QCN: Stably improving transient response

Abdul Kabbani, Rong Pan,
Balaji Prabhakar and Mick Seaman

Outline

- This is a presentation about
  - A simple, unified redefinition of QCN
  - Two methods for grabbing available bandwidth
    1. SONAR
    2. Active paths
       - Does not increase gain
       - Probes the path
       - Very quickly recovers bandwidth
       - Simplifies 2-QCN
- We will also discuss the pros-cons of the different methods for grabbing available bandwidth

Redefine QCN

- It is convenient and simpler to redefine QCN 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 TR and CR

- Equilibrium
  - 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 goes to 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 + R_i Mbps for AI, or TR <--- TR + R_i*cycle_cnt Mbps for HAI
    - CR <--- (CR+TR)/2

- Downward Transience: Target Rate Reduction
  - At the end of the FR1, if TR > 10*CR
    - Then TR <--- TR/8; CR <--- (TR+CR)/2

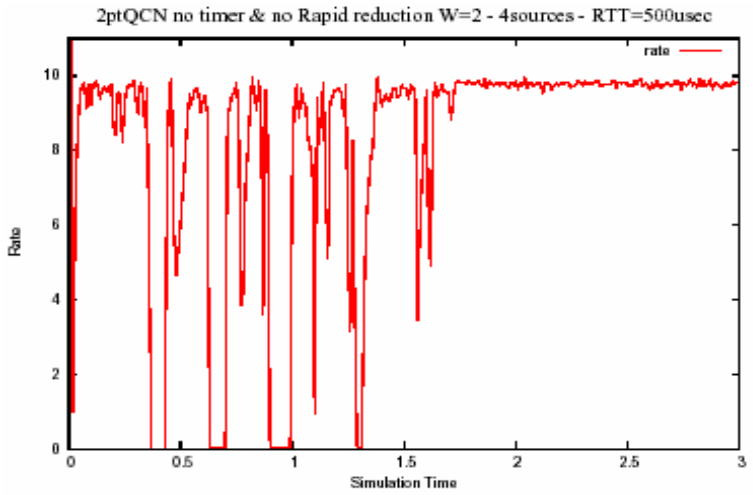
- We will consider ***upward transience*** later

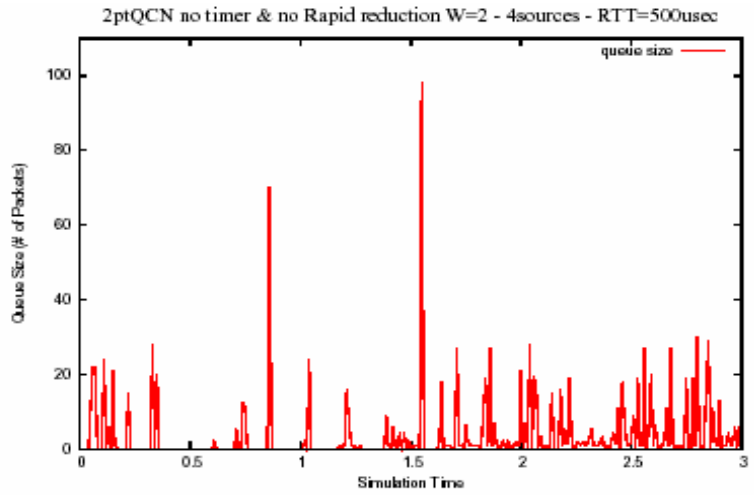
Downward Transience

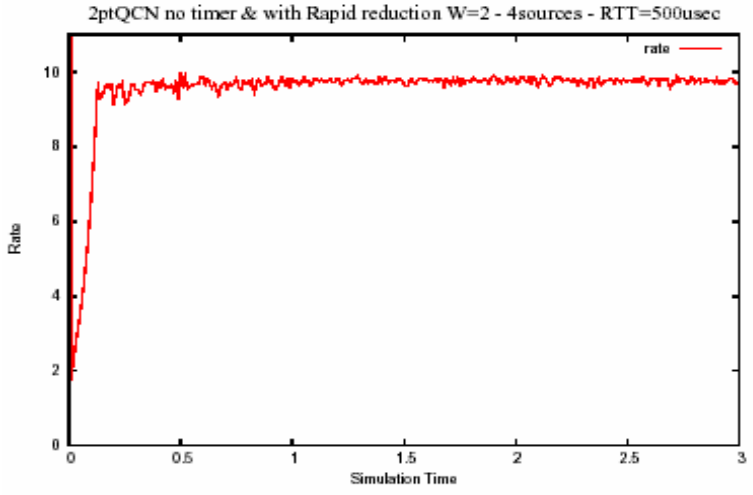
- When a severe bottleneck appears, or when PAUSE is asserted and a saturation tree begins to form, it is important to settle RLs quickly to a lower rate

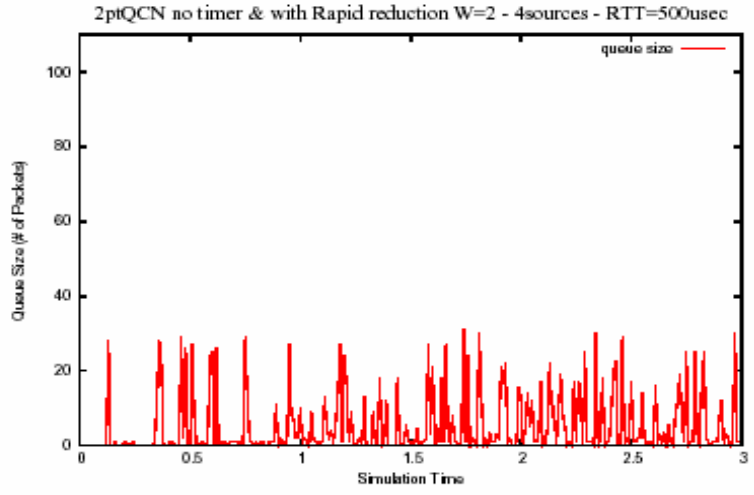
- By reducing the downward transience time
  - Packet drops or long transients occurring during congestion episodes are highly reduced
  - The effect is most noticeable when the RTT is large, because bursty dings are quite likely in this case, and the RLs take a long time to get into steady-state

