The Origin, Evolution and Current Status of QCN

Berk Atikoglu, Abdul Kabbani, Rong Pan,
Balaji Prabhakar, Mick Seaman

Overview

- Review the development and current status of QCN
  - Stability, responsiveness, robustness
  - The role of BIC: byte-counter and timer
  - Convergence
- Understanding the role of gain parameters

QCN: Evolution Summary

- Goal: To develop a simple, stable, responsive, robust CM scheme
  - ***Robust*** means there are no tunable parameters; all parameters fixed regardless of number of sources (N) or round trip time (RTT)

- We began with BCN
  - First, just quantized it and removed the RLT
  - Later, rediscovered BIC and hence improved the self-increase feature
  - This is pretty much what we know as 2-pt QCN
  - We obtained a stable scheme

- Response time
  - Since this is important, tried various things
    - 3-pt QCN, Fb-hat, SONAR, Fb99
  - 3-pt QCN impeded by multipath; others either had poor response time (Fb-hat) or were hard to make universally stable (robust)

- Finally: used a timer at the source in conjunction with the byte-counter, and put HAI in series with AI to get stability + good response time + robustness

QCN: Evolution Timeline

Feb '07

Begin with BCN
-- Quantize
-- Remove RLT
Option 2
-- Rediscover BIC
-- Define basic 2-QCN

Mar '07

2- and 3-QCN
-- FR, AI, EFR, …

May '07

Multipathing
-- Shared RLs are a problem for 3-QCN

Jul '07

2-QCN, stability with N
-- Simulations and MC model show stability
-- Poor response time

Sep '07

Fb-hat
-- Better response
-- Stable with N, RTT
-- Response time depends on current rate

Nov '07

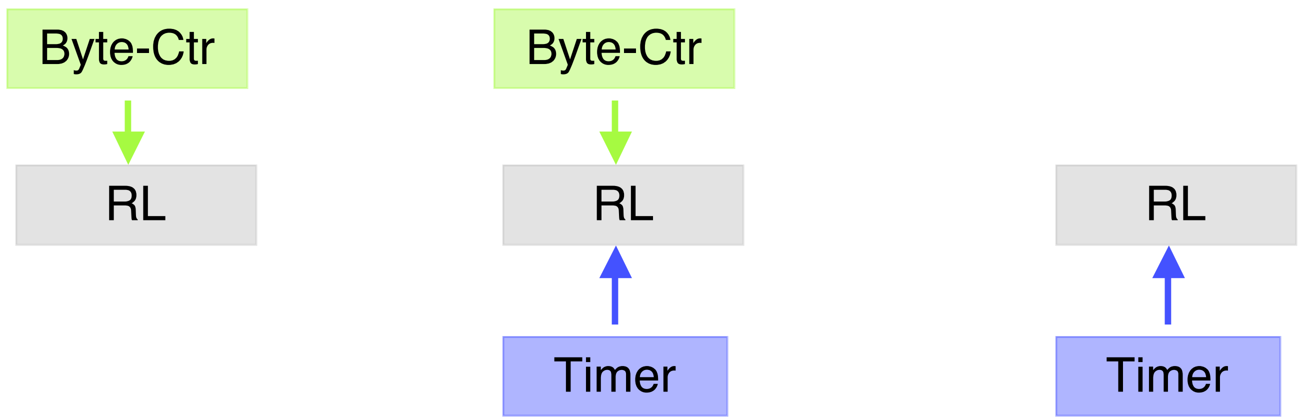
SONAR, Fb99
-- Response independent of current rate
-- Requires detecting available bandwidth accurately; hence not robust

Current

2-QCN with Serial HAI
-- **Stable** due to byte-ctr
-- **Responsive** due to timer
-- **Works in multipath** environment
-- Performance is **robust** with N and RTT

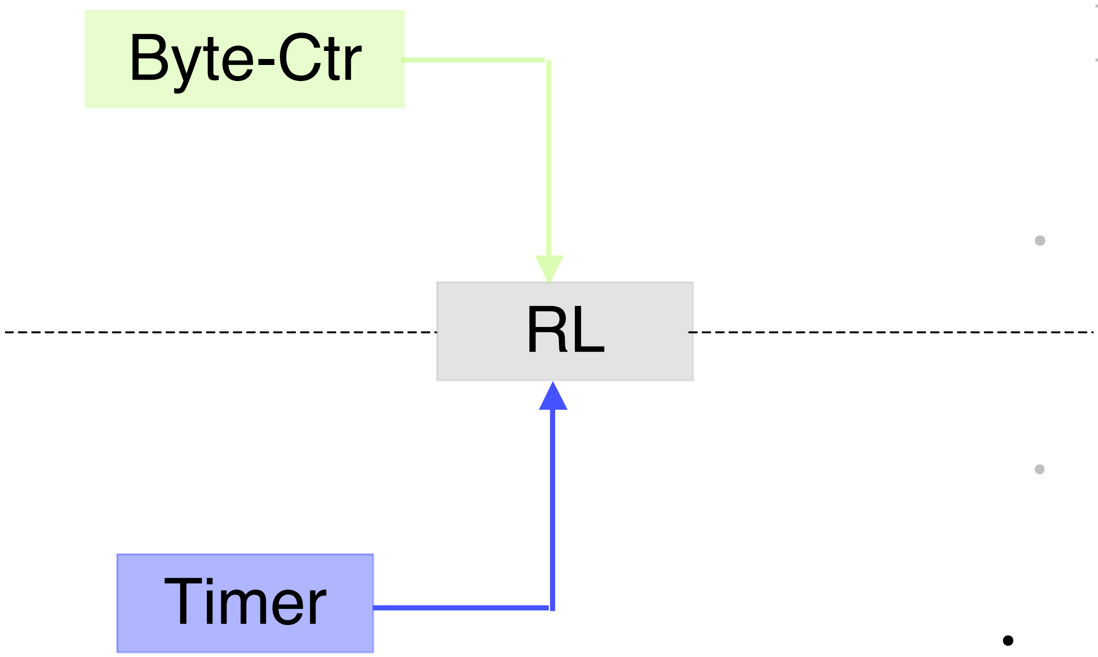
A Synthesis

- Initial version of 2-QCN just had the byte-ctr
- Now, we have a byte-ctr ***and*** a timer
- We can also consider using just the timer



- Thus, the byte-ctr and the timer just provide “events of increase”
  - At these events we use either FR or AI, as appropriate
- NOTE: All three versions are QCN because they all have BIC in common
- We have already seen how the byte-ctr version performs
  - Let us see what the timer-only version means
  - This exercise is for understanding the scheme better, QCN will have ***both*** the timer and the byte-counter

Timer-only QCN



- Byte-Counter
  - 5 cycles of FR (150KB per cycle)
  - AI cycles afterwards (75KB per cycle)
  - Fb < 0 sends timer to FR
- RL
  - In FR if ***both*** byte-ctr and timer in FR
  - In AI if ***only one of*** byte-ctr or timer in AI
  - In HAI if ***both*** byte-ctr and timer in AI
- Note: RL goes to HAI only after 500 pkts have been sent
- Timer
  - 5 cycles of FR (T msec per cycle)
  - AI cycles afterwards (T/2 msec/cycle)
  - Fb < 0 sends timer to FR

Timer-only QCN = ECM+

- The main issue is: choosing the timer value
  - Too small makes it aggressive; too large makes it sluggish
  - Essentially, need the “self-clocking” feature of the byte-counter

- Adaptive timer: a simple idea suggested by Berk Atikoglu
  - Suppose current timer value is T
  - If timer expires, make next timer value T- a or T.c, where c < 1
  - If dinged before timer expires, make next timer value T + b or T.d, where d > 1

- If we now look at the timer-only version, it is not that different from
  - Taking ECM
    - Ignoring Fb > 0 values
    - Using the drift timer to do all the self-increase as above
  - If we call this version of ECM as, say ECM+, then we see the following major point

- The effort of developing QCN has been to shift BCN from an AIMD scheme to a BIC-based scheme with good stability (via byte-ctr) and responsiveness (via timer)
  - This is how I see the convergence as having occurred

Robustness

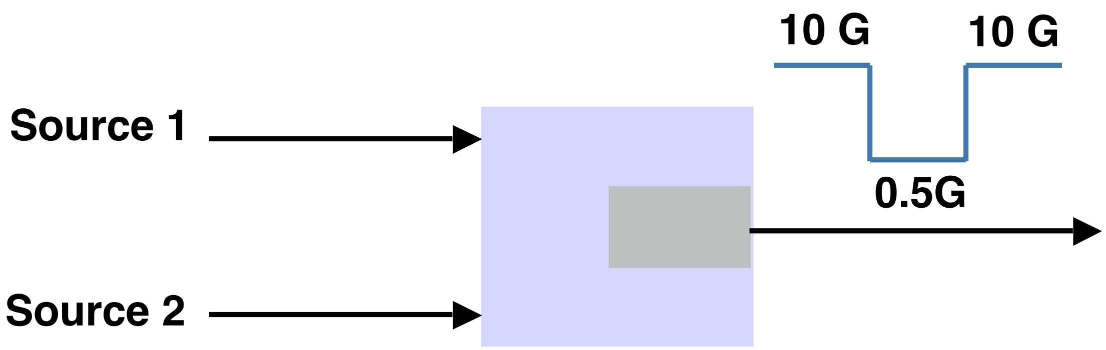
- Worth understanding this some more…

- AIMD schemes like TCP don't possess it; feedback compensation needed
  - Negative side effect: Choice of parameters which stabilize scheme for long RTT make it sluggish
  - As we shall see, this is also true for BCN (which is AIMD)

