QCN: Algorithm with Drift

Rong Pan, Balaji Prabhakar,
Ashvin Lakshmikantha, Mick Seaman

Outline of presentation

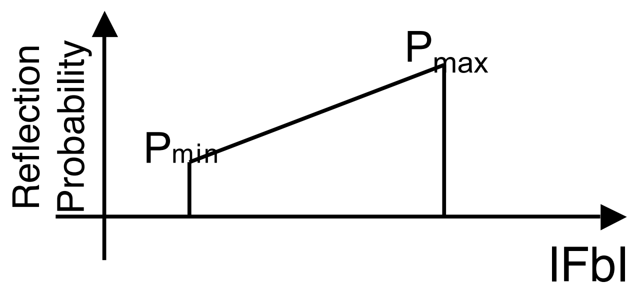
- High-level description of QCN already provided
  - Current slides have updated version of how the DE bit is used
- Drift has now been added
  - For failsafe operation
  - For reclaiming rate limiters
- Simulations with drift
  - Infinitely long-lived flows: throughput, backlog, fairness, drops
  - Finite flows: FCT (short and long flows), drop percentages

Basic QCN

- 2-point architecture: Reaction Point -- Congestion Point

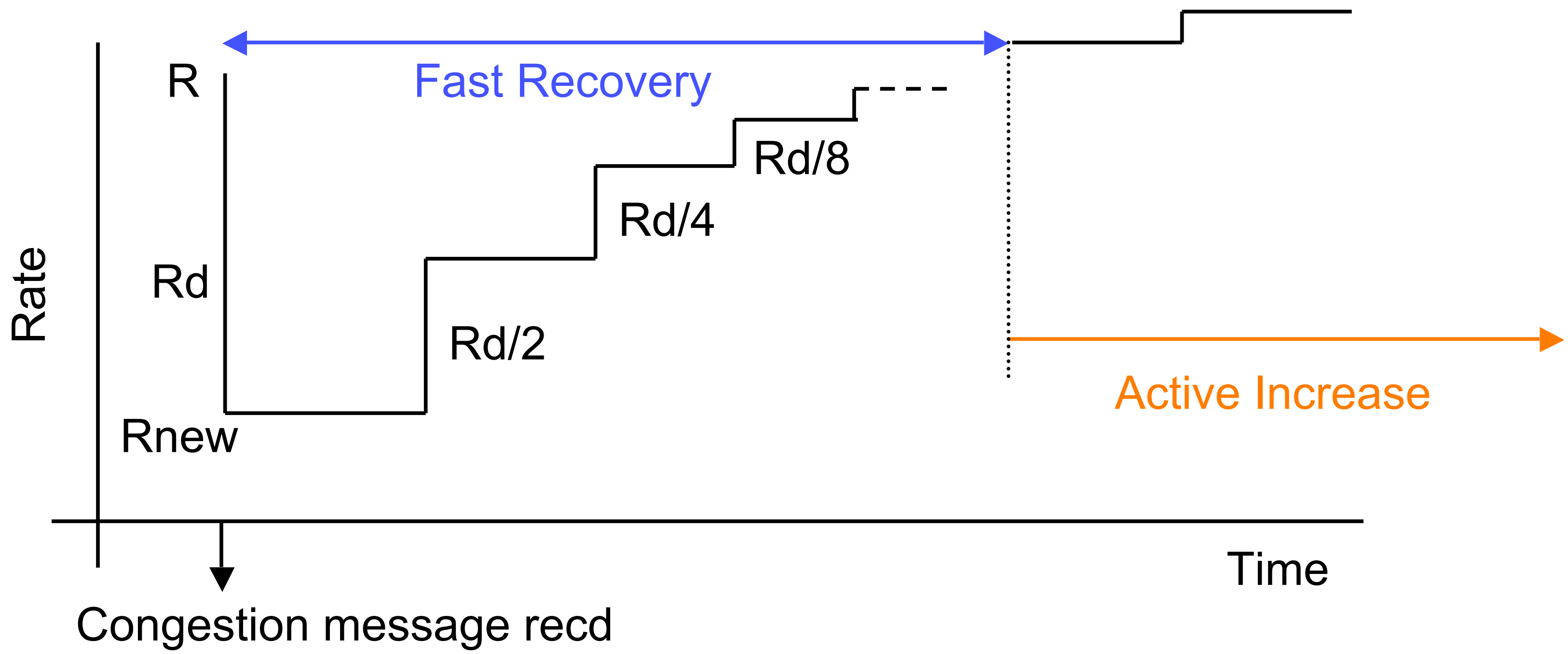
1. **Congestion Points:** Sample packets, compute feedback (Fb), quantize Fb to 6 bits, and reflect only *negative* Fb values back to Reaction Point with a probability proportional to Fb.

$$Fb = -(q_{off} + w\ q_{delta})$$
$$= -(\text{queue offset} + \text{w.rate offset})$$



2. **Reaction Points:** Transmit regular Ethernet frames. When congestion message arrives: perform multiplicative decrease, fast recovery and active probing.
   – Fast recovery similar to BIC-TCP: gives high performance in high bandwidth-delay product networks, while being very simple.

Fast Recovery and Active Increase



Basic QCN: Outcomes/results

- Easy to deploy, light resource requirement
  - No header modifications, no tags, **immediately deployable**.
  - Can work with a *single* rate limiter.
    - Alias all flows which have received negative feedback onto the rate limiter. RL becomes "meta-flow" with fast recovery + active probing ensuring good performance.
    - The algorithm is well-defined; i.e. does not rely on the existence of multiple rate limiters for correctness of specification since it has no tags or probes.

- Quantizing Fb simplifies implementation
  - Fb value used to index into a small table to find the decrease factor.
    - No potentially expensive hardware resources needed for computations.
  - Lookup table also makes the scheme **easily reconfigurable** (if Fb --> Rate relation changes), a useful workaround.

QCN: 3-point architecture

- ReaP--CP--RefP
  - Allows signaling Fb=0 values to ReaP, which indicate *lack* of congestion. Only the RefP can do this without the use of RP-->CP association tags.
  - When a ReaP receives an Fb=0 signal, it just skips to the next cycle of Fast Recovery or Active Increase; i.e. it increases the rate appropriately and it restarts the byte counter
    - Simple behavior, no increase gains or parameters.
  - Two flavors of signaling
    - In-band: Using packet headers
    - Out-of-band: Using probe packets (as in E2CM and FECN)
- In-band signaling
  - In the pseudocode released, we showed how the 6-bit Fb field in the packet header can be modified at the switch for sampled packets and how reflection occurs at CP and RefP.
  - A probe version of this scheme can also be done.

