NADA: A Unified Congestion Control Scheme for Real-Time Media

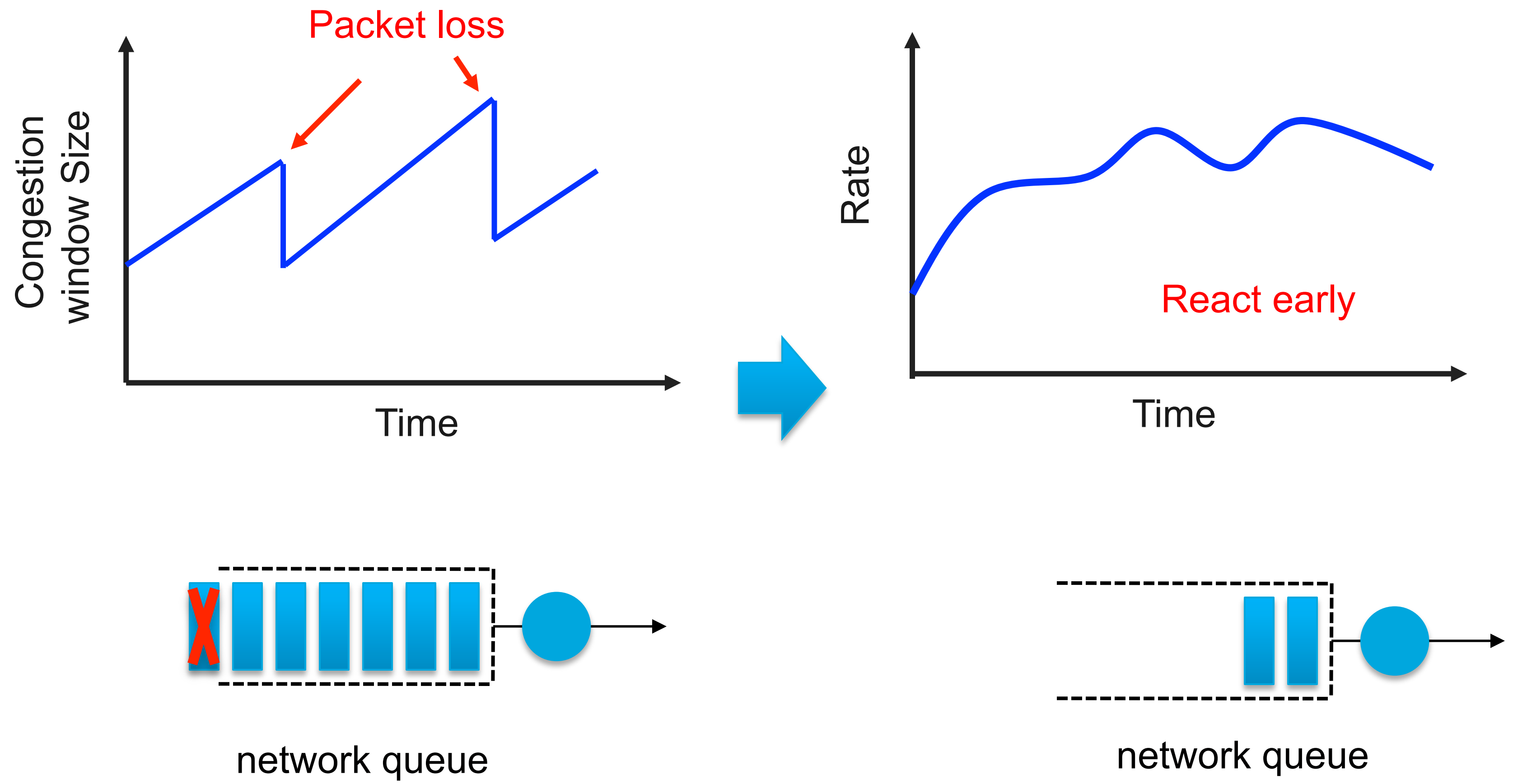
Xiaoqing Zhu and Rong Pan
Advanced Architecture & Research
Cisco Systems

August 2012

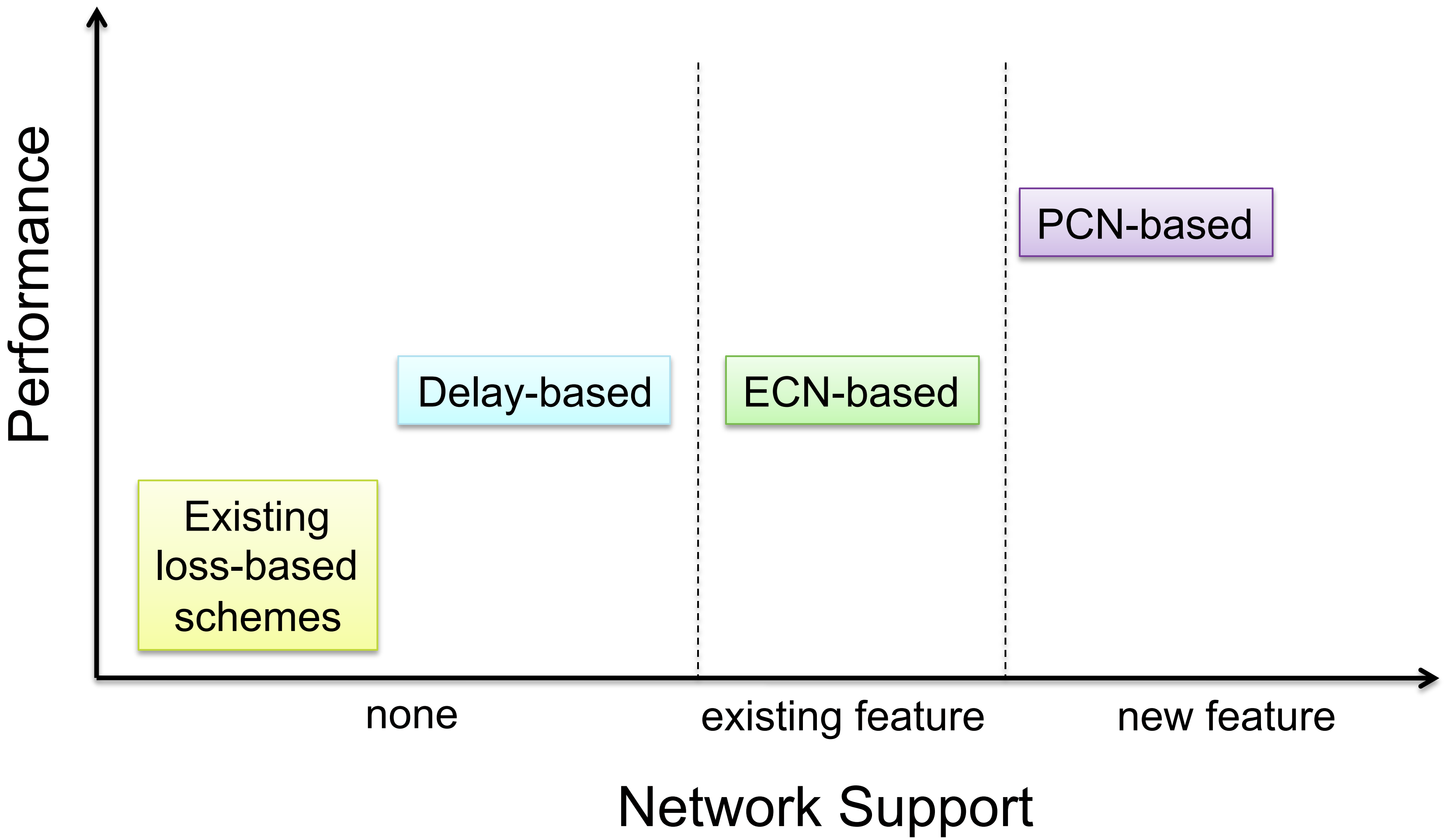
Agenda

- Design goals
- System model
- Network node operations
- Sender/Receiver behavior
- Evaluation results
- Open issues

