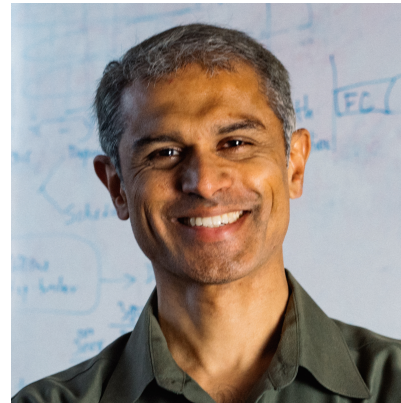


Copa: Practical Delay-Based Congestion Control



Venkat Arun and Hari Balakrishnan

MIT, CSAIL
web.mit.edu/copa

The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

Higher bandwidth-delay product

Greater bandwidth $\Rightarrow$ Lower tolerance for non-congestive loss

Greater flow-churn

The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

Higher bandwidth-delay product

Greater bandwidth $\Rightarrow$ Lower tolerance for non-congestive loss

Greater flow-churn

Large flows (e.g. video streaming) co-exist with short-flows

The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

Higher bandwidth-delay product

Greater bandwidth $\Rightarrow$ Lower tolerance for non-congestive loss

Greater flow-churn

Large flows (e.g. video streaming) co-exist with short-flows

Wireless links with variable bandwidths are commonplace

The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

Higher bandwidth-delay product

Greater bandwidth $\Rightarrow$ Lower tolerance for non-congestive loss

Greater flow-churn

Large flows (e.g. video streaming) co-exist with short-flows

Wireless links with variable bandwidths are commonplace

Simultaneously, users are more sensitive to performance!

Loss-based schemes have long-standing problems

- Buffer-filling
 - Vulnerable to non-congestive loss
 - Loss is a coarse signal
- 
- Worsens** with increasing bandwidth

Delay-based congestion control?

Benefits

Challenges

Delay-based congestion control?

Benefits

Maintain low delay

Challenges

Not competitive with
buffer-filling schemes

Delay-based congestion control?

Benefits

Maintain low delay

Robust to misleading loss

Challenges

Not competitive with
buffer-filling schemes

Delay can mislead too!

Delay-based congestion control?

Benefits

Maintain low delay

Robust to misleading loss

Rich signal

Challenges

Not competitive with
buffer-filling schemes

Delay can mislead too!

Delay-based congestion control?

Benefits

Maintain low delay

Robust to misleading loss

Rich signal

Challenges

Not competitive with
buffer-filling schemes

Delay can mislead too!

Finding true minimum RTT
is hard

Delay-based congestion control?

Challenges

Not competitive with
buffer-filling schemes

Delay is noisy too!

Finding true minimum RTT
is hard

Our Solution

Delay-based congestion control?

Challenges

Not competitive with
buffer-filling schemes

Delay is noisy too!

Finding true minimum RTT
is hard

Our Solution

If buffer-fillers are present,
give up on low delay

Delay-based congestion control?

Challenges

Not competitive with buffer-filling schemes

Delay is noisy too!

Finding true minimum RTT is hard

Our Solution

If buffer-fillers are present, give up on low delay

MIN: a more robust statistic for queuing delay

Delay-based congestion control?

Challenges

Not competitive with buffer-filling schemes

Delay is noisy too!

Finding true minimum RTT is hard

Our Solution

If buffer-fillers are present, give up on low delay

MIN: a more robust statistic for queuing delay

Empty queues periodically

Basic Goals

Avoid **congestion collapse**

+

Efficient and **Fair** allocation of bandwidth

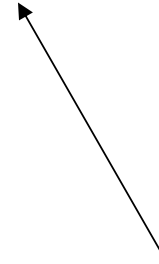
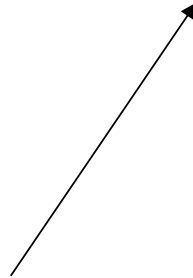
+

Low delay

$$\text{Target rate} = r_t = \frac{1}{\delta \cdot d_q}$$

Adjustable Parameter
default = 0.5

Queuing delay



Target Rate $\equiv$ Nash Equilibrium

Selfishly optimize for:

$$Utility_i = \log(tput) - \delta_i \log(d_q)$$

Assuming Poisson arrivals (more details in paper)

Target Rate $\equiv$ Nash Equilibrium



Unique and Efficient

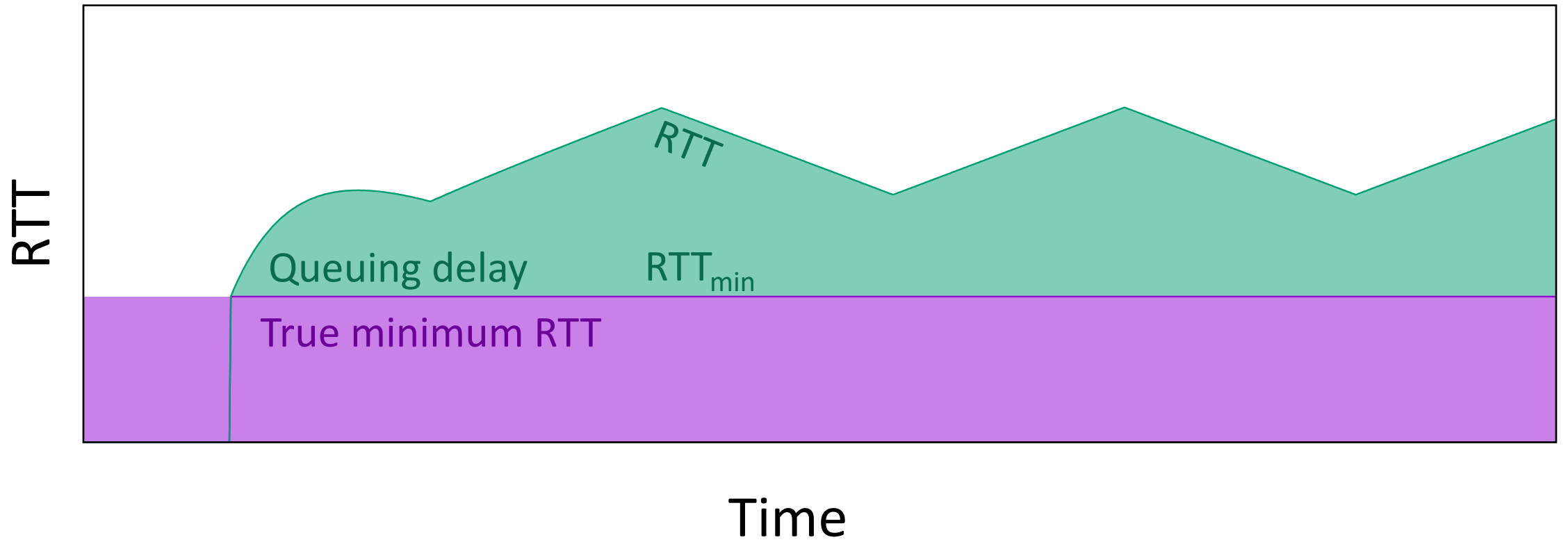
Selfishly optimize for:

$$Utility_i = \log(tput) - \delta_i \log(d_q)$$

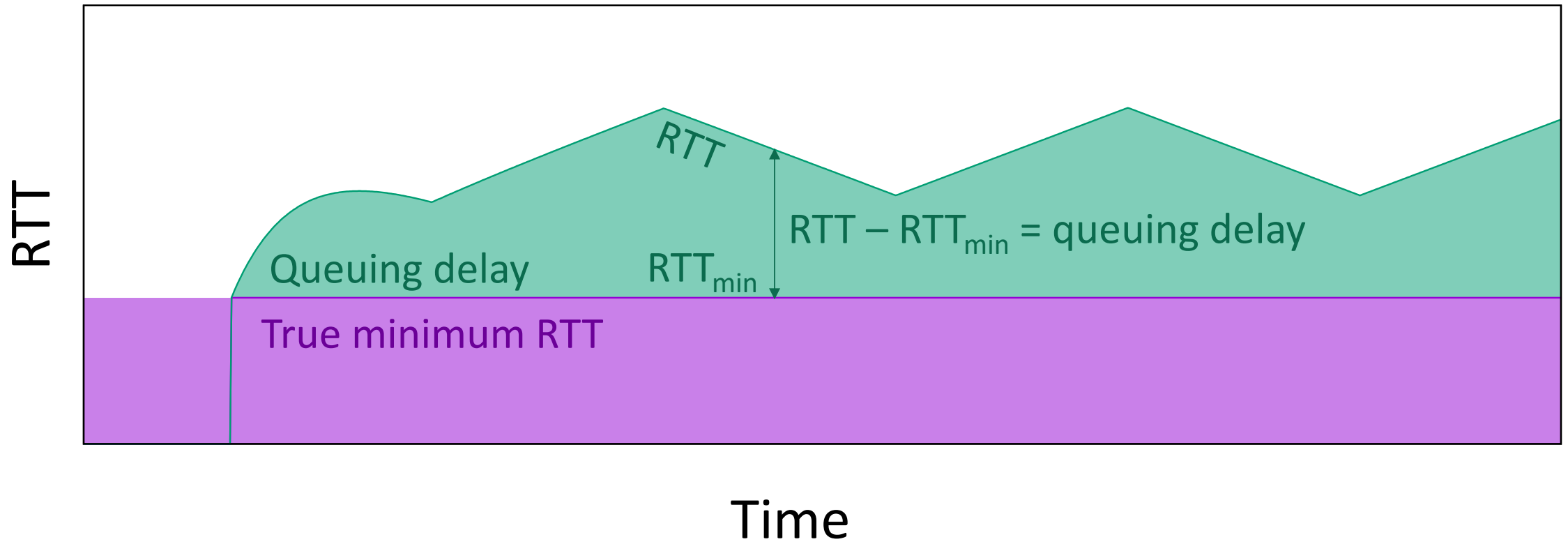
Assuming Poisson arrivals (more details in paper)