- However, BIC and QCN are robust with respect to N and RTT

Simulations

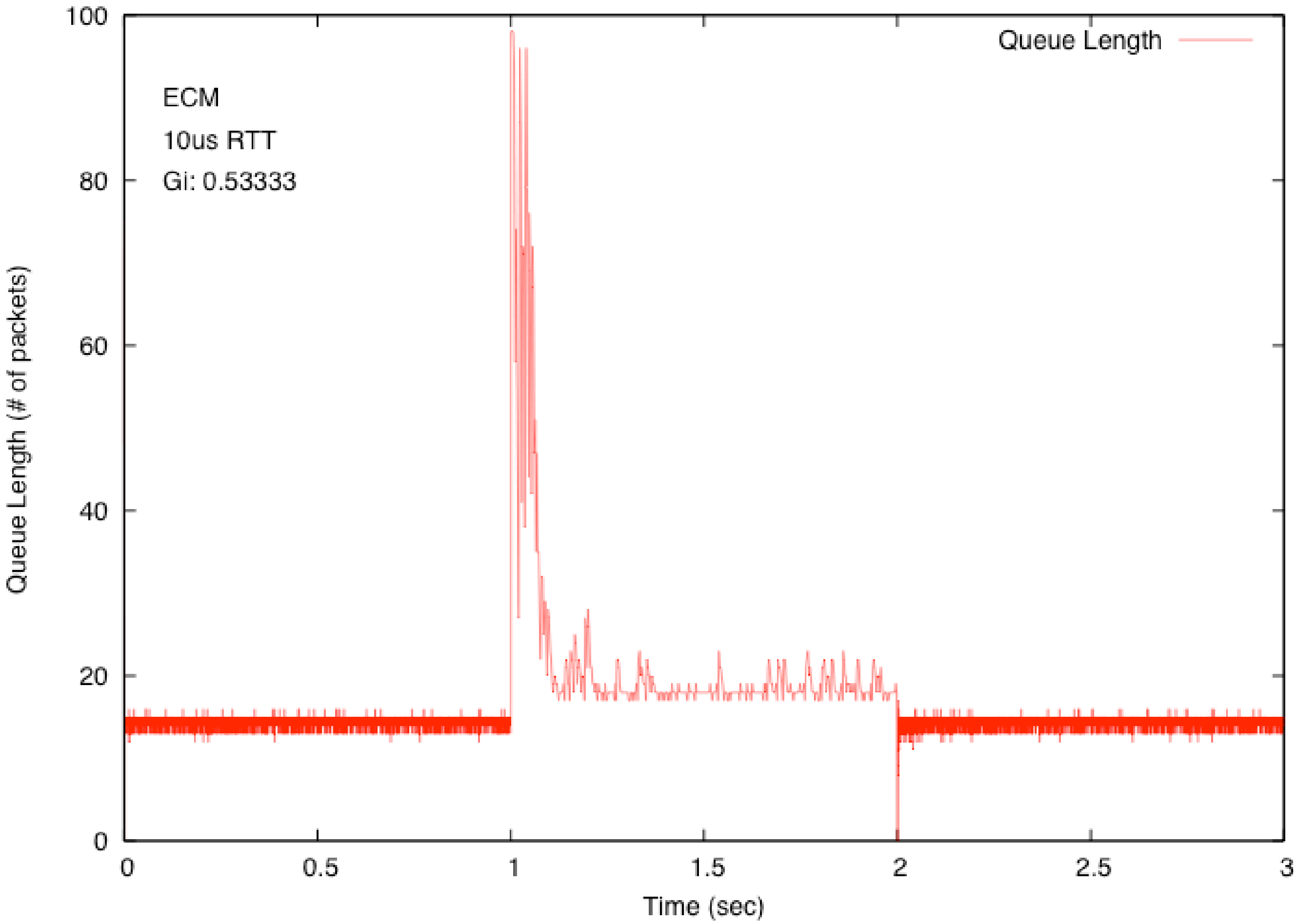
- Consider the Baseline scenario
  - Single link, 2 sources
  - OG hotspot; hotspot severity: 0.5G; hotspot duration 1s
  - Vary RTT: 10 us, 200 us
  - Study: behavior of QCN and BCN: stability and response time



Simulation Parameters

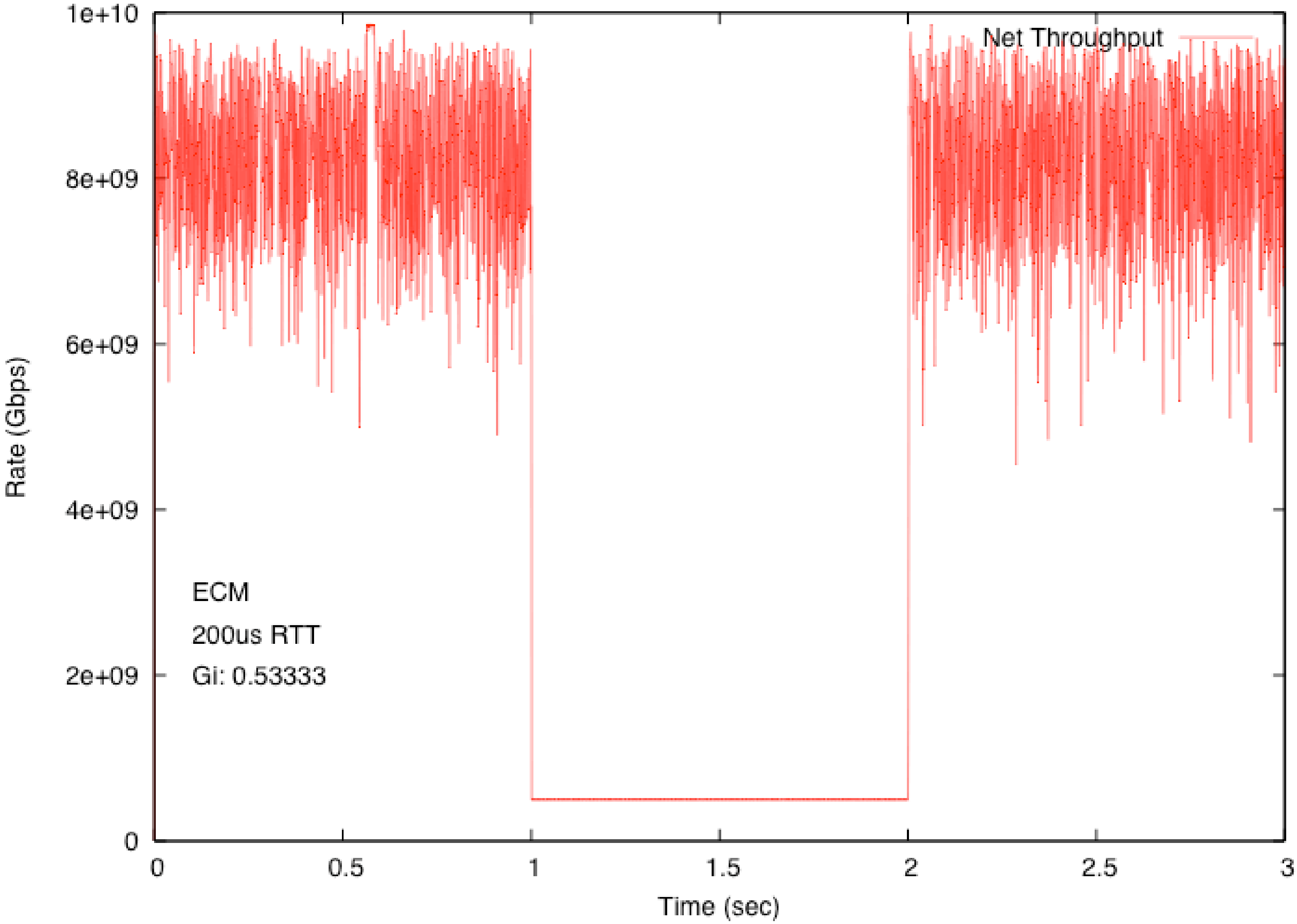
- QCN
  - W = 2.0
  - Q_EQ = 33 KB
  - GD = 0.0078125
  - Base marking: once every 150 KB
  - Margin of randomness: 30%
  - R_{unit} = 1 Mb/s
  - MIN_RATE = 10 Mb/s
  - BC_LIMIT = 150 KB
  - TIMER_PERIOD = 15 ms
  - R_AI = 5 Mbps
  - R_HAI = 50Mbps
  - FAST_RECOVERY_TH = 5
  - Quantized_Fb: 6 bits
- ECM
  - Qeq = 375
  - Qsc = 1600
  - Qmc = 2400
  - Qsat disabled
  - Ecm00 disabled
  - Gi = 0.53333 (varies with RTT)
  - W=2
  - Gd = 0.00026667
  - Ru = 1,000,000
  - Rd = 1,000,000
  - Td = 1ms
  - Rmin = 1,000,000