Simplifying signaling further

- Note that
  - To maintain low drops while allowing sources to come on at 10 Gbps, we need negative Fb values to be signaled backward; the forward path has a larger delay.
  - To grab extra bandwidth, it is useful to signal Fb=0. We can employ forward signaling to do this without tags.

- Therefore, we propose
  - All Fb-negative signals generated probabilistically by CPs
  - RefP reflects only Fb=0 signals
  - This elegantly extends the 2-point architecture to the 3-point architecture
  - As we will see in the simulations, it also performs excellently

Signaling in the 3-point architecture

- Two concrete signaling methods based on this proposal are…

1. Use probe packets, say 1 in K packets from the source
    - Probe enters network with a single Fb0-bit set to 0 and passes through the CPs
    - If a CP has Fb < 0 value, it sets the Fb0-bit to 1
  - When RefP receives a probe
    - If Fb0-bit is set to 1, do nothing
    - Else, reflect probe with small probability (e.g. 1-3%)

2. Using the DE (Discard Eligible) bit in the packet header
  - DE bit set to 0 when packet leaves source
  - When packet arrives at CP
    - If Fb value at CP is negative
    - Set DE bit to 1
    - Reflect Fb value to ReaP with probability biased by Fb value
    - Else, do nothing
  - When RefP receives a packet
    - If DE bit is set to 1, do nothing
    - Else send Fb=0 signal to ReaP with small probability (e.g. 1-3%)

Drift

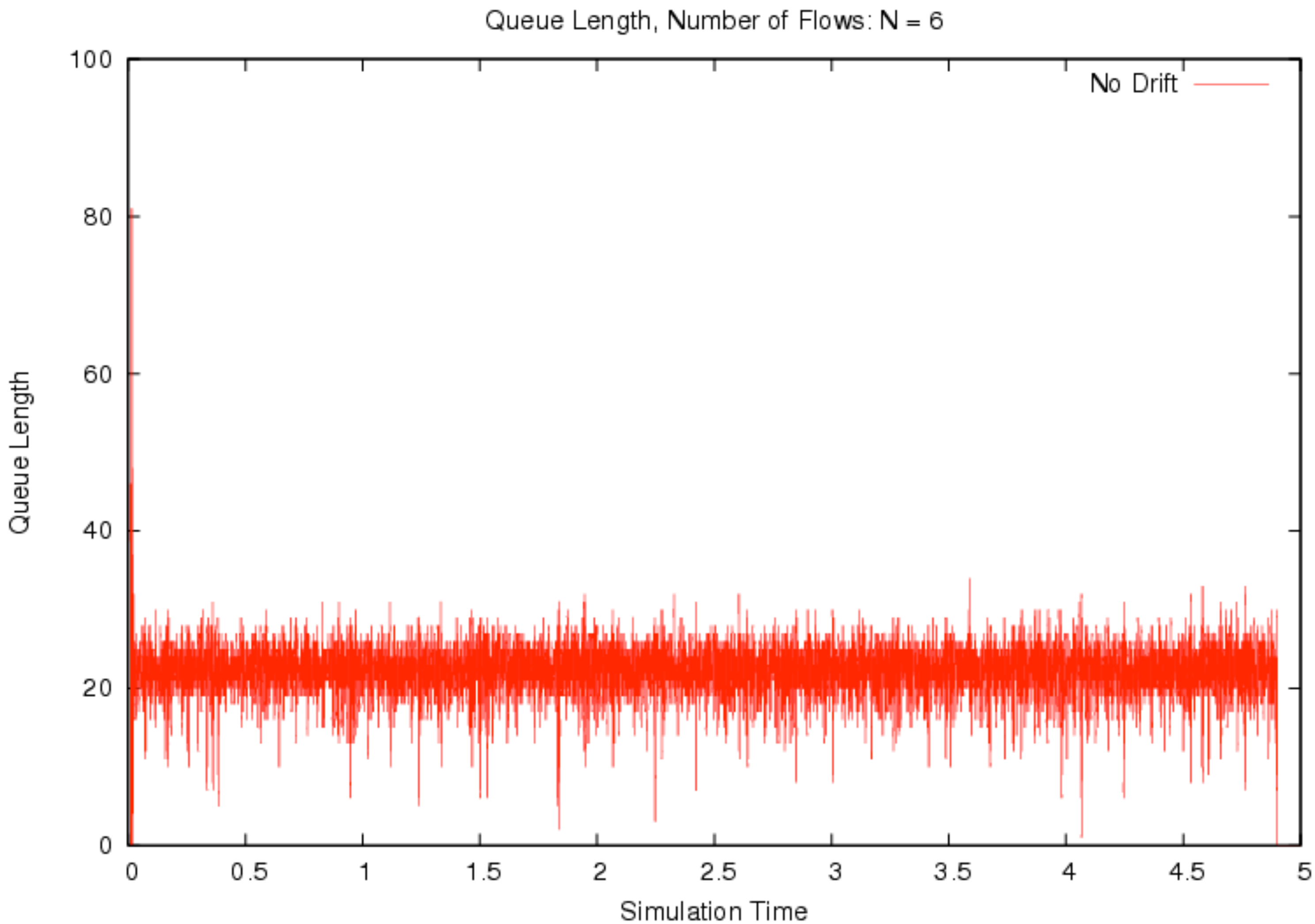
- Since both FR and Active Increase use byte counters for self-clocking, it is advisable to have a time-driven “rate drift”
  - Provides failsafe operation, allows rate limiters to be decommissioned
  - Note: We’ve seen drift earlier in BCN

- Drift
  - Drift clock corresponding to RL expires every T units of time
  - When clock expires
    - Increase transmission rate from R to R.X, where X > 1
    - Restart clock
  - Any time an Fb<0 signal is received by RL, restart the drift clock
    - Note: this ensures drift is used only minimally and when network is uncongested
    - Also note it makes drift inversely proportional to a flow’s sending rate, since larger sources get more Fb<0 signals relative to small sources

