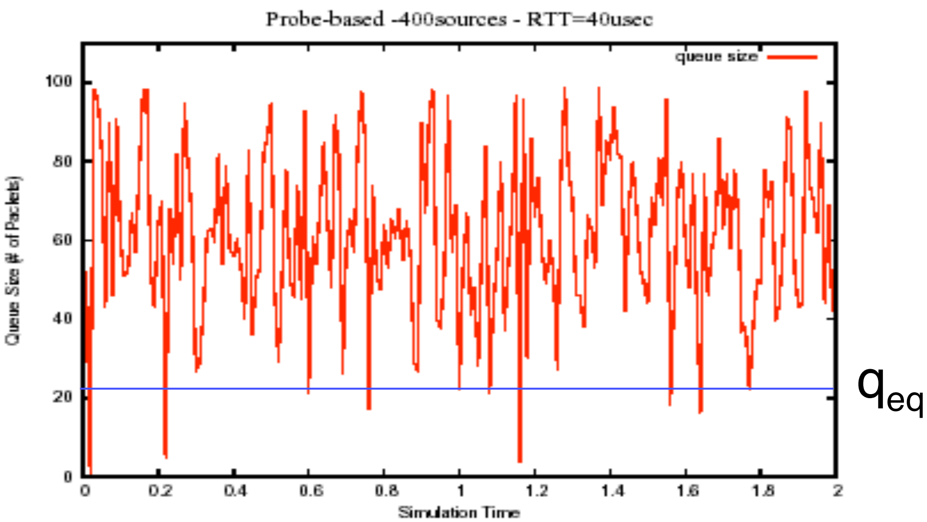


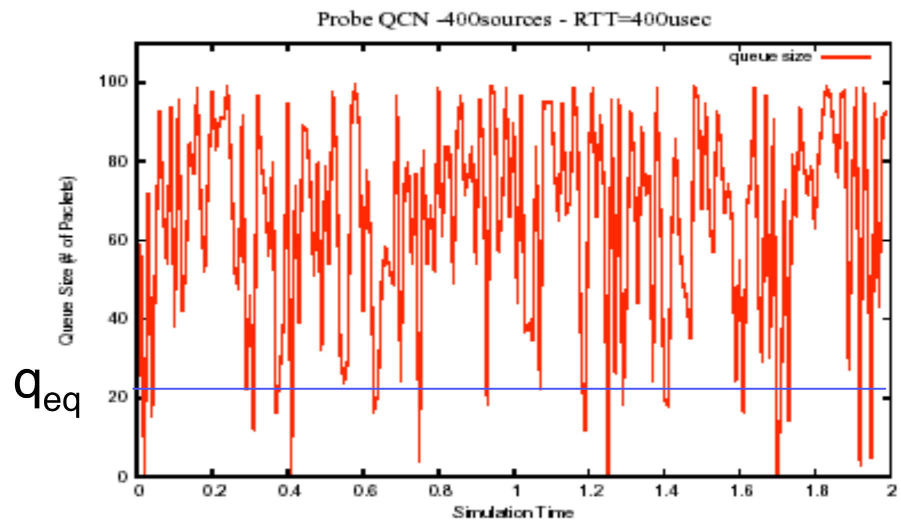


QCN-P: queue size









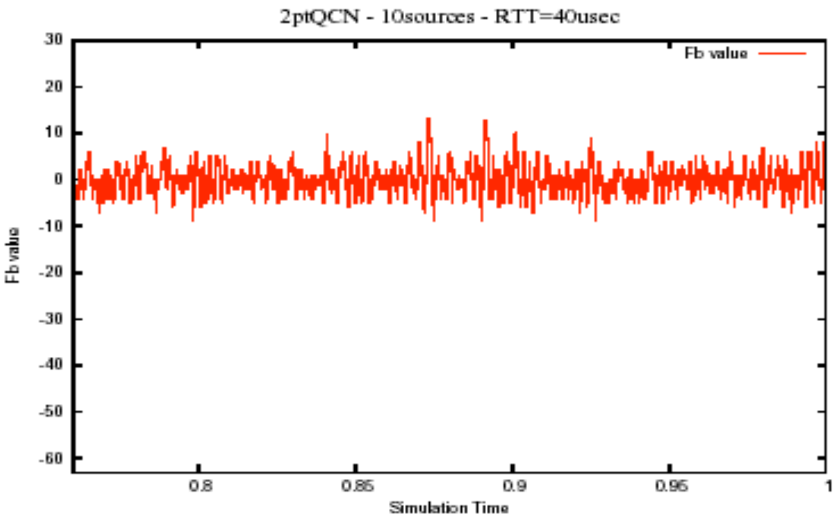
What’s going on

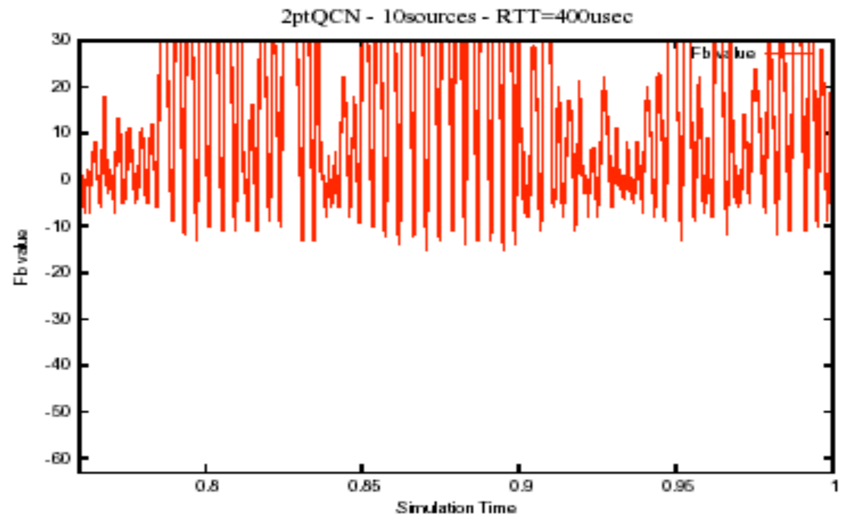
- QCN-P does the following (at least)
  - When a source gets Fb<0 signal, same as QCN
  - When a source gets Fb>0 (in response to a probe)
    - R <-- R + (constant*Fb + cycle_number - 5)*12 Mbps, where
      constant = link_rate*QCN_MAX_INC/(12*63) >= 13
    - So, if Fb = +10, cycle_number = 3, then the rate increase = 146 Mbps
      and if Fb = + 30, cycle_number = 3, rate increase = 464 Mbps
    - The source then goes to the next cycle of FR or AI, etc
  - Probes are launched at least once in 100 pkts