ECM, RTT=10 usecs

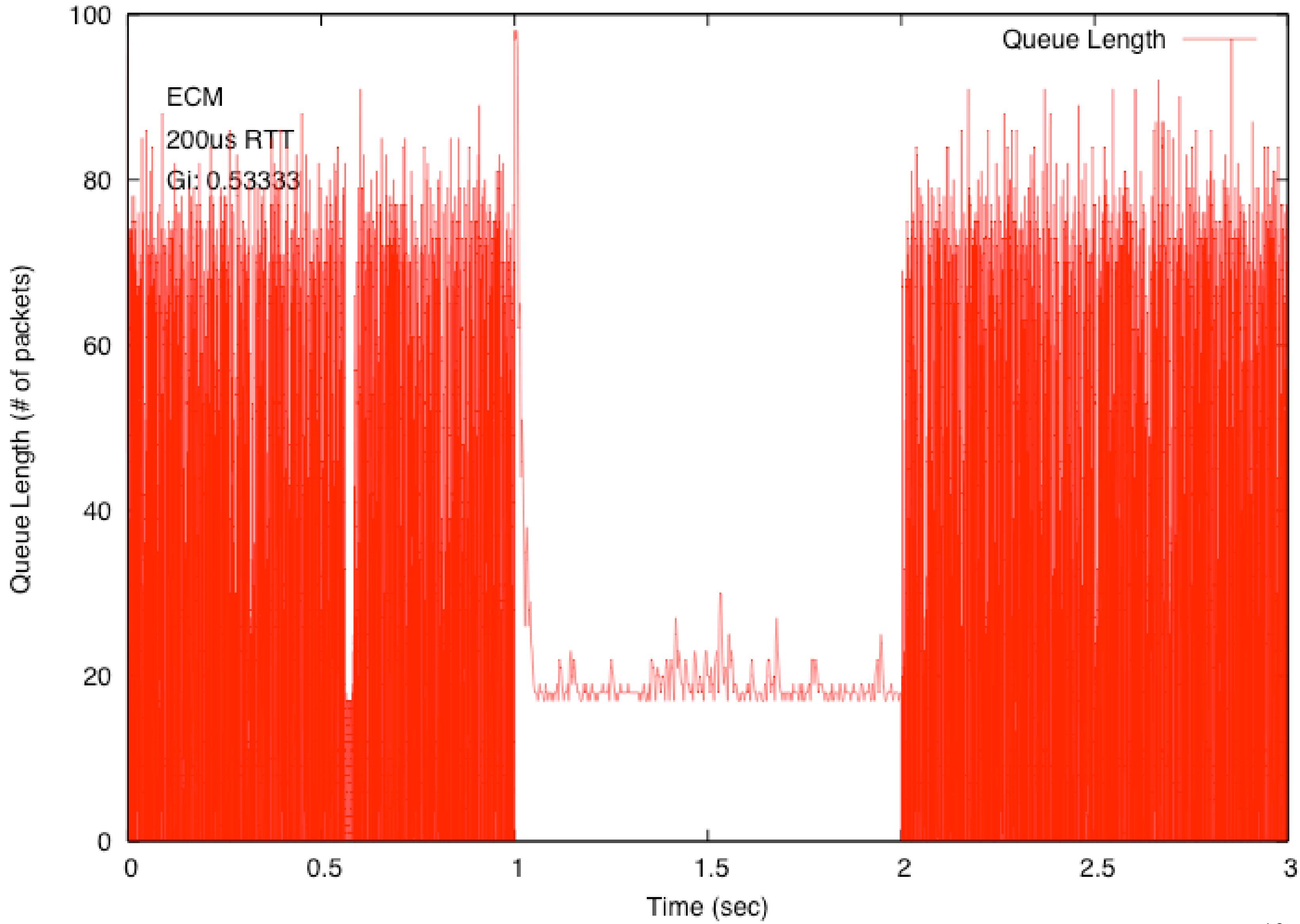


Recovery time = 3 msec

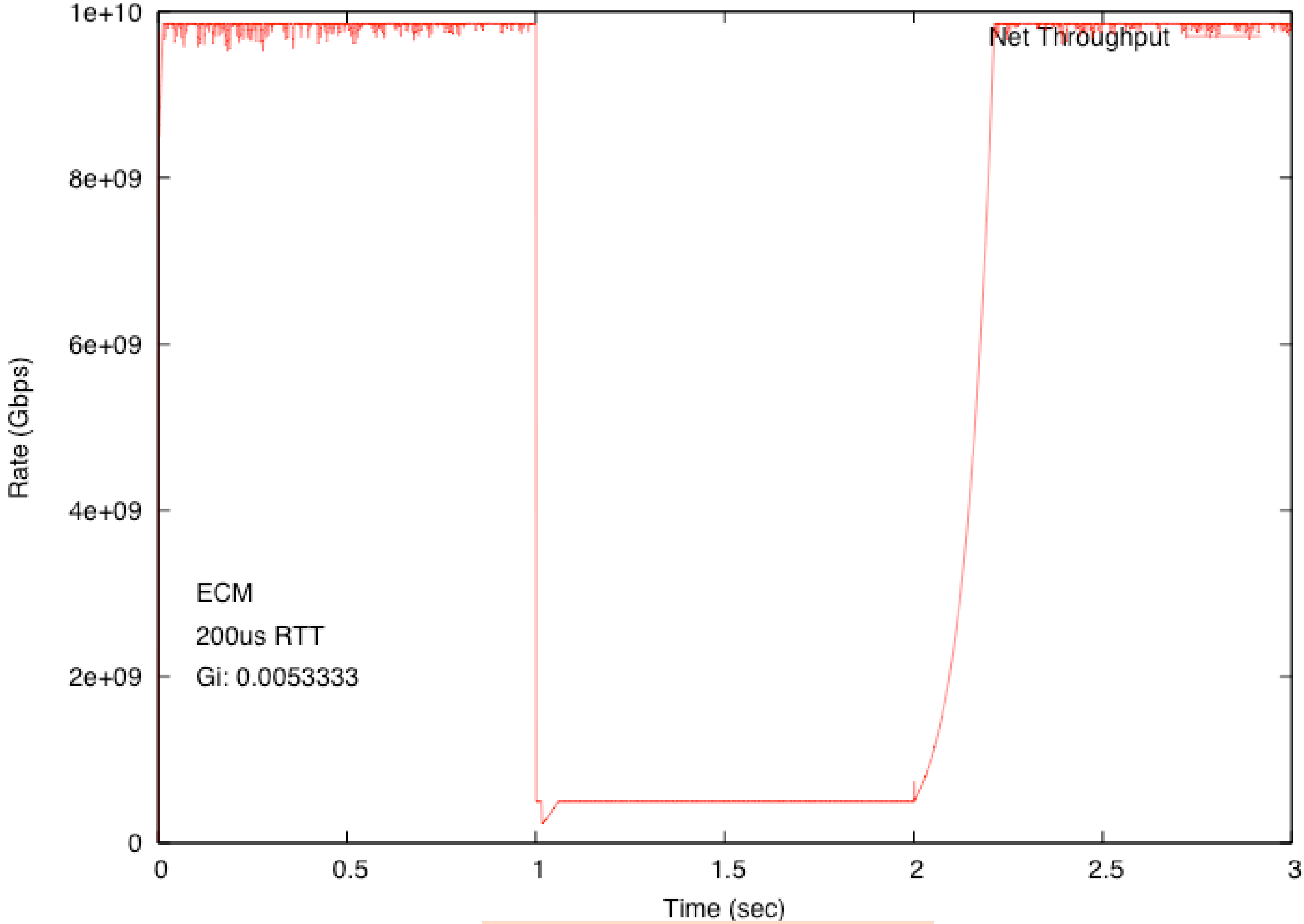
ECM, RTT=200 usecs, Throughput Gi = 0.53333