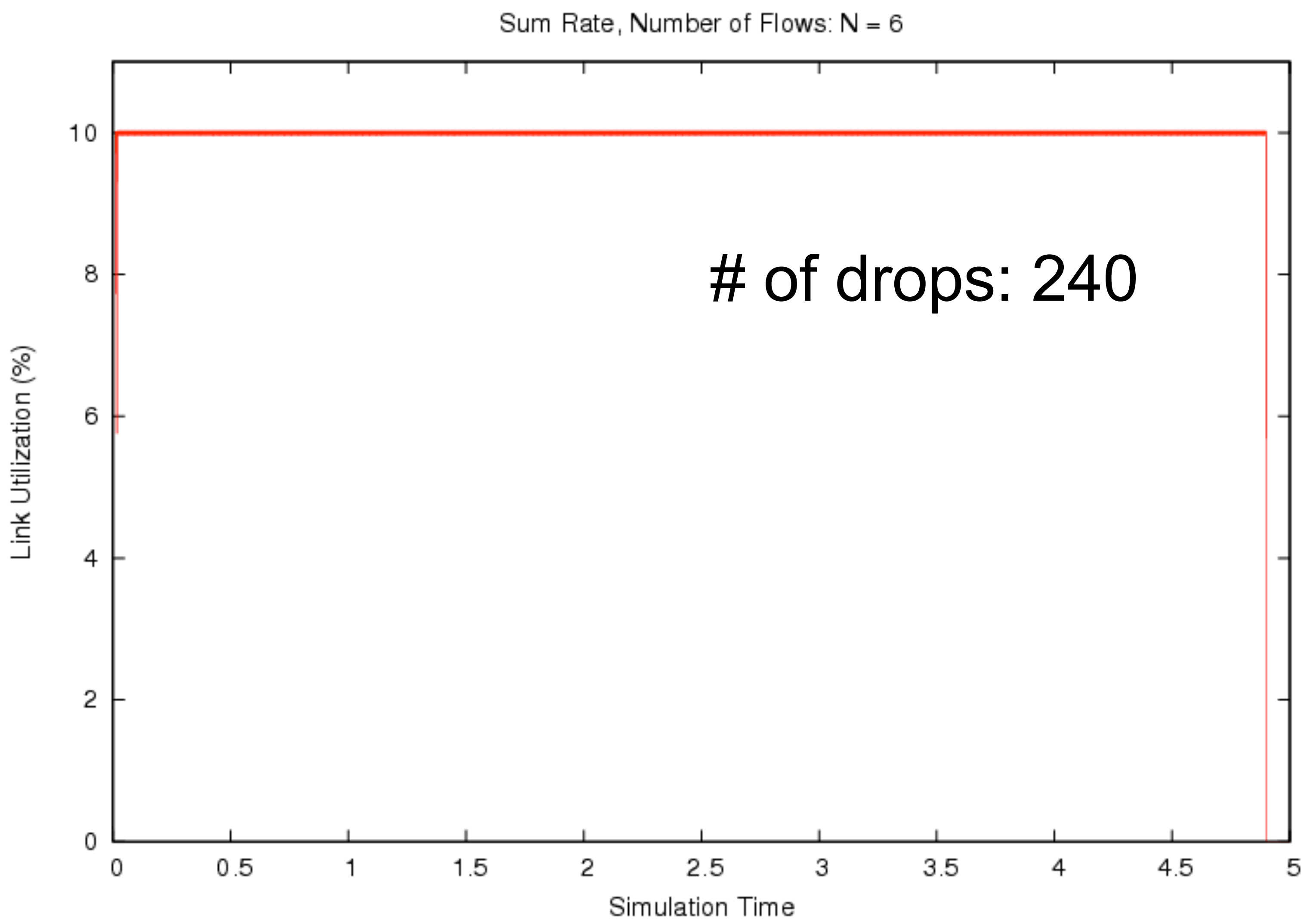
Simulation: 2-point Architecture

- Infinitely long-lived flows: simultaneous starts
  - Single link, 6 flows on at 10 Gbps at time 0
  - Link delay (RTT): 40 microseconds
  - Gd = 1/128
  - w = 2
  - Ri = 12 Mbps
  - Drift: X = 1.005, T = 500 musecs
  - Sampling function = linearly increases with lFbl from 1--10%