Computing the Target Rate

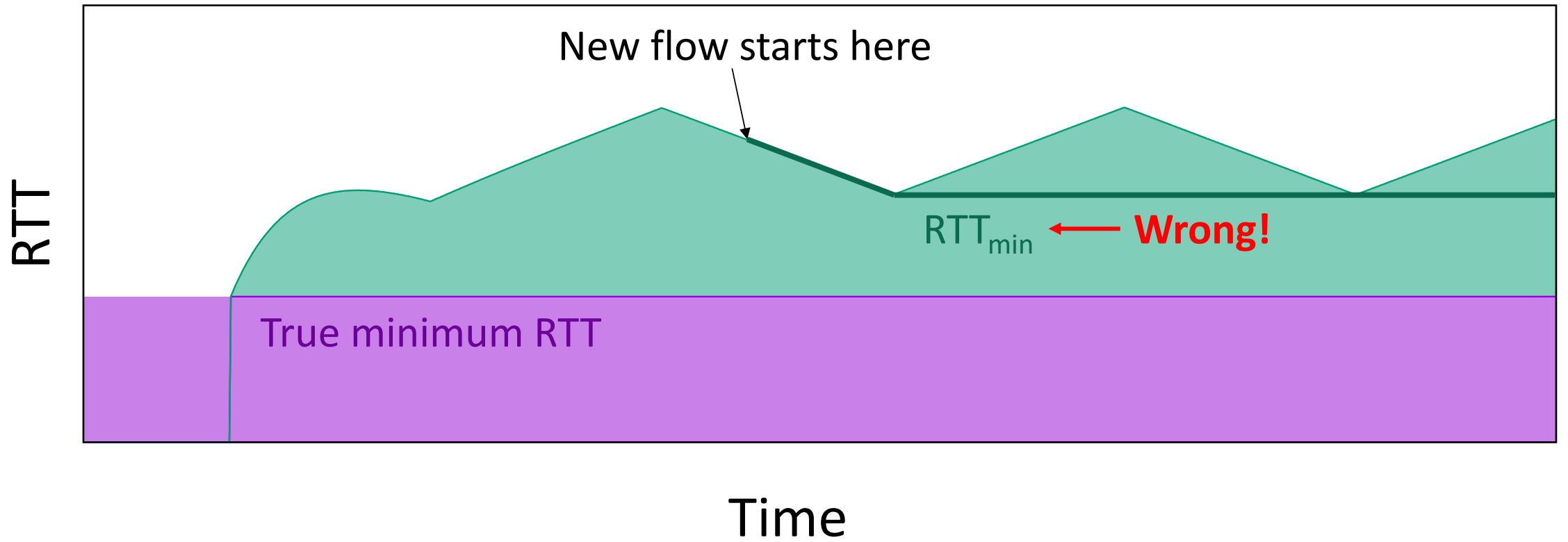
Estimating queuing delay from RTT



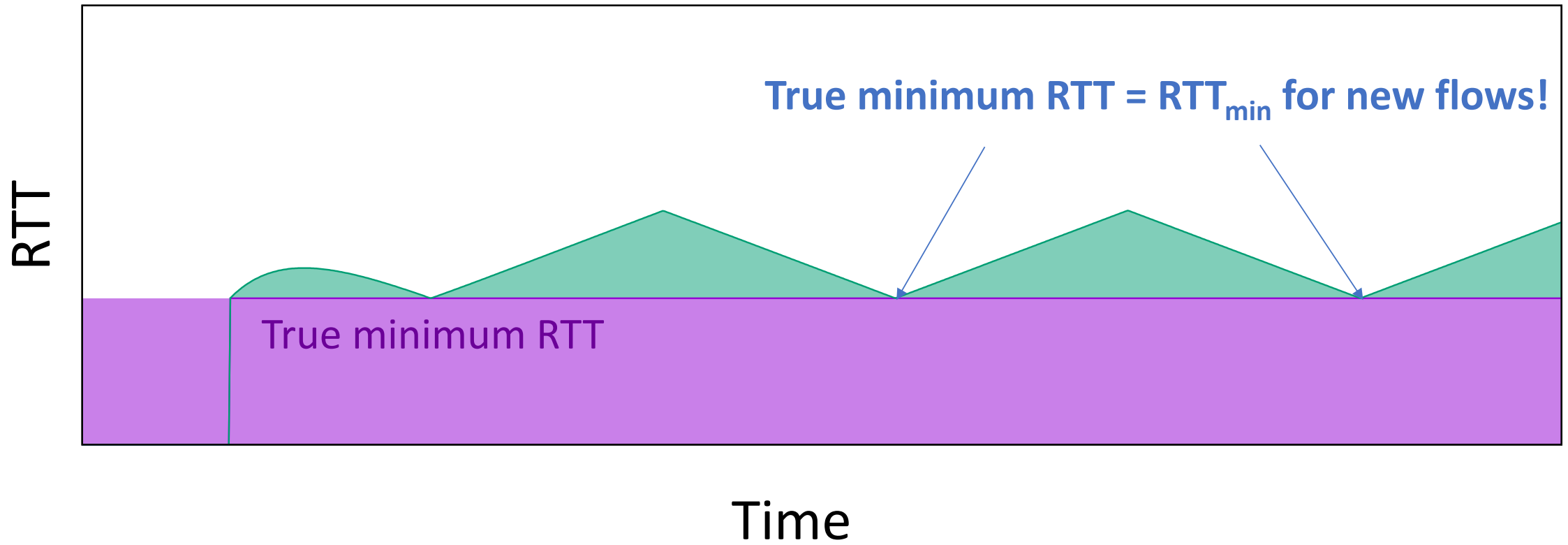
Estimating queuing delay from RTT



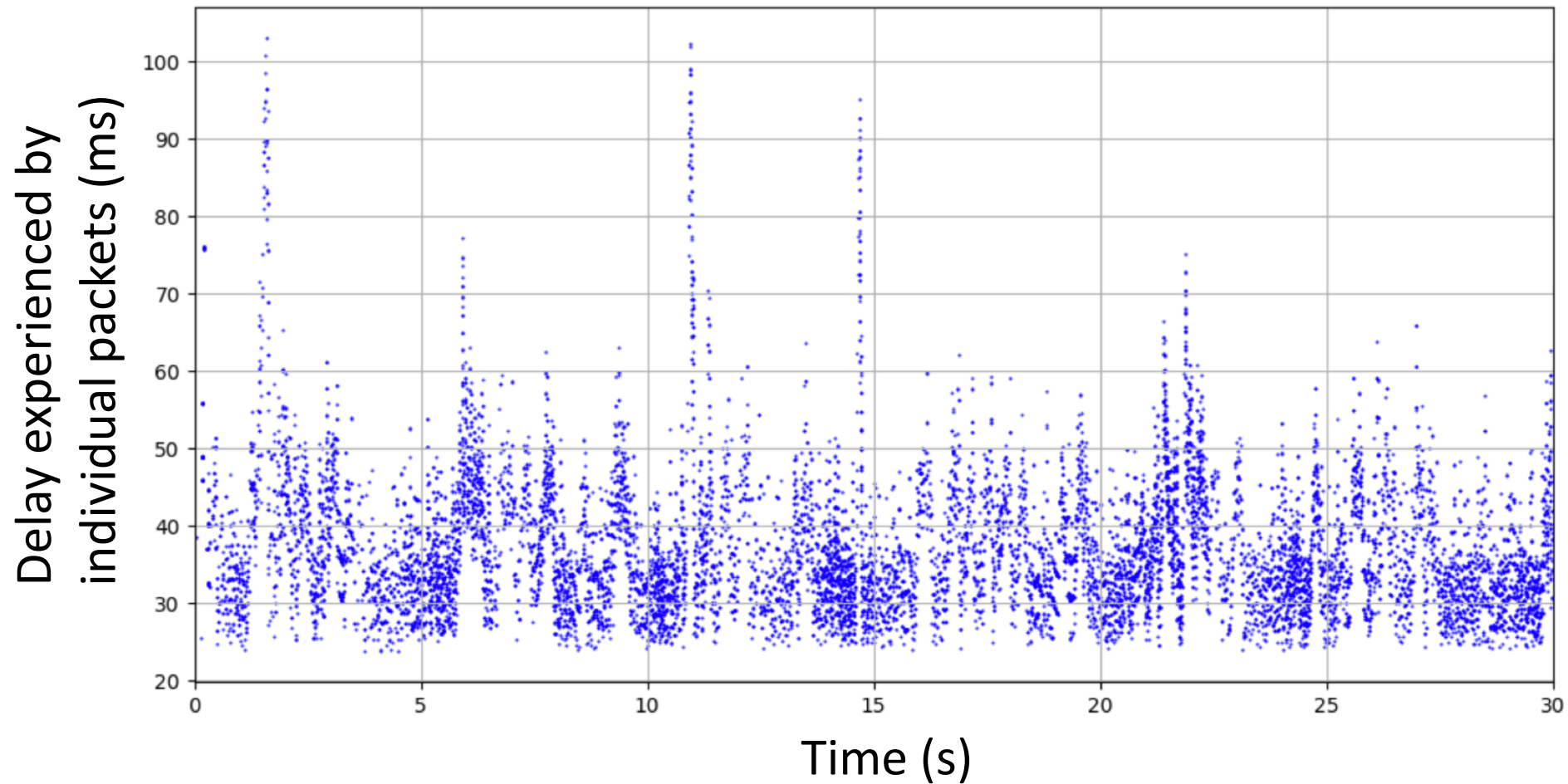
Estimating queuing delay from RTT



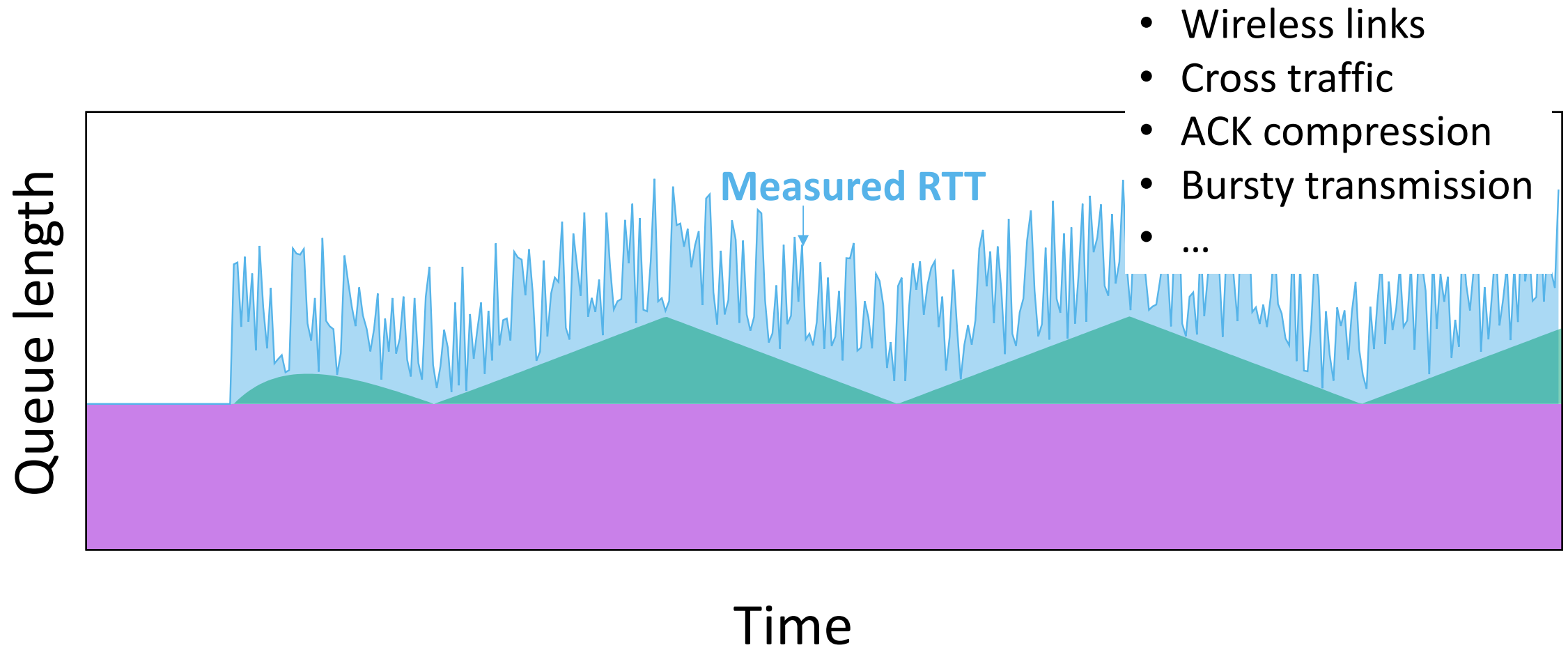
Estimating queuing delay from RTT



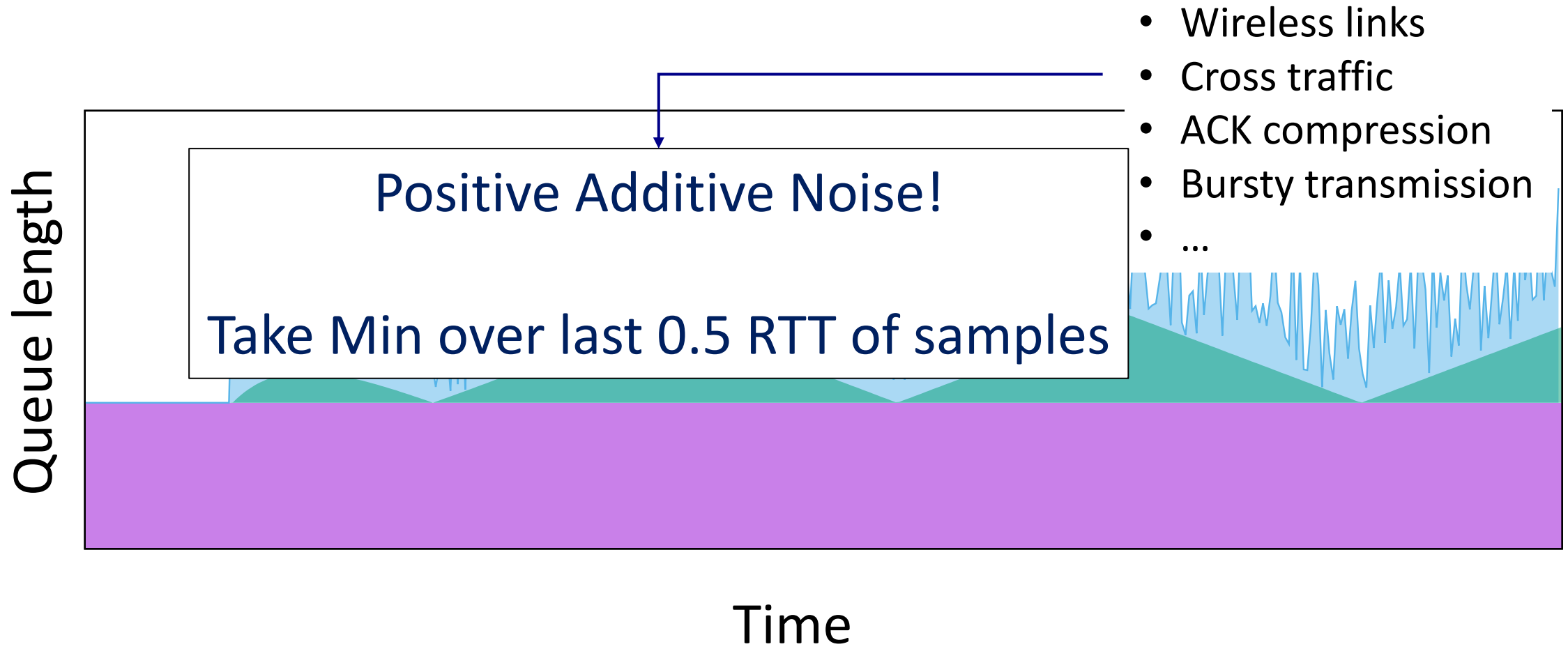
A “noisy” cellular link: Stanford to AWS