ECM, RTT=200 usecs, Queue size Gi = 0.53333

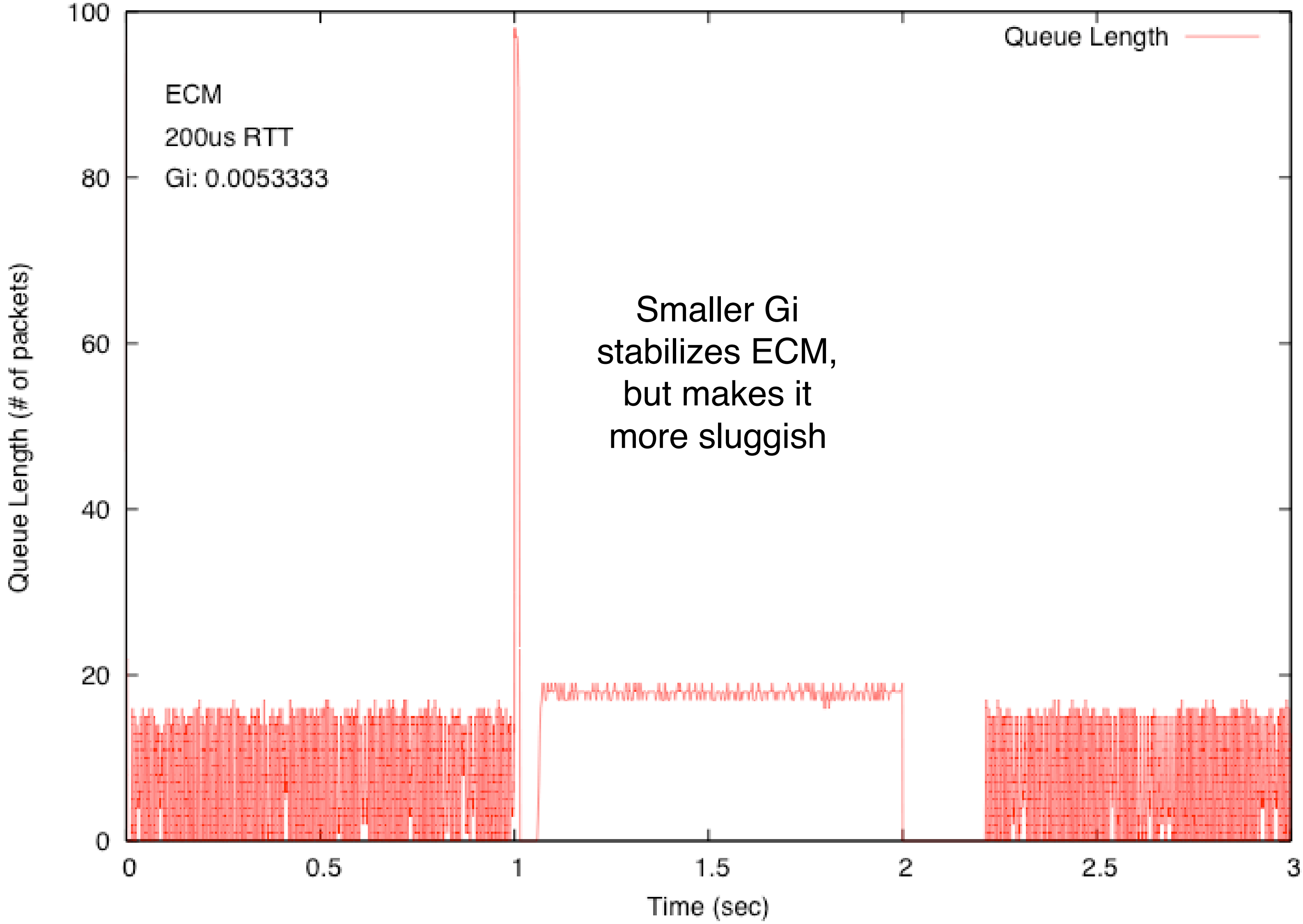


ECM, RTT=200 usecs, Throughput Gi = 0.0053333

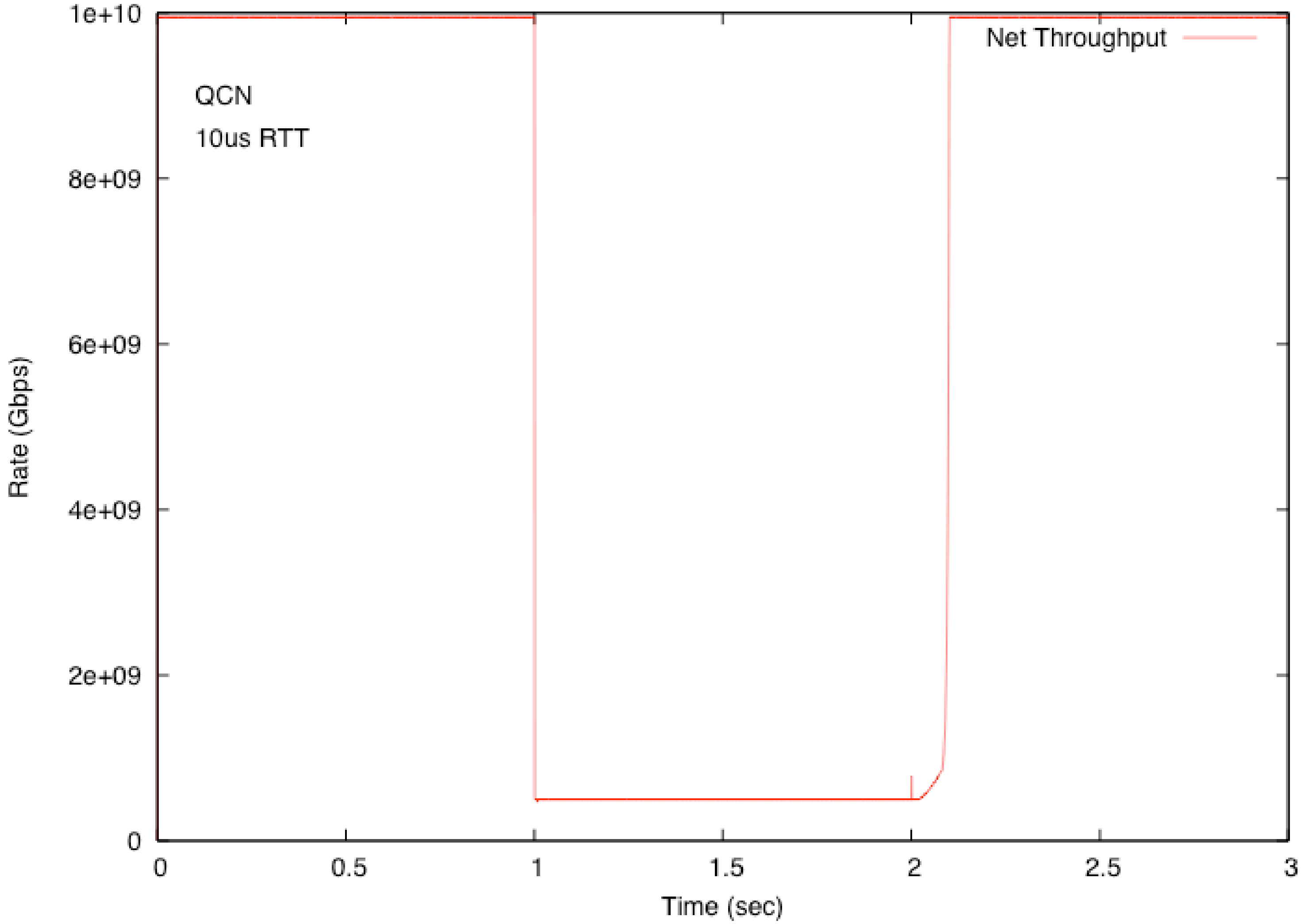


Recovery time = 214 msec

ECM, RTT=200 usecs, Queue size Gi = 0.0053333

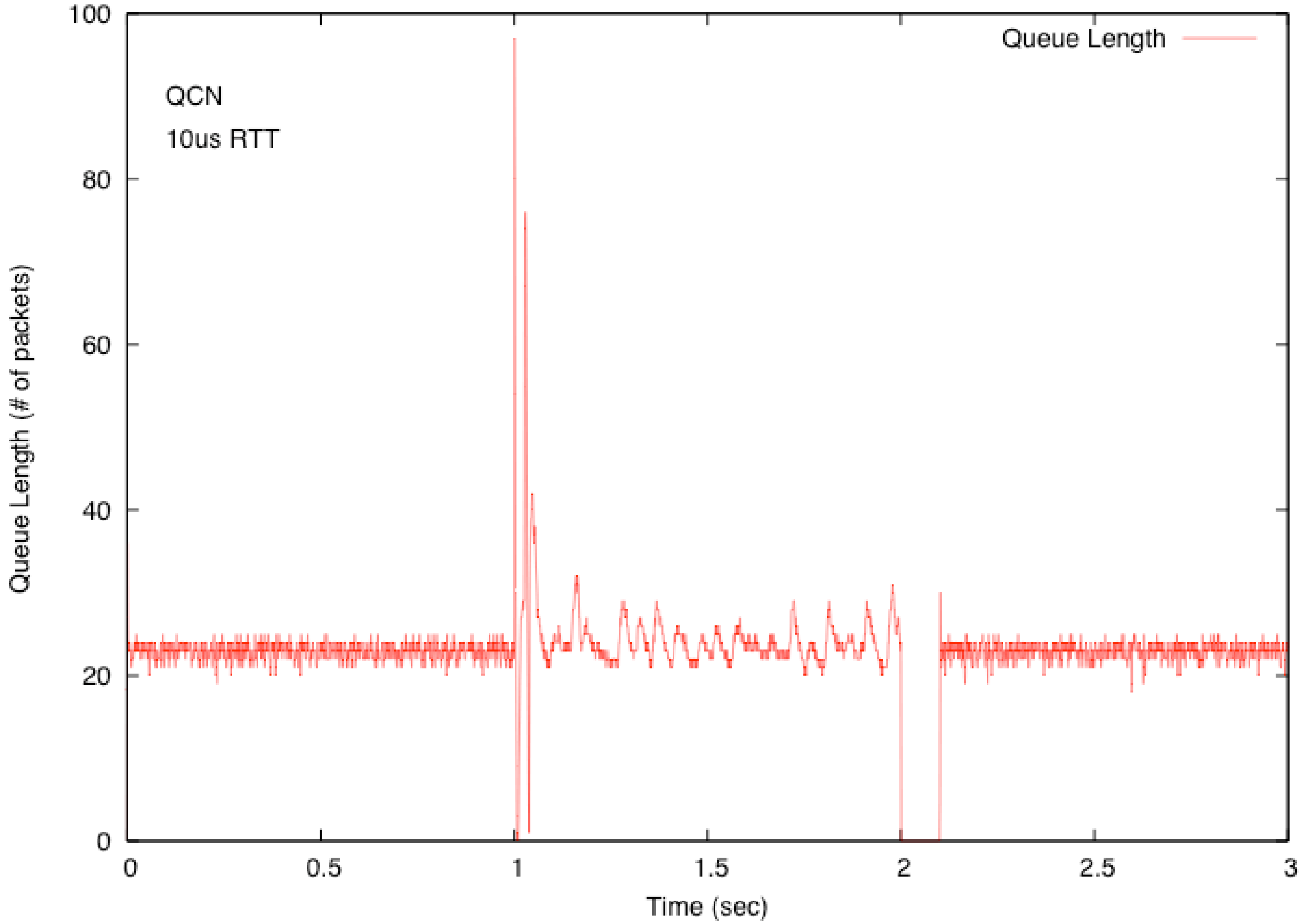


QCN, RTT = 10 us, Throughput

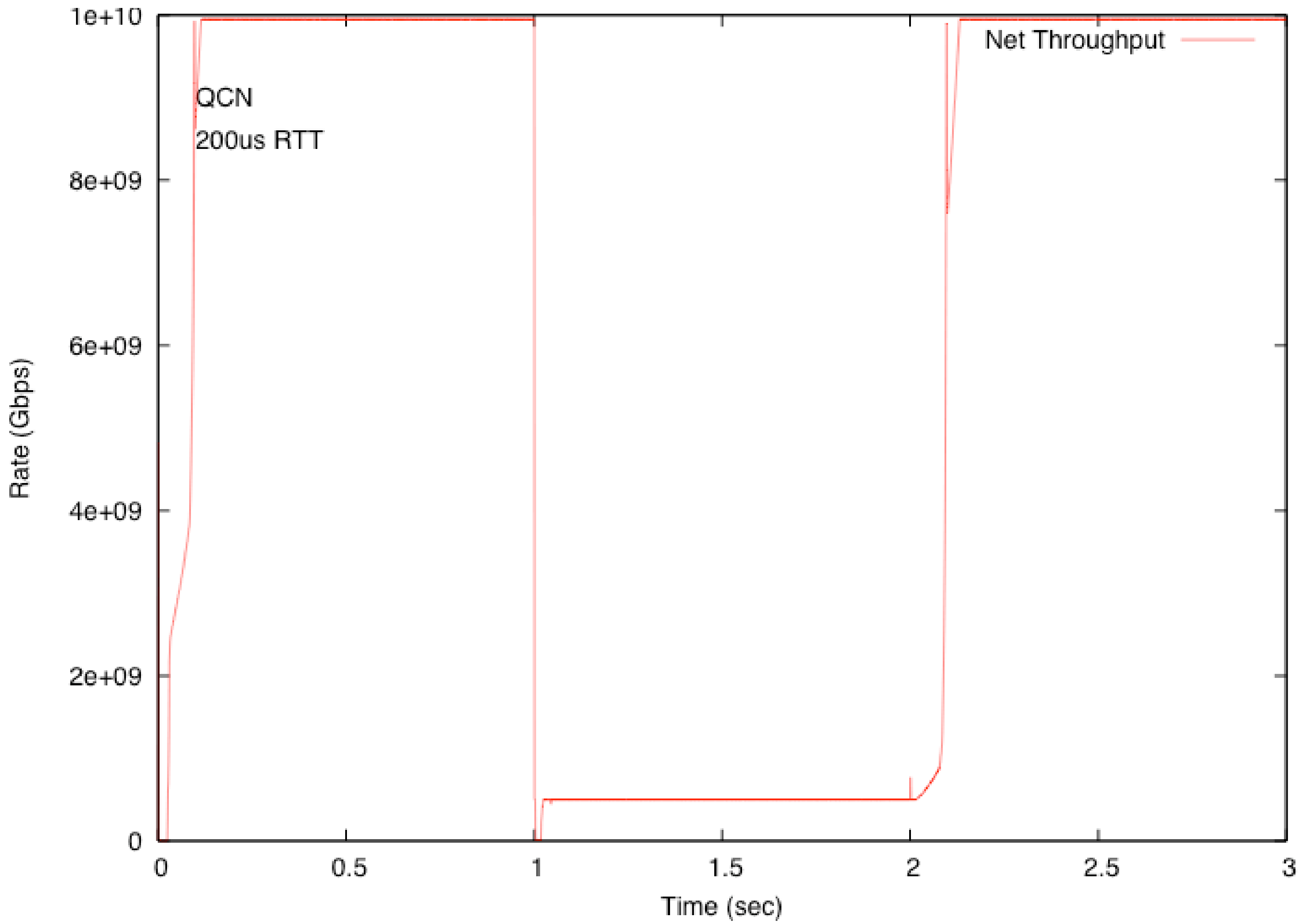


Recovery time = 94 msec

QCN, RTT=10 usecs, Queue size

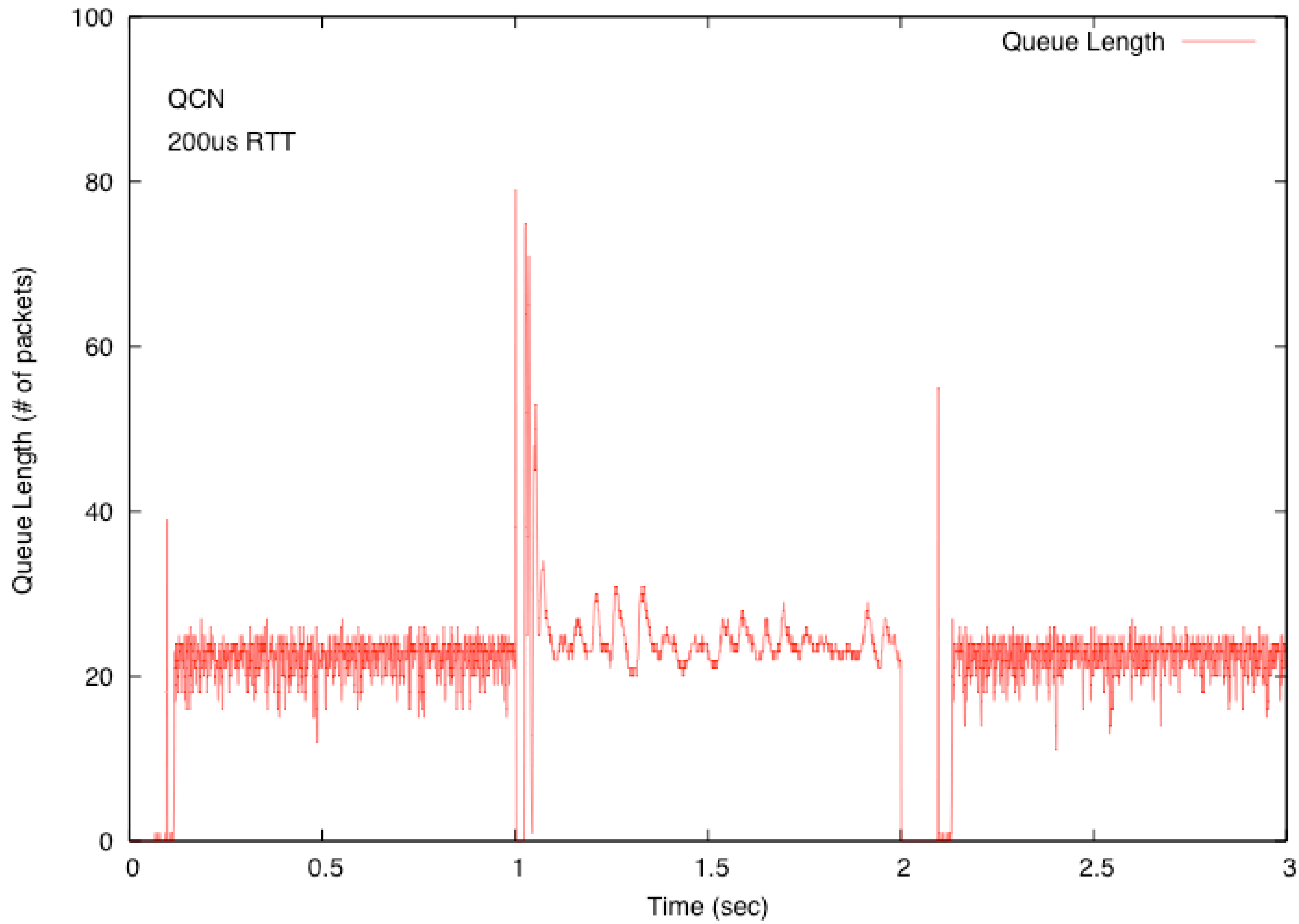


QCN, RTT = 200 us, Throughput



Recovery time = 108 msec

QCN, RTT=200 usecs, Queue size

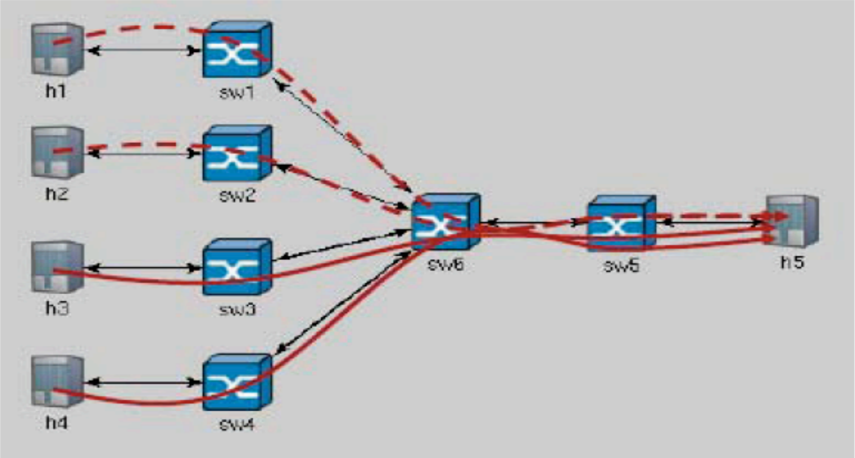


Summary of Robustness

- Robustness is important property of QCN
  - BCN, like other AIMD schemes, doesn't have it