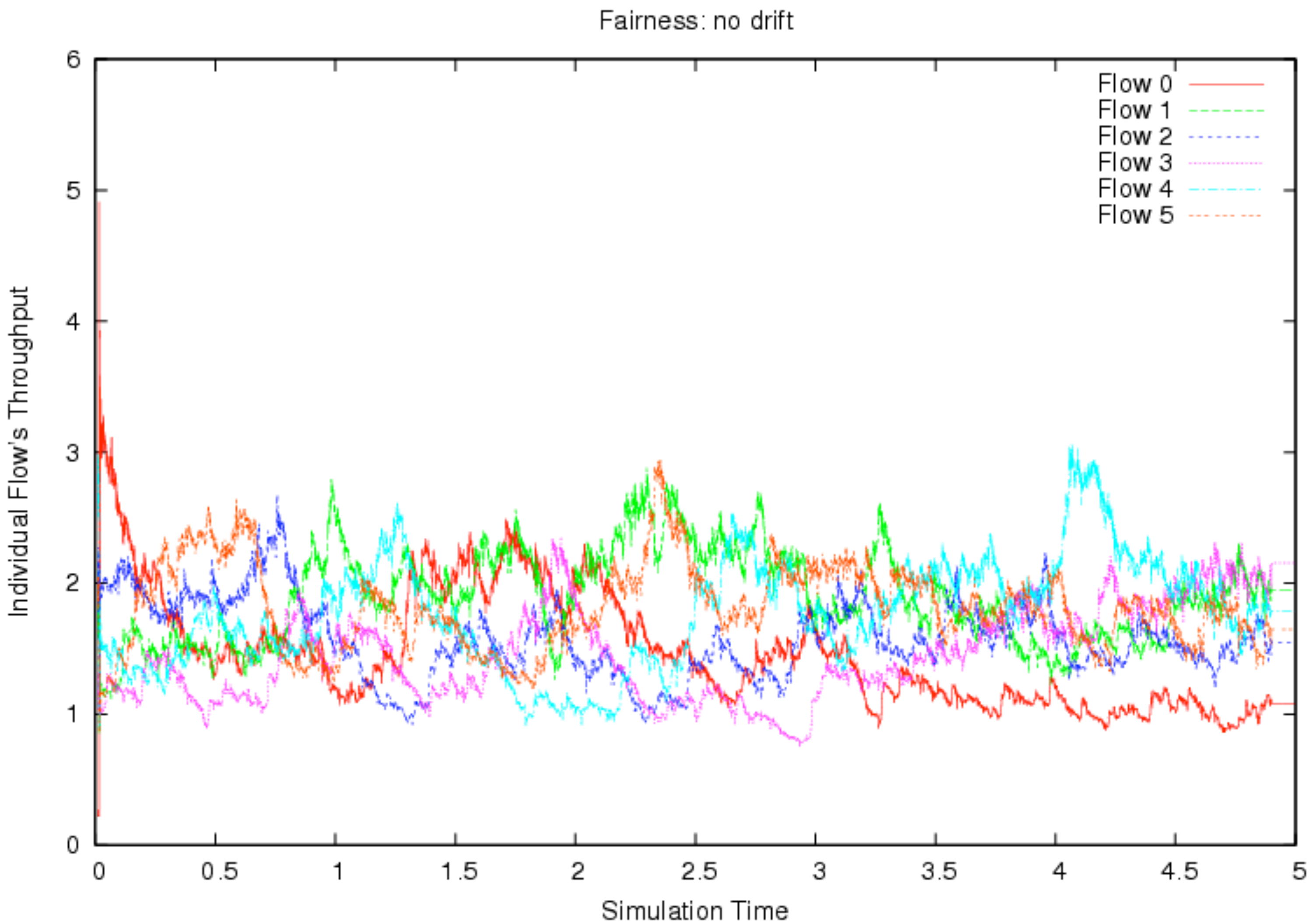
Queue Length: No Drift



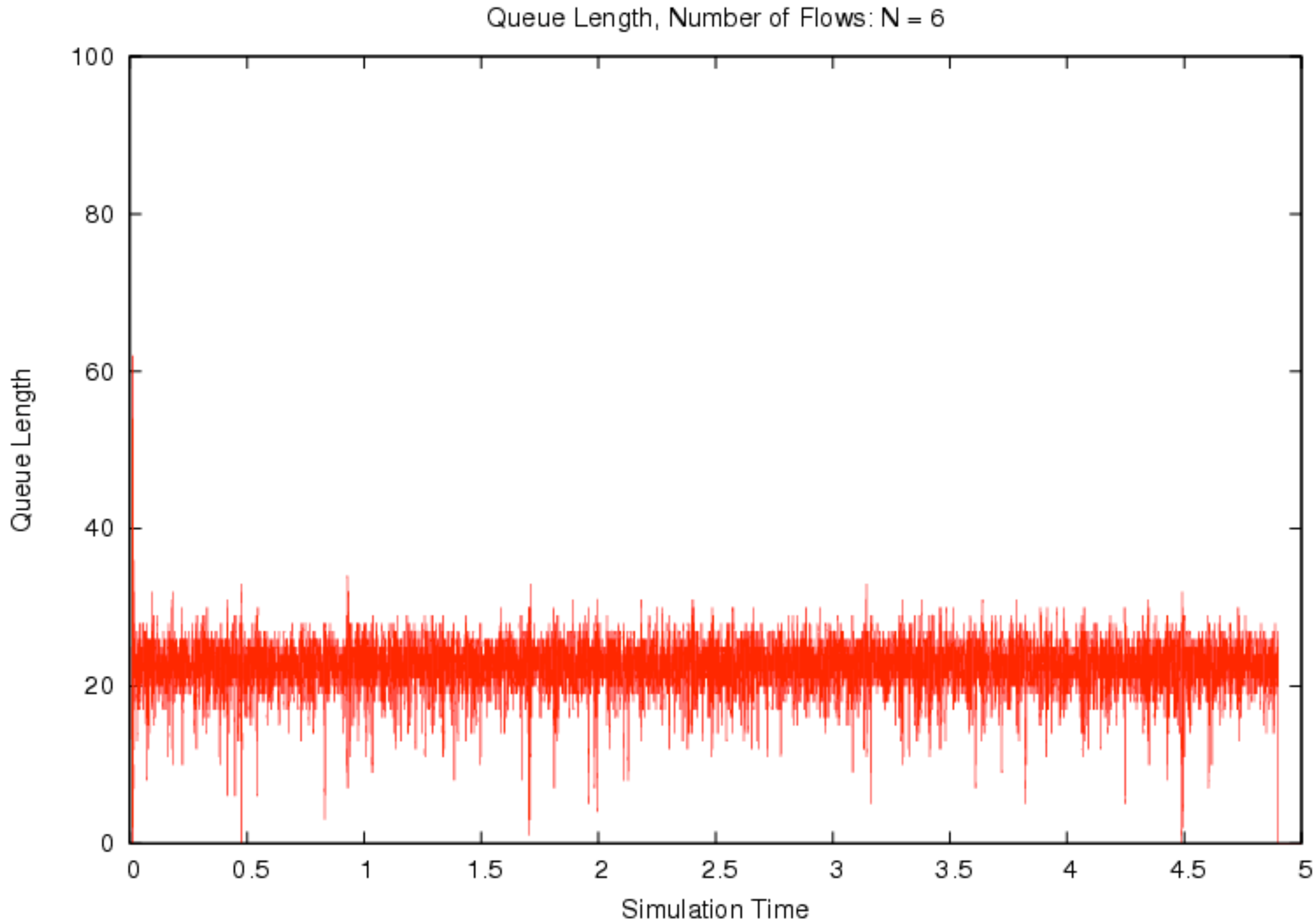
Aggregate Rate: No Drift



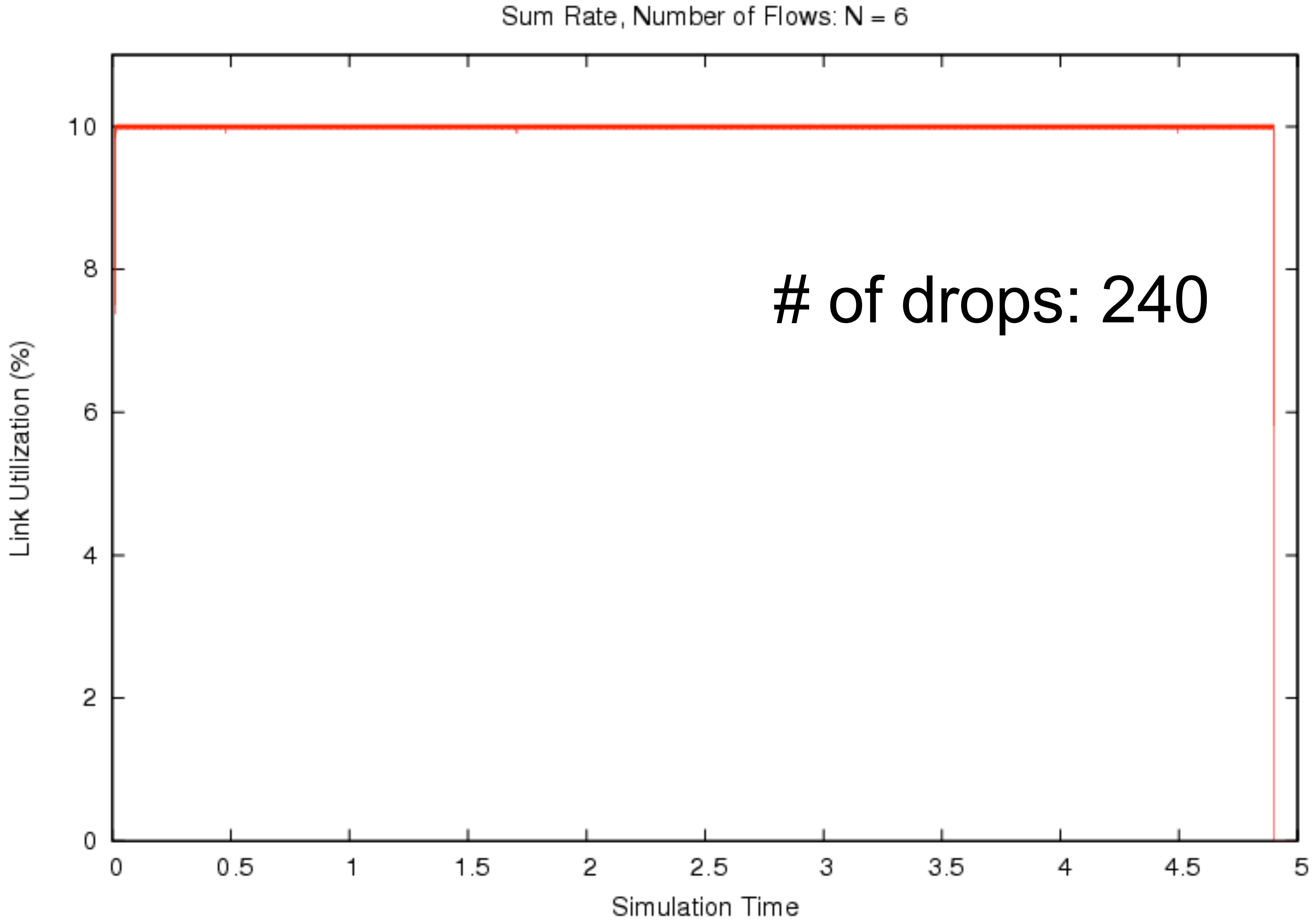
Instantaneous Rates, No Drift



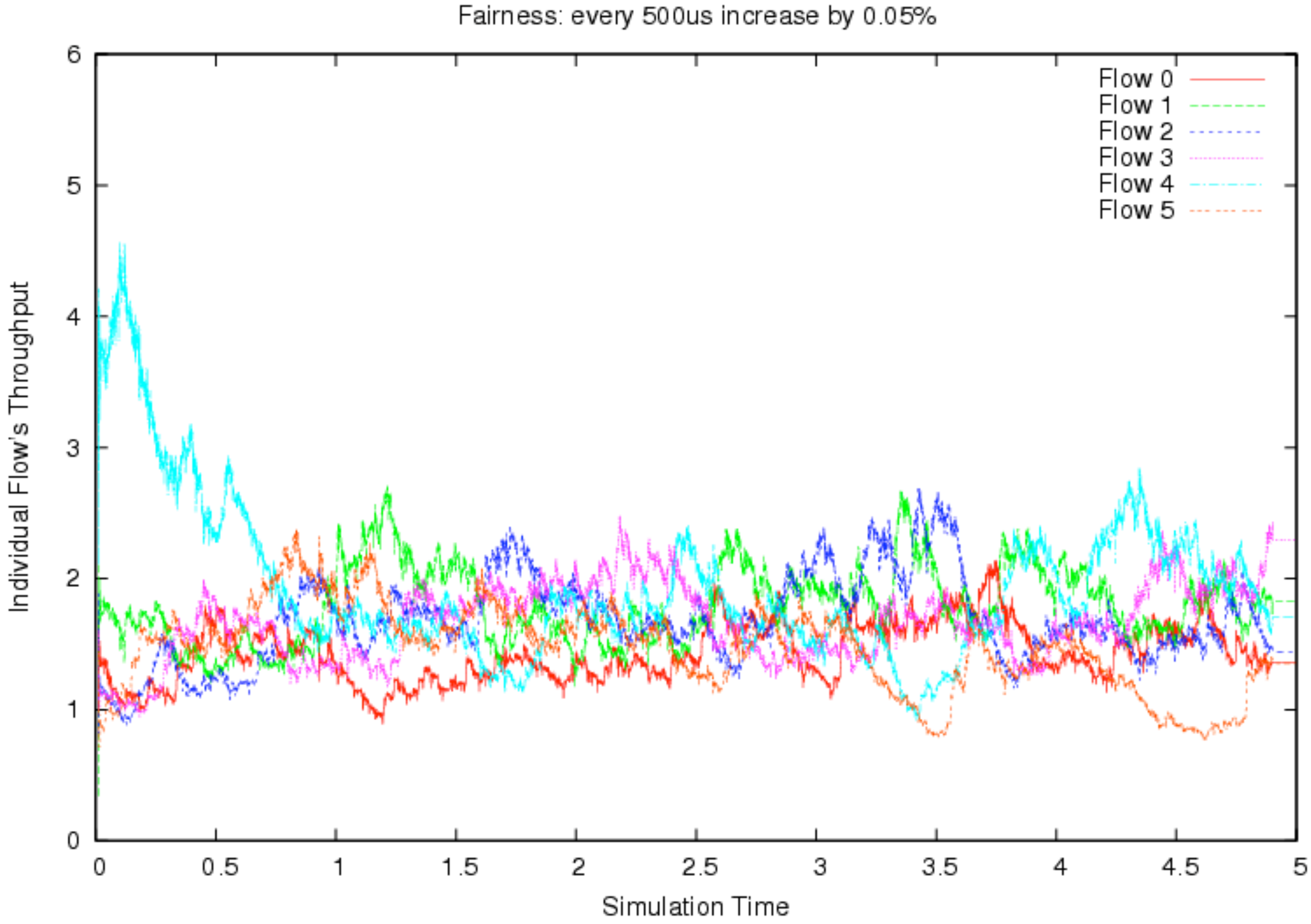