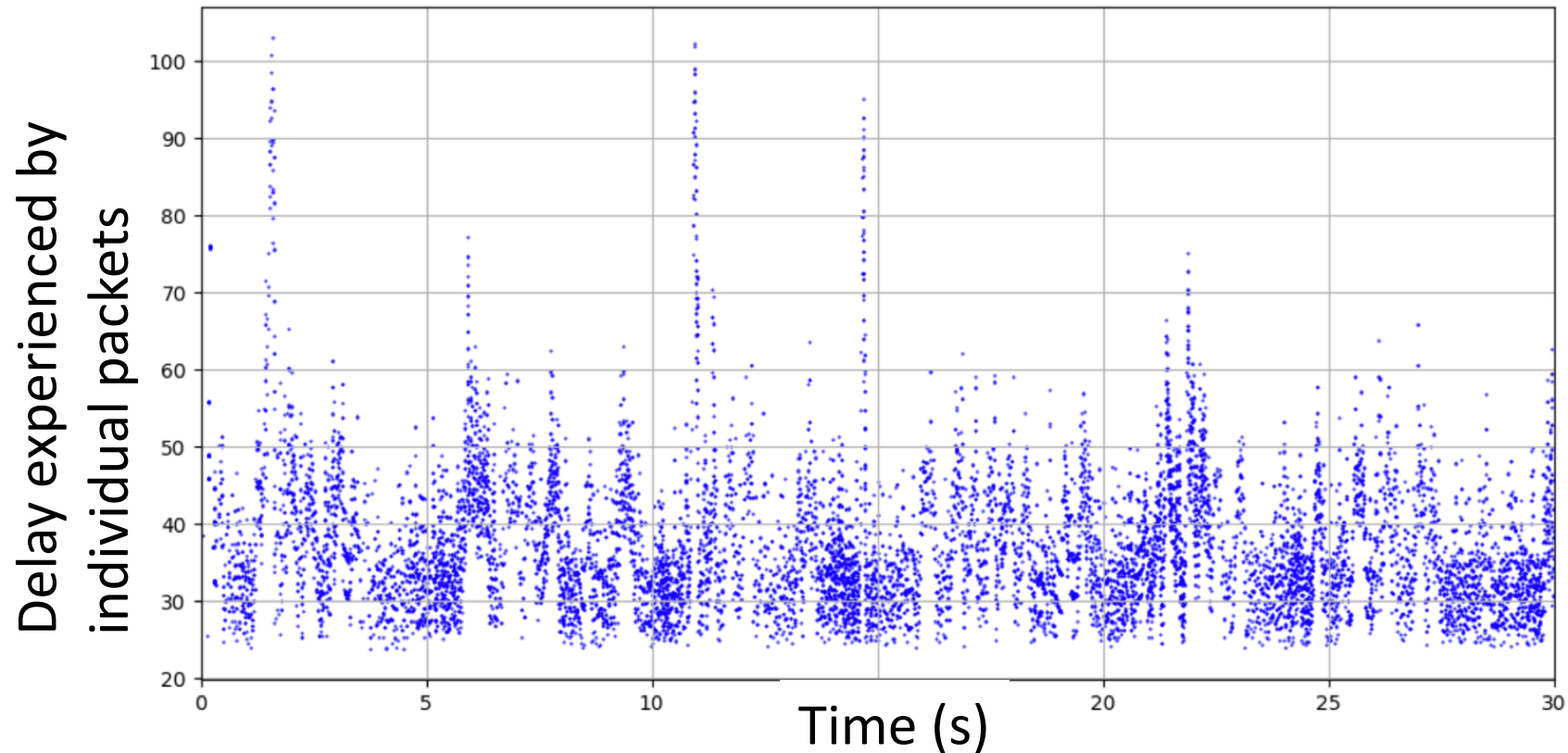
Decoupling queuing delay from other delay variation



Decoupling queuing delay from other delay variation



A “noisy” cellular link: Stanford to AWS



Using the MIN delay estimator improves throughput from
0.5 Mbits/s to 3.9 Mbits/s

Attaining the Target

The Copa Algorithm

Calculate target rate = $r_t = \frac{1}{\delta d_q}$

If current rate $< r_t$: additively increase by $\frac{v}{\delta}$ pkts/RTT
Else: additively decrease by $\frac{v}{\delta}$ pkts/RTT

The Copa Algorithm

Calculate target rate = $r_t = \frac{1}{\delta d_q}$

If current rate $< r_t$: additively increase by $\frac{v}{\delta}$ pkts/RTT

Else: additively decrease by $\frac{v}{\delta}$ pkts/RTT

Velocity for
faster convergence



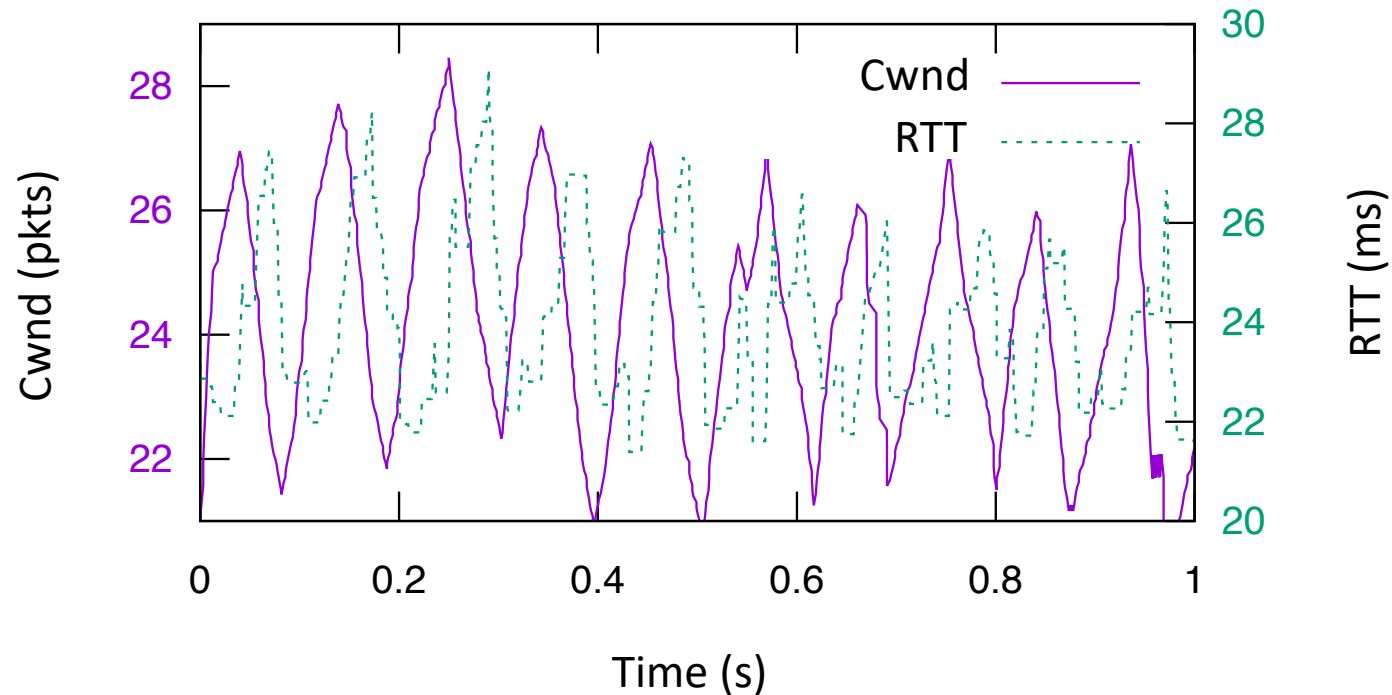
The Copa Algorithm

Calculate target rate $= r_t = \frac{1}{\delta d_q}$

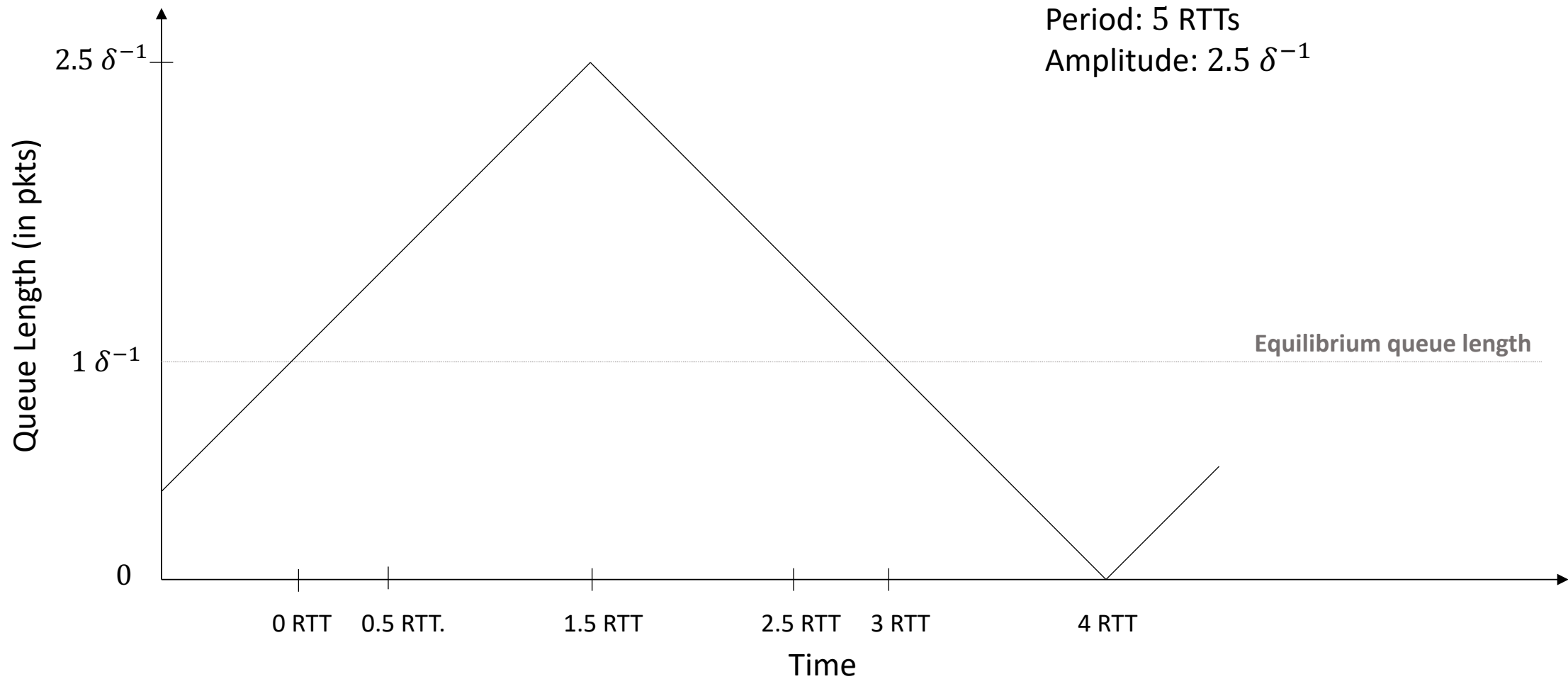
If current rate $< r_t$: additively increase by $\frac{v}{\delta}$ pkts/RTT