  - So, stability at large RTT comes at cost of sluggish response
- Therefore, it is worth considering benchmark simulations
  - With different hotspot durations (Rong's presentation)
  - Different RTTs and number of sources
  - As an example, we consider Benchmark 5, under different ECM parameters

Benchmark #5

5. Symmetric Topology Single HS – Bursty



Workload:

- Point-to-point from h1-4 to h5
- Load: 100%
- H1 and H2 on-off sources (Ton = Toff = 20 ms)
- On/Off period exponential distribution

Scenarios:

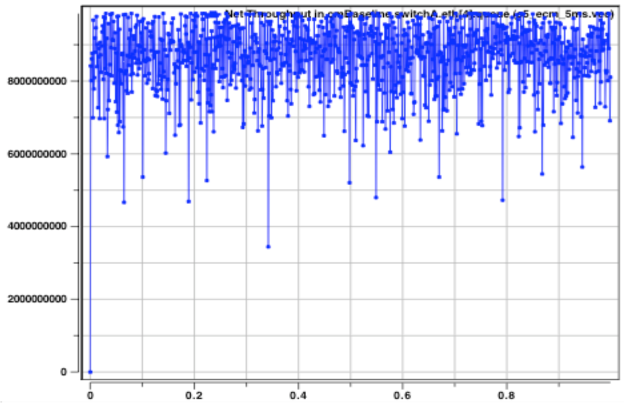
- Burst periods: 20, 10, 5mS

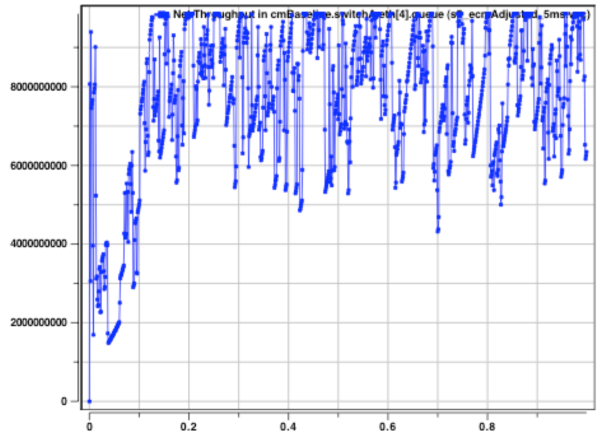
Verdana regular 7pt.
Legal text goes here