Design Goal #1:
Limit Self-Inflicted Delay



Design Goal #2: Leverage A Suite of Feedback Mechanisms



Design Goal #3:
Weighted Bandwidth Sharing

Relative bandwidth

$$\bar{R}_i = \frac{R_i - R_i^{min}}{R_i^{max} - R_i^{min}}$$

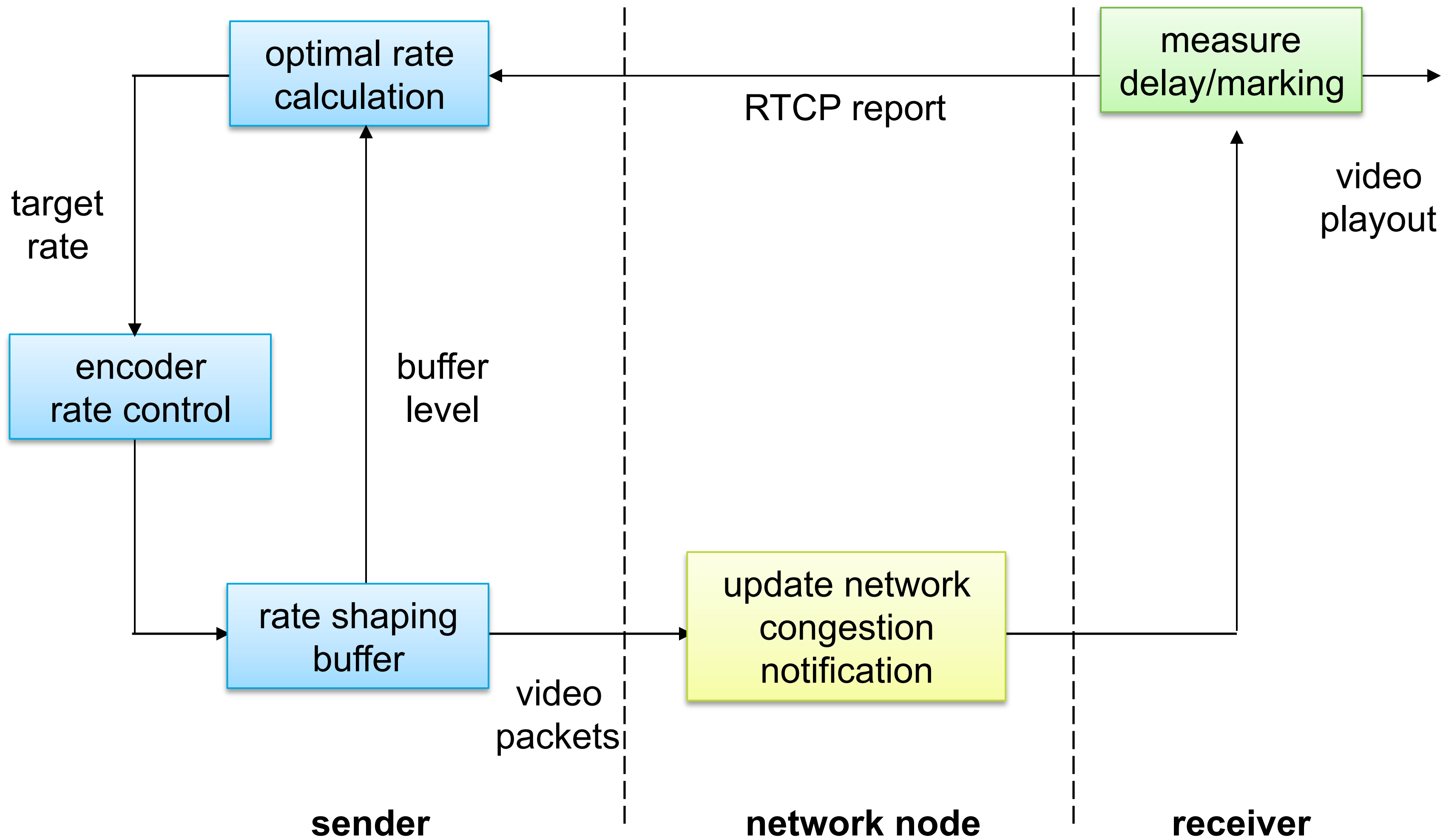
Application-level priority

$$w_i$$

Bandwidth sharing among flows:

$$\frac{\bar{R}_i}{w_i} = \frac{\bar{R}_j}{w_j}$$

System Overview



Network Node Behavior

- Queuing discipline: FIFO
- Congestion notification via:
  - Delay: no special operation at the queue
  - ECN: queue-based random marking
  - PCN: token-bucket-based random marking

Queue-based ECN Marking

if $q_{avg} < q_{lo}$:

no marking

else if $q_{avg} < q_{hi}$:

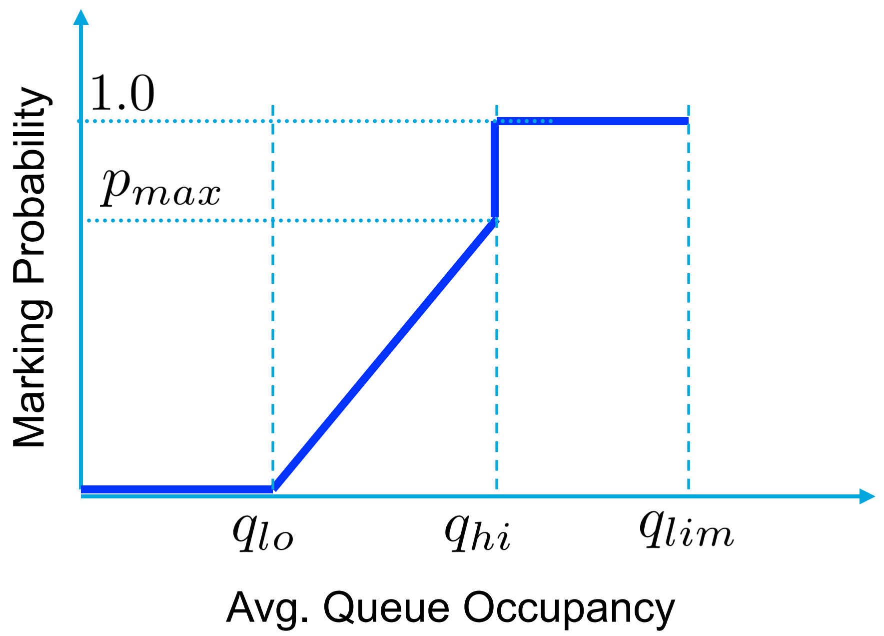
$$p = \frac{q_{avg} - q_{lo}}{q_{hi} - q_{lo}} p_{max}$$

else:

mark all packets

update:

$$q_{avg} = \alpha q + (1 - \alpha) q_{avg}$$

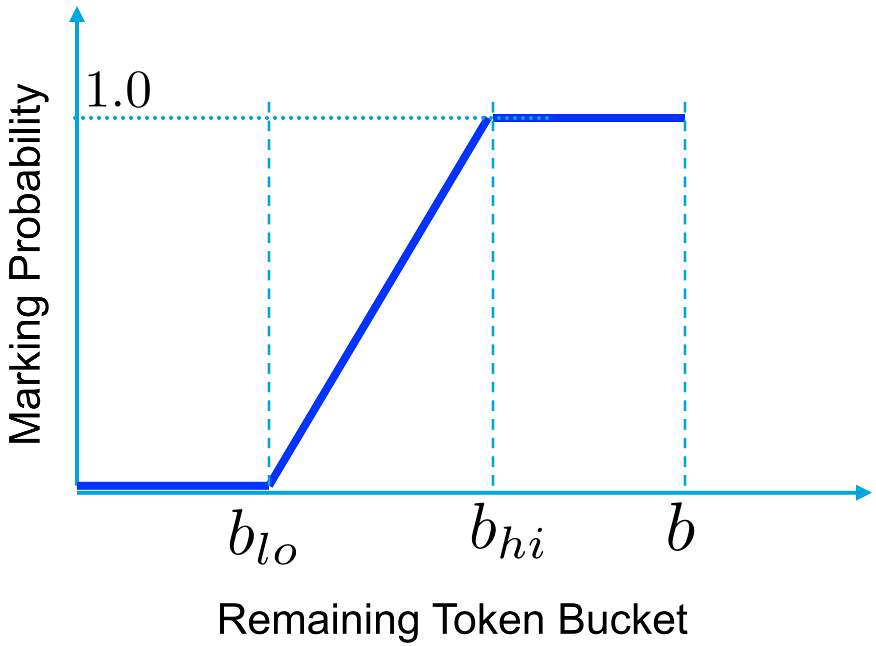


Token-bucket-based PCN Marking

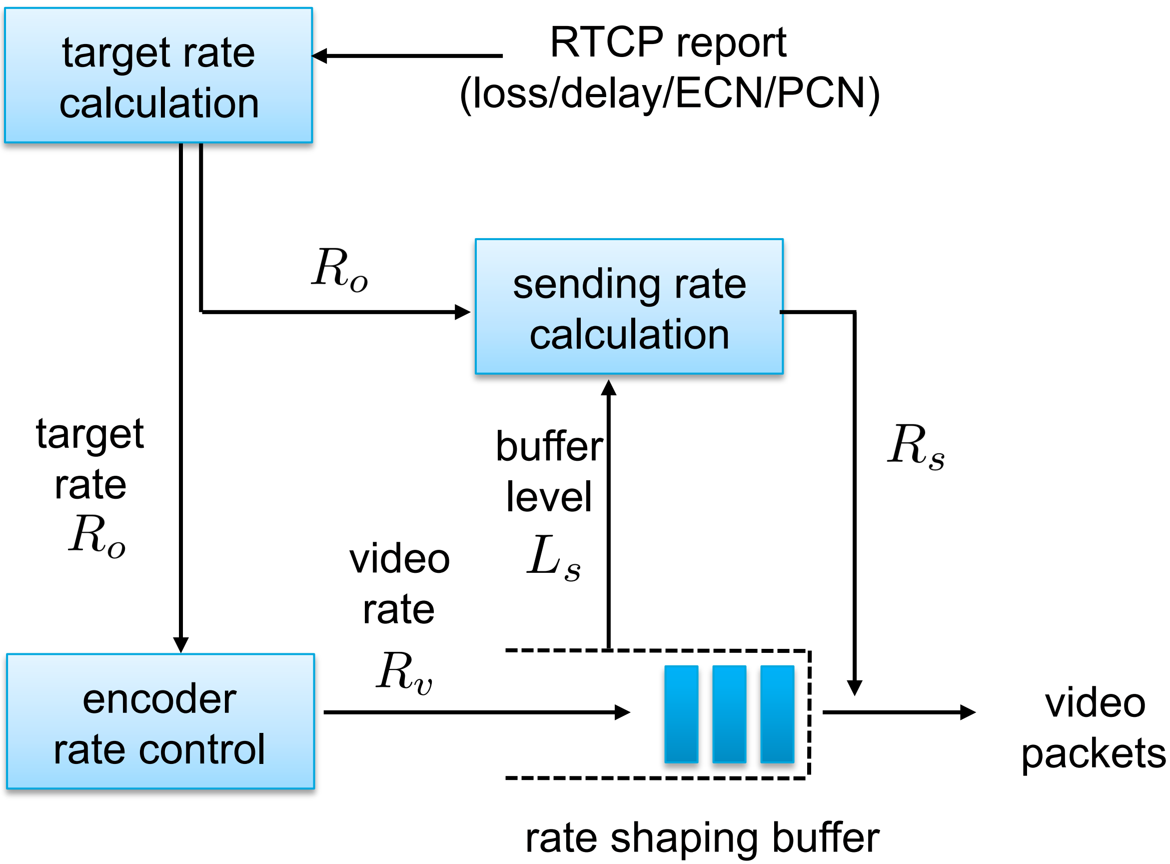
```
upon packet arrival:

  meter packet against token
  bucket (r, b) ;
  update token level b_tk

if b - b_tk < b_lo:

      no marking

else if b - b_tk < b_hi :
```

$$p = \frac{(b - b_{tk}) - b_{lo}}{b_{hi} - b_{lo}}$$

```
else:
      mark all packets
```



Sender Structure



Target Rate Calculation

- Reacting to delay:

$$R_o = R_{min} + \kappa \frac{w(R_{max} - R_{min})}{d}$$

session priority: w; scaling parameter: κ; queuing delay: d

- Reacting to ECN/PCN marking:

$$R_o = R_{min} + \eta \frac{w(R_{max} - R_{min})}{p}$$

scaling parameter: η; marking ratio: p

Sending Rate Calculation

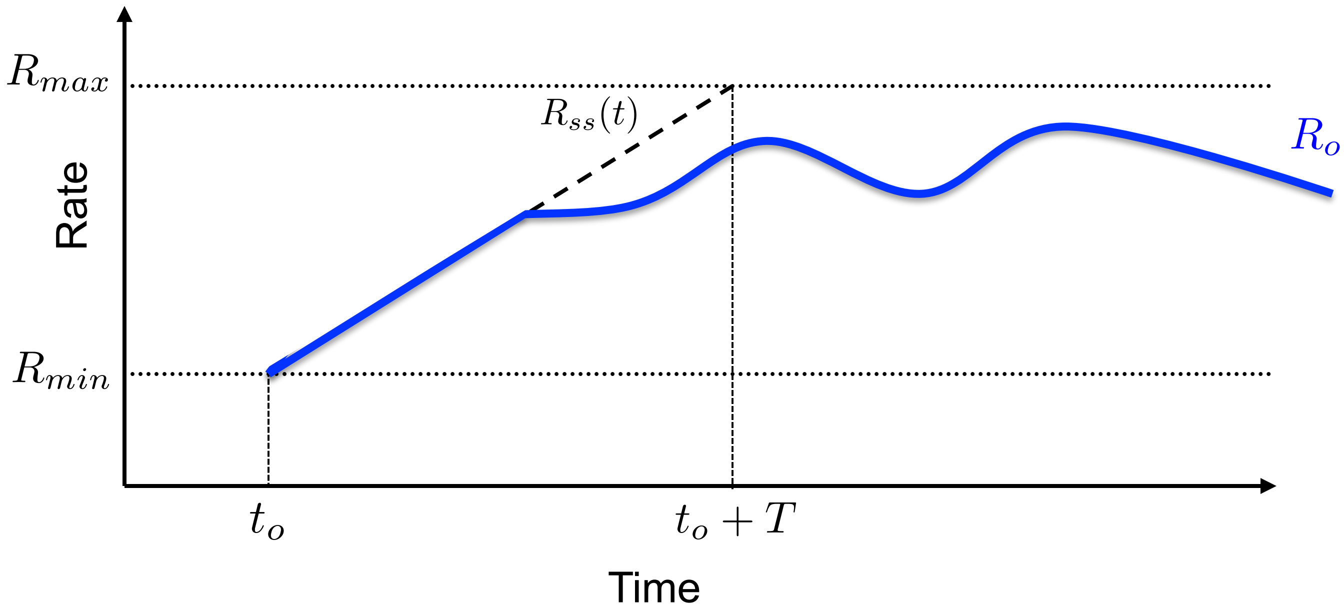
scaling parameter

$$R_s = R_o + \beta \frac{L_s}{\tau_v}$$

encoder reaction time

- Accommodate lag in encoder reaction
- Trade-off between between network queuing and rate shaping delay

Slow-Start Rate



$$R_{ss}(t) = R_{min} + \frac{t - t_o}{T}(R_{max} - R_{min})$$

start time

time horizon

Receiver Behavior

- Observe instantaneous end-to-end per packet statistics:

  Queuing delay: $d_n = t_{r,n} - t_{s,n} - \min_{n' \leq n} (t_{r,n'} - t_{s,n'})$

  ECN/PCN marking: $p_n \in \{0, 1\}$

- Obtain time-smoothed estimations:

$$d_{avg} = \alpha d_n + (1 - \alpha) d_{avg}$$

$$p_{avg} = \alpha p_n + (1 - \alpha) p_{avg}$$

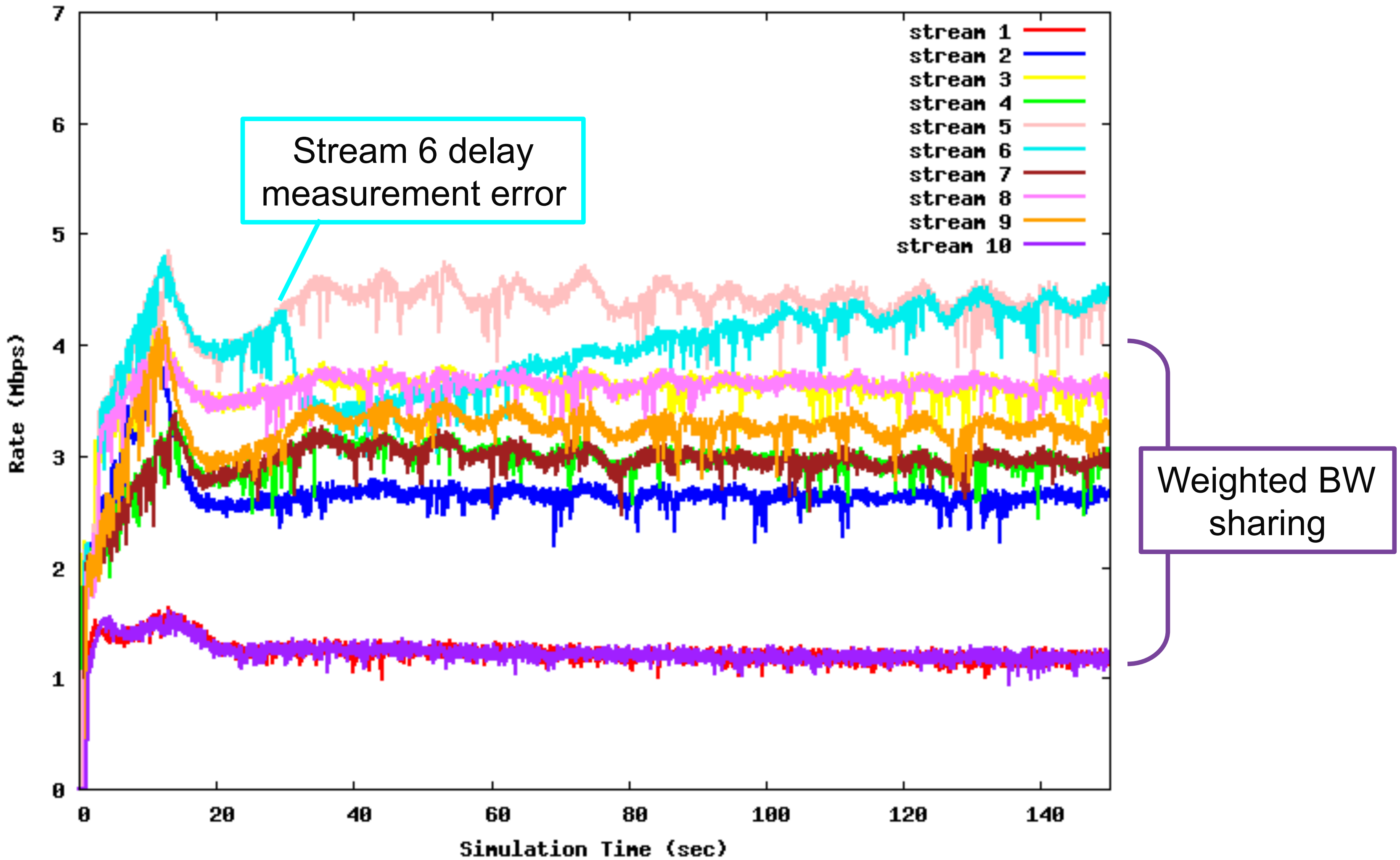
- Periodic RTCP reports (e.g., at 3% of received packets)

Test Scenario

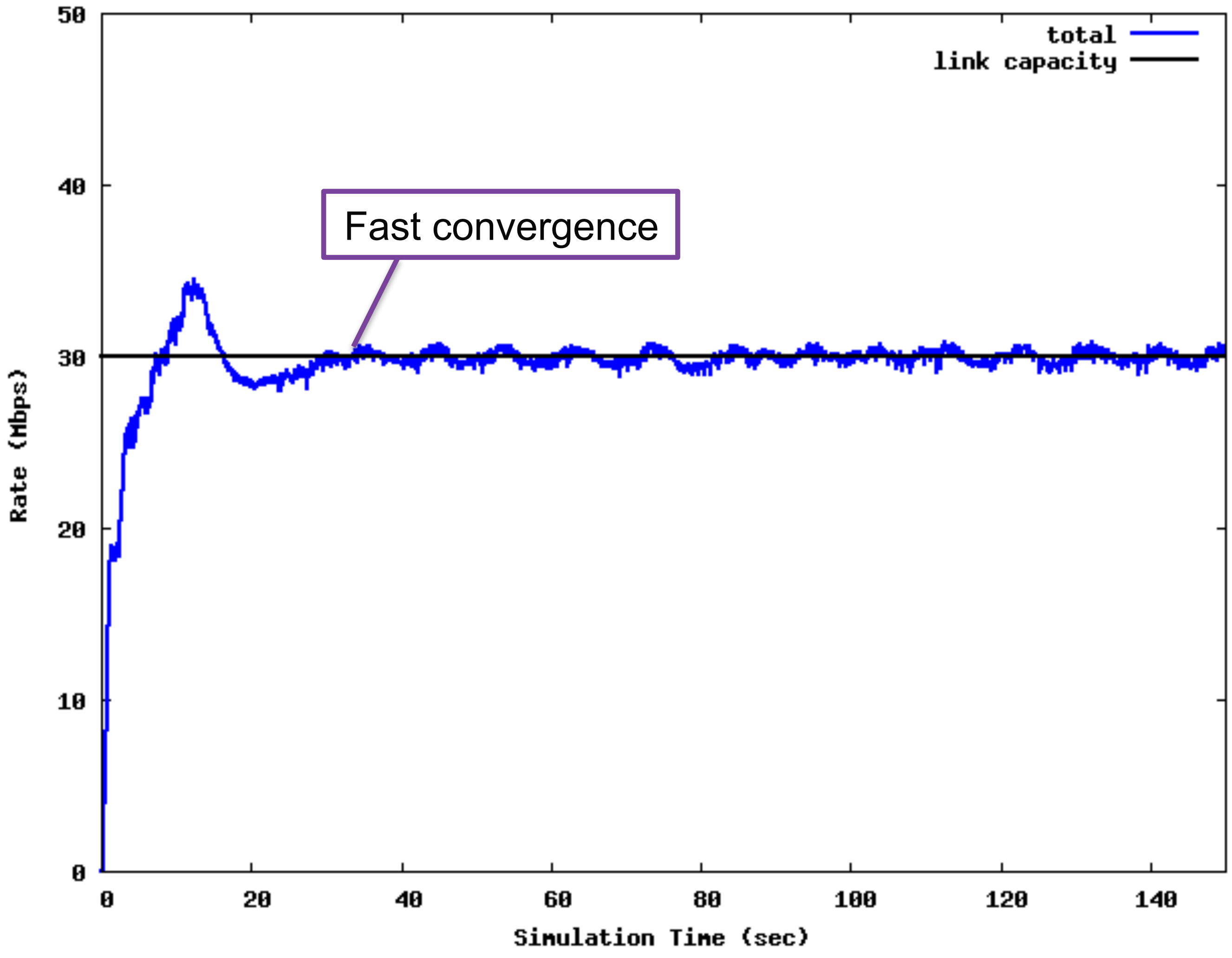
Flow ID	R_i^{min}	R_i^{max}	w_i
1	1	2	1
2	1	2	1
3	2	6	1
4	2	6	2
5	3	5	2
6	3	5	2
7	2	4	3
8	2	4	3
9	3	6	3
10	3	6	3

- Bottleneck bandwidth: 30Mbps
- Random delay measurement error for stream 6, at time t=30s