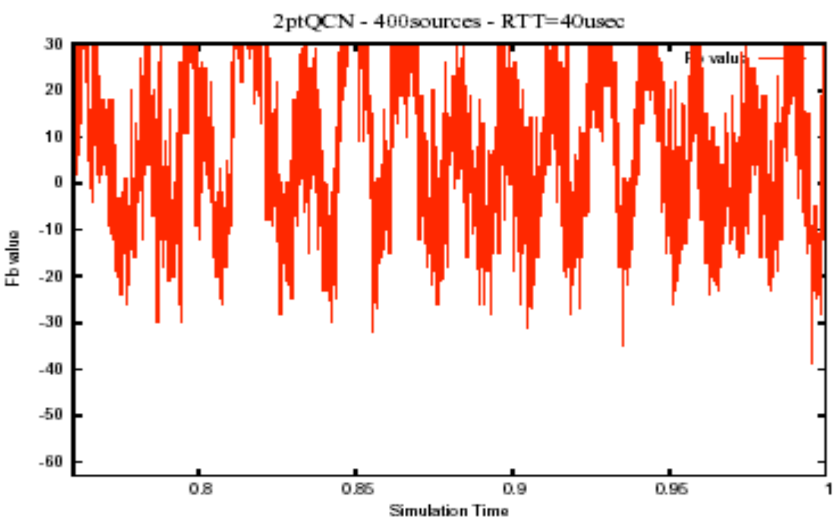
  - The point is: due to the aggressive increase in rate when Fb>0 signal arrives, the large N and large RTT regime adversely affects the stability of the scheme