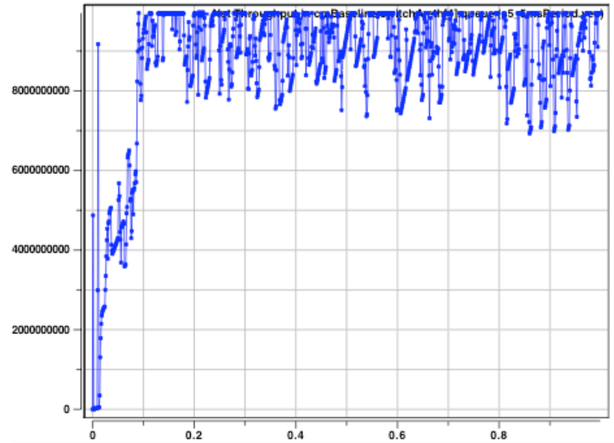
Required

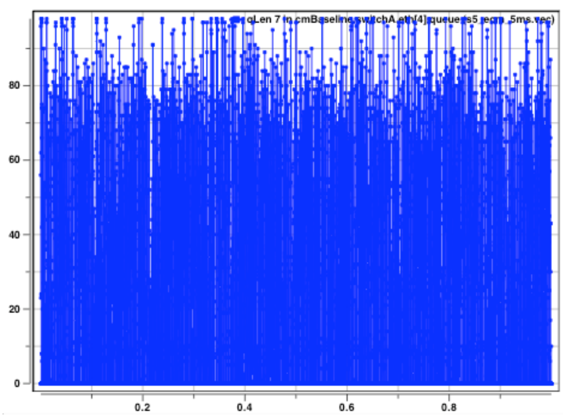
intel

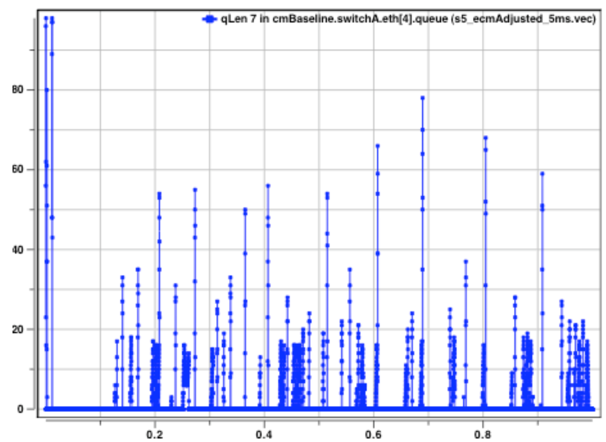
5 msec average burst period

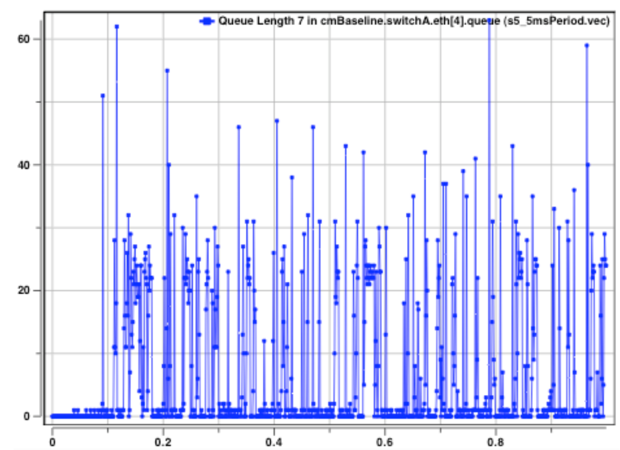










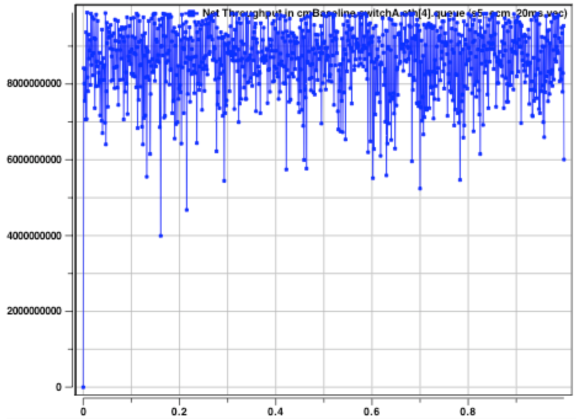


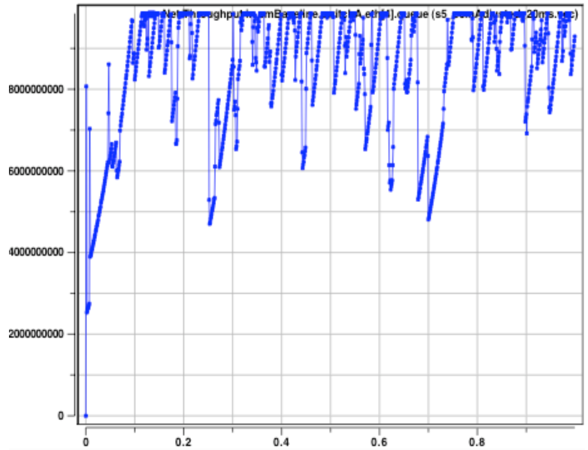
ECM: Standard Parameters
(Gi = 0.53333)

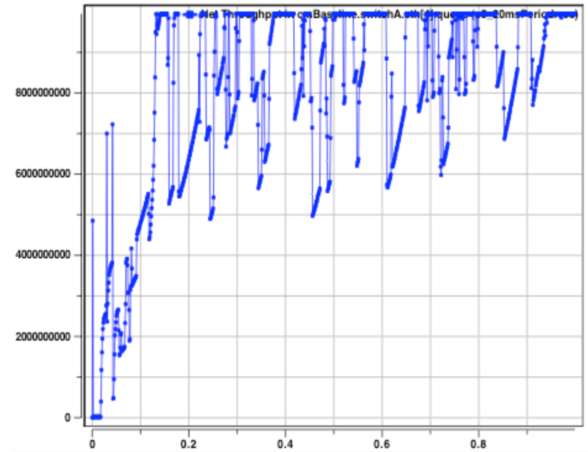
ECM: Stability Adjusted Parameters
(Gi = 0.0053333)

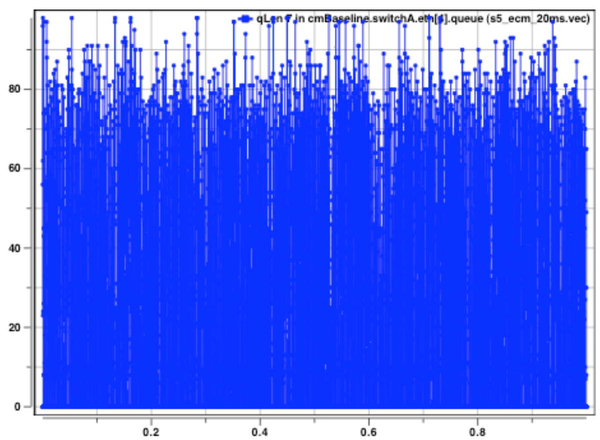
QCN

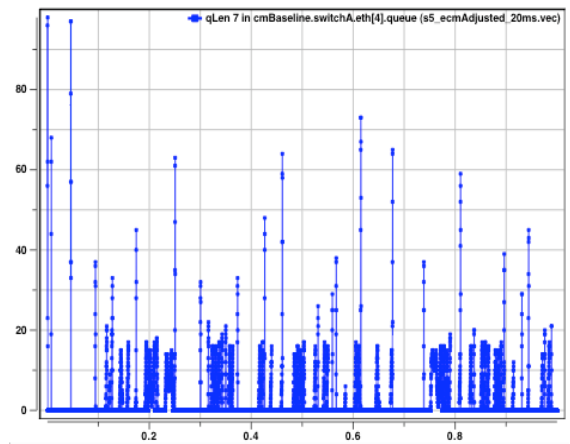
20 msec average burst period

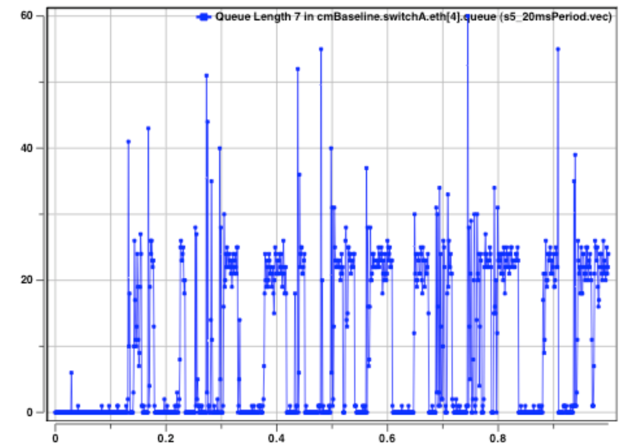












ECM: Standard Parameters
(Gi = 0.53333)

ECM: Stability Adjusted Parameters
(Gi = 0.0053333)

QCN

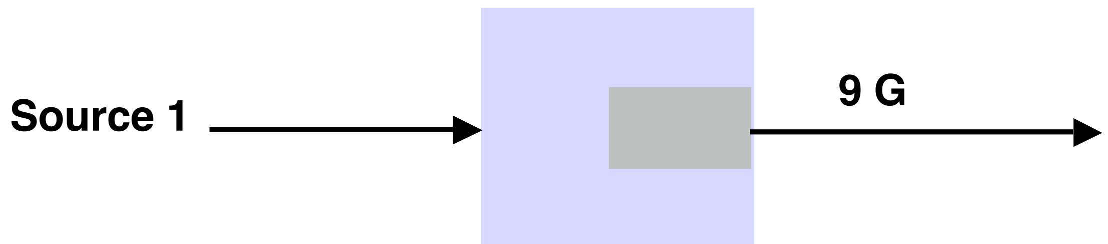
Summary of Presentation

- Overviewed the evolution of QCN
  - Showed the important and complementary roles of the timer and byte-counter
  - Outlined ECM+ as an evolution of ECM toward QCN/BIC
- Highlighted the role of the gain parameters in AIMD schemes