Improvement in transient time due to Target Rate Reduction









Upward Transience

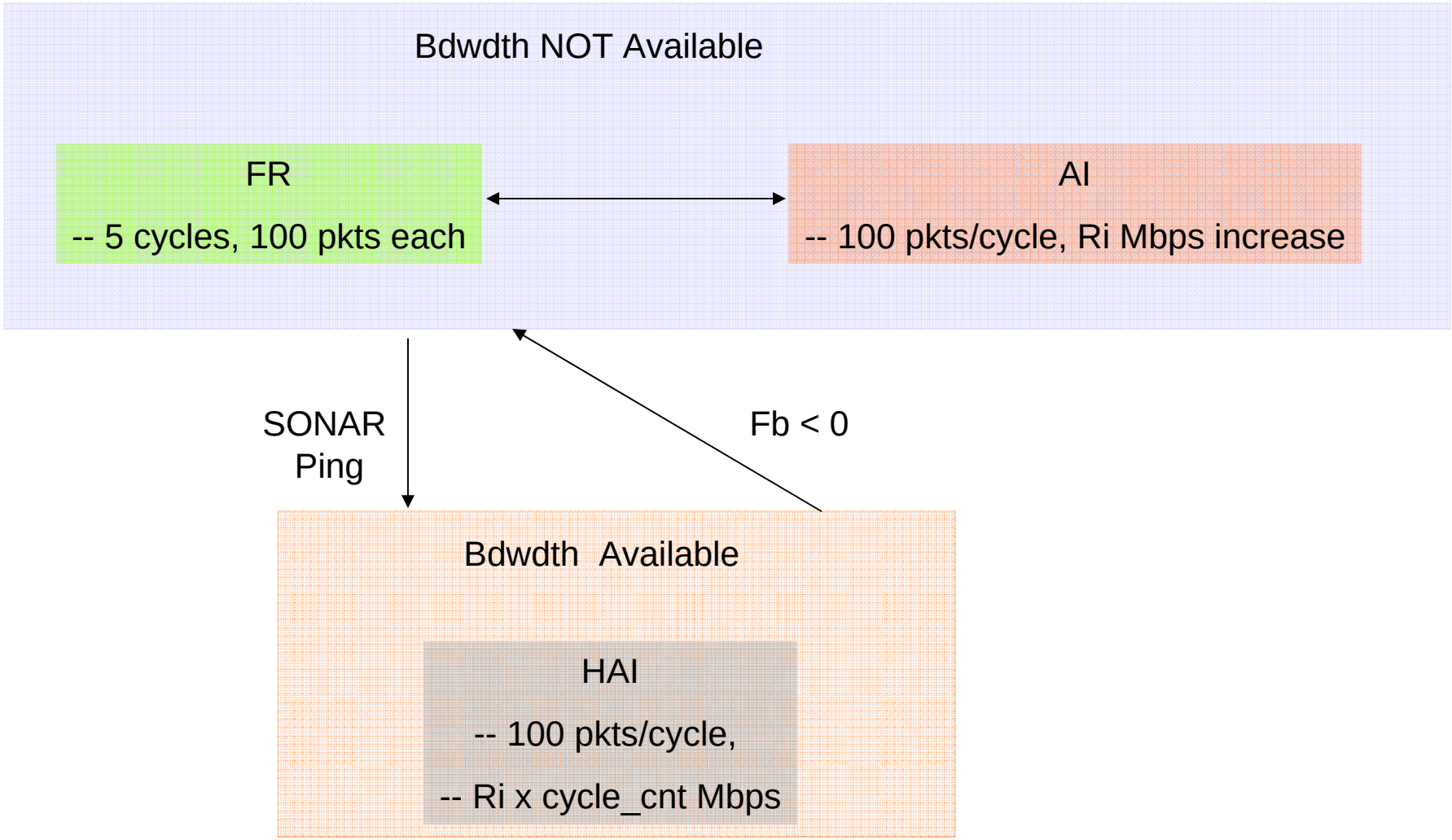
- For stable recovery of available bandwidth, we need the following
  - A stable indication of “available bandwidth”
    - We cannot increase transmission rate based on instantaneous value of Fb > 0 (or Q_{off} or Q_{delta})
    - Because they will all swing from positive to negative as RTT increases
  - To probe the path

- Two methods
  - SONAR
  - Path-based congestion indication
    - Path-based method is preferred, but we’ll see SONAR briefly first

SONAR

- The main idea
  - 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)

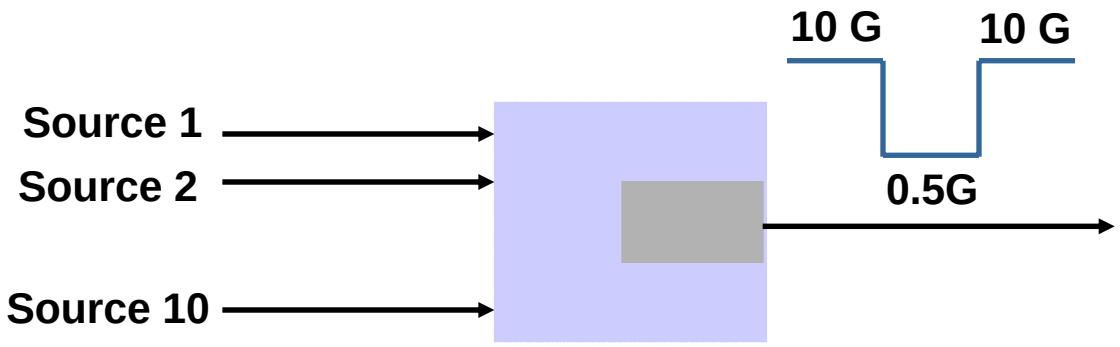
- The operation
  - The RL goes to the WP state whenever it receives an Fb<0 signal
  - If the WP timer expires, the next pkt sent by RL is a “special pkt”
    - Spl pkt == data packet with 1 bit set to indicate special
  - After Spl pkt is launched, RL goes to WE
  - If RL hears an echo for the SP
    - 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

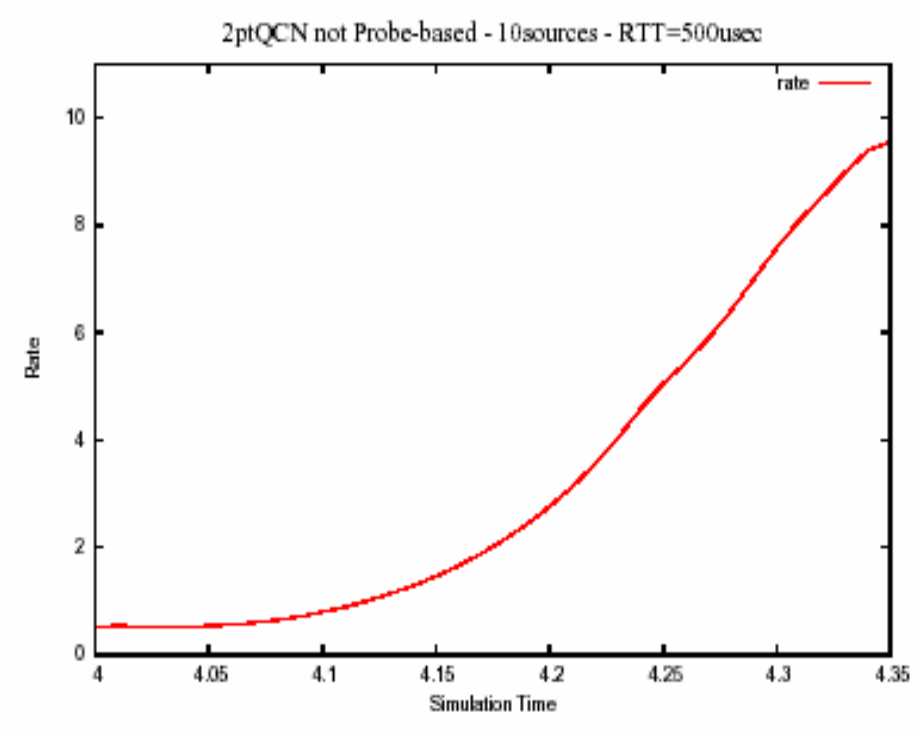
- Bdwdth available if queue-length < 6 pkts (say) for at least 10 msecs
  - Q_len < Q_eq (= 22 pkts) means input rate < output rate
  - So every time Q_len < 6 pkts, swith starts congestion timer
  - If timer expires, bdwdth available; else timer restarted when Q_len < 6 pkts again

