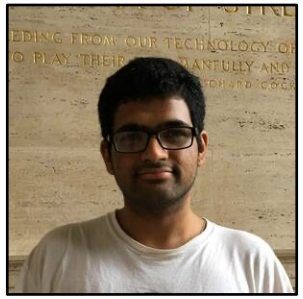
FRCC: Towards fair and robust congestion control

Anup Agarwal
108anup@gmail.com



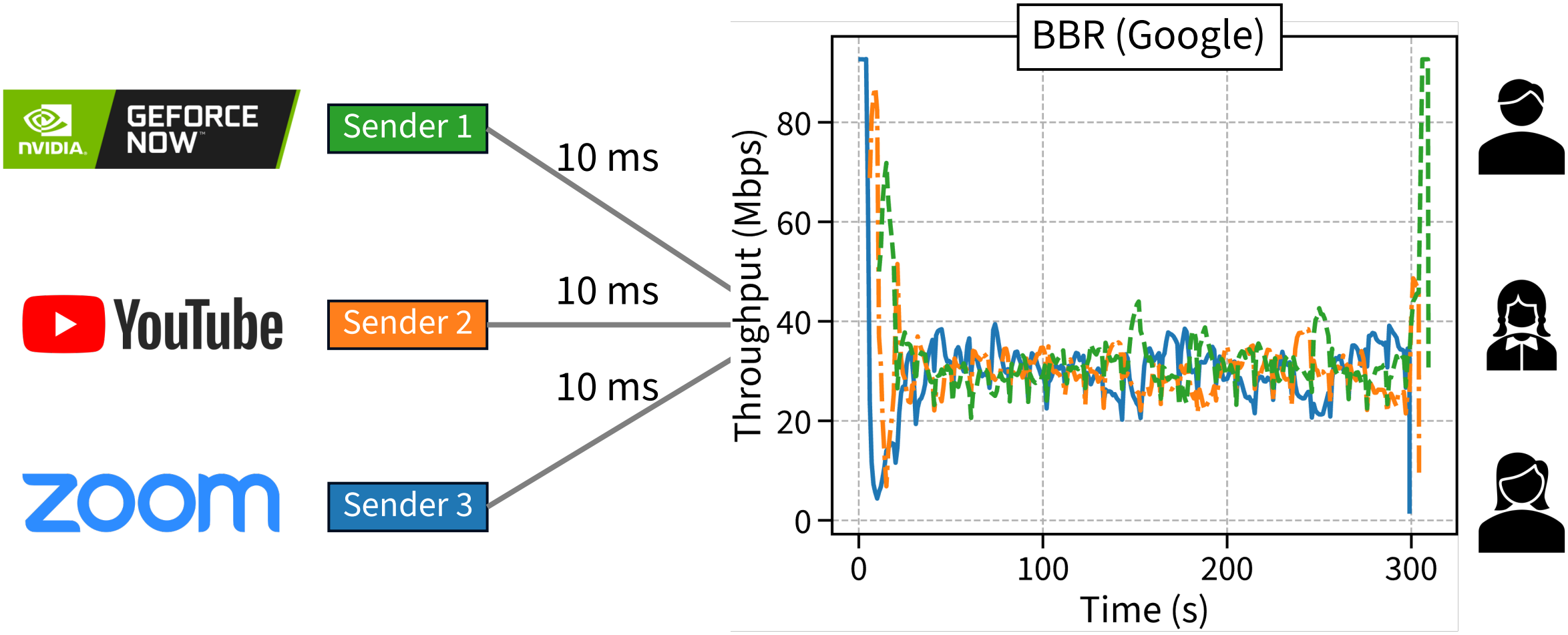
Venkat Arun

Srinivasan Seshan

Carnegie Mellon

THE UNIVERSITY OF TEXAS AT AUSTIN
PRAESIDIUM
DISCIPLINA
CIVITATIS

All past congestion control algorithms (CCAs) starve flows
E.g., BBR, deployed at Google (40% Internet traffic)



All past congestion contr flows
E.g., BBR, deployed at c)