Queue Length: With Drift



Aggregate Rate: With Drift

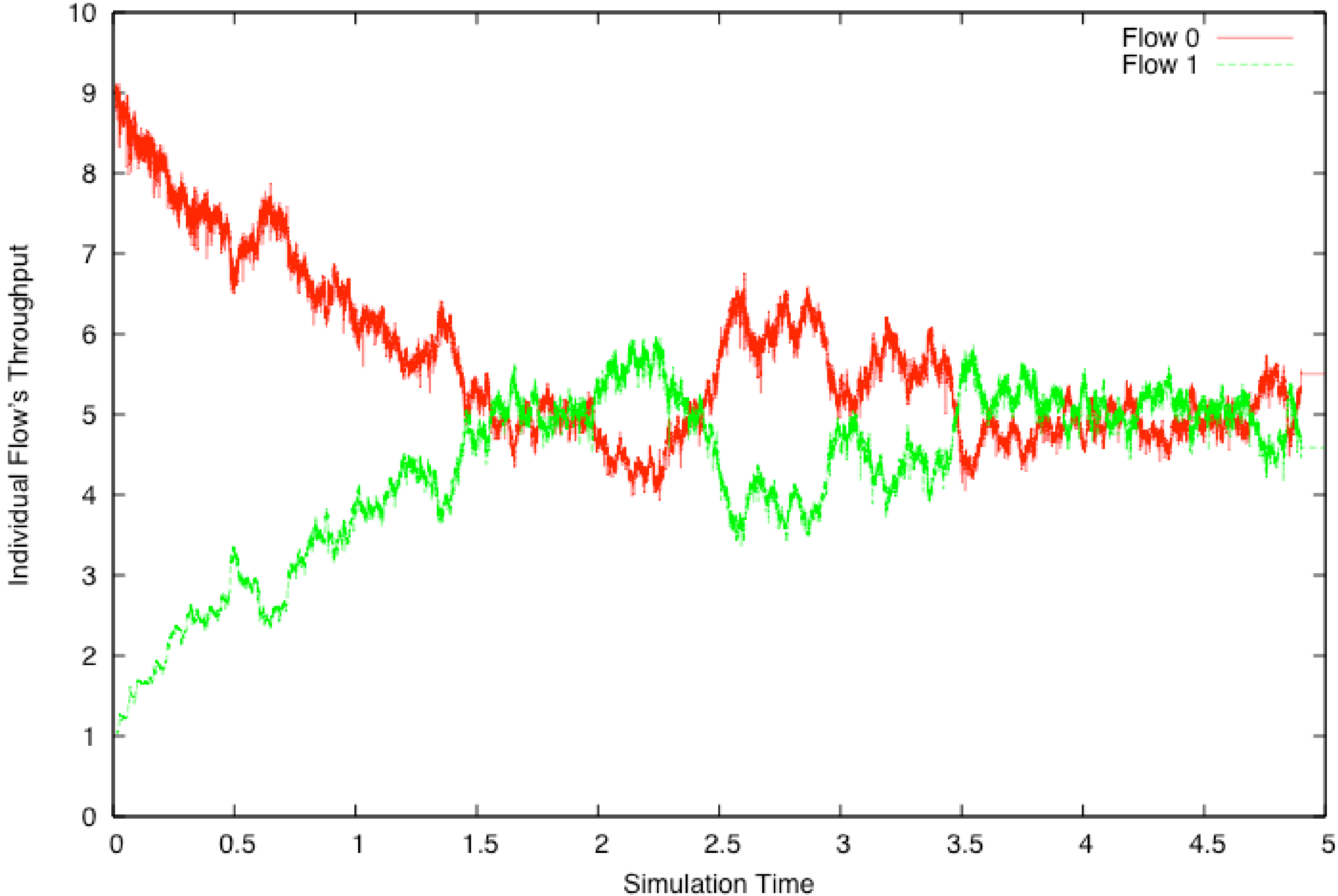


Instantaneous Rates: With Drift



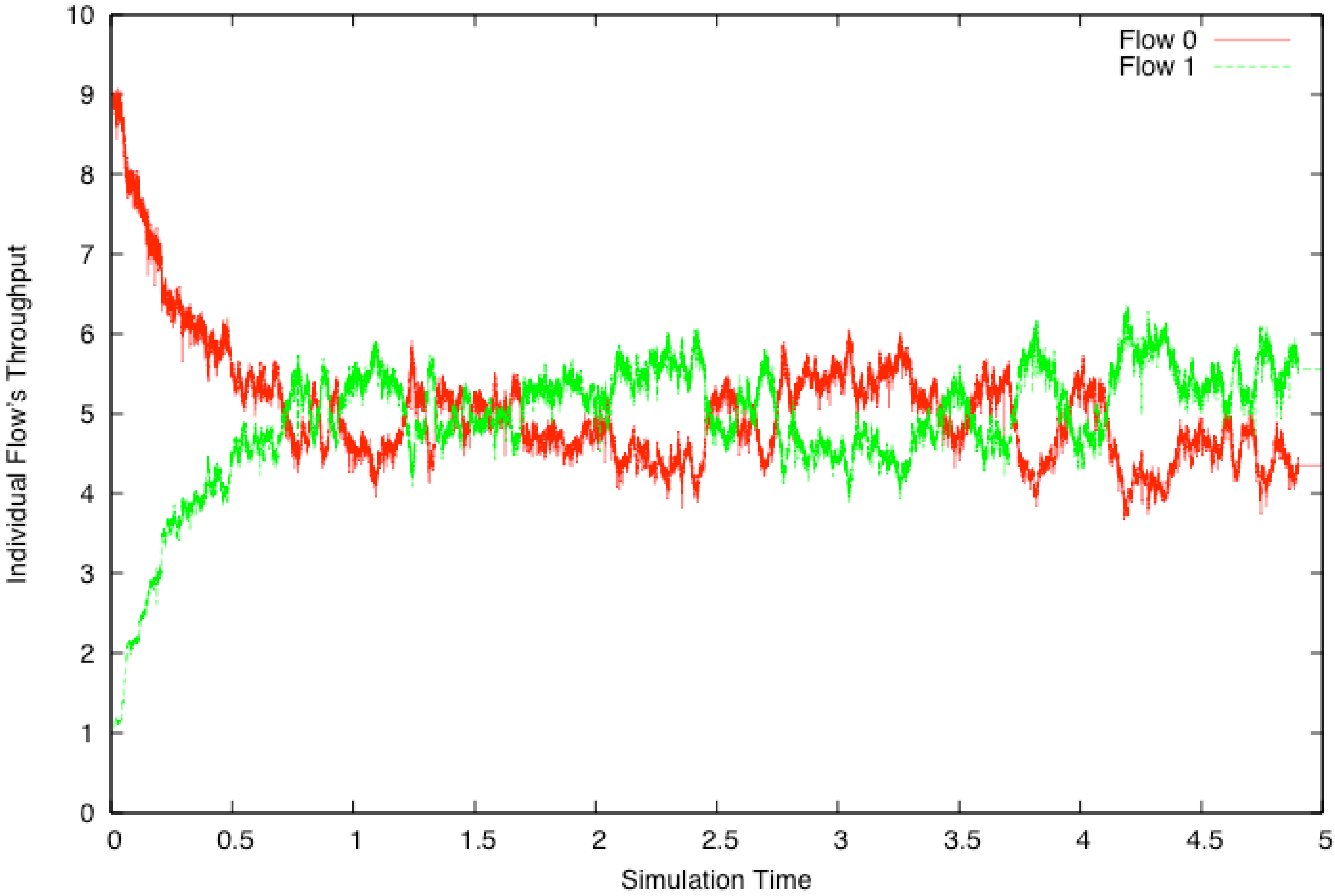
Fairness: No Drift

2 flows: 1 starting at 1Gbps, 1 starting at 9 Gbps