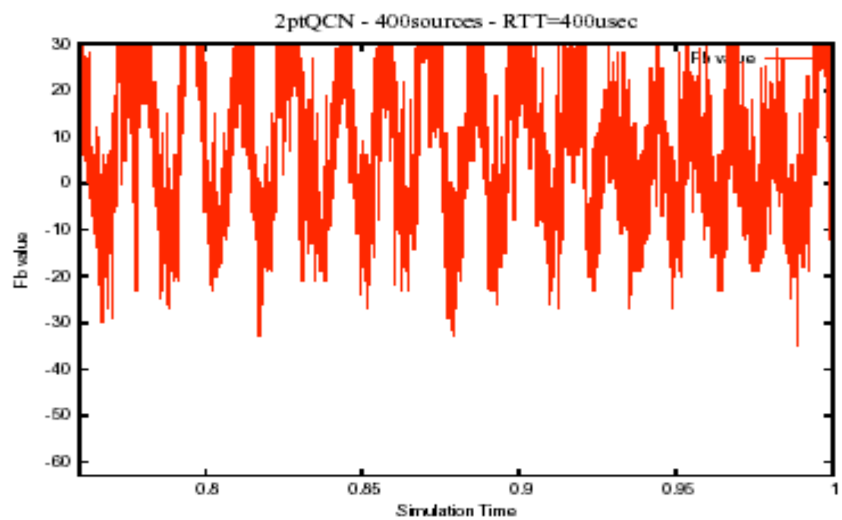
  - Let’s look at the Fb values

QCN: Fb values

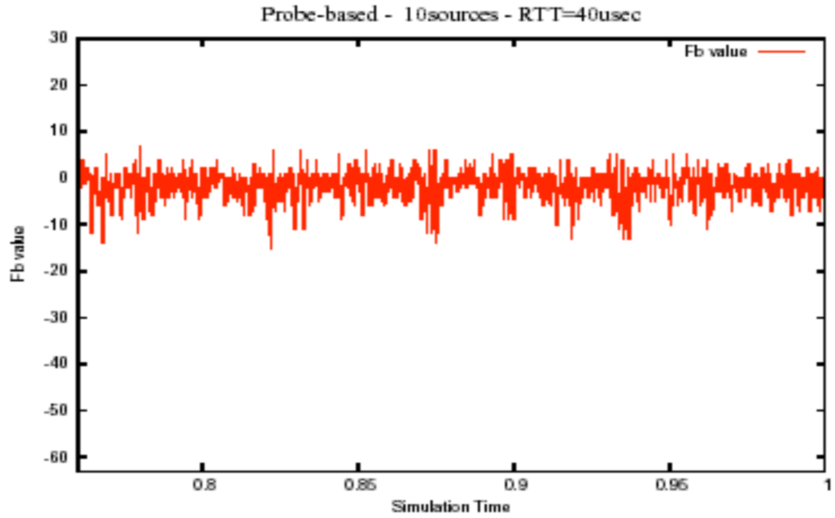


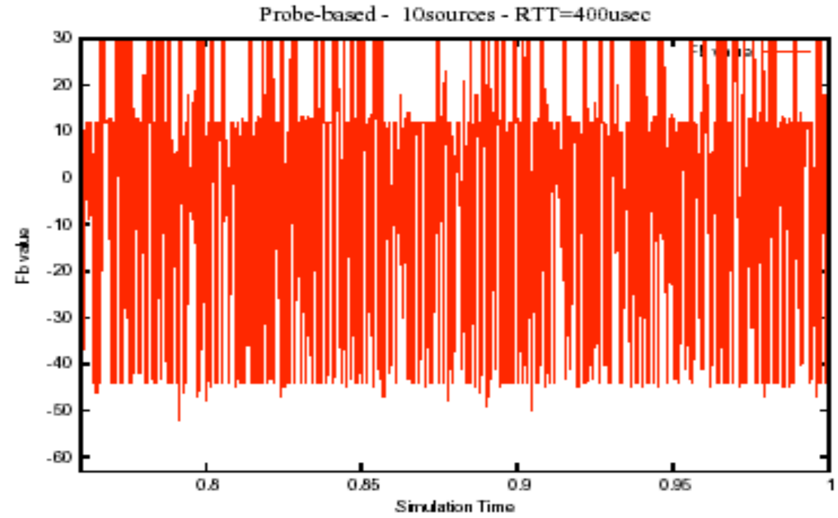


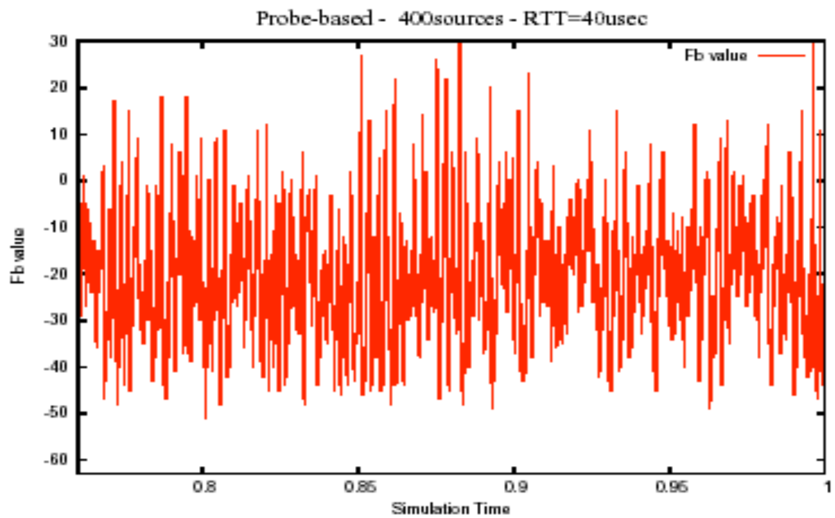


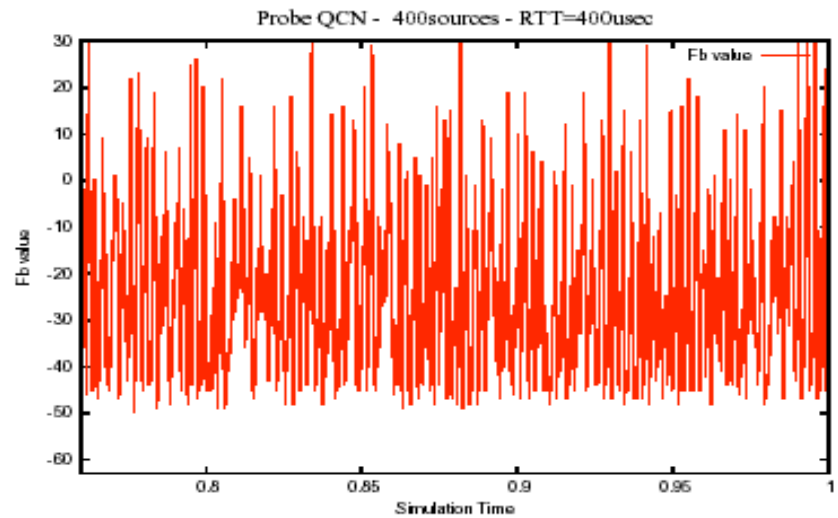


QCN-P: Fb values

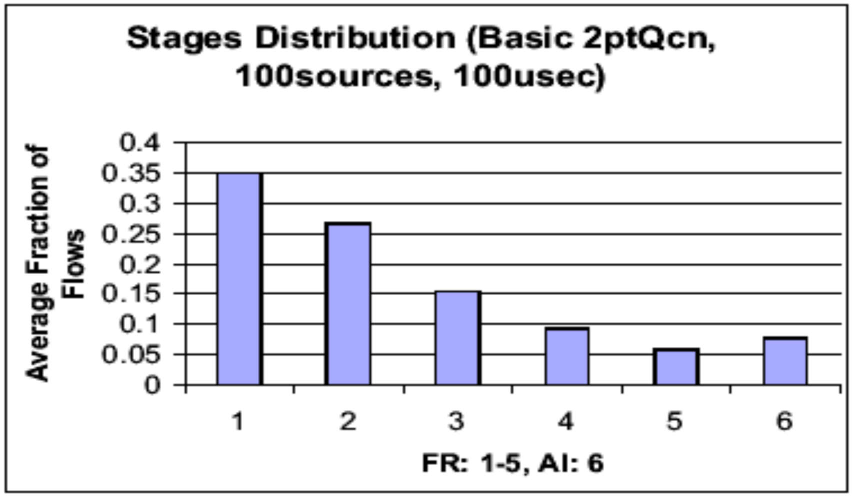


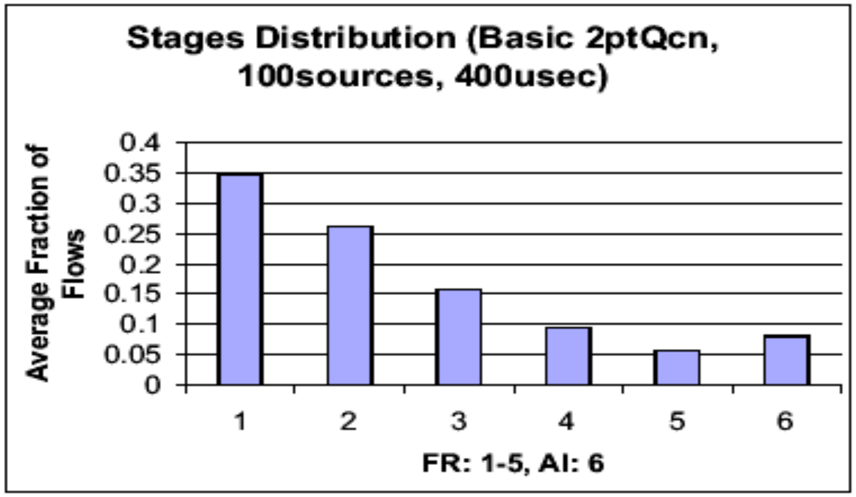


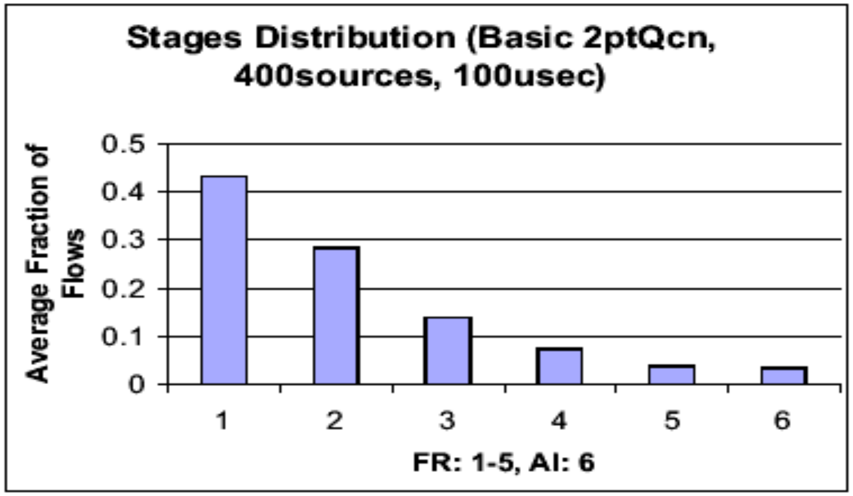


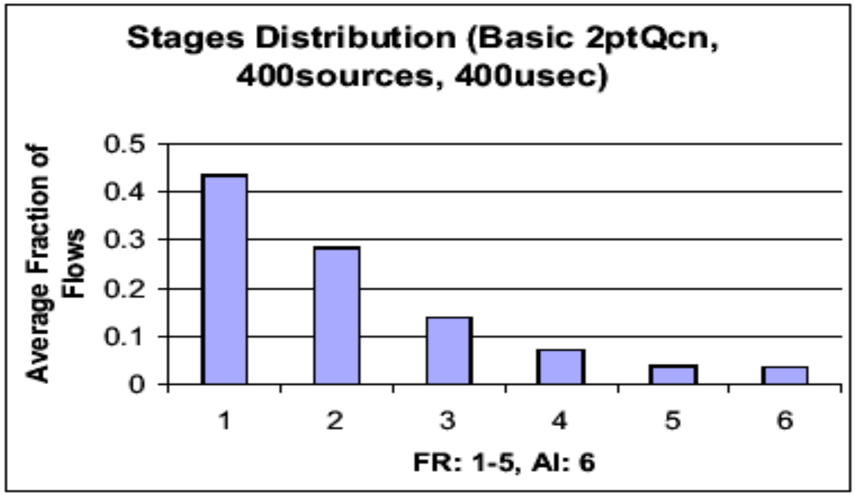