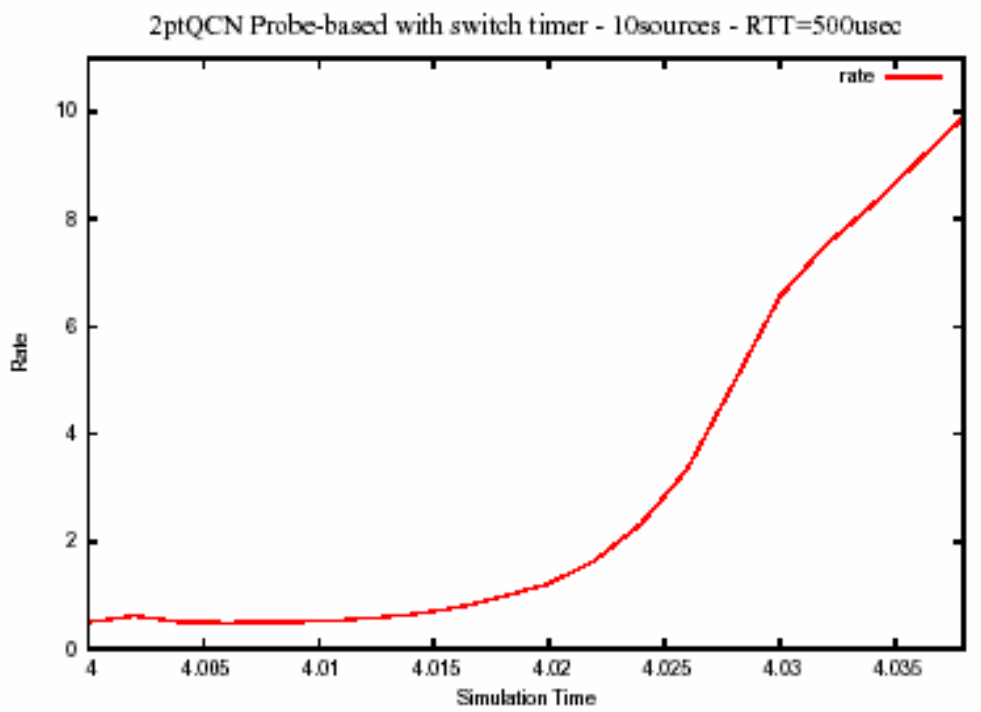
Simulations: OG Hotspot

- 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



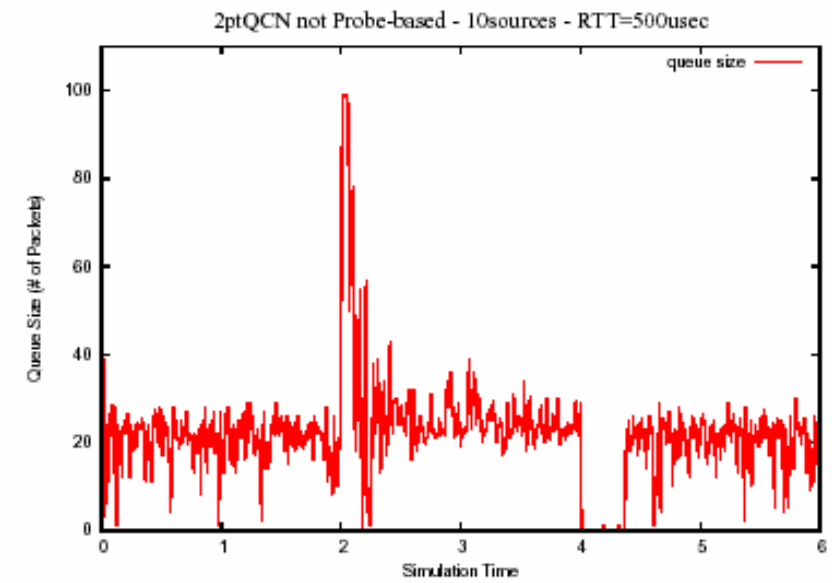


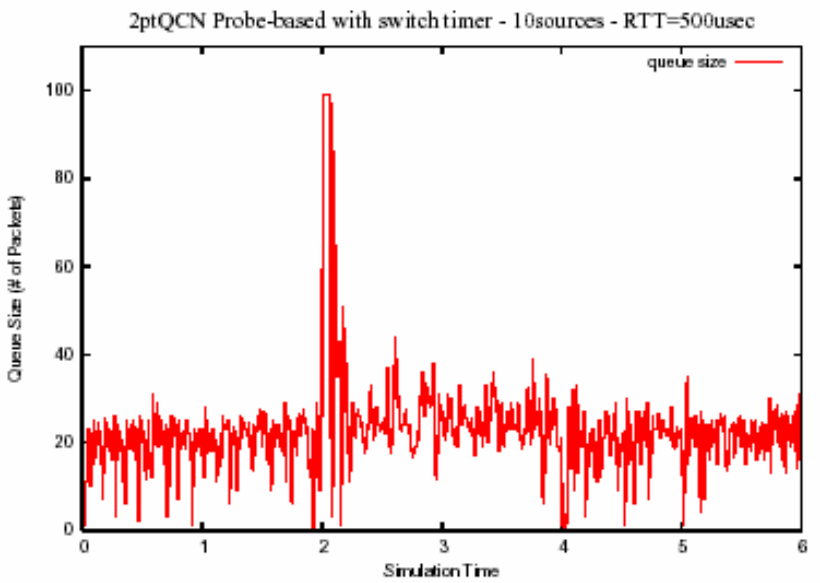
Time improvement

-- 350+ msces down to 38 msecs

-- 0 false alarms

Queue size: No effect on stability





Summary of SONAR

- The SONAR idea is a simple way of discovering available bandwidth without compromising stability
- However, QCN-SONAR has a drawback of not exploring all the paths in a multipath scenario, and moreover
  - Ping messages could get stuck in paused queues (at least something special might need to be don to deal with this)
  - It might send back-to-back SONAR pings to a switch
- This leads us to the next approach, which improves on SONAR

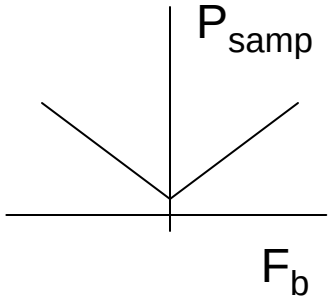
Method 2: Path-based congestion notices

- 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 a trade-off, the method is fundamentally correct