Sender 1

30 ms
~~10 ms~~

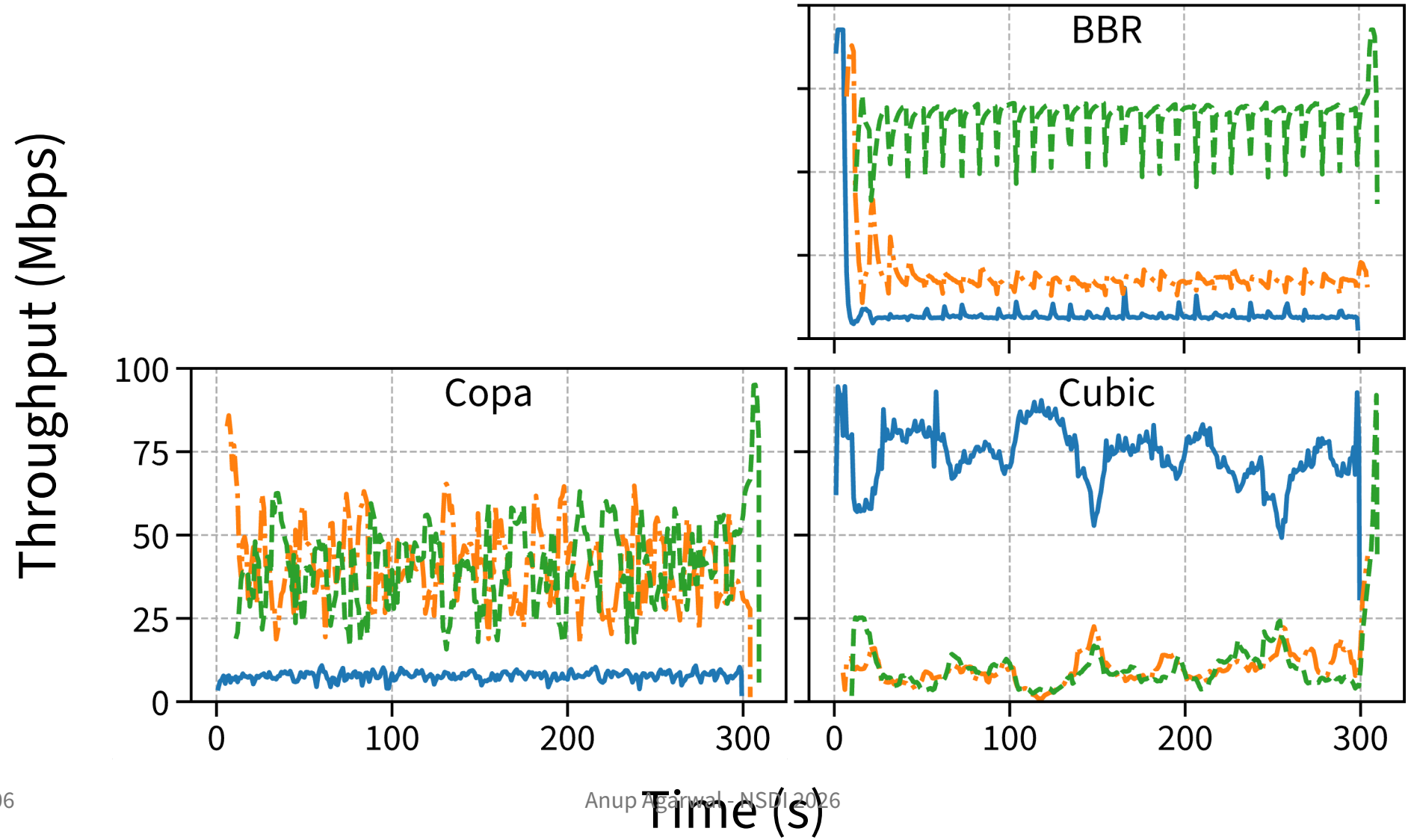
Sender 2

20 ms
~~10 ms~~

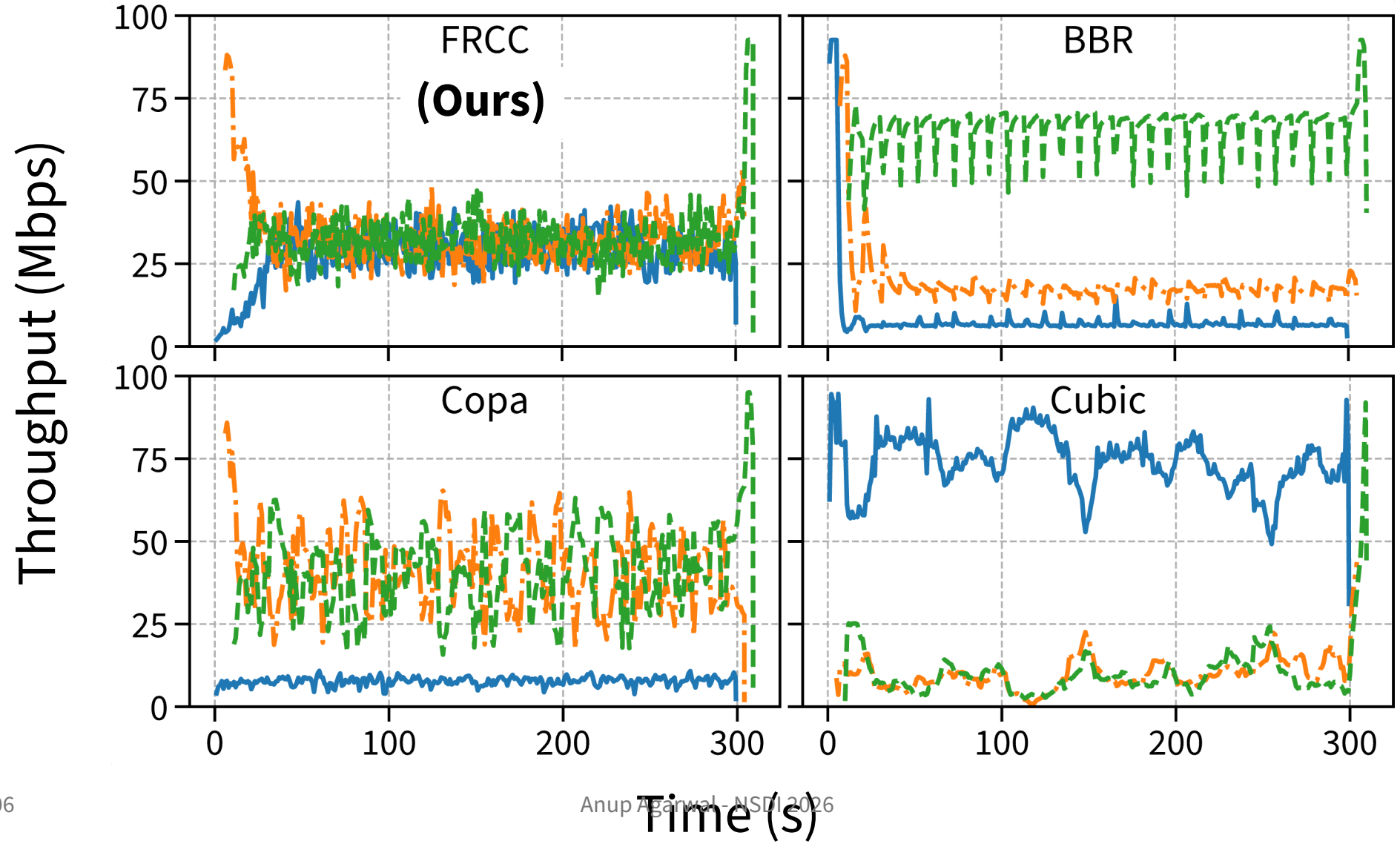
Sender 3

10 ms

All past CCAs starve flows under noise (Open problem before our work)

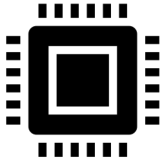


FRCC: first CCA that provably ensures fairness

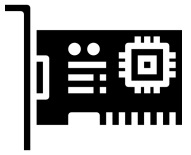


Sources of noise or delay jitter

RTS/CTS,
ACK aggregation



OS Scheduling
delays



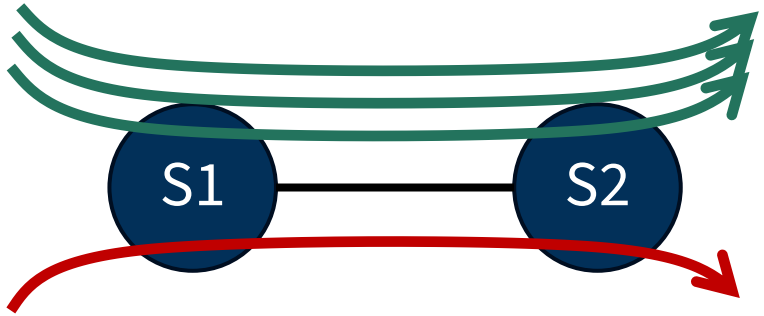
Batching
of packets

Complex
cellular service

Why does starvation occur?