QCN: Histogram of FR/AI

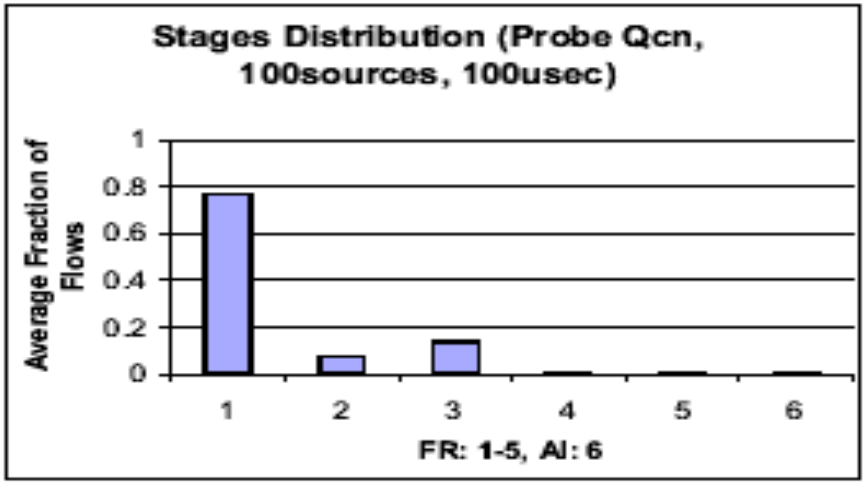


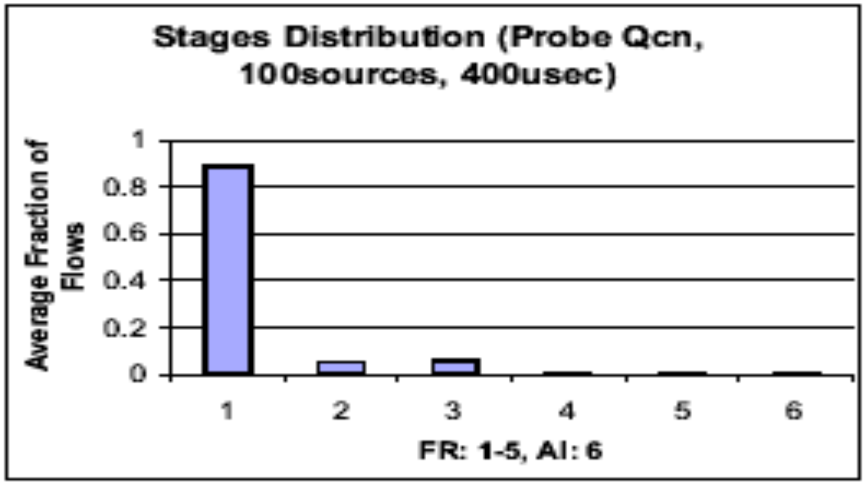


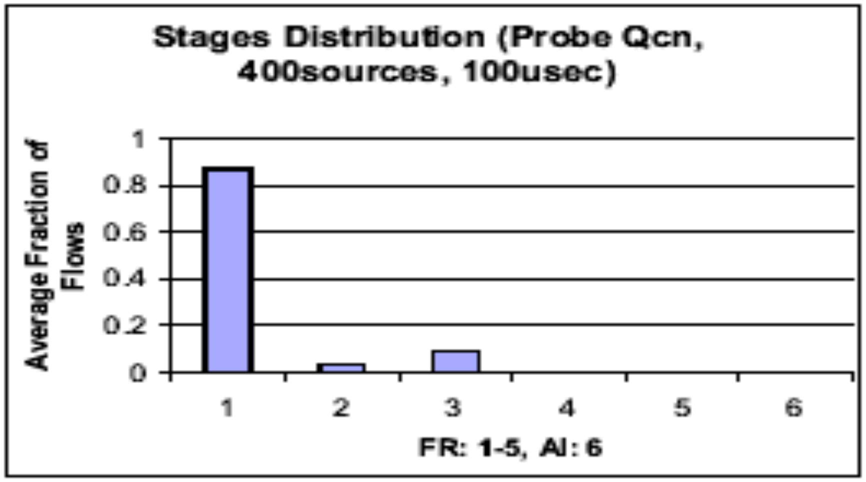


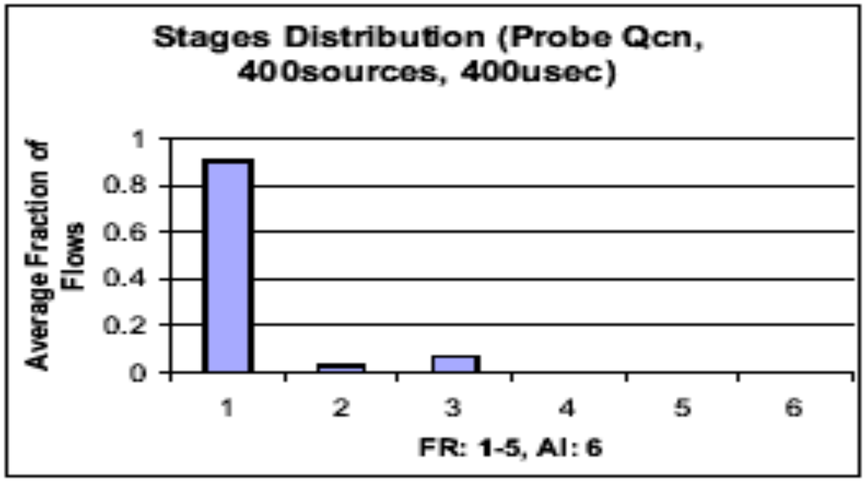


QCN-P: Histogram of FR/AI









Summary of positive feedback

- Since Fb becomes positive even when bandwidth is not available, and large N and RTT only makes this more likely, reacting to Fb>0 signals increases gain, compromises stability
  - We have seen this with QCN-P
  - Even the Orlando version of 3-QCN, which reflects Fb>0 values with increasing probability the more positive Fb is, increases the loop gain
    - Hence we changed the reflection probability in the Geneva meeting to 1%
  - 2-QCN (without Hyperactive Increase) is stable across a large range of RTT and N values
    - We need to improve its transient response ***without*** compromising stability
    - Need a “stable” indicator of “path bandwidth available”