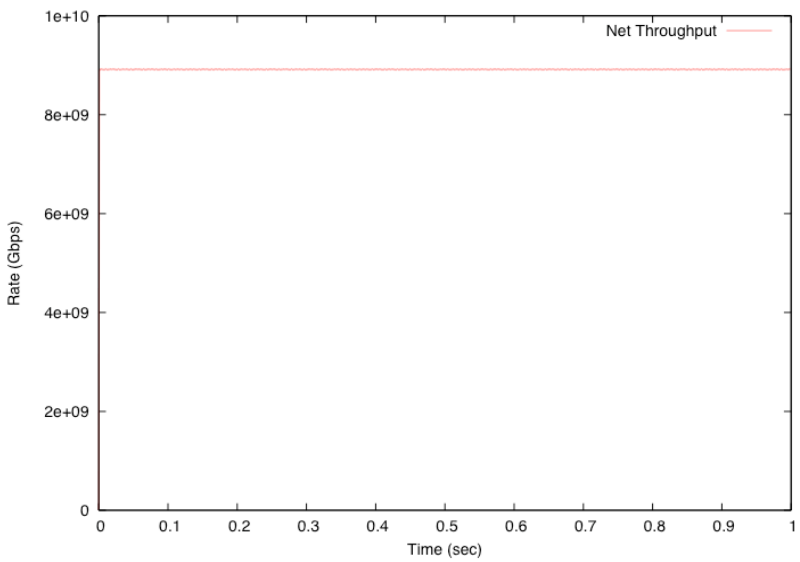
Appendix: The role of Gi

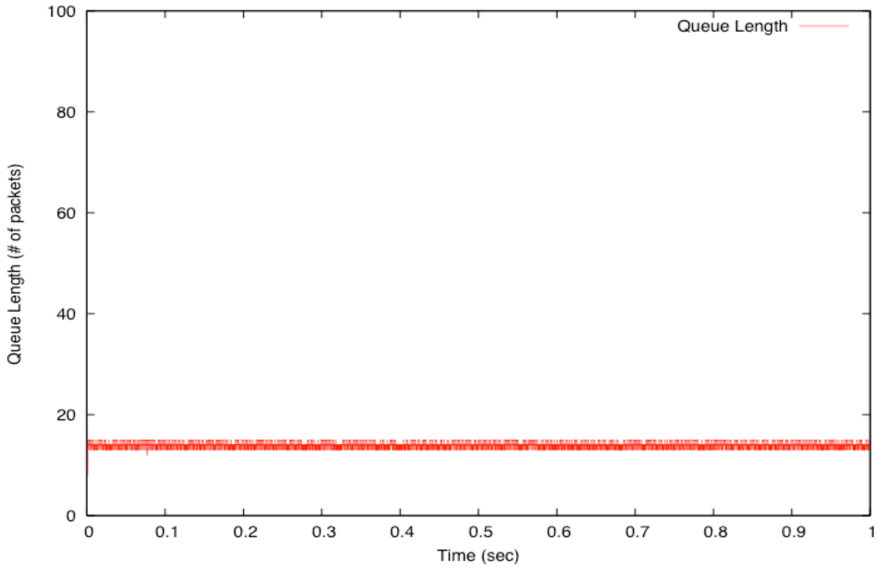
- It is worth understanding why AIMD schemes are not robust wrt RTT
  - Specifically, the gain parameter Gi depends on RTT
  - We will see that it is not possible to "set it" for all RTTs to have good stability and responsiveness

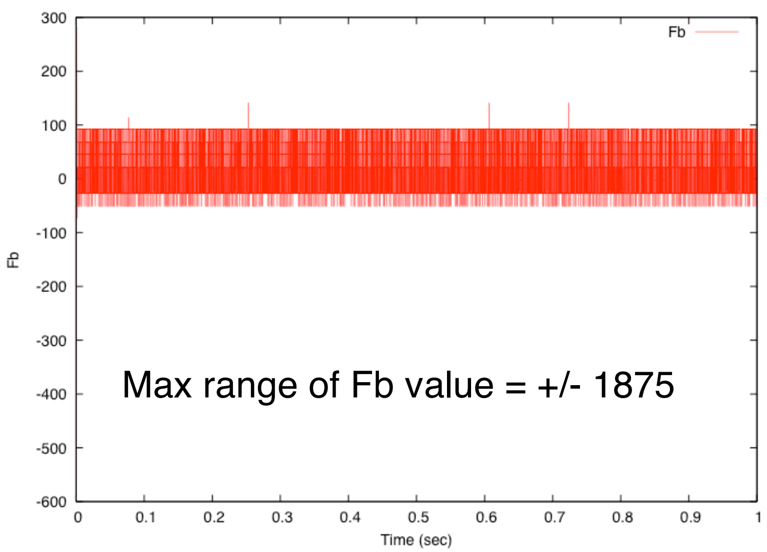
- Consider Baseline scenario
  - 1 source, 9G link
  - Source can send upto 10 G
  - Vary RTT: 10 usecs and 200 usecs