Else: additively decrease by $\frac{v}{\delta}$ pkts/RTT

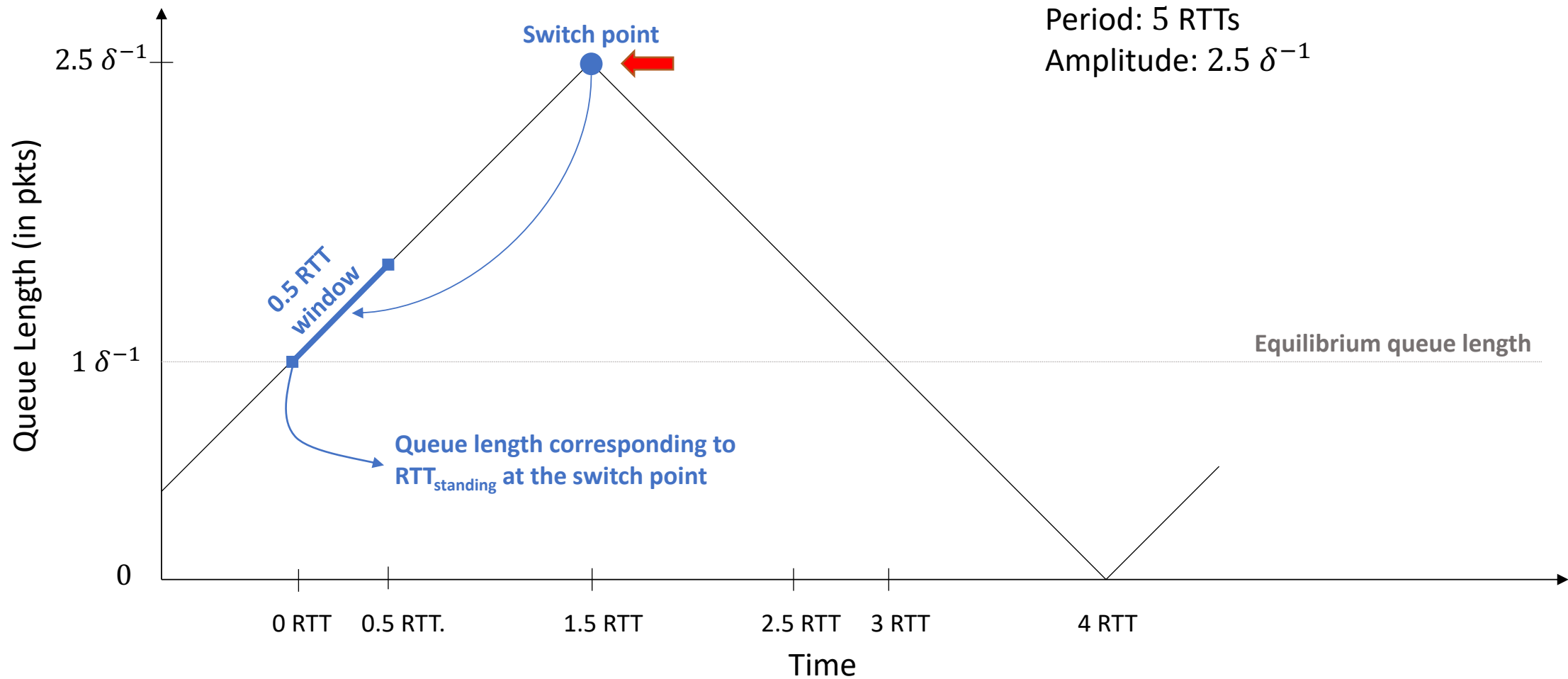
Velocity for
faster convergence



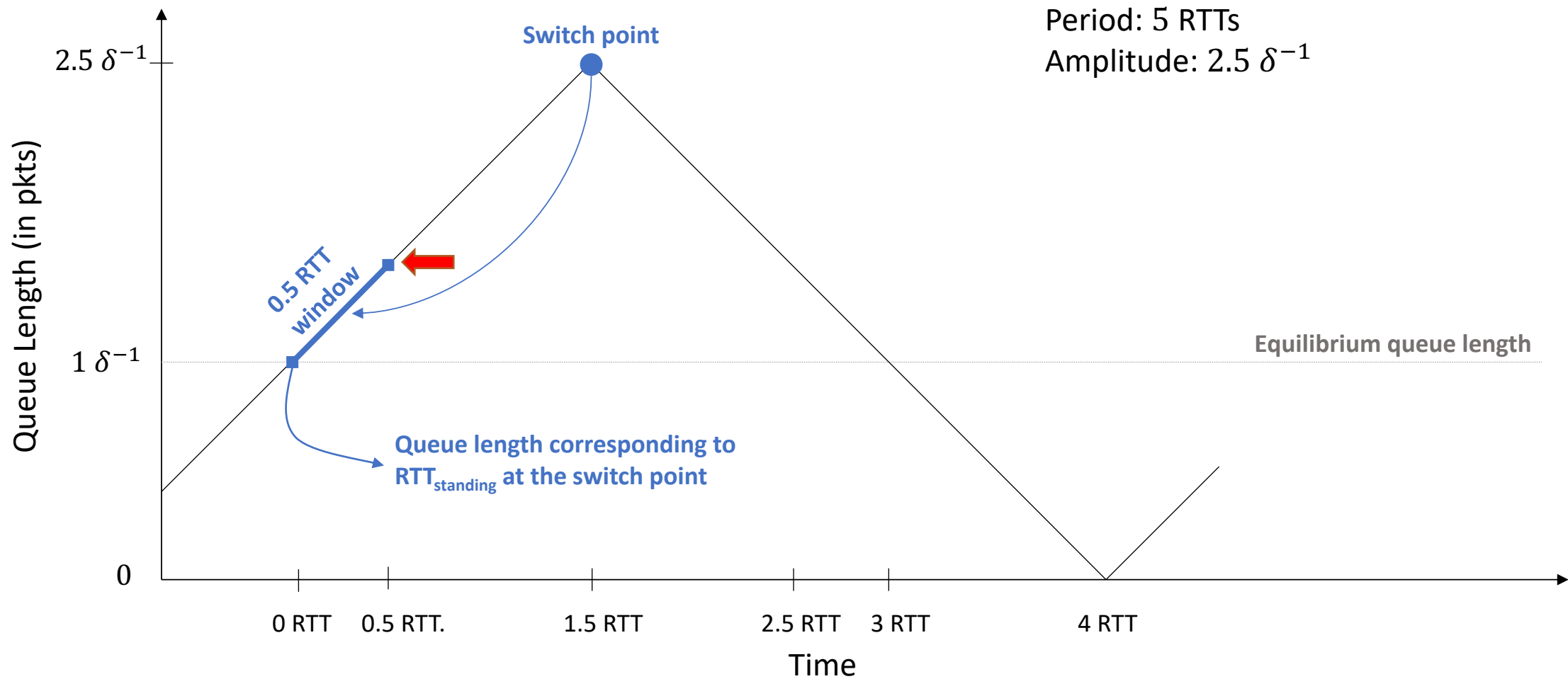
Steady-State Dynamics of Copa



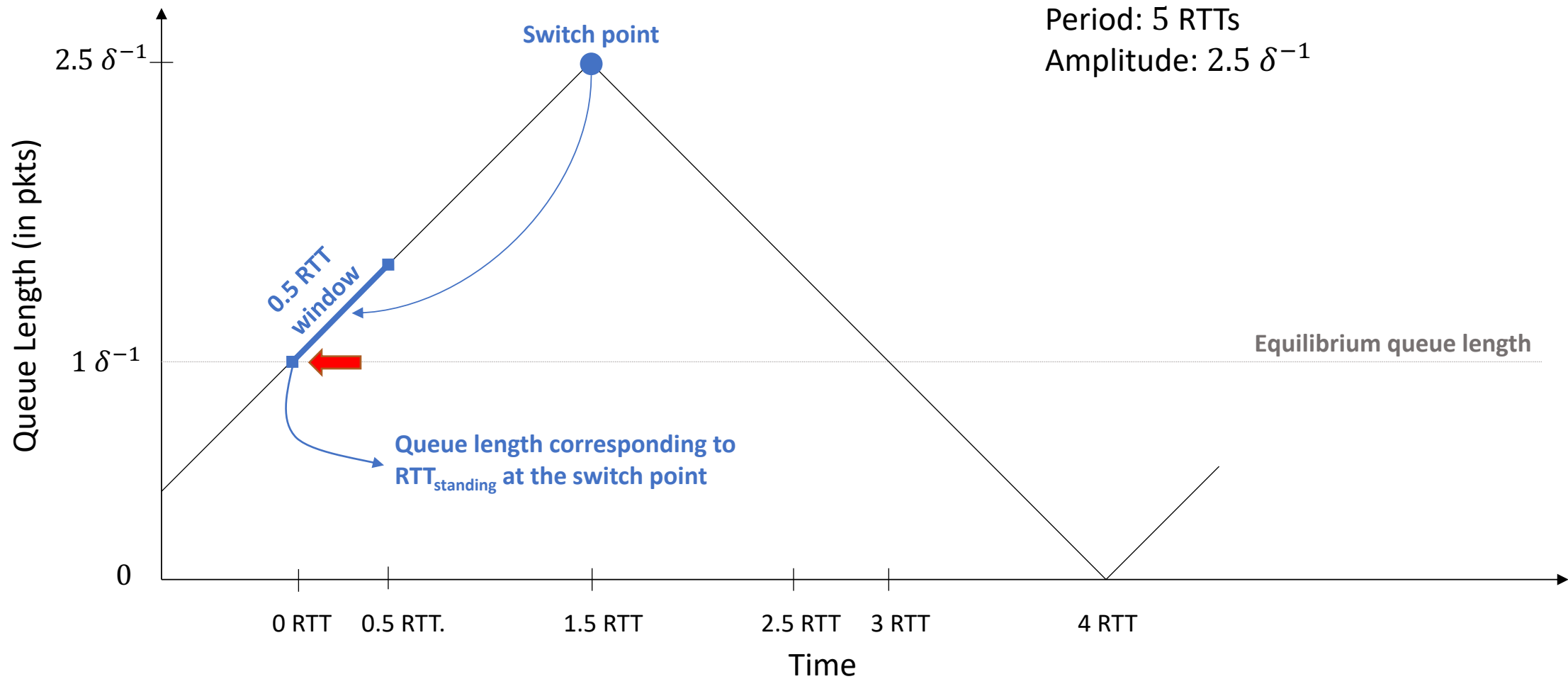
Steady-State Dynamics of Copa



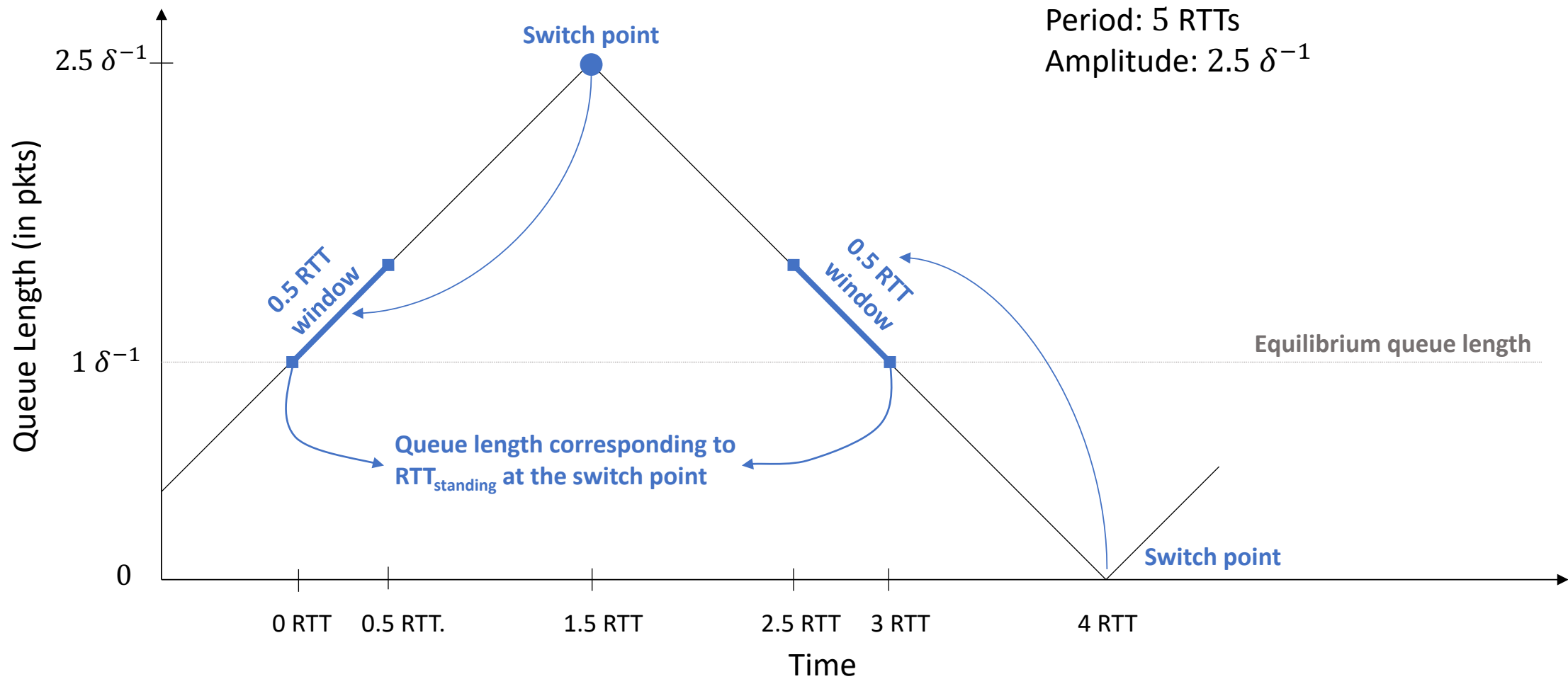
Steady-State Dynamics of Copa



Steady-State Dynamics of Copa



Steady-State Dynamics of Copa



Estimate true minimum RTT

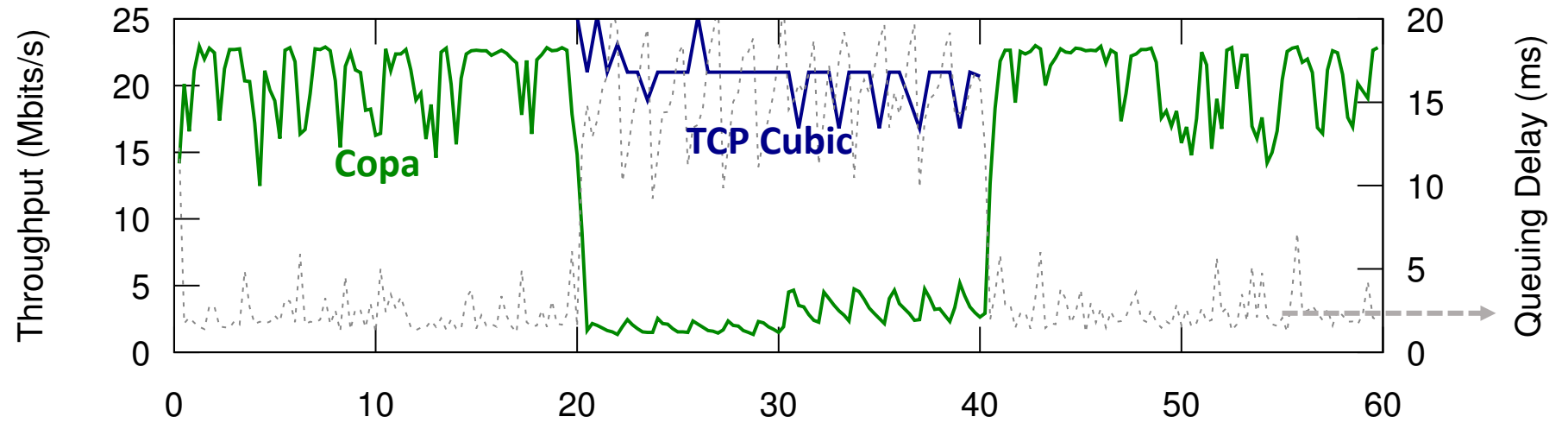