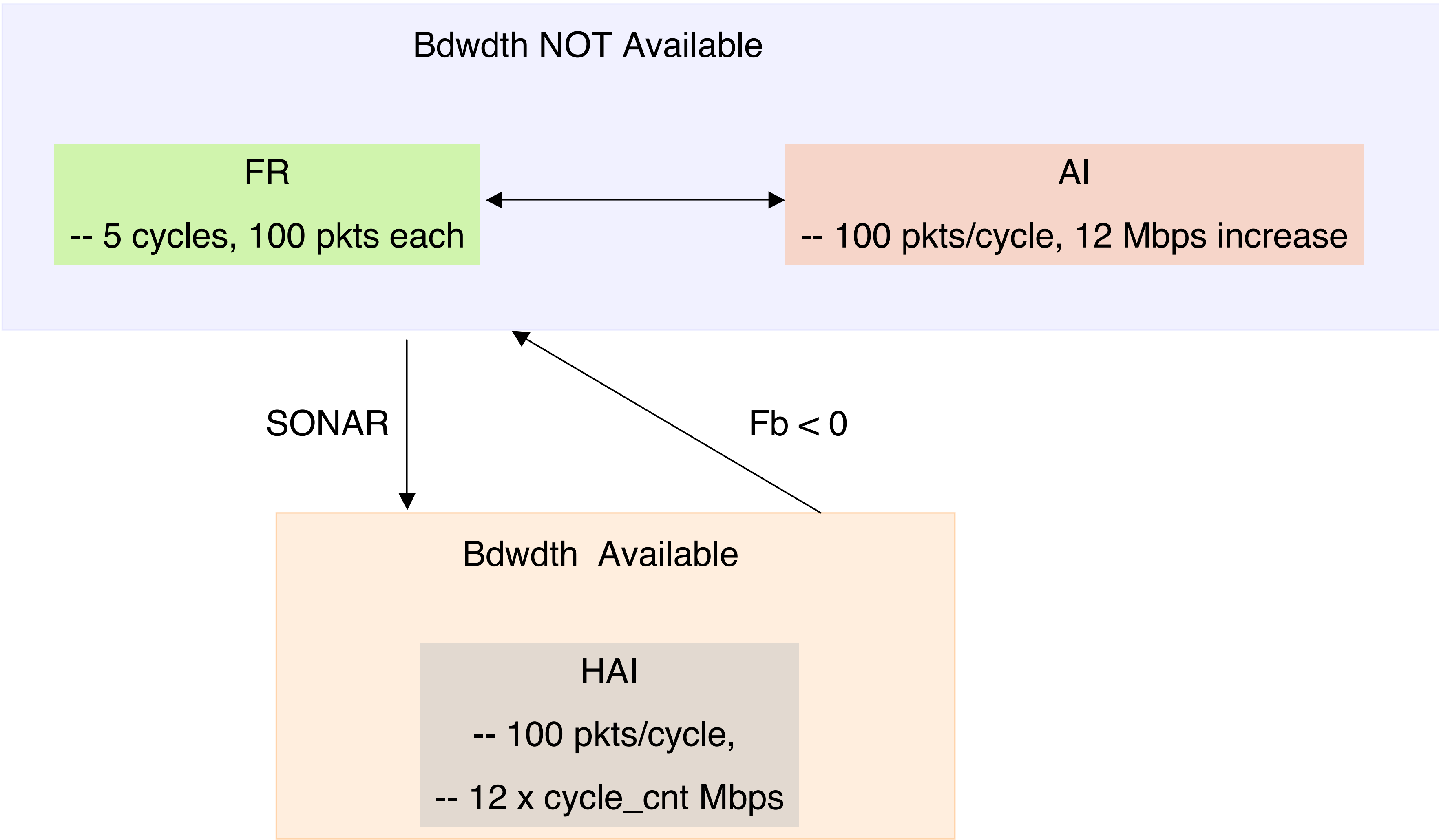
Our approach

- We need to ensure that we don't increase the gain in the loop
  - This means not increasing rate whenever Fb > 0
  - Because this will occur in large N and RTT scenarios

- Second, we need to probe the path

- Our main idea: Analogous to SONAR
  - RL sends periodic pings (details later) probing for extra bandwidth
  - A switch which "has no extra bandwidth" responds indicating this; else, it does not respond
  - If no switch responds, then the path has extra bandwidth available
  - RL infers this whenever a ping elicits no "echo"

The Algorithm at RL



SONAR

- The Ping Timer
  - The ping timer is in one of 3 states: Waiting to probe (WP), waiting for echo (WE), short fuse (SF)
  - WP is 10 msec duration, WE is 2 msec, SF is 0.5 msec

- The operation
  - The RL goes to the WP state whenever it receives an Fb<0 signal
  - If the WP timer expires, the next pkt sent by RL is a “special pkt”
    - Spl pkt == data packet with 1 bit set to indicate special
  - After Spl pkt is launched, RL goes to WE
  - If RL hears an echo for the SP
    - The ping timer returns to WP; RL continues operation (I.e FR or AI)
  - If the WE clock expires
    - Ping timer goes to SF; RL goes to HAI
    - In HAI, RL increases rate due to 100-pkt byte ctr ***and*** the ping timer

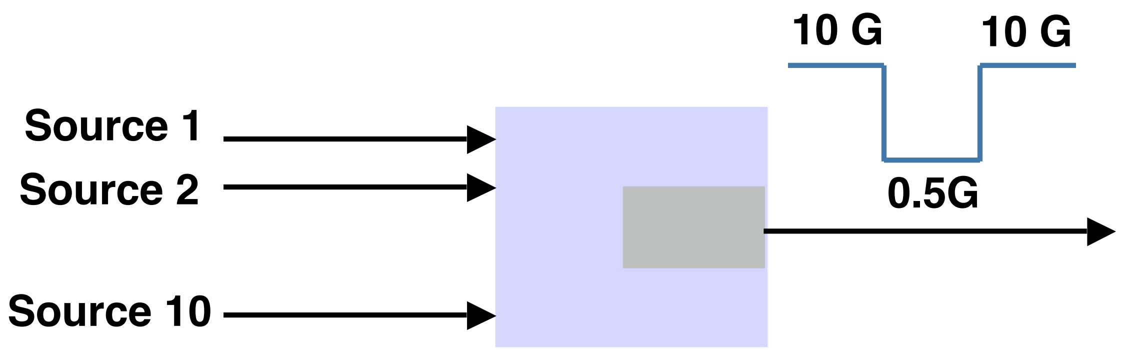
At the Switch

- Need to determine if switch is congested or not
  - Can't simply look if current value of Fb is positive
    - Again, this is why we changed 3-QCN to have constant reflection probability, instead of Fb-dependent reflection probability

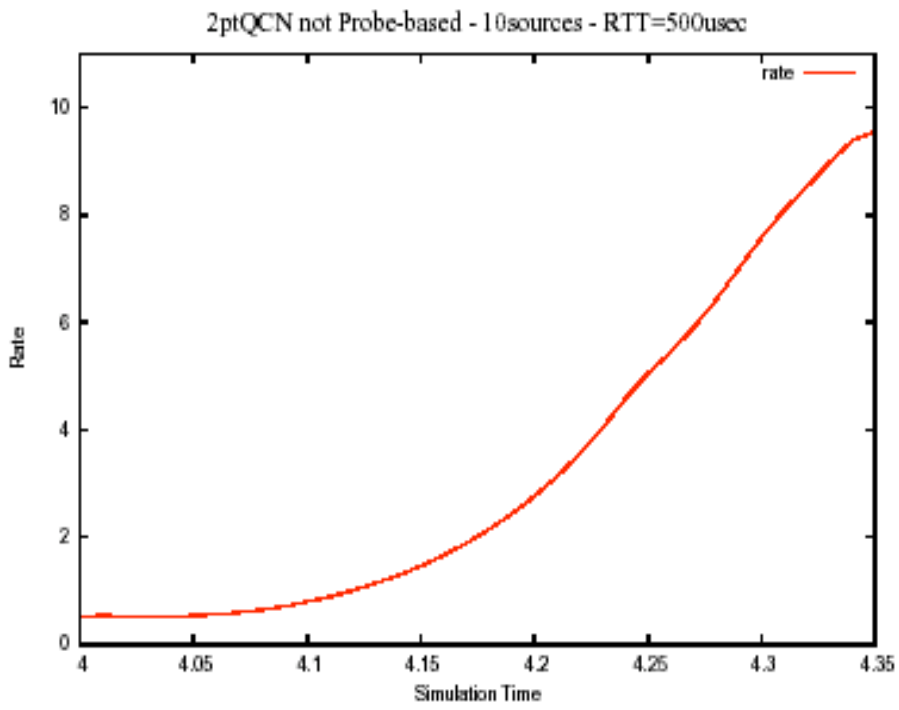
- Bdwdth available if queue-length < 6 pkts (say) for at least 10 msecs
  - Q_len < Q_eq (= 22 pkts) means input rate < output rate
  - So every time Q_len < 6 pkts, swith starts congestion timer
  - If timer expires, bdwdth available; else timer restarted when Q_len < 6 pkts again