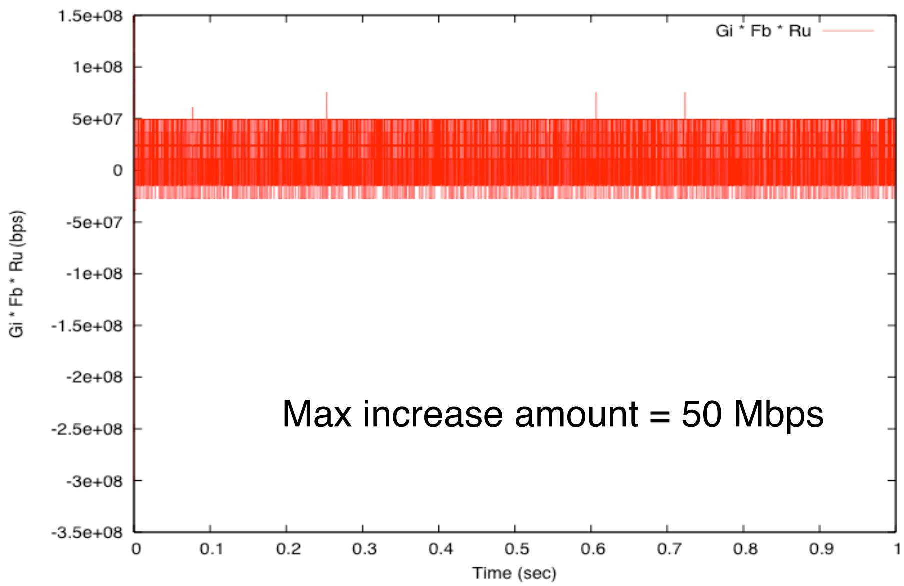


ECM, RTT=10 usecs

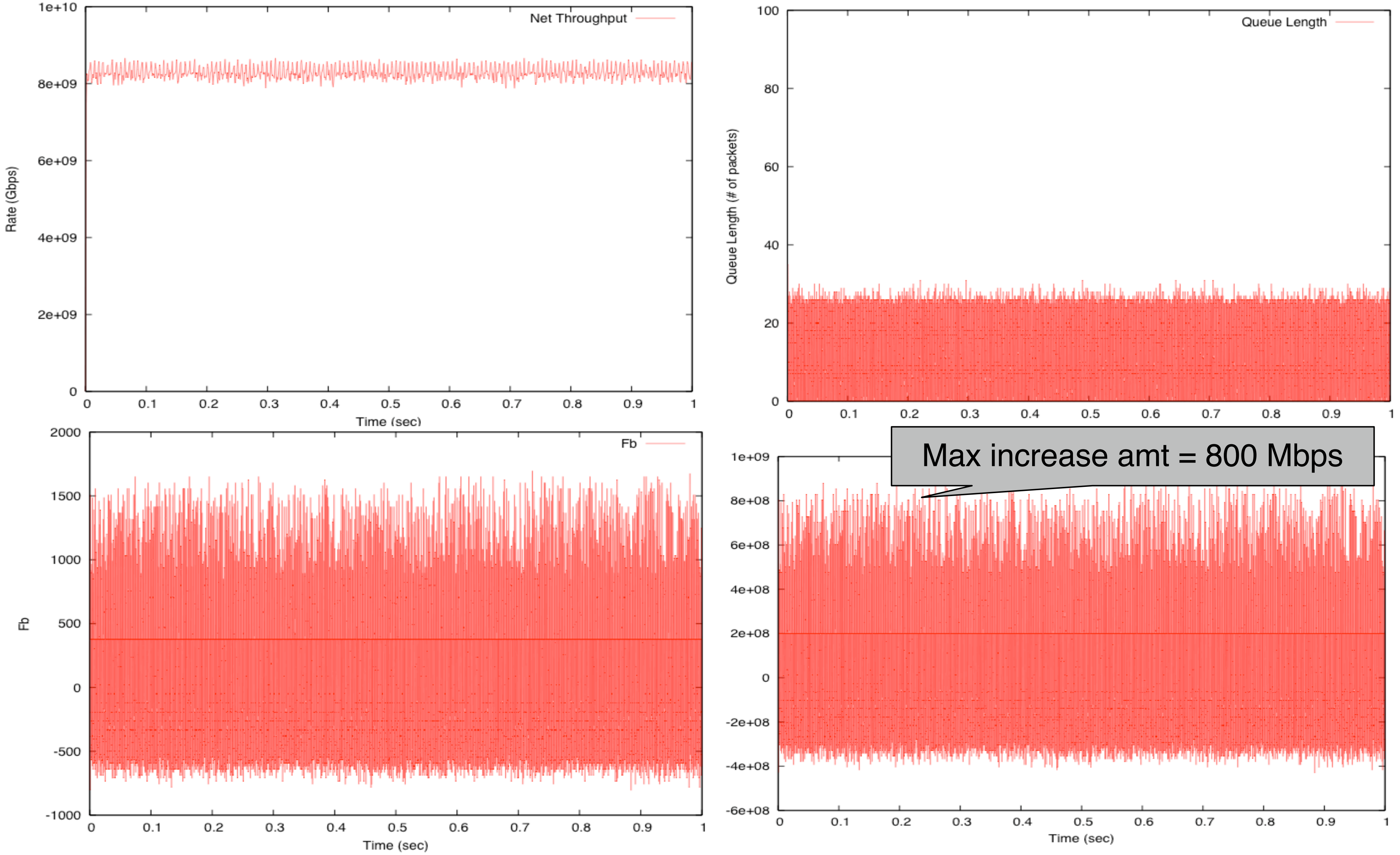




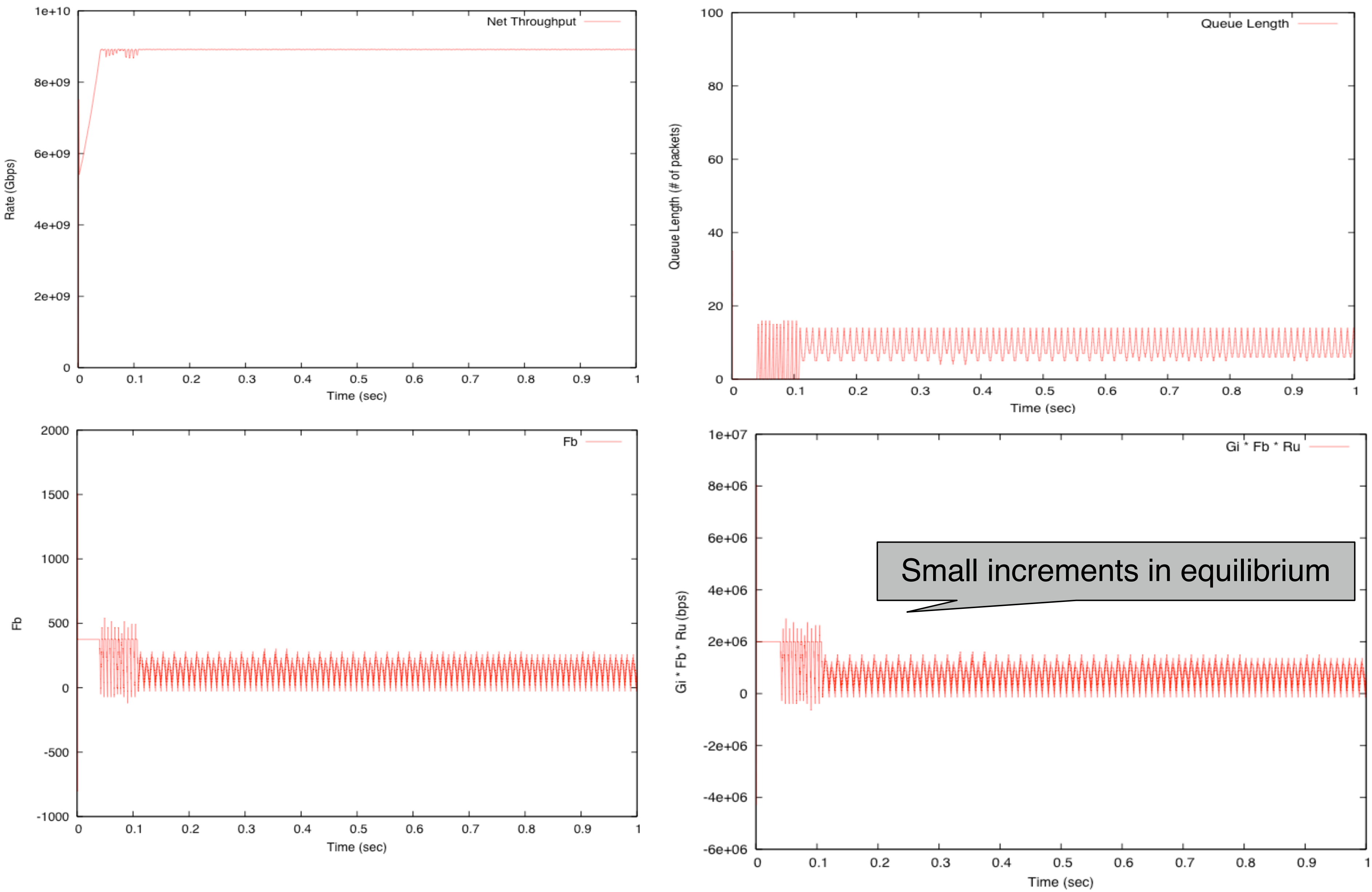




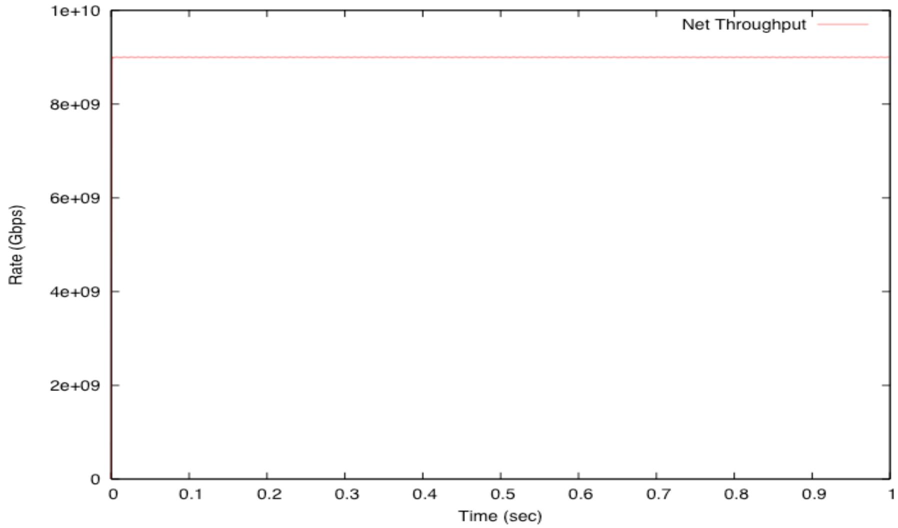
ECM, RTT=200 usecs, Gi = 0.53333

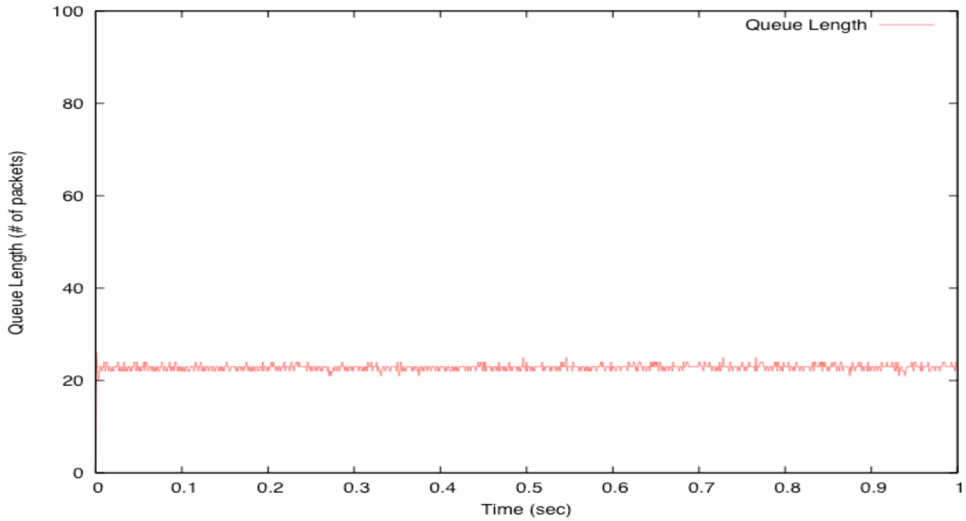


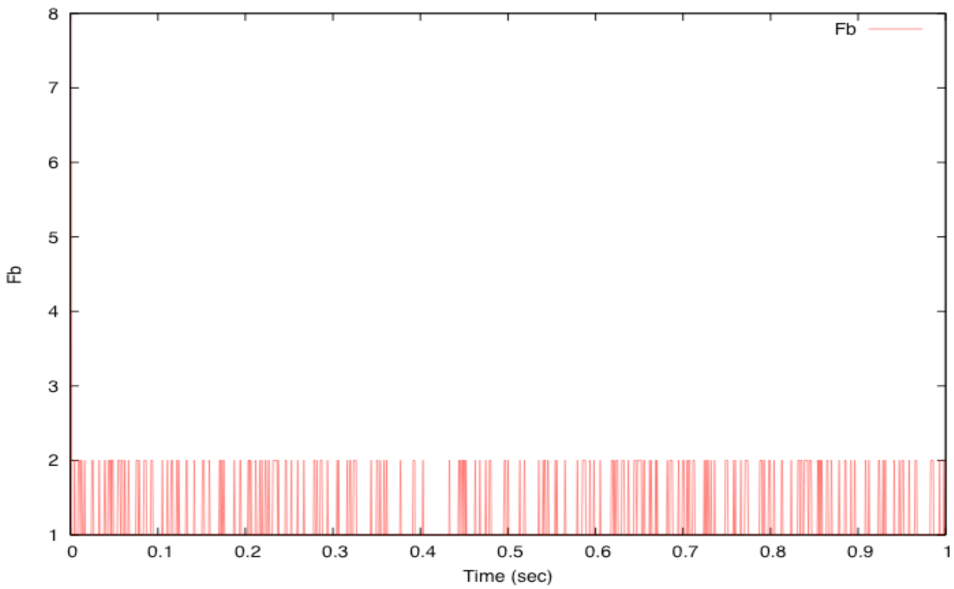
ECM, RTT=200 usecs, Gi = 0.0053333



QCN, RTT = 10 us


Net Throughput
Rate (Gbps)
1e+10
8e+09
6e+09
4e+09
2e+09
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time (sec)


Queue Length
Queue Length (# of packets)
100
80
60
40
20
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time (sec)


Fb
Fb
8
7
6
5
4
3
2
1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time (sec)

QCN, RTT = 200 us