- Recall: 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

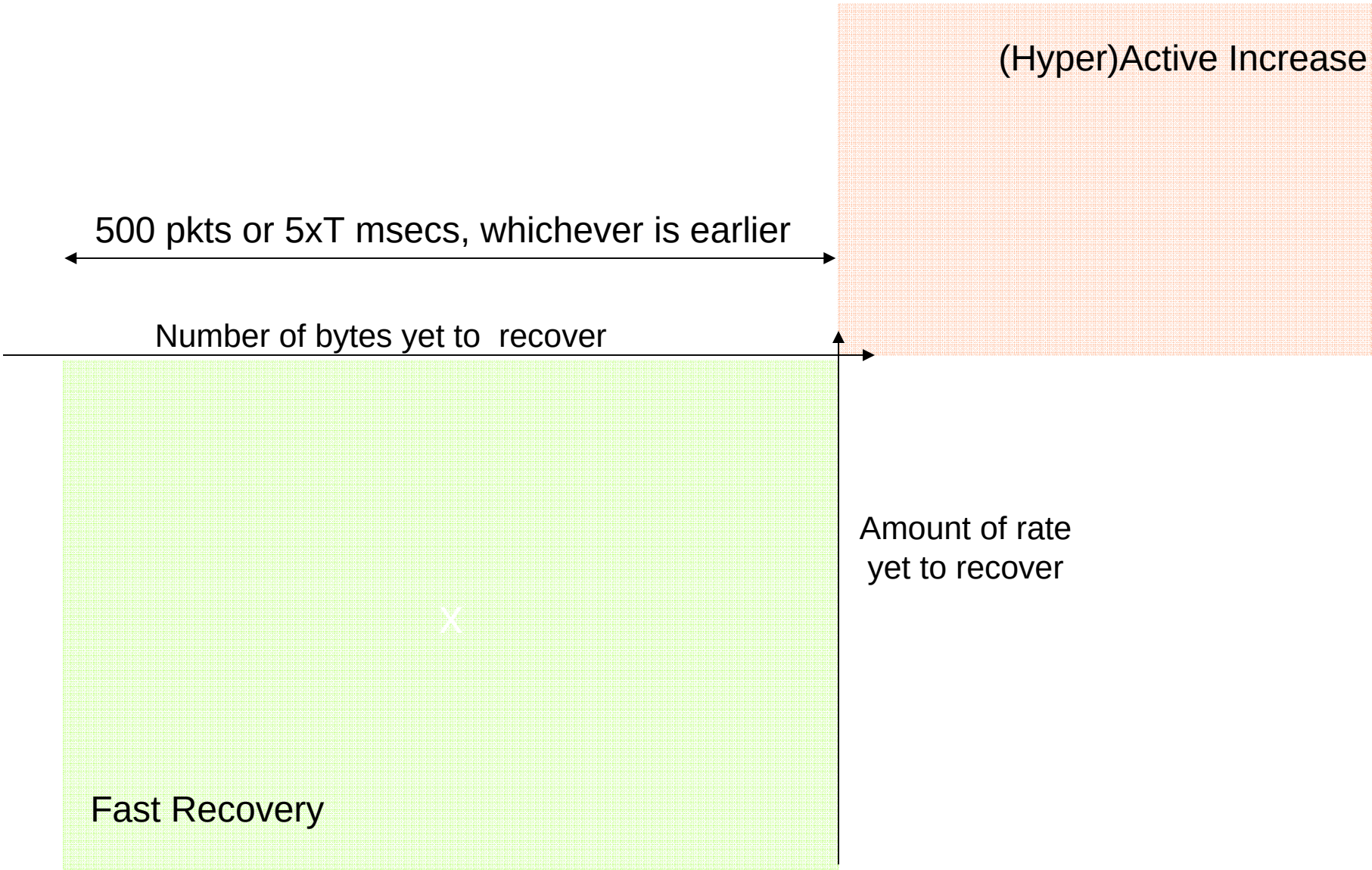
Method 2: The Details

- 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

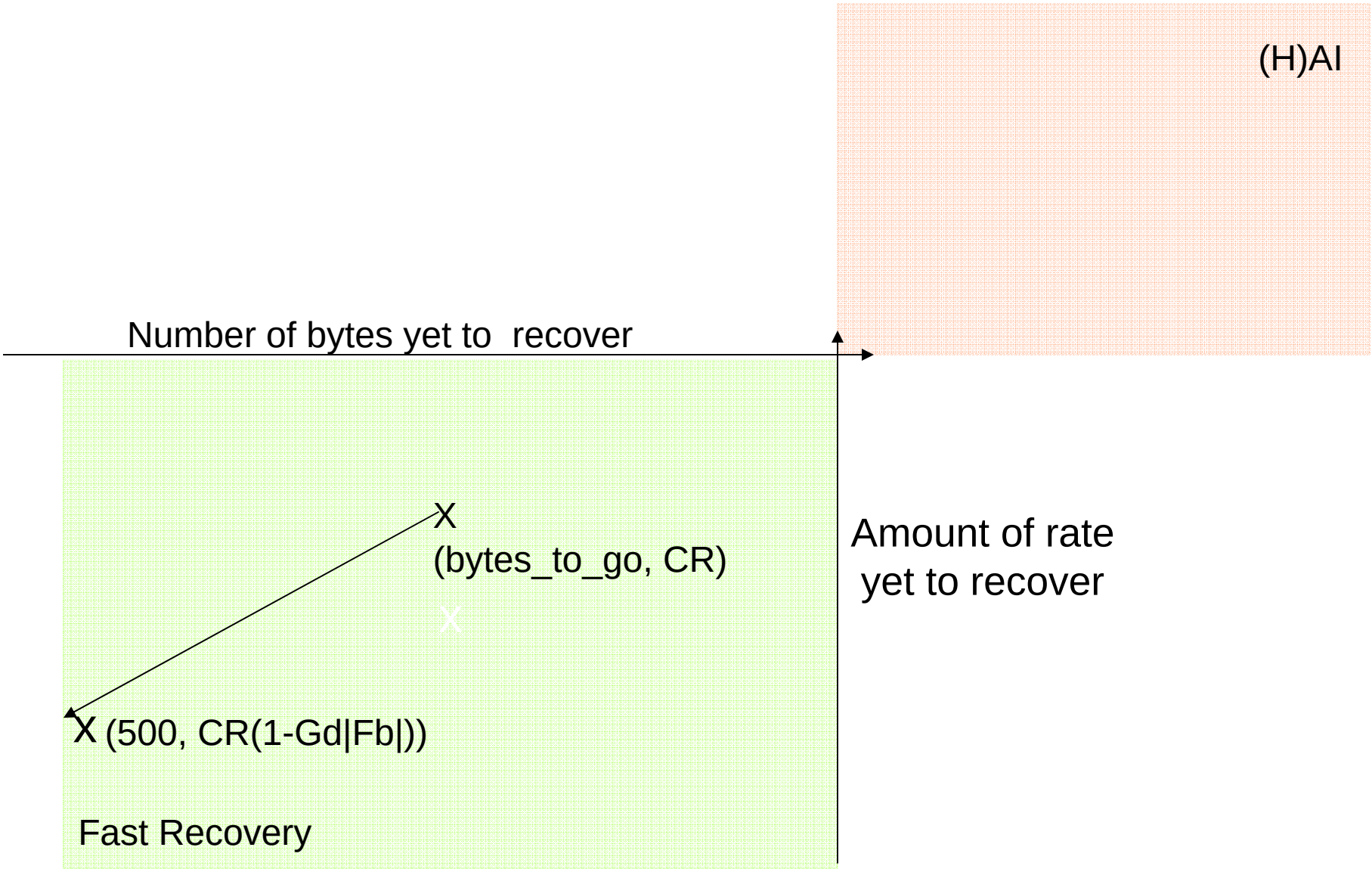


- At the RL
  - There is a timer which runs for T msecs
    - Timer is reset every time an Fb<0 or Fb99 message is received
  - When Fb<0 signal is received, same actions as before
  - When Fb99 signal is received
    - TR and CR remain unchanged
    - Go back one cycle in FR
  - When timer or byte-counter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