Queue empties every 5 RTTs! 

Detect buffer-filling TCP

TCP-Competitiveness

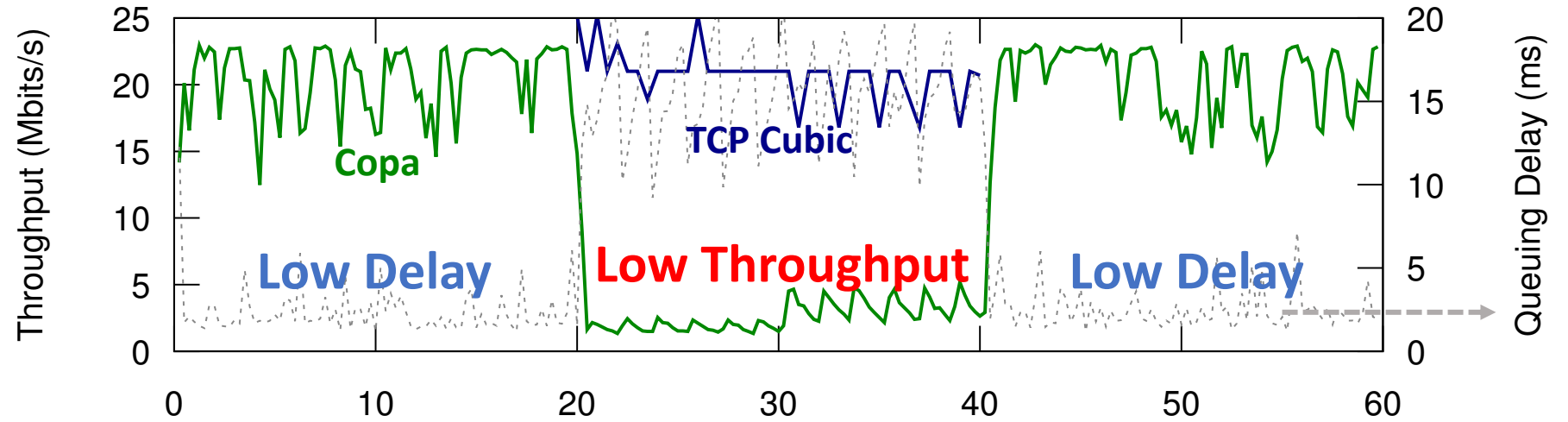
Mode switching for TCP competitiveness

Delay sensitive
($\delta = 0.5$)



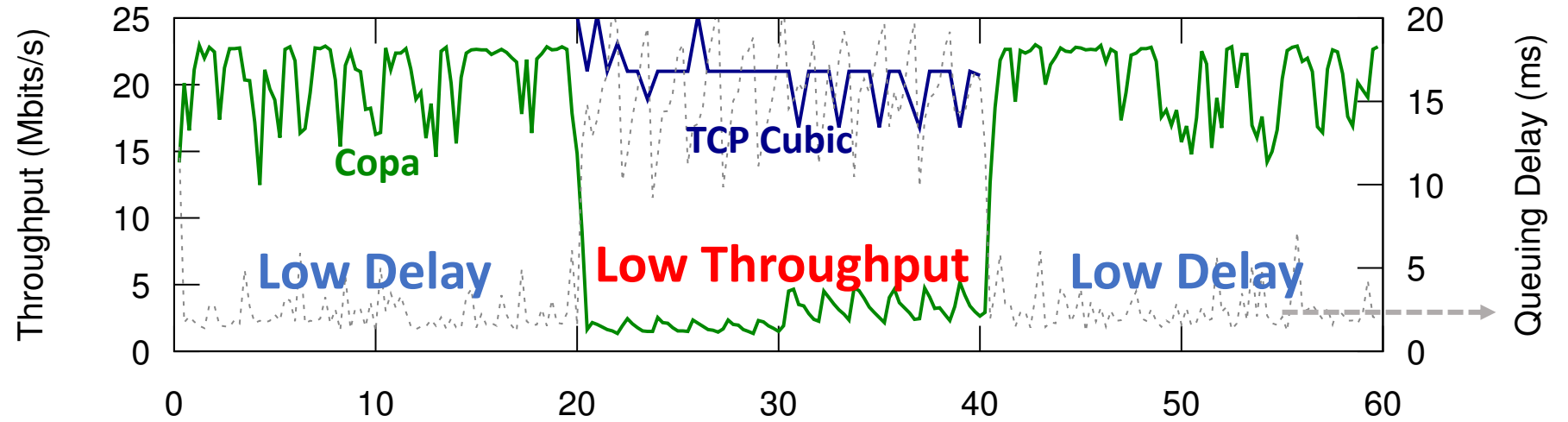
Mode switching for TCP competitiveness

Delay sensitive
($\delta = 0.5$)

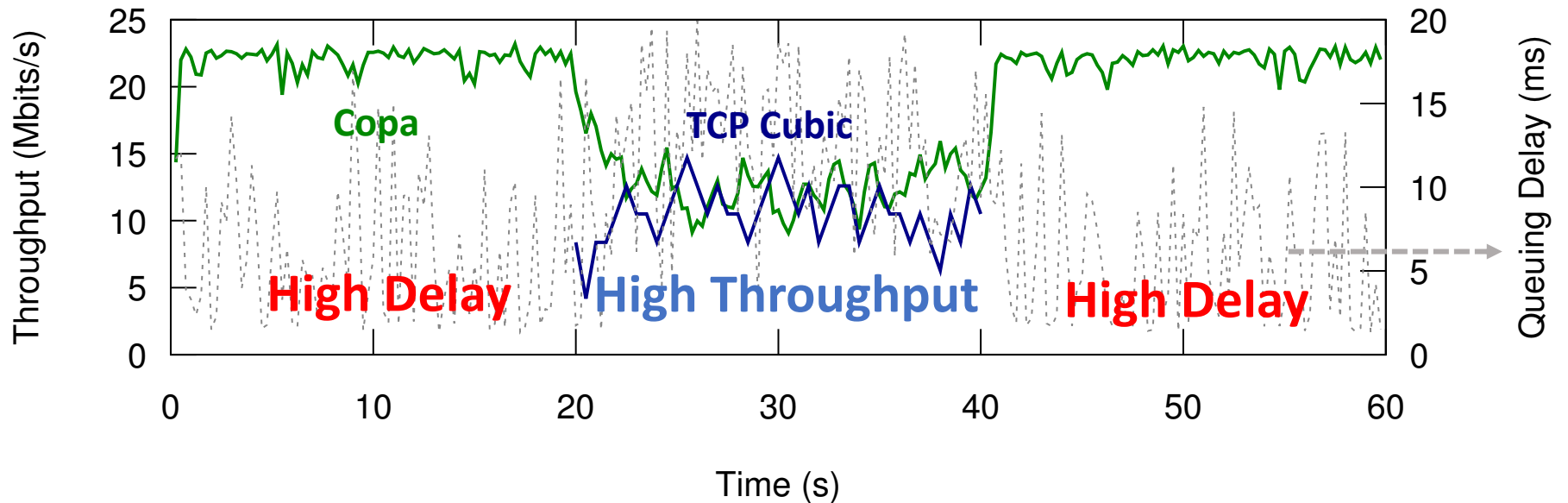


Mode switching for TCP competitiveness

Delay sensitive
($\delta = 0.5$)

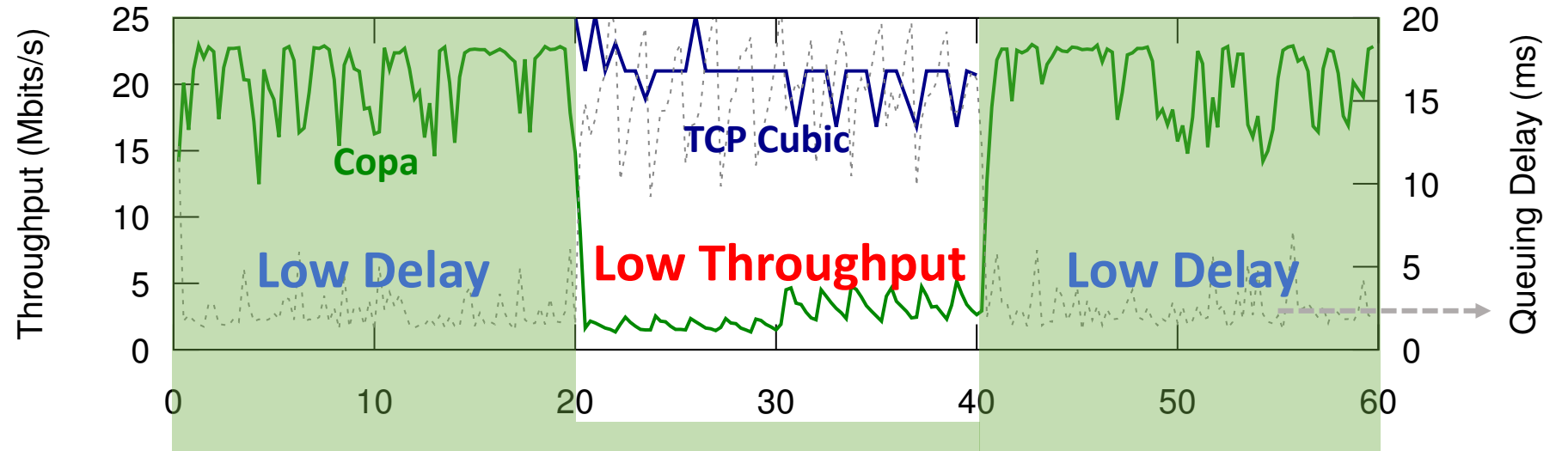


TCP Competitive
(AIMD on δ^{-1})