1. CCAs coordinate by encoding info in congestion signals
2. Noise corrupts the coordination causing starvation

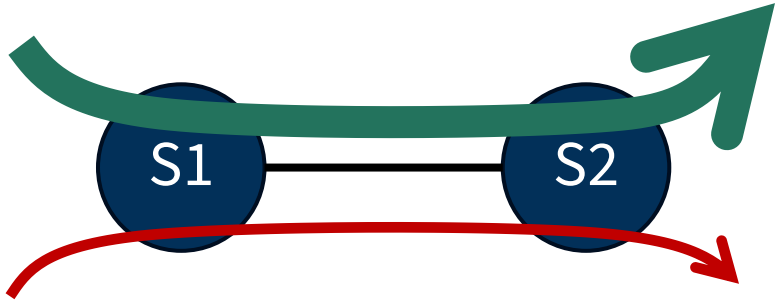
1. CCAs coordinate by encoding info in congestion signals
2. Noise corrupts the coordination causing starvation



3x 25 Mbps

25 Mbps

Fair-share = 100/4 = **25 Mbps**



75 Mbps

25 Mbps

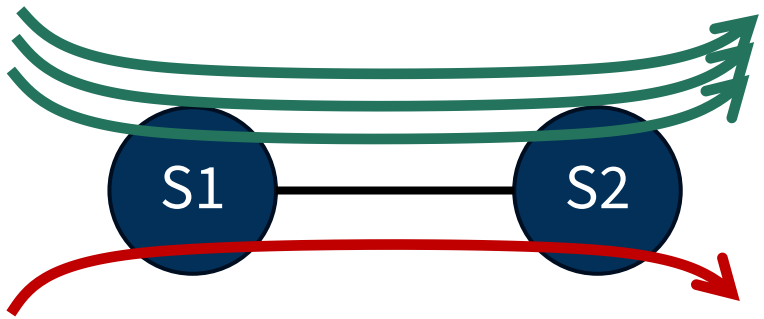
Fair-share = 100/2 = **50 Mbps**

RED FLOW

Same observations

Different fair share

1. CCAs coordinate by encoding info in congestion signals
2. Noise corrupts the coordination causing starvation

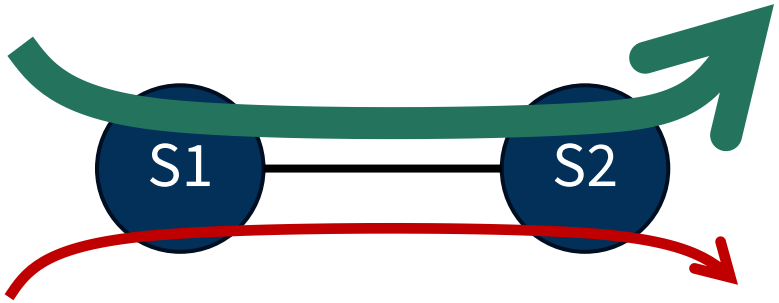


3x 25 Mbps

25 Mbps

Fair-share = 100/4 = **25 Mbps**

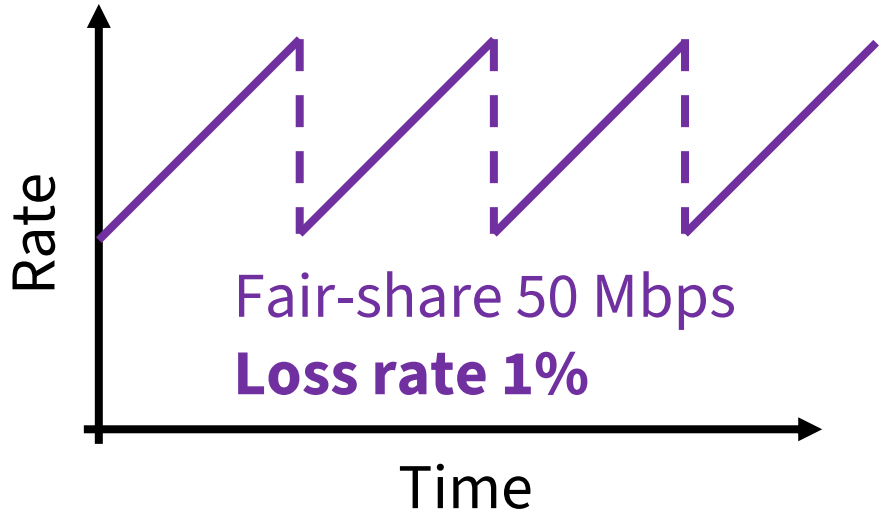
Reno (AIMD)
$Fair\ share = \frac{1}{\sqrt{loss\ rate}}$