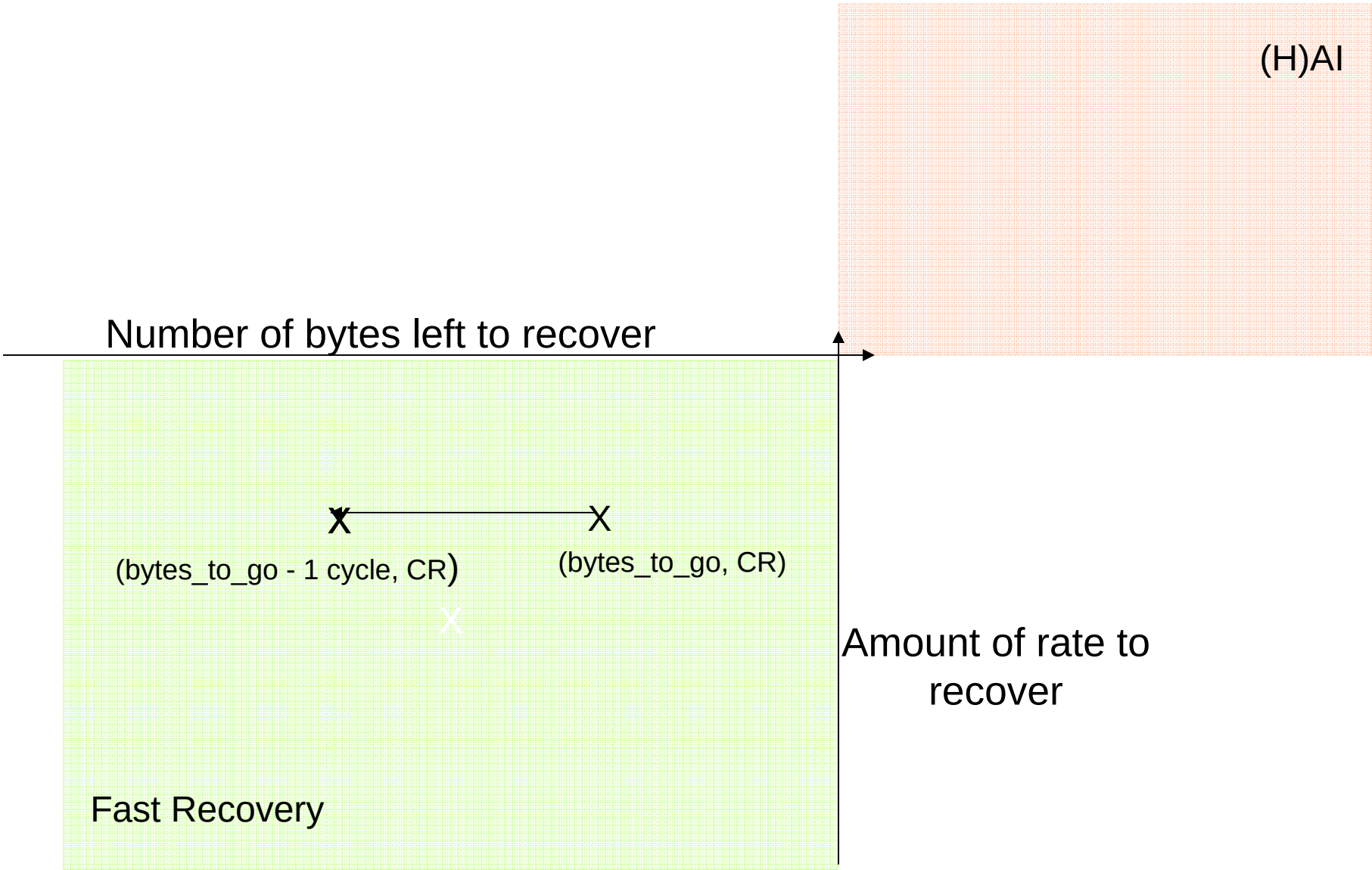
A useful mental image



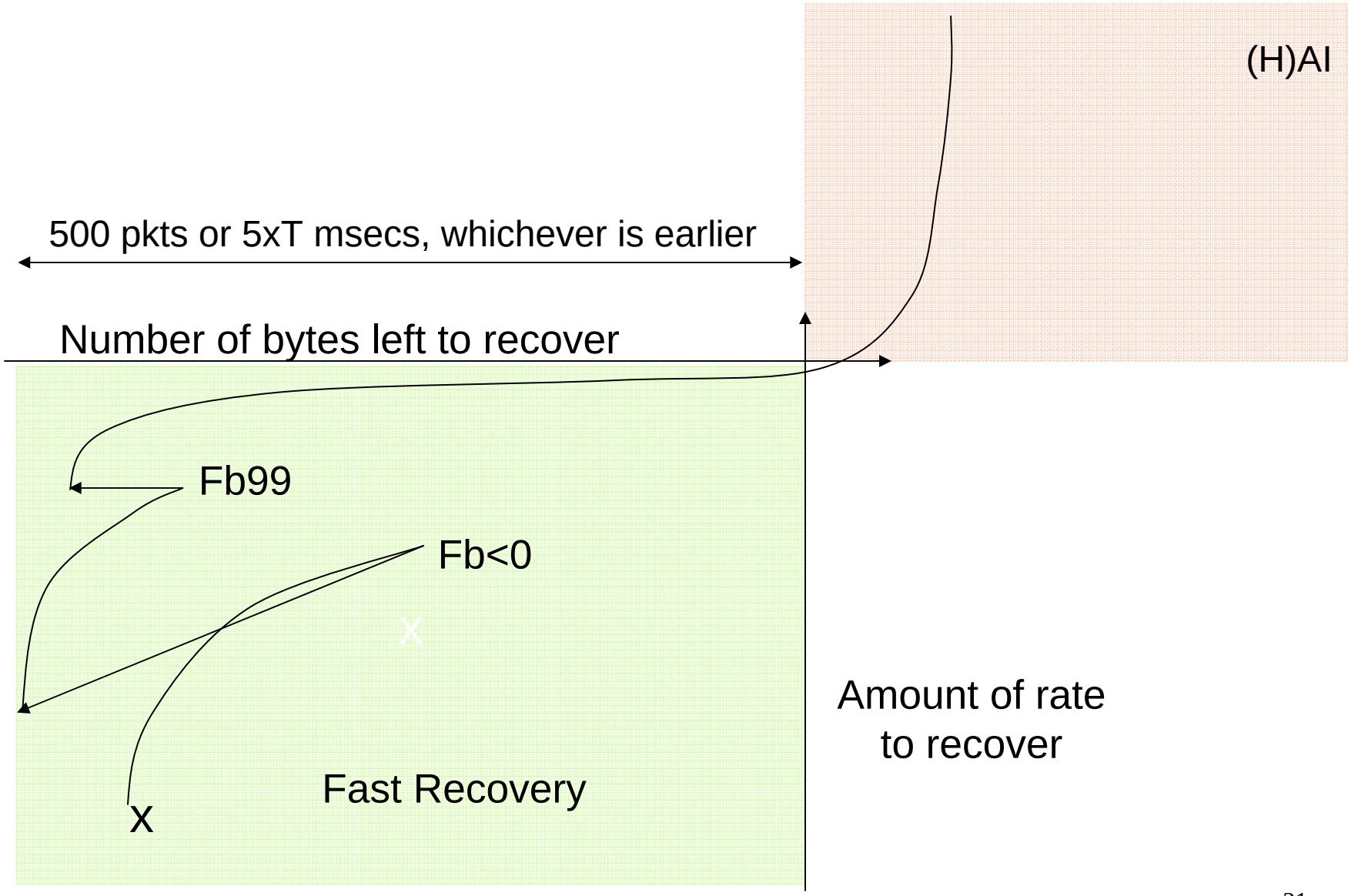
When Fb<0 signal arrives



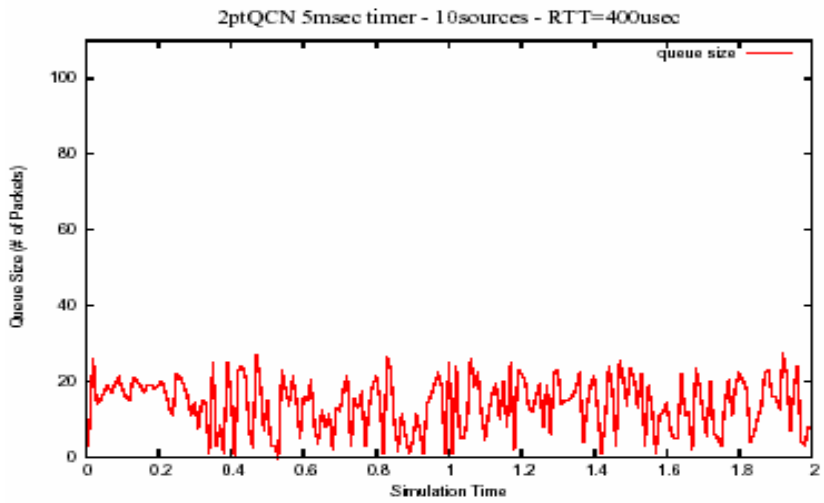
When Fb99 signal arrives

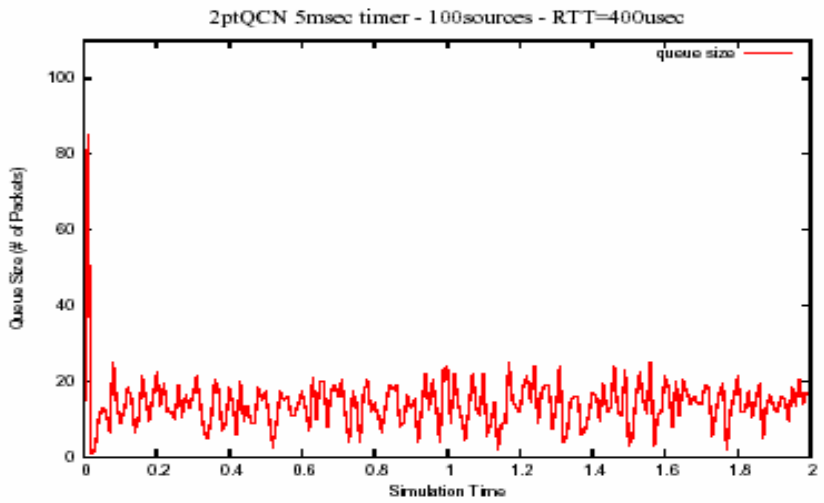


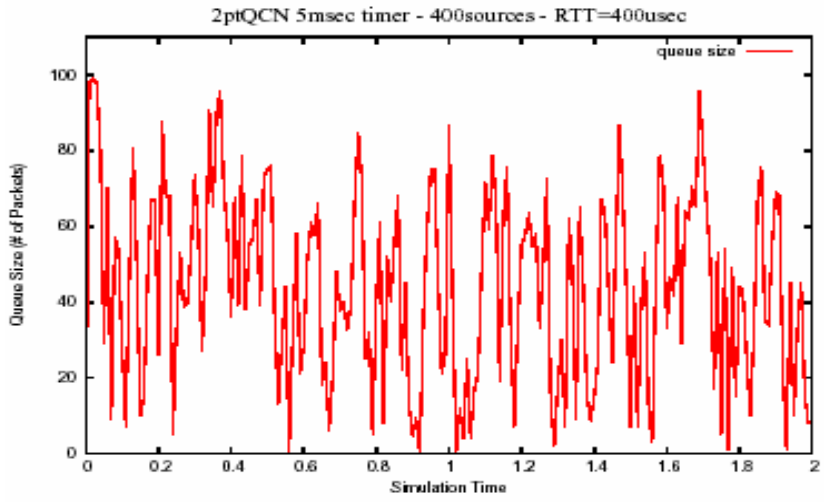
An evolution in phase space