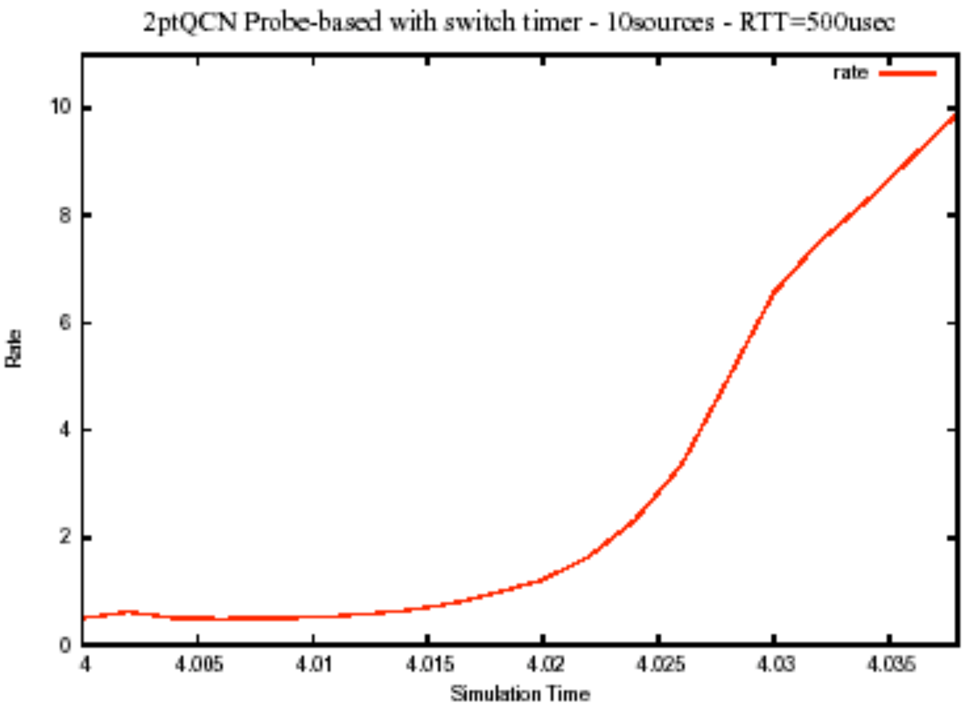
Simulations: OG Hotspot

- 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 = 500 usec
  - Buffer size = 100 pkts; Qeq = 22
  - Drift timer disabled