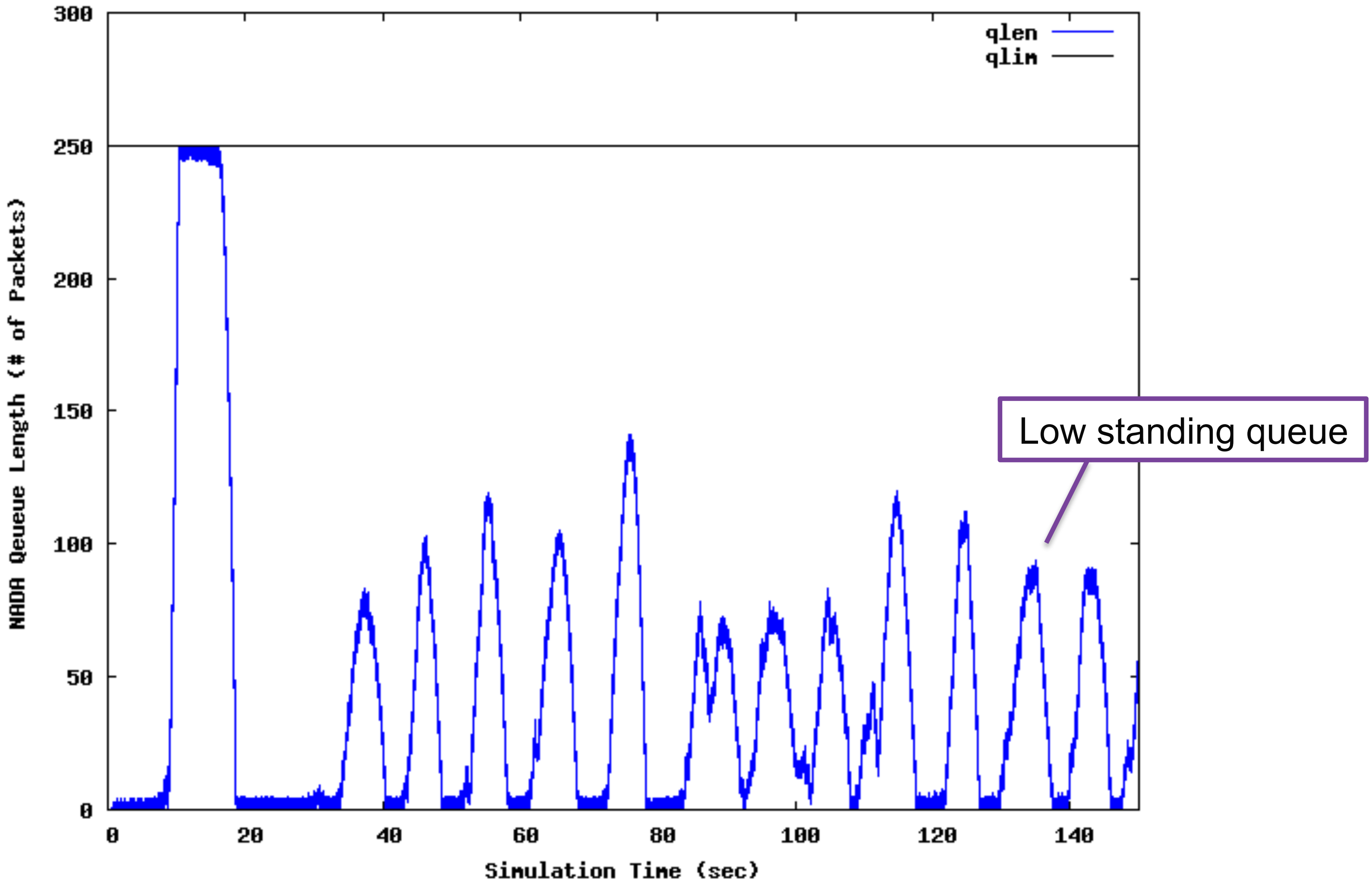
Delay-Based Adaptation: Per-flow Rate



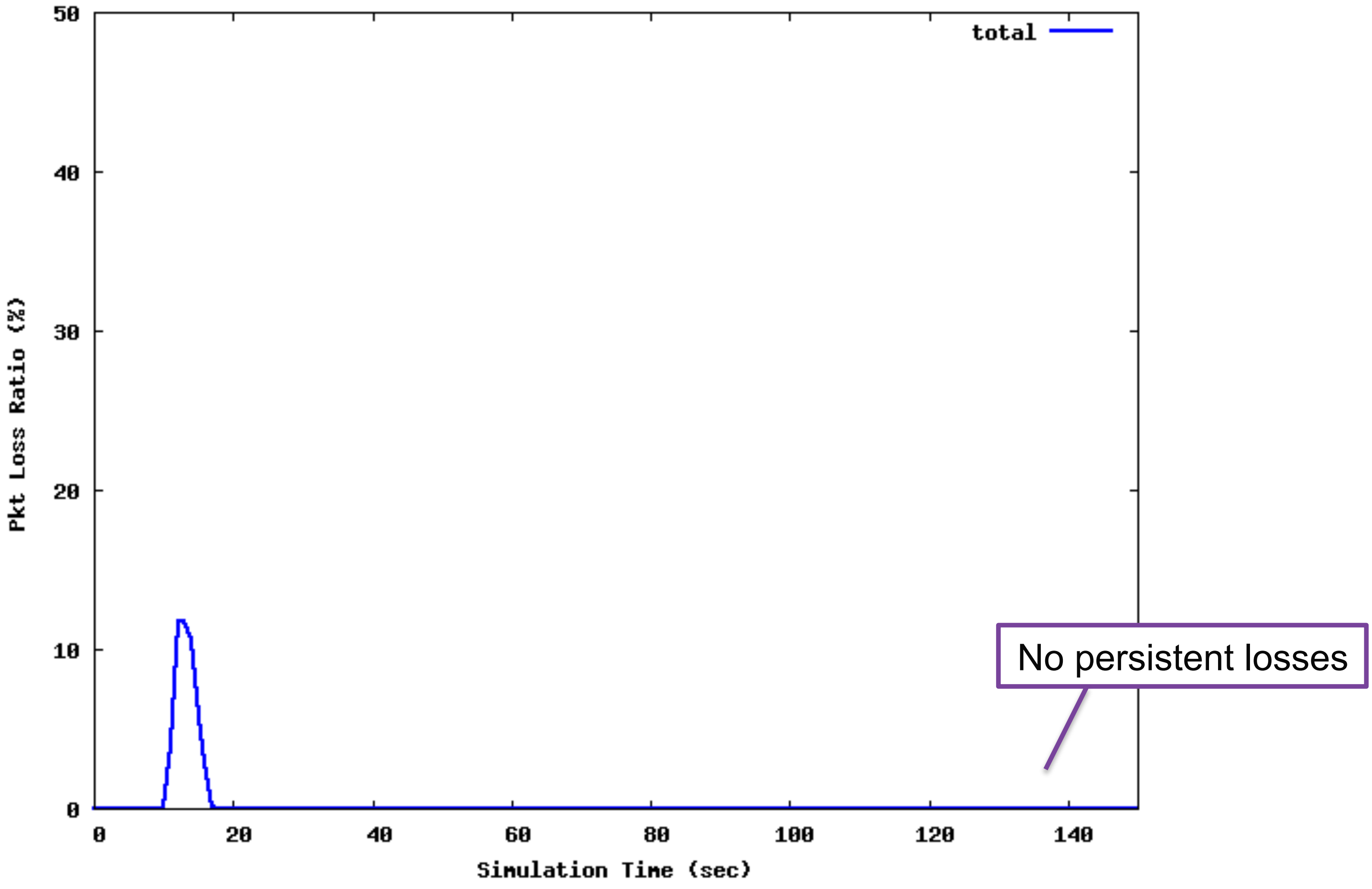
Delay-Based Adaptation: Total Rate



Delay-Based Adaptation: Bottleneck Queue

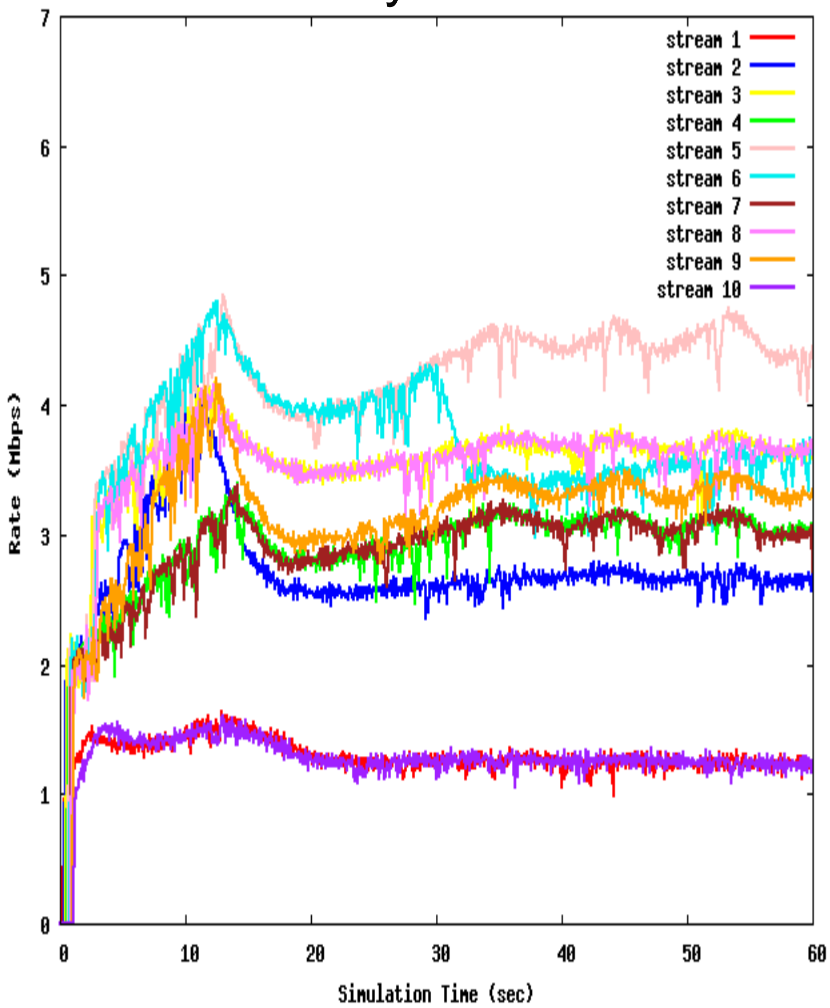


Delay-Based Adaptation: Packet Loss Ratio

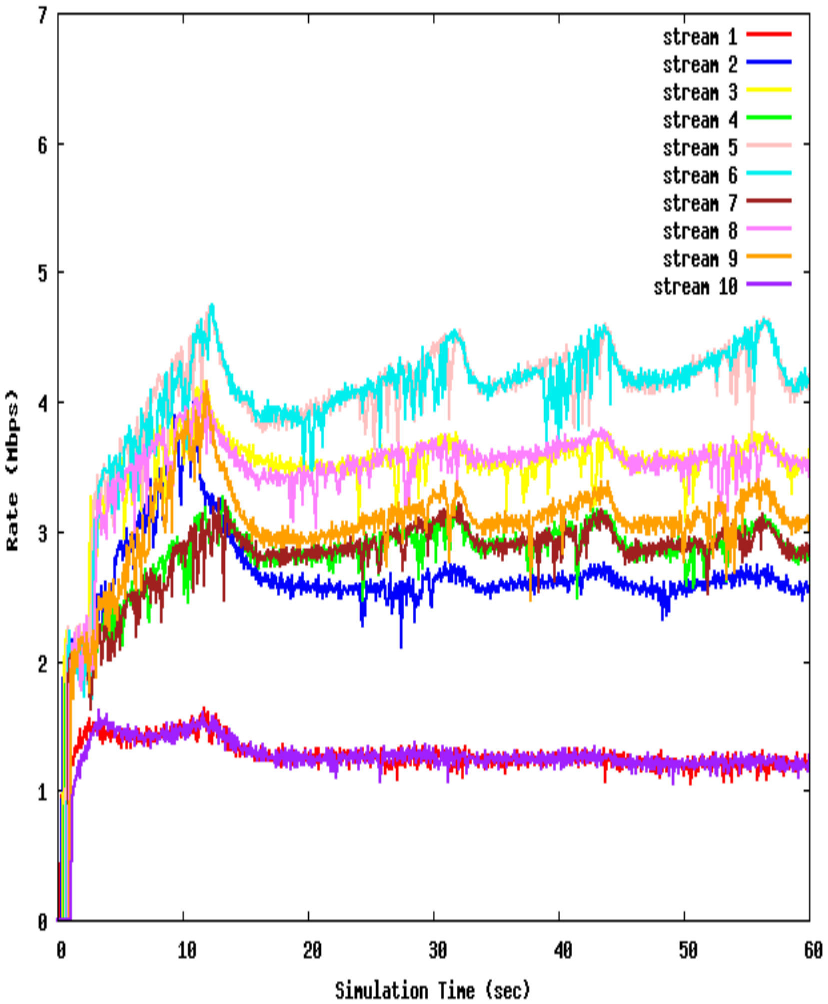