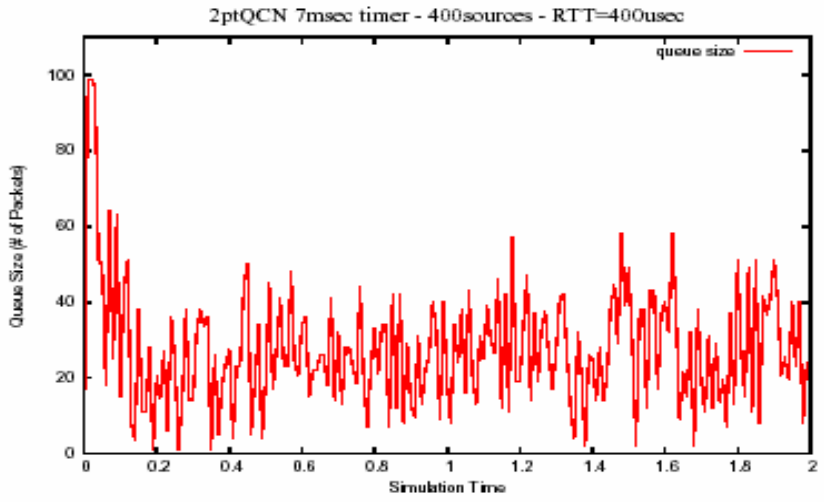


Simulations: Stability with Method 2

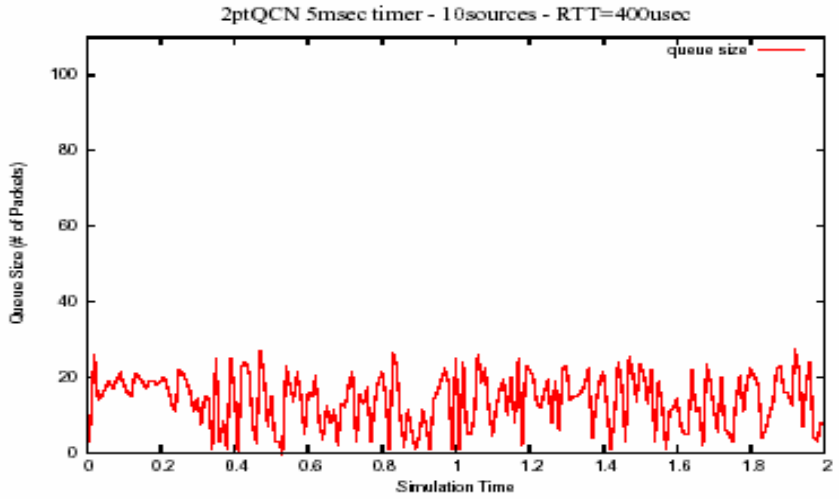


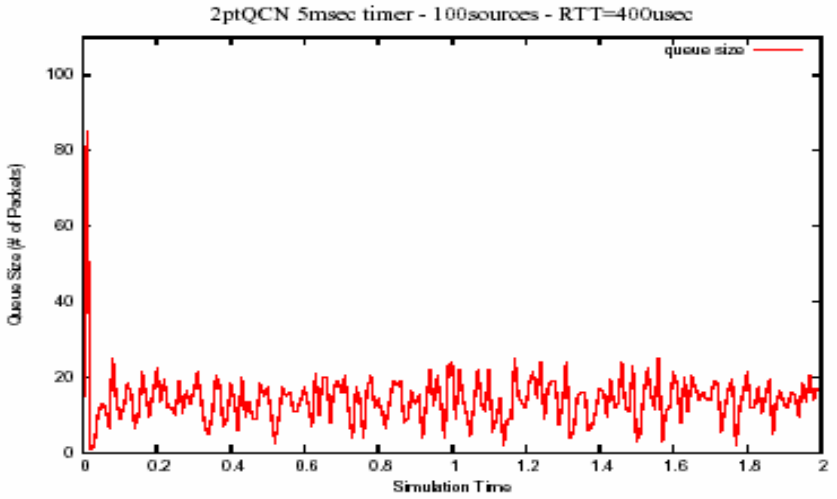




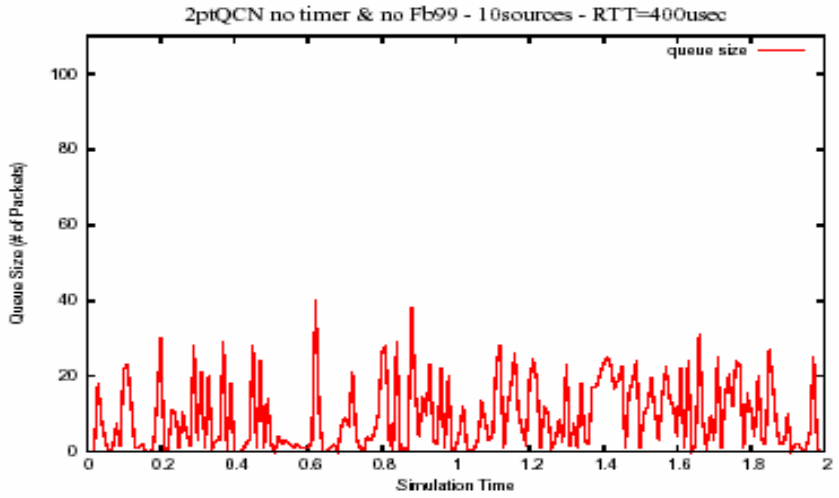


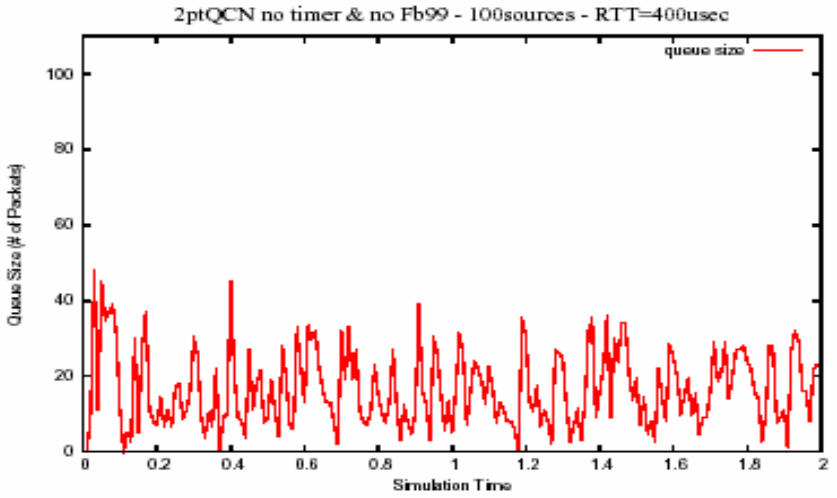
Stability improves due to cycle-stretching when Fb99 is received