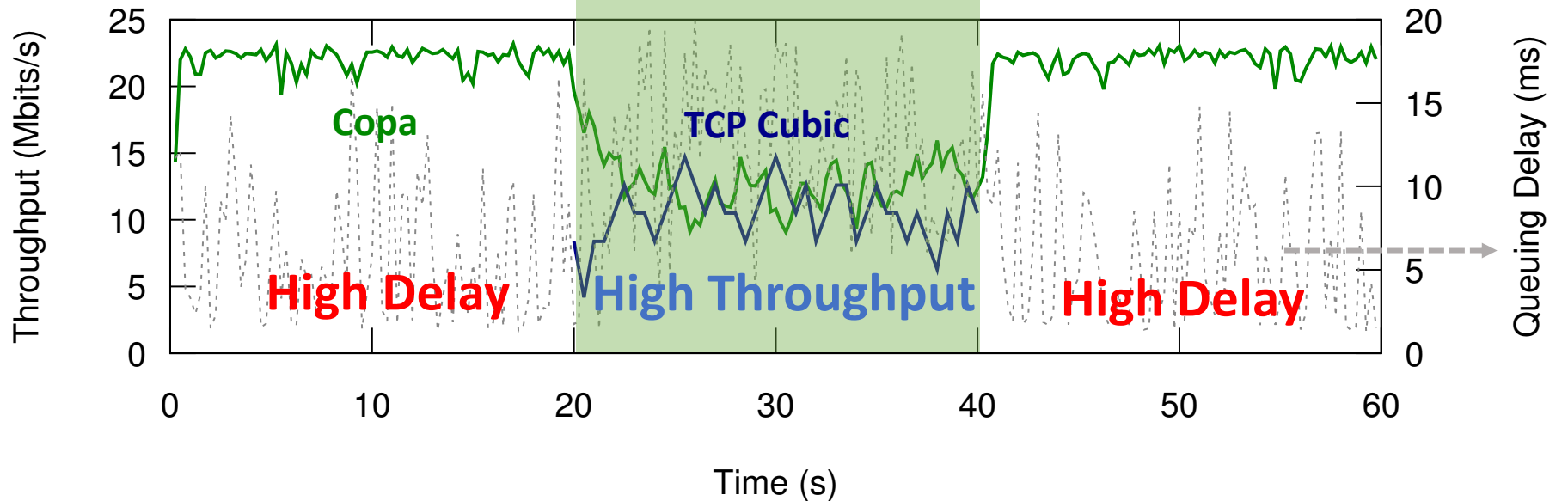


Mode switching for TCP competitiveness

Delay sensitive
($\delta = 0.5$)

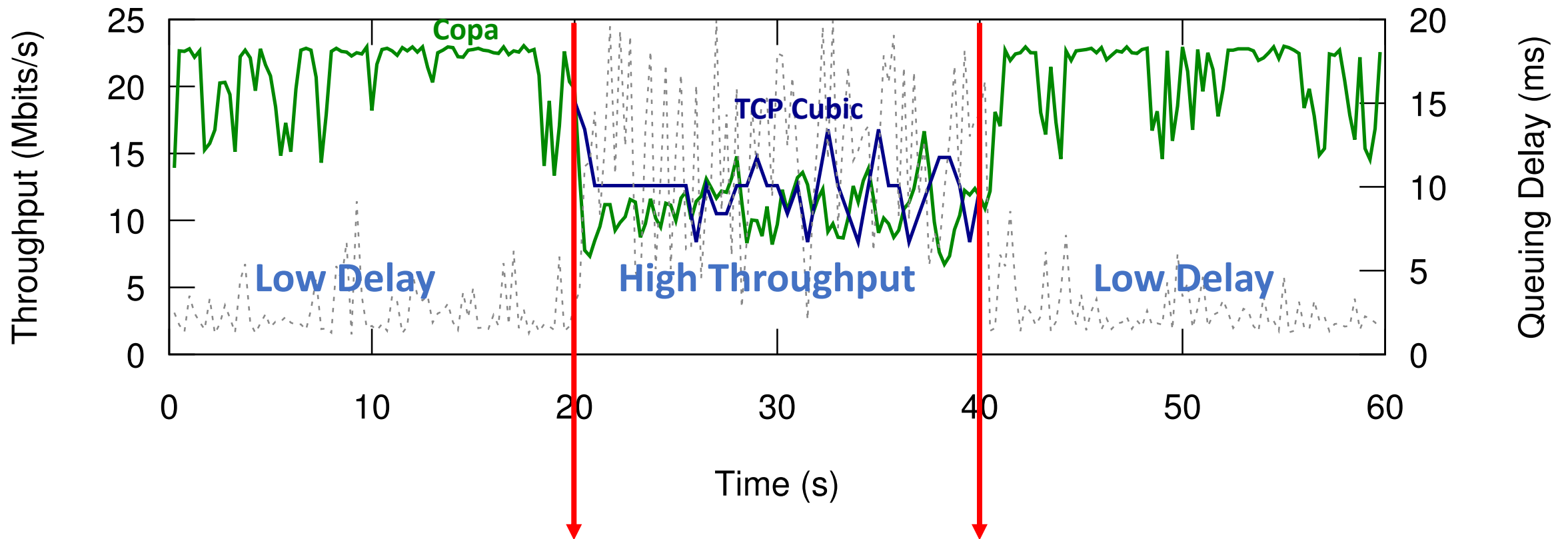


TCP Competitive
(AIMD on δ^{-1})



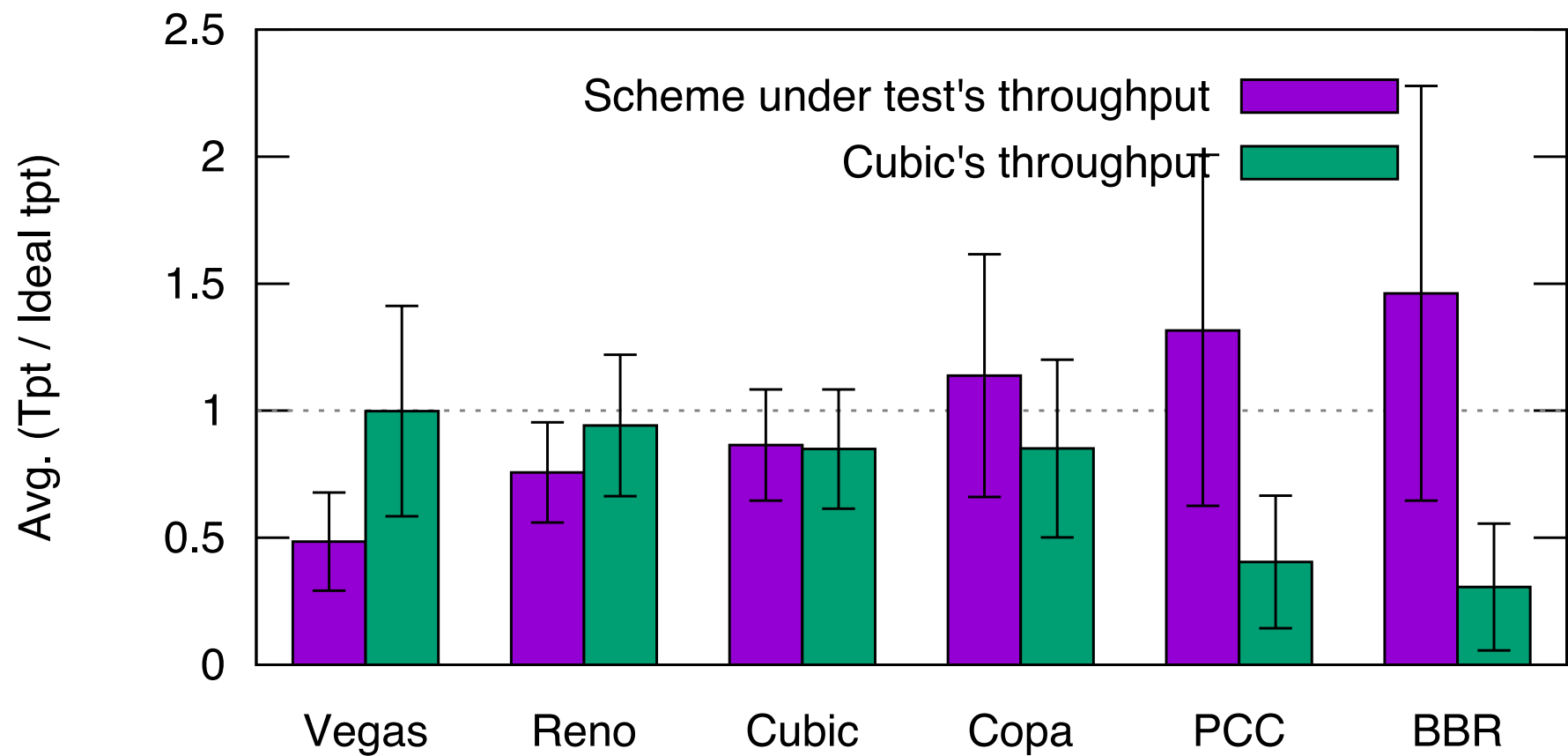
Mode switching for TCP competitiveness

Best of both worlds!

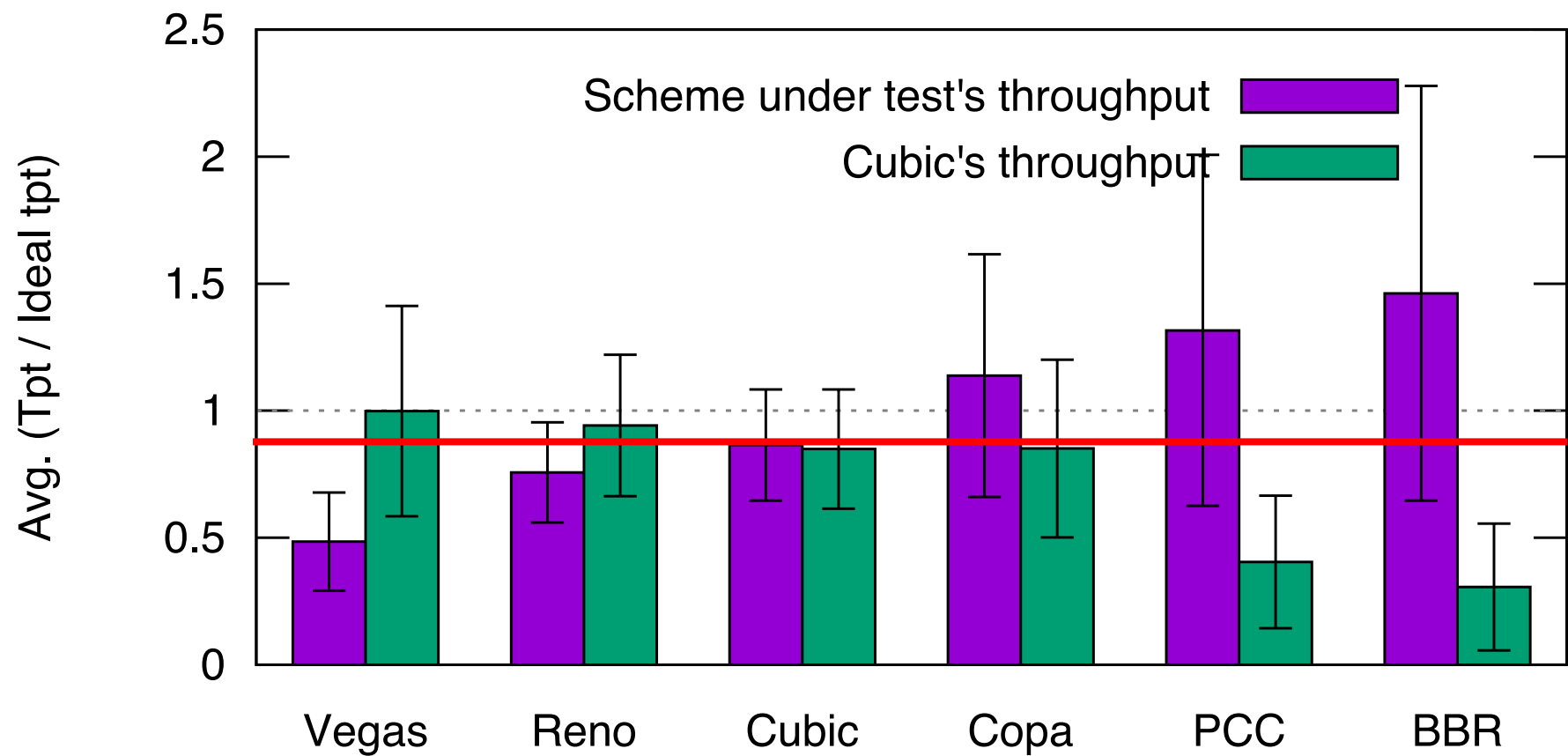


When queue doesn't empty once every 5 RTTs, switch to TCP Competitive mode!