Delay vs. ECN: Per-Flow Rate

Delay-Based

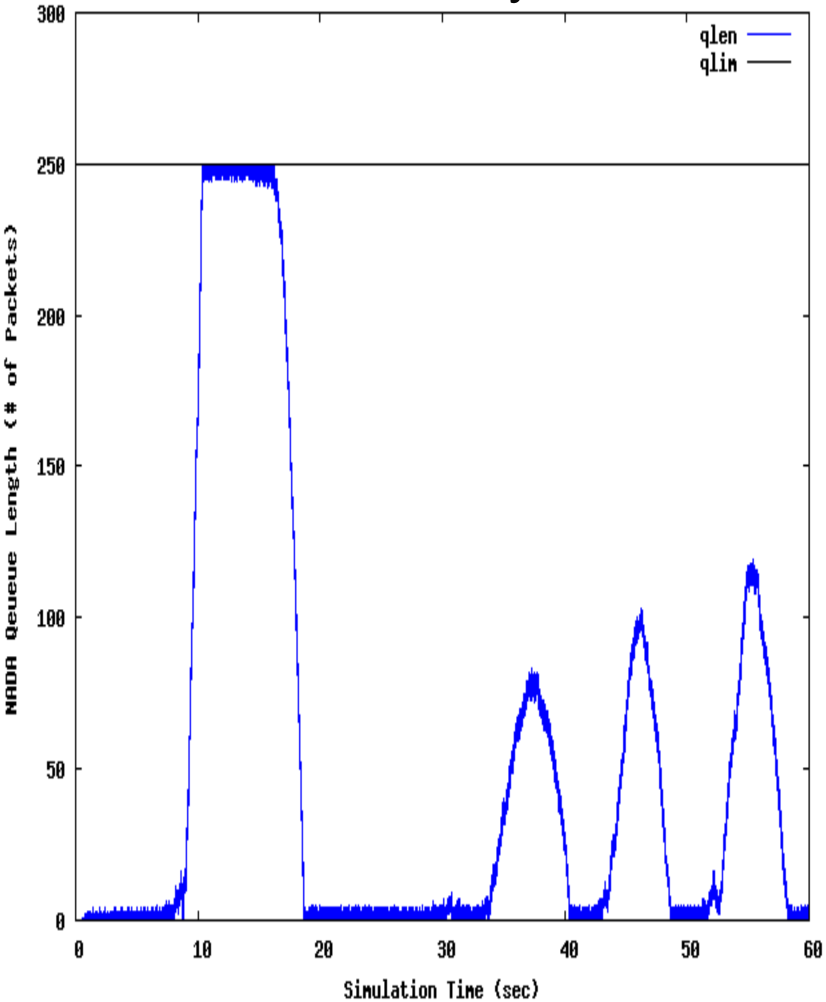


ECN-Based

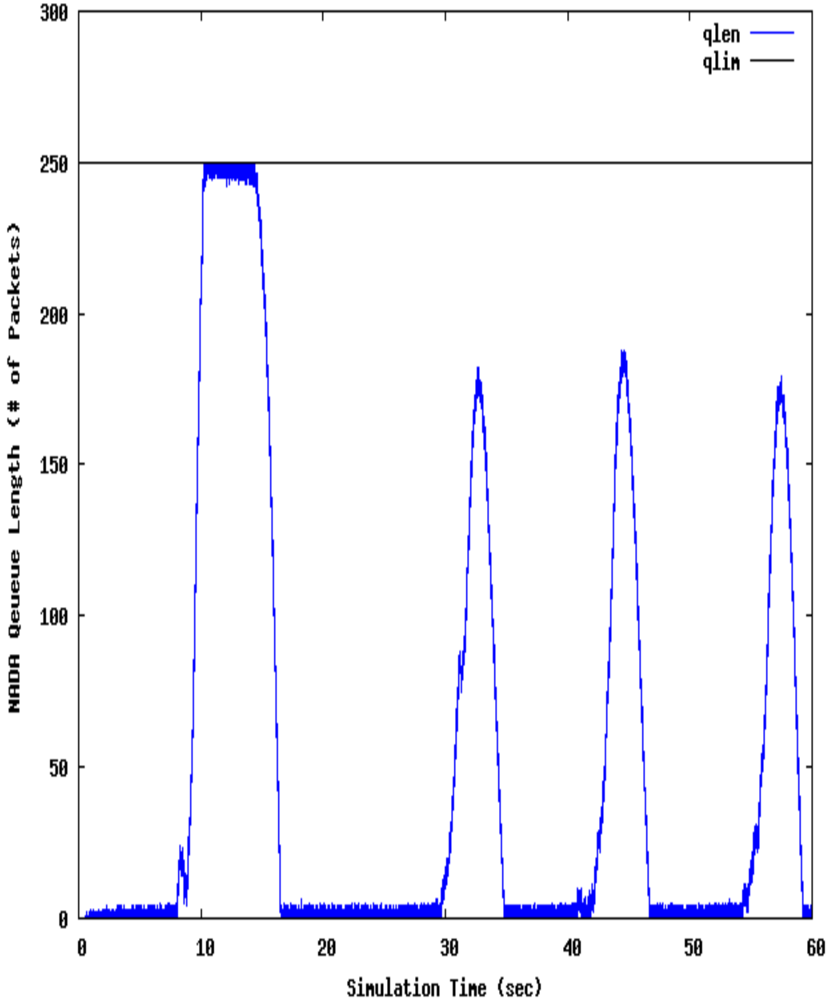


Delay vs. ECN: Bottleneck Queue

Delay

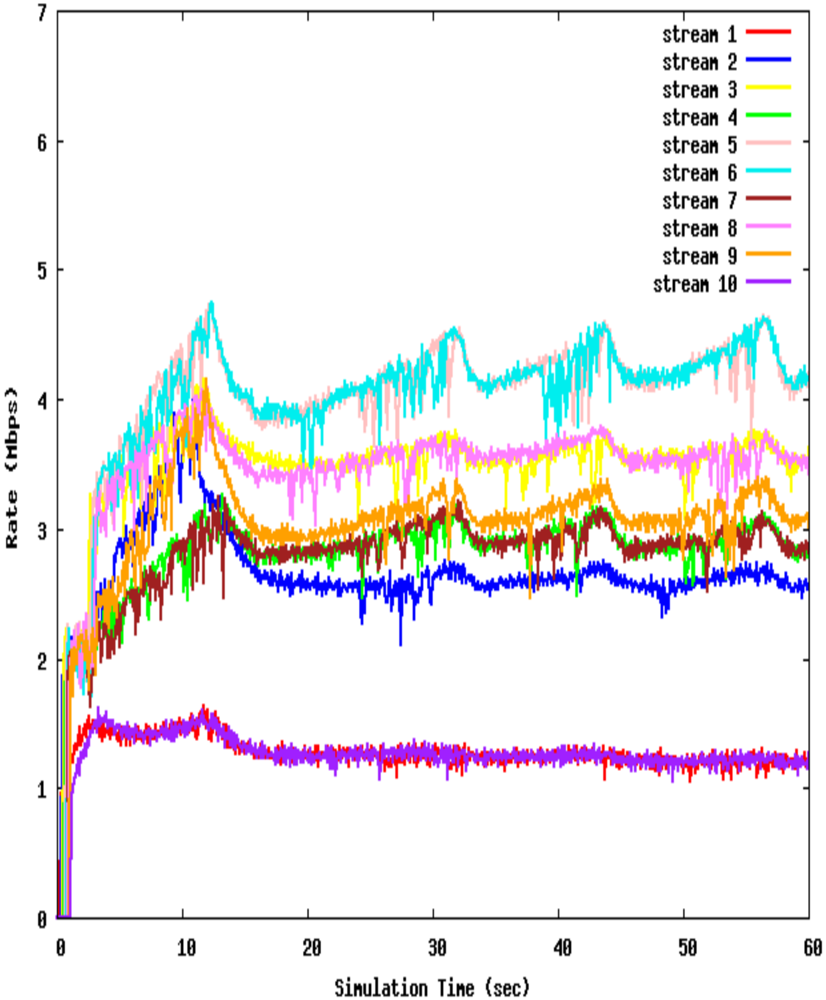


ECN

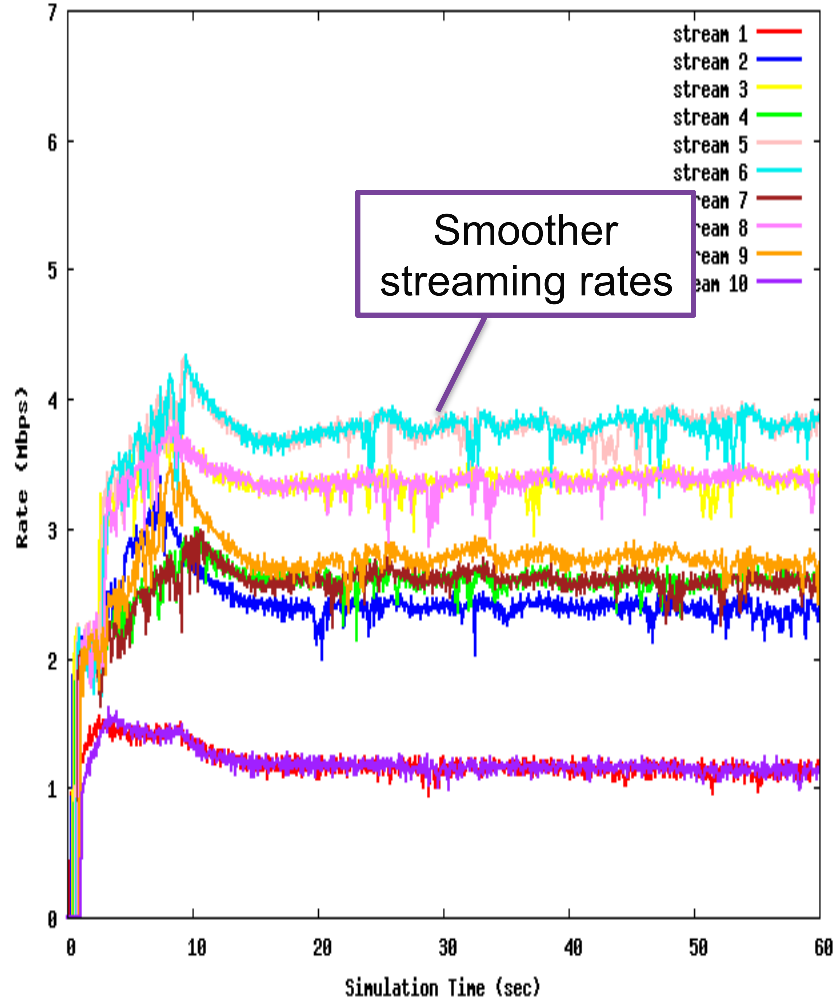


ECN vs. PCN: Per-Flow Rate