Fairness: With Drift

2 flows: 1 starting at 1Gbps, 1 starting at 9 Gbps

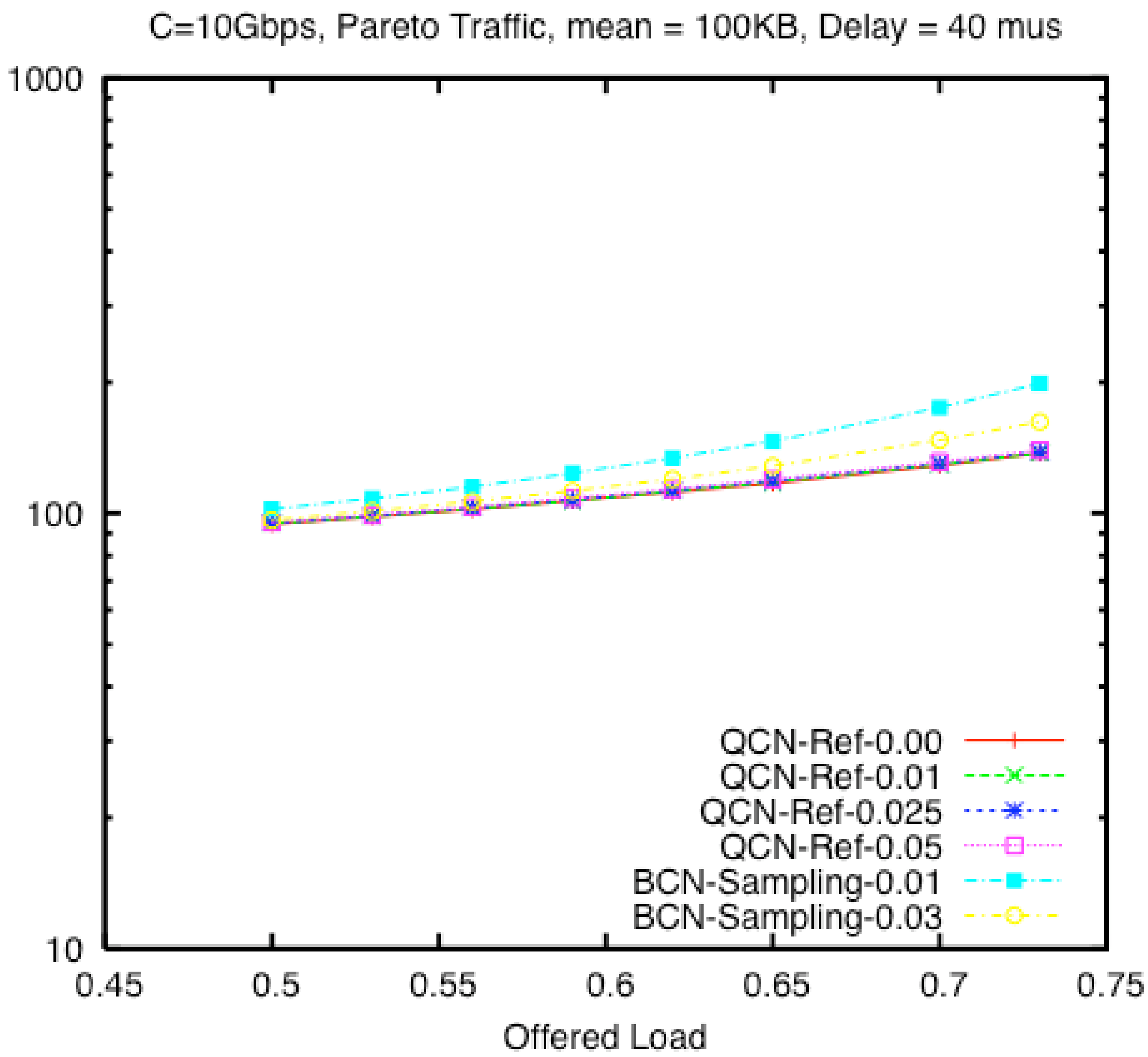


Dynamic flows: FCT and Drops

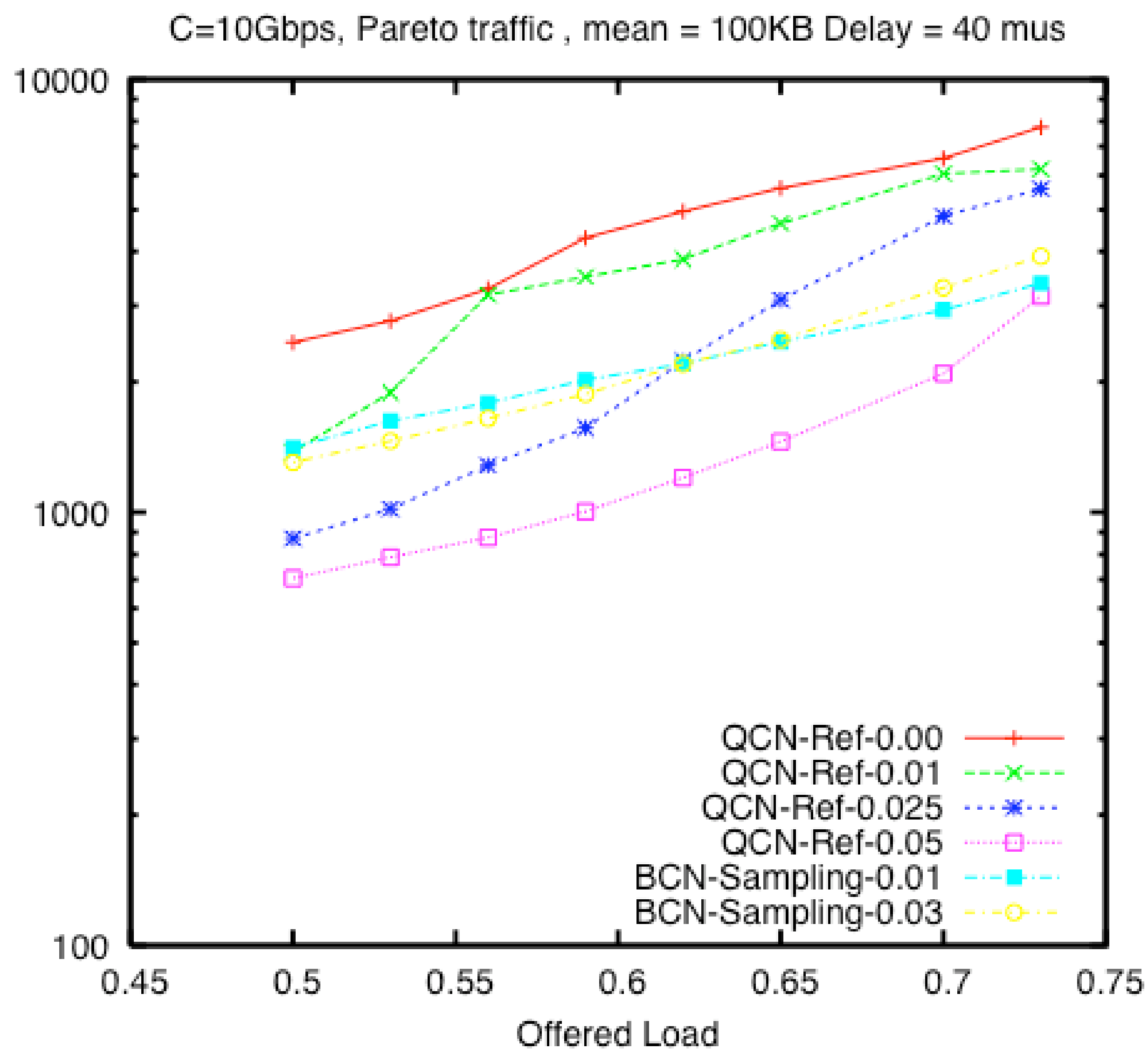
- Workload