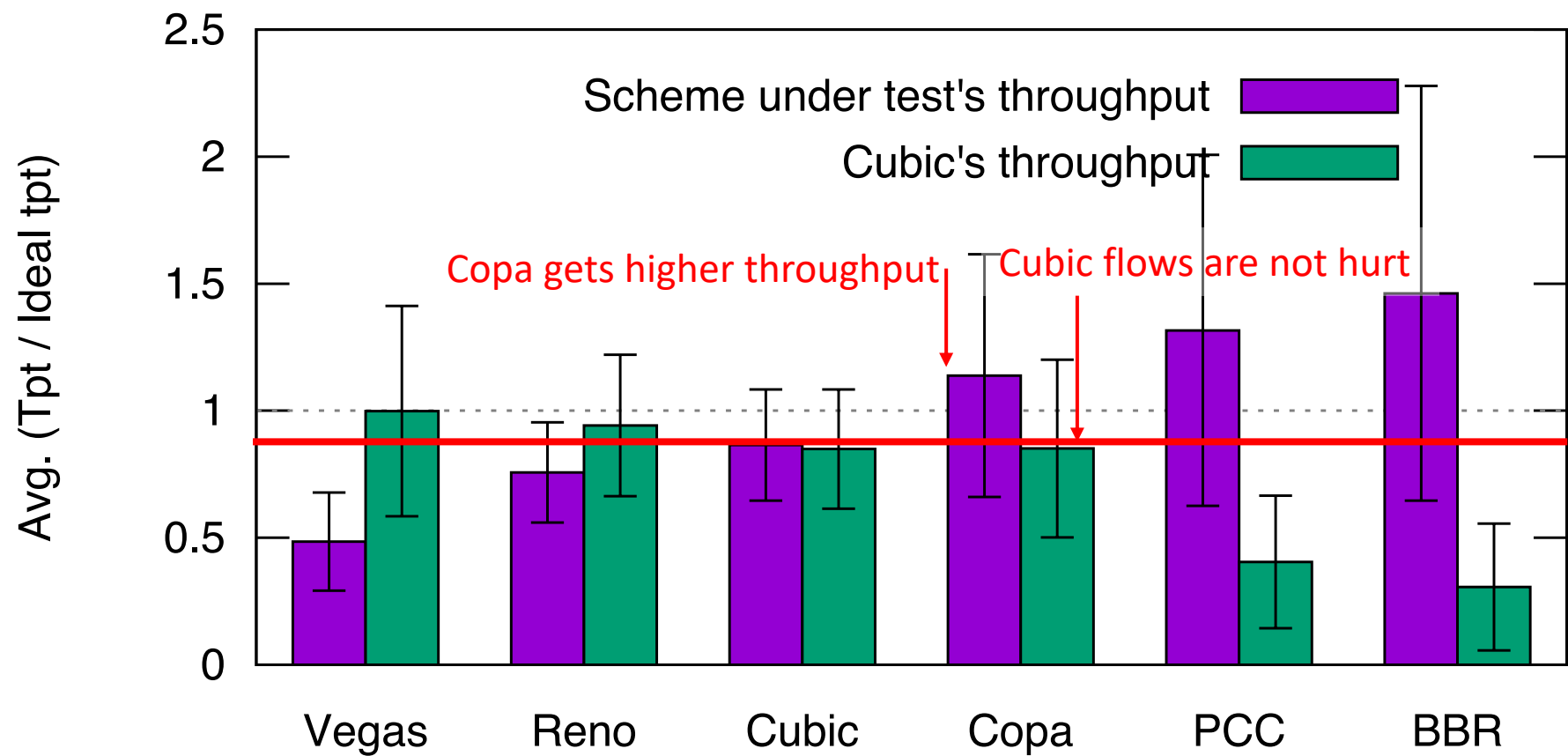
Copa gets higher throughput without hurting TCP Cubic!



Copa gets higher throughput without hurting TCP Cubic!



Copa gets higher throughput without hurting TCP Cubic!

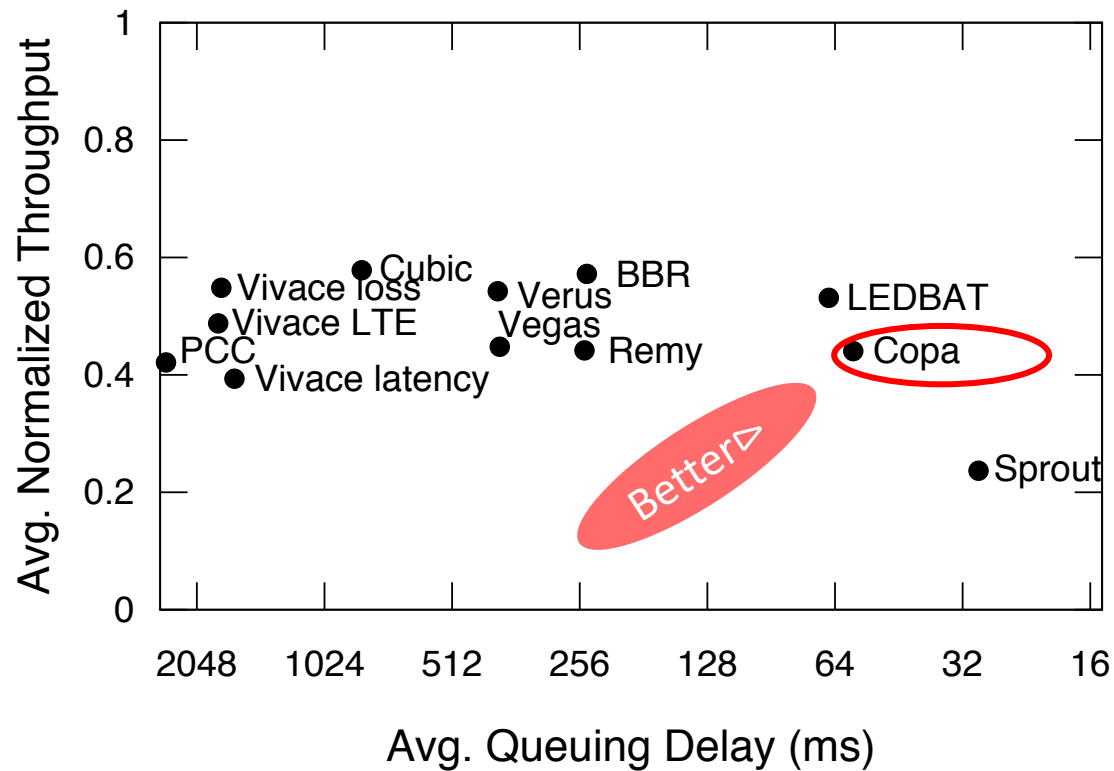


Limitations

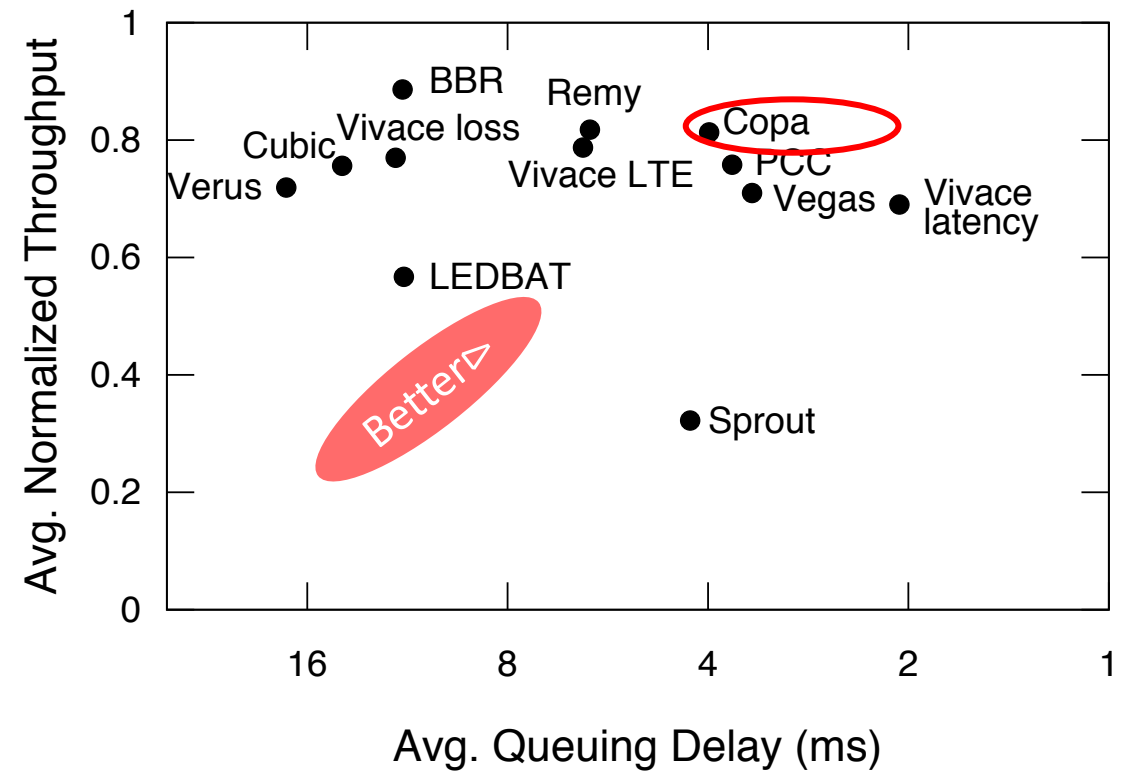
- Cannot ignore low frequency noise
- Queues don't empty periodically if:
 - Propagation delay is much smaller than queuing delay
 - Flows with very different propagation delays share a bottleneck queue
- Needs precise RTT measurements