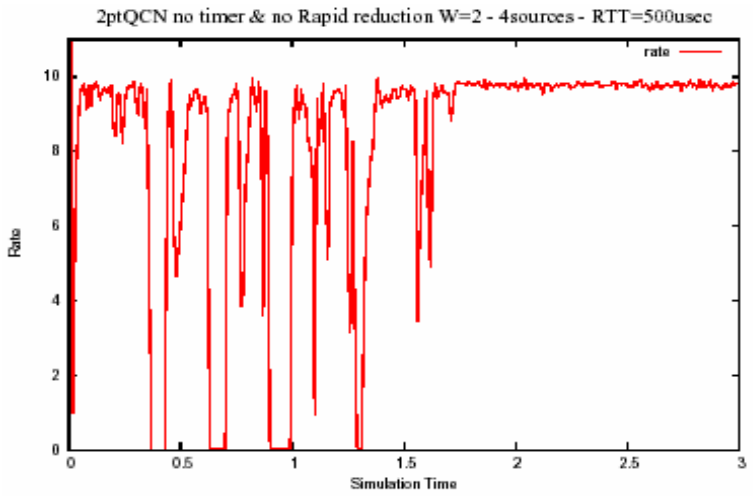


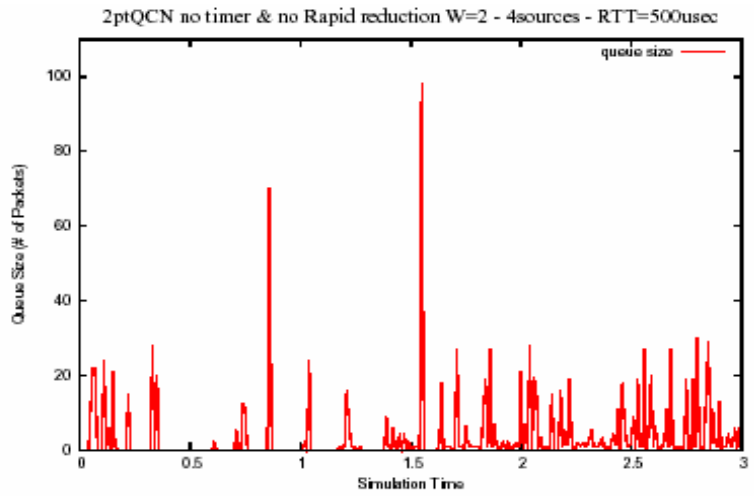
This is better

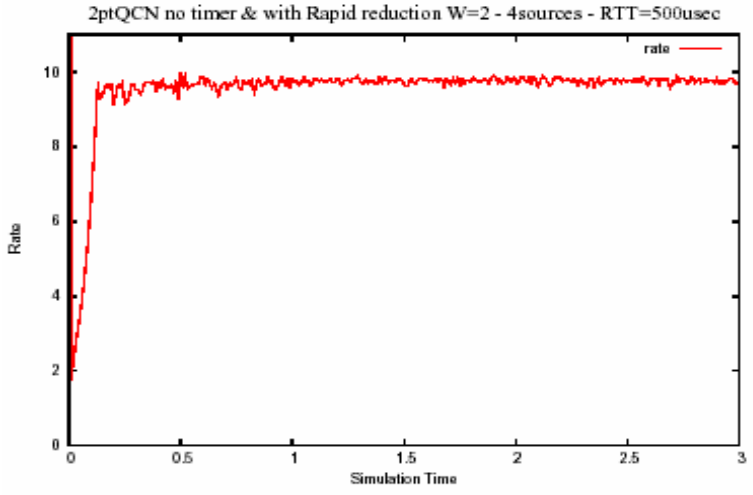


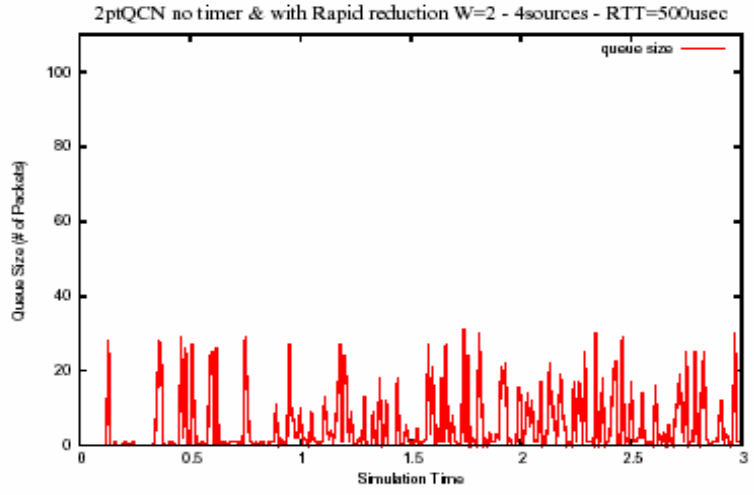


Improvement in transient time due to Target Rate Reduction (recall slide 6)