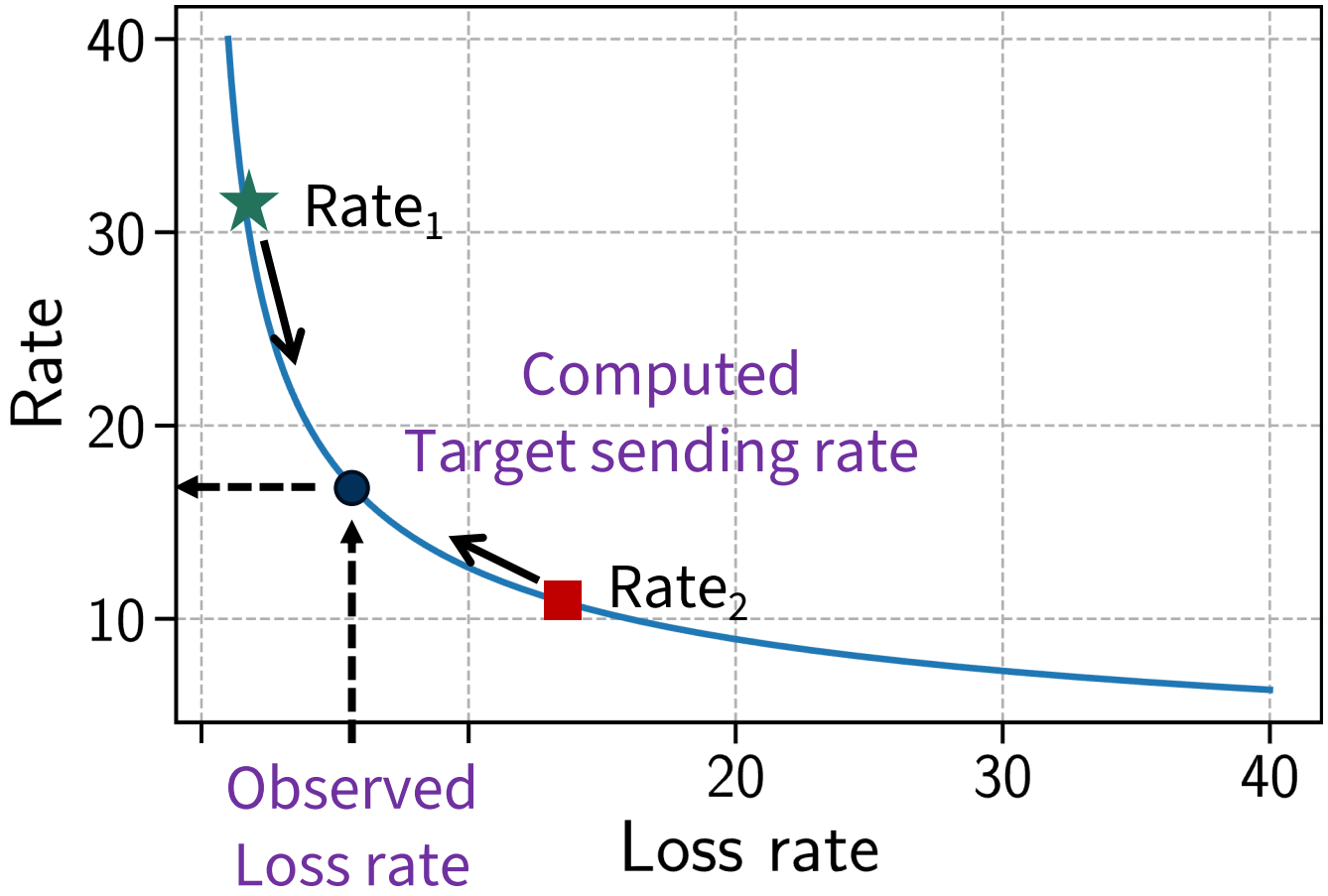
75 Mbps

25 Mbps

Fair-share = 100/2 = **50 Mbps**



1. CCAs coordinate by encoding info in congestion signals
2. Noise corrupts the coordination causing starvation