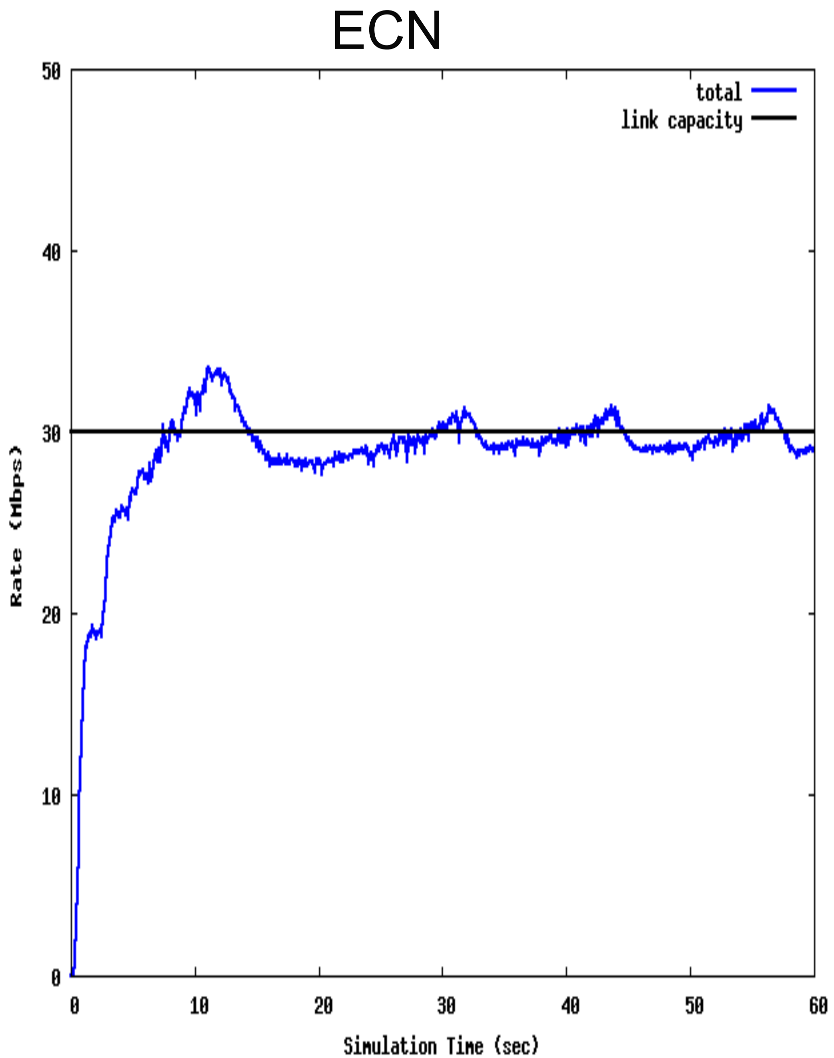
ECN

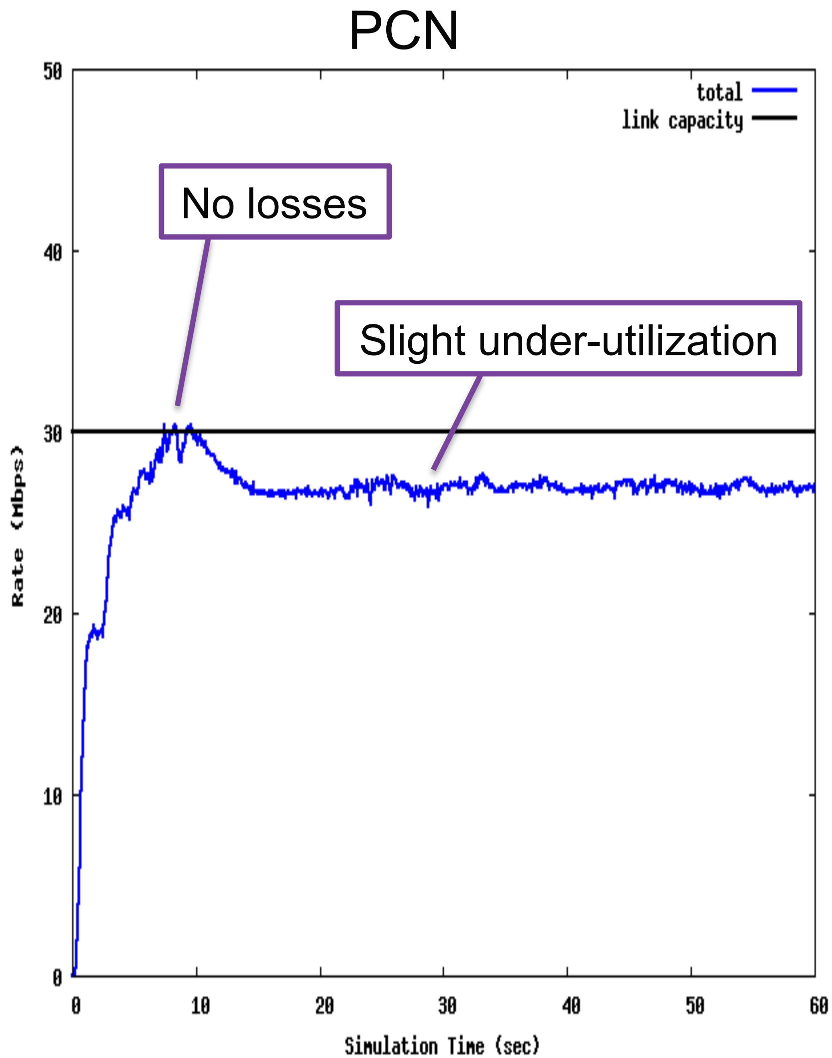


PCN



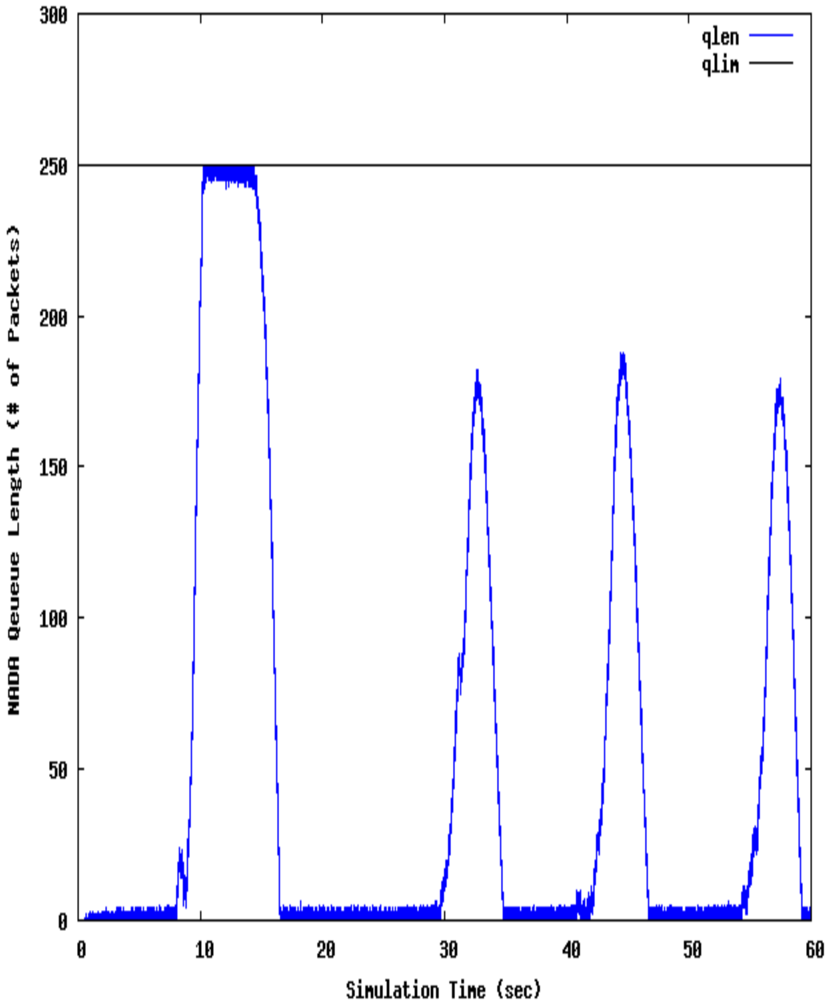
ECN vs. PCN: Total Rate



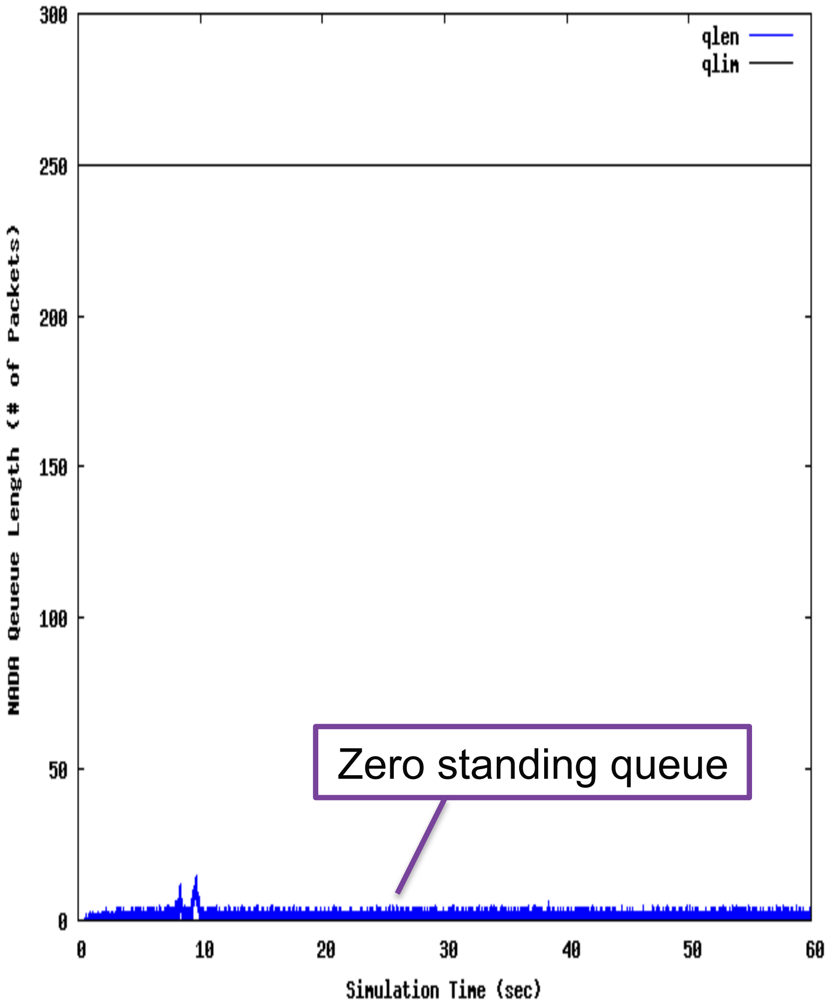


ECN vs. PCN: Bottleneck Queue

ECN



PCN



Conclusions and Next Steps

- Key benefits of NADA:
  - Fast rate adaptation
  - Weighted bandwidth sharing
  - Can work with a range of congestion signals
  - In case of PCN: zero standing queue and smoother streaming rates
- Next steps:
  - Future evaluations in linux-based implementations
  - Graceful transition between different congestion signals
  - Compete robustly against loss-based schemes