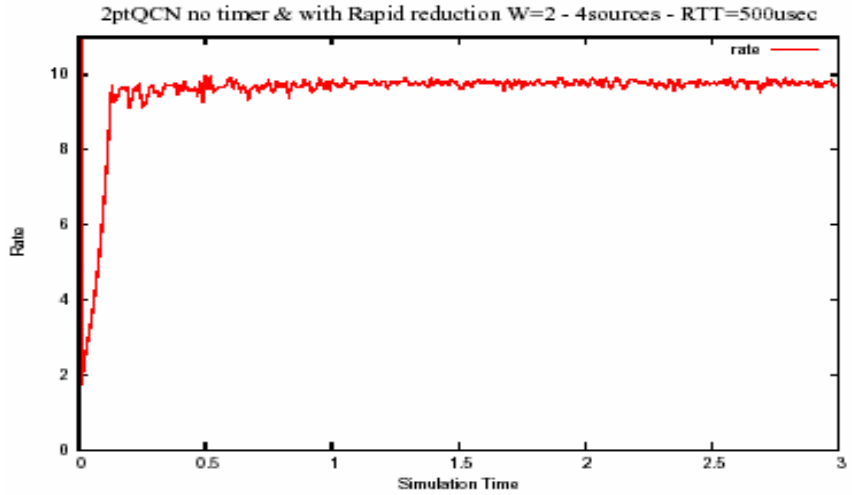


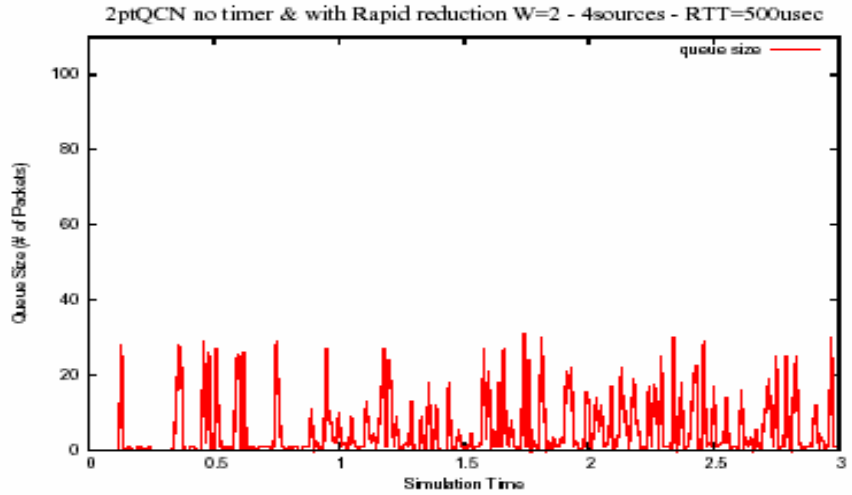


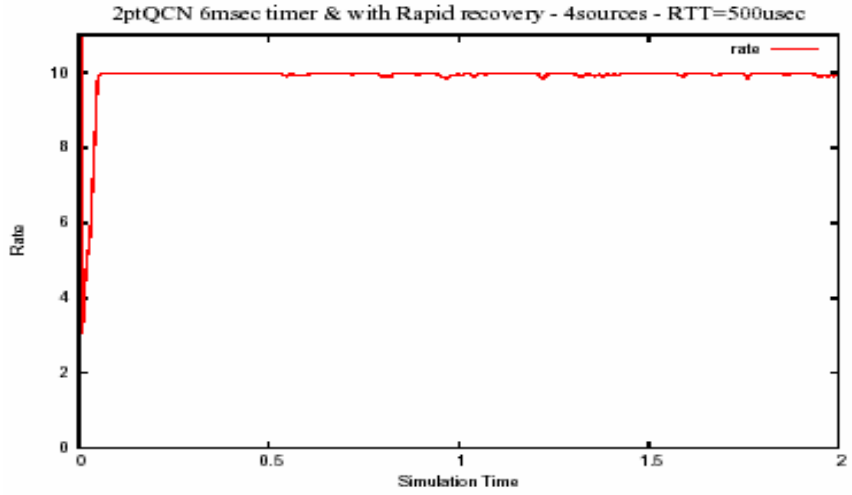


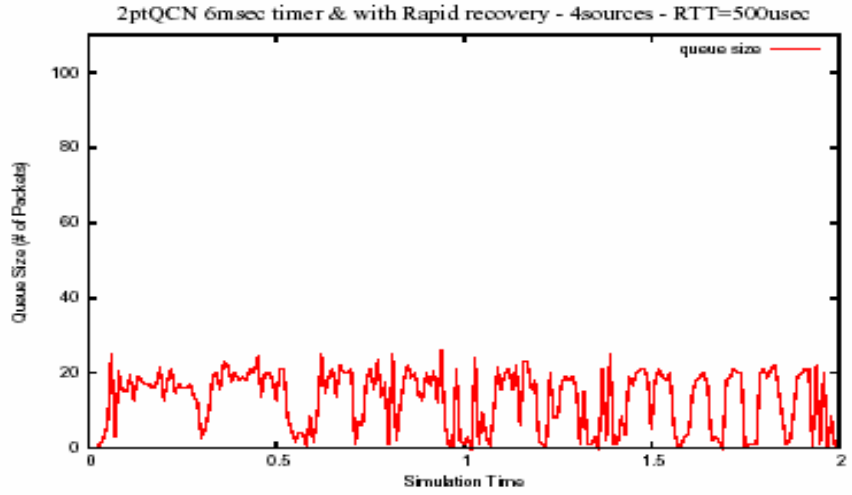


Improvement in transient time due to Target Rate Reduction and Timer



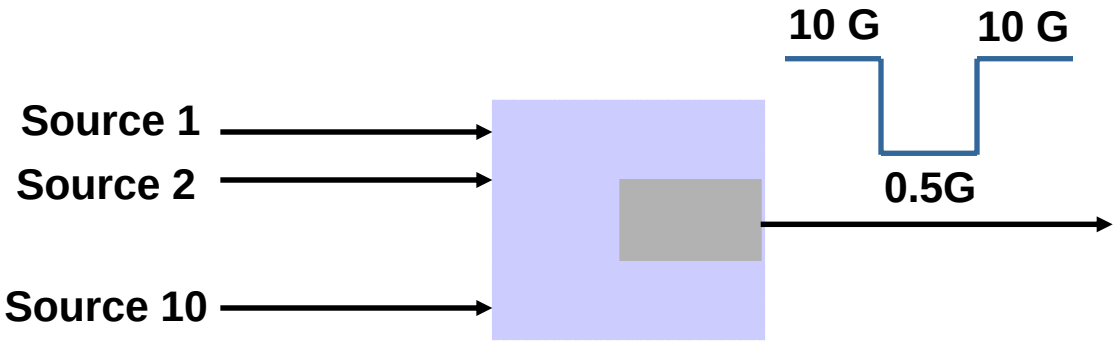




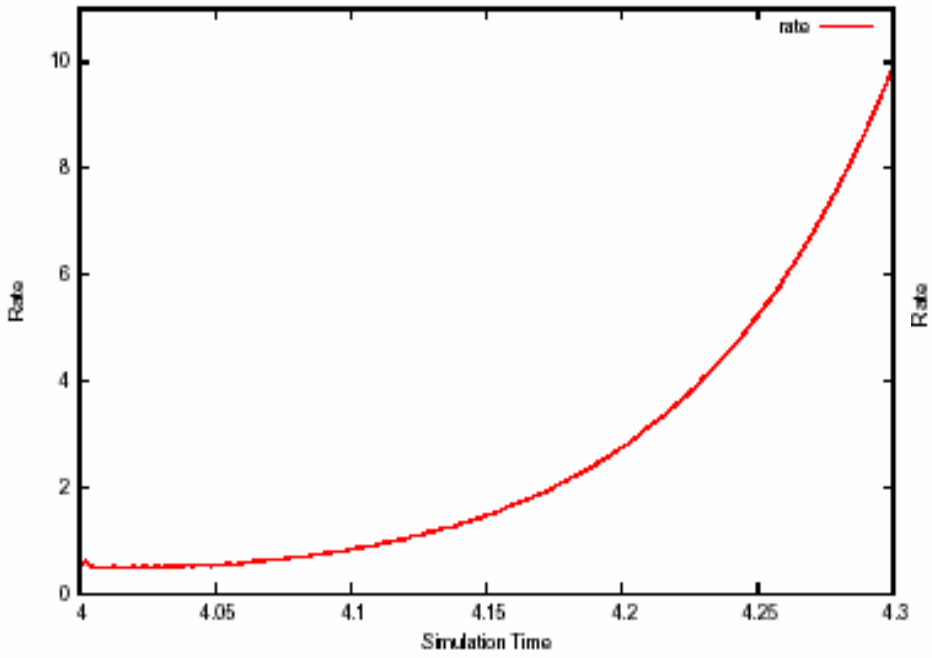


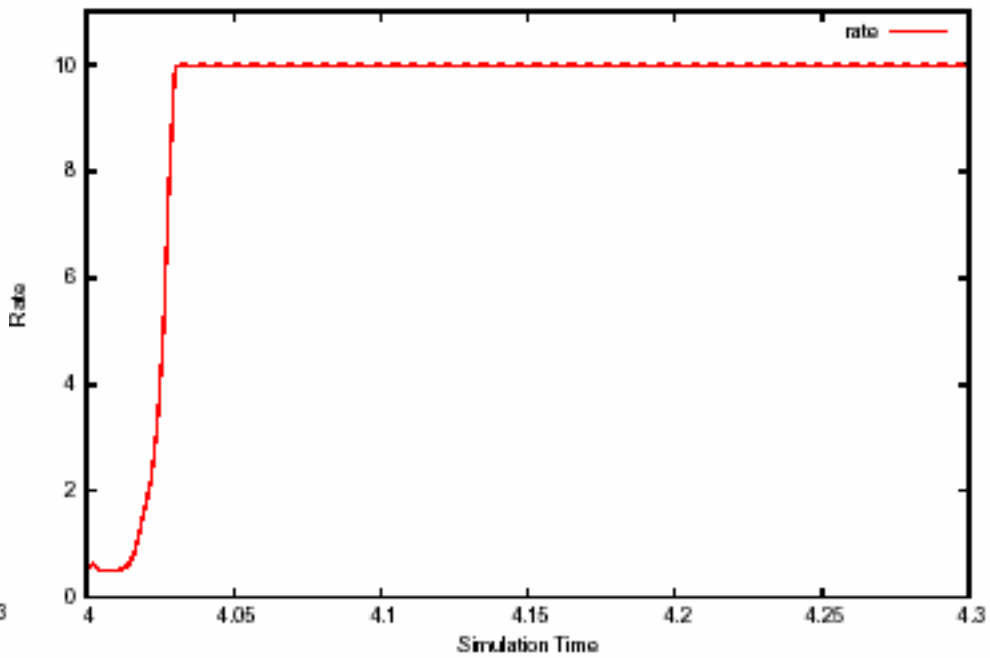
Recovery time: OG Hotspot

- 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

In summary

- We have a simple, unified redefinition of QCN
  - Uses the TR--CR formalism
  - Stability does not require any modifications or parameter changes (set w=2)
  - Deals with upward and downward transience
    - Congestion transience is shortened
    - Recovery times are improved
    - Multipathing is also dealt with, since all paths report BA status
    - Does not affect stability

- Note that the above attributes are also true for the Fb-hat approach
  - The difference is that it is all at the source
  - There was no timer, so its recovery times were poor

- We (Berk Atikoglu and Abdul Kabbani) are also beginning to get Omnet going

CN Mantras

- Short buffers are not a problem
  - This is where multibit Fb and springiness of QCN help
- Swinging queues (during transience and with large RTT) are ok
  - This is consistent behavior for control schemes
- Keep a simple control loop
  - Stay with gain parameters once they are chosen
  - Detect transient conditions quickly and adapt operation
    - Better not to adapt gains dynamically, environment likely to change quickly
- Look at flow completion time (FCT): that is what matters eventually