Bdwdth Recovery



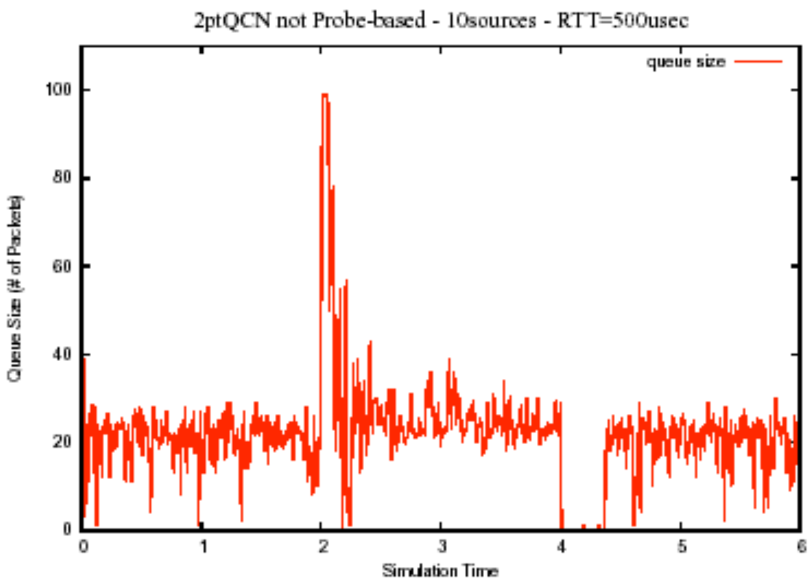


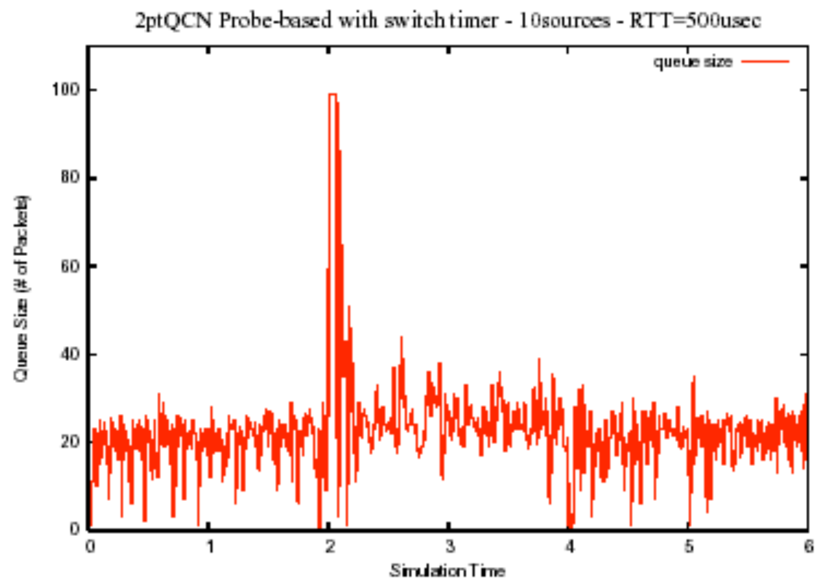
Time improvement

-- 350+ msces down to 38 msecs

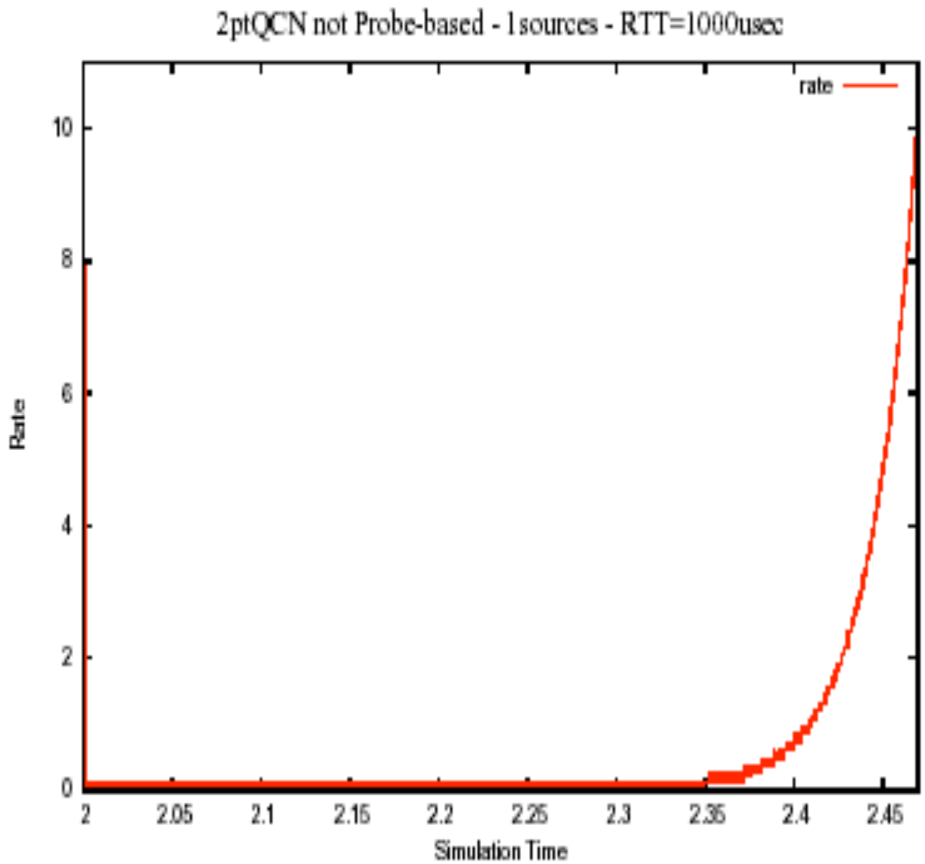
-- 0 false alarms

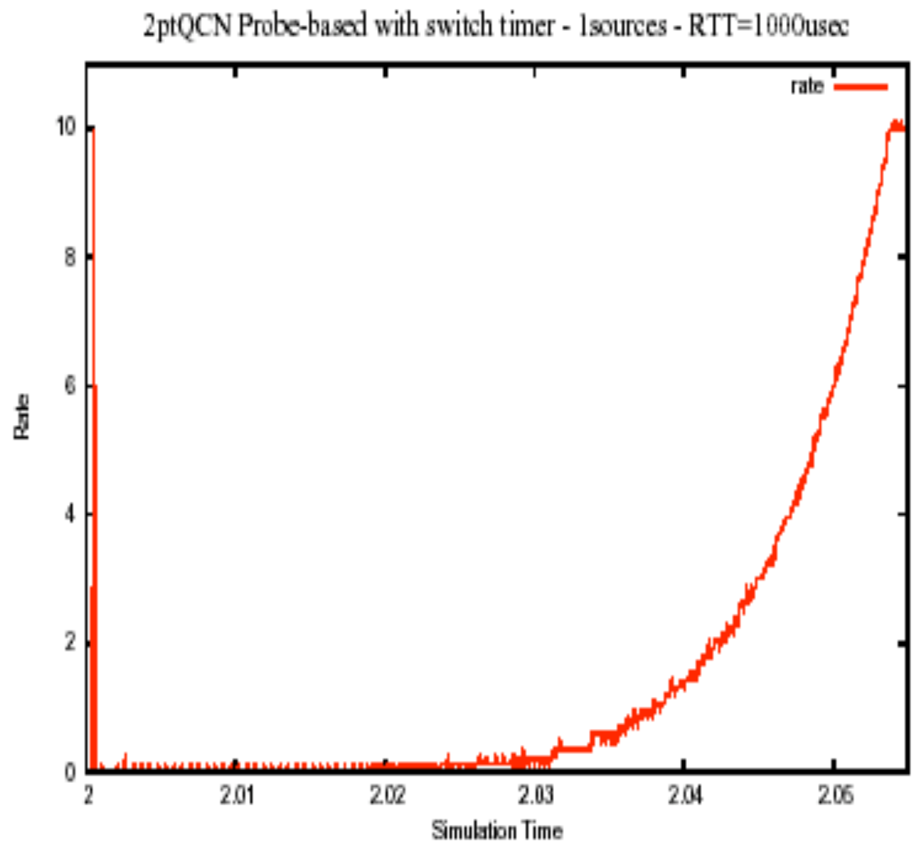
Queue size: No effect on stability





Bdwdth Recovery: 1 source, 1 msec RTT
Starting rate 10 Mbps





Time improvement

-- 450+ msecs to 55 msecs

Conclusions

- The SONAR idea is a simple way of discovering available bandwidth without compromising stability
  - Clearly, we have been very conservative in grabbing available bandwidth; we can make the recovery time less than 20-25 msecs

- We have several other things to present/discuss
  - Large RTT and large N simulations of SONAR for stability checked
  - Further simplification of the QCN algorithm (no drift timer)
  - Dealing with multipathing in a simple fashion