  - IPC traffic: Mean = 5 KB (uniform distribution)
  - Data traffic: Pareto, shape 2, mean 100 KB
  - Parameters (Gd, w, etc): same as before
  - Reflection probability = 0, 2.5 and 5%

- BCN parameters
  - Gd = 1/128
  - Gi = 2.0
  - w = 2.0
  - Sampling Probability = 1% and 3%

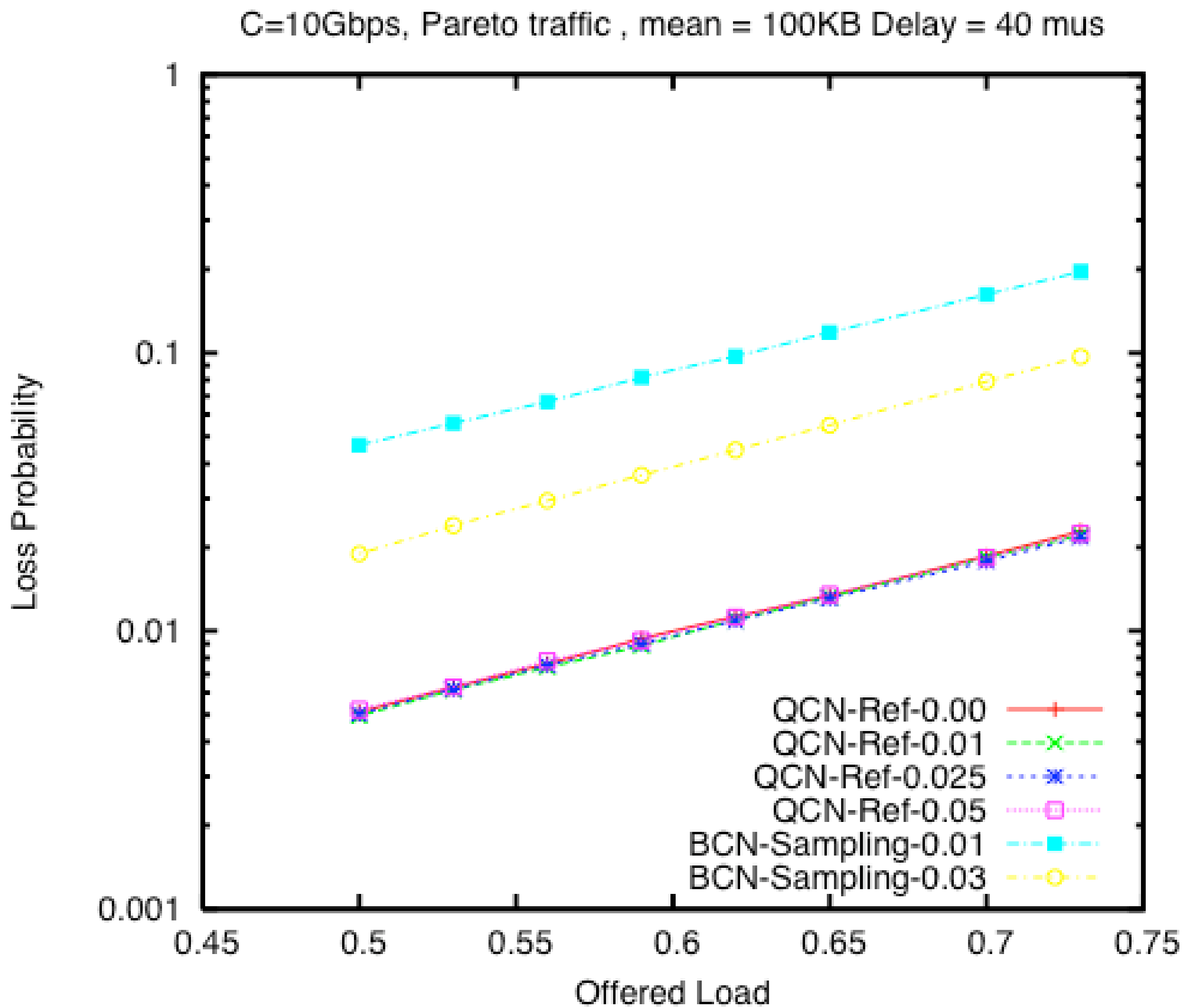
Completion Time of Short Flows



Completion Time of Long Flows



Packet Drops



Conclusion

- Drift needed for failsafe behavior
  - Very mild amount sufficient
  - Improves fairness