Consistent Performance on Real Paths

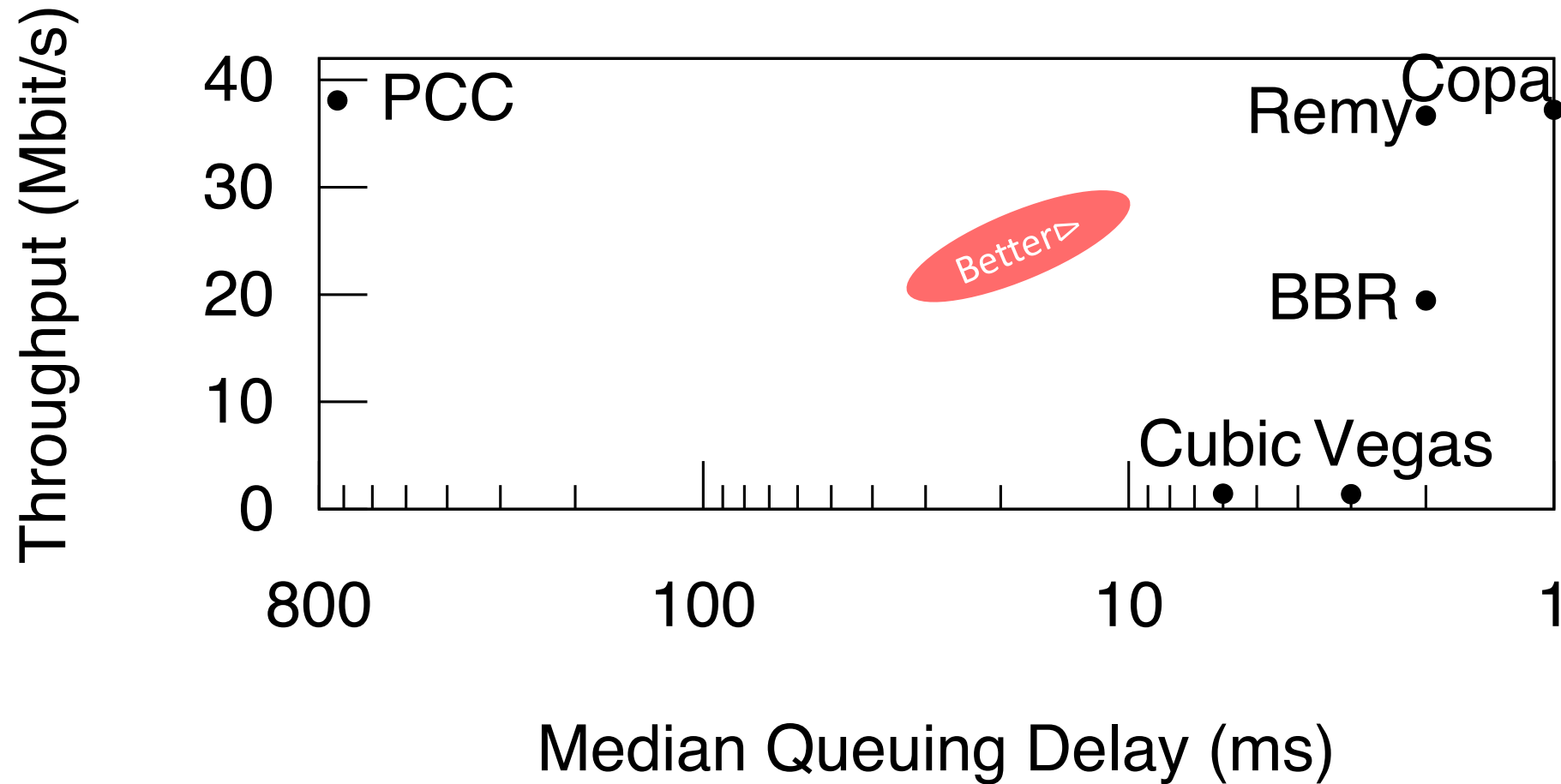
Cellular Networks



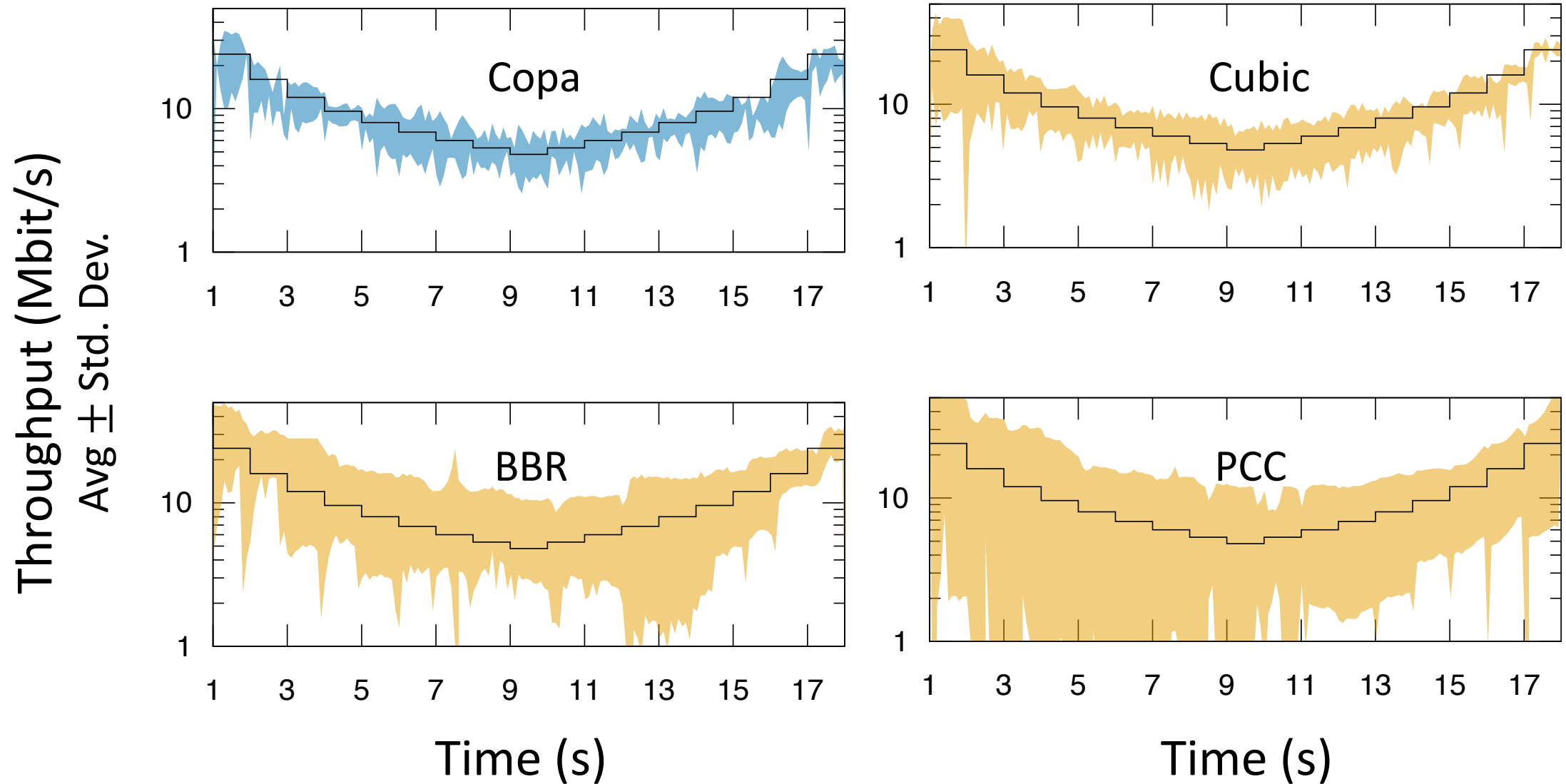
Wired Networks



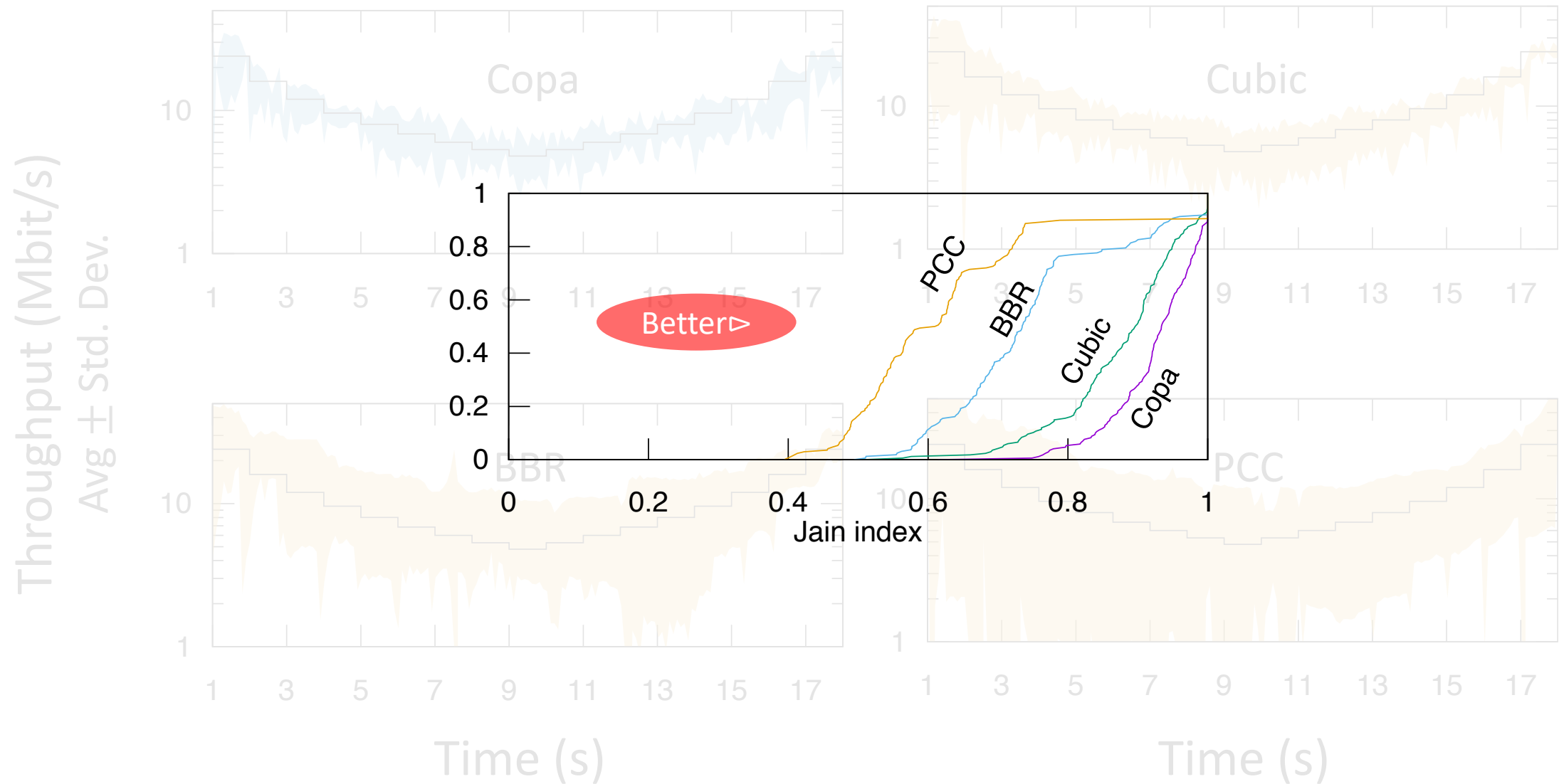
Satellite link: High BDP, high loss



Fairness during flow-churn



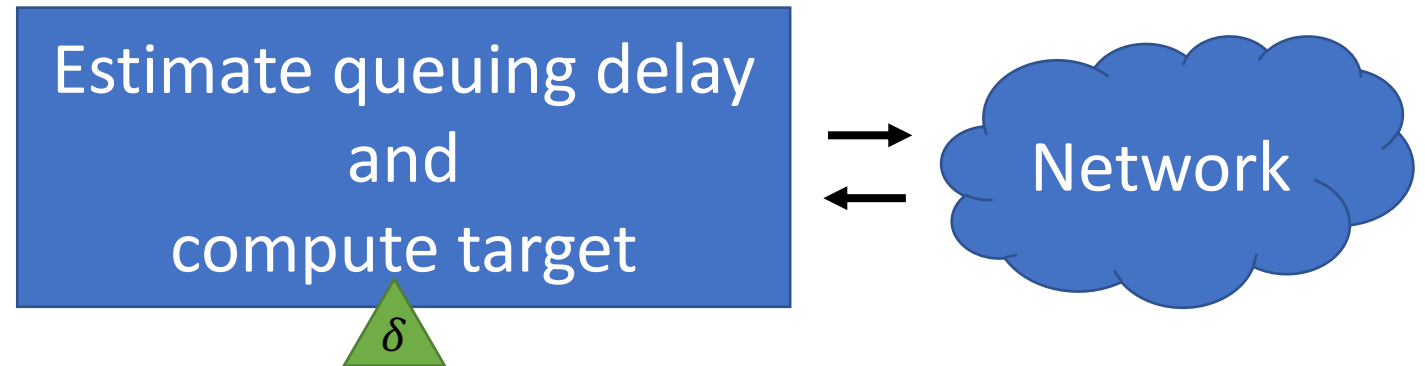
Fairness during flow-churn



Summary

A *practical* delay-based congestion control algorithm

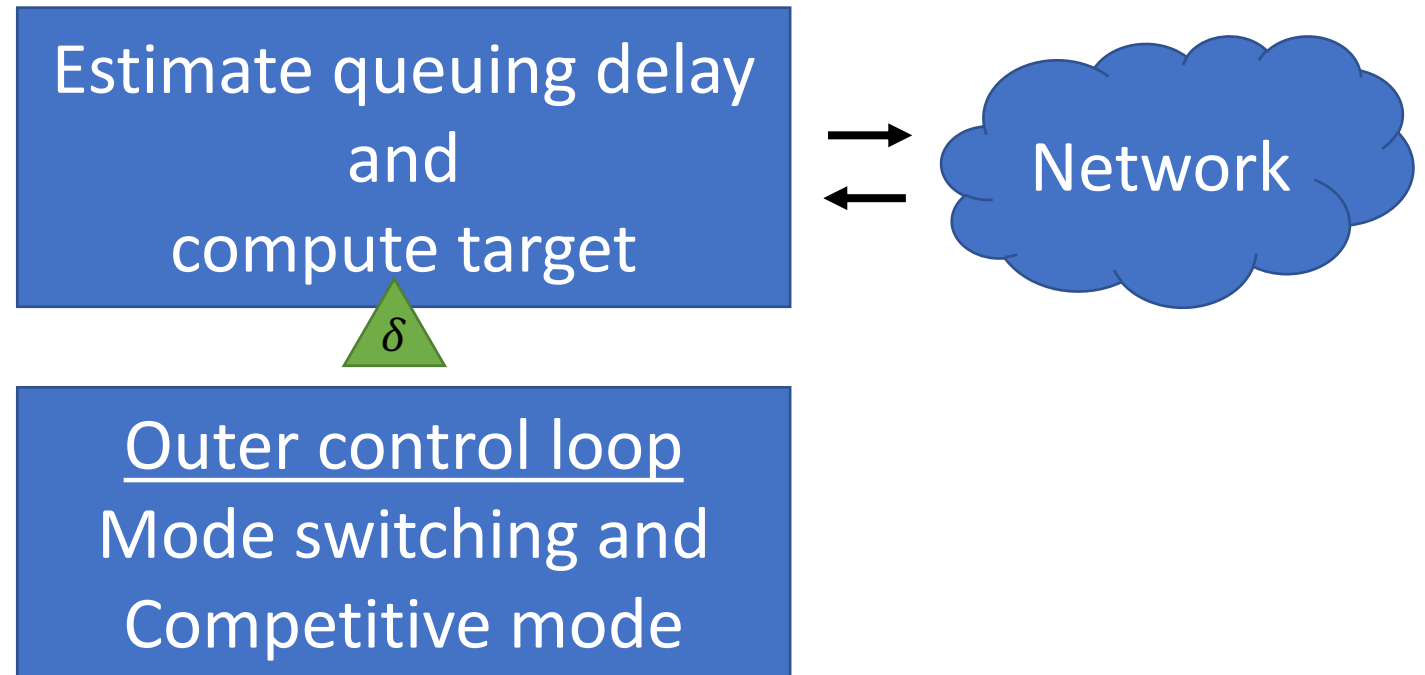
<https://web.mit.edu/copa>



Summary

A *practical* delay-based congestion control algorithm

<https://web.mit.edu/copa>



Summary

A *practical* delay-based congestion control algorithm

<https://web.mit.edu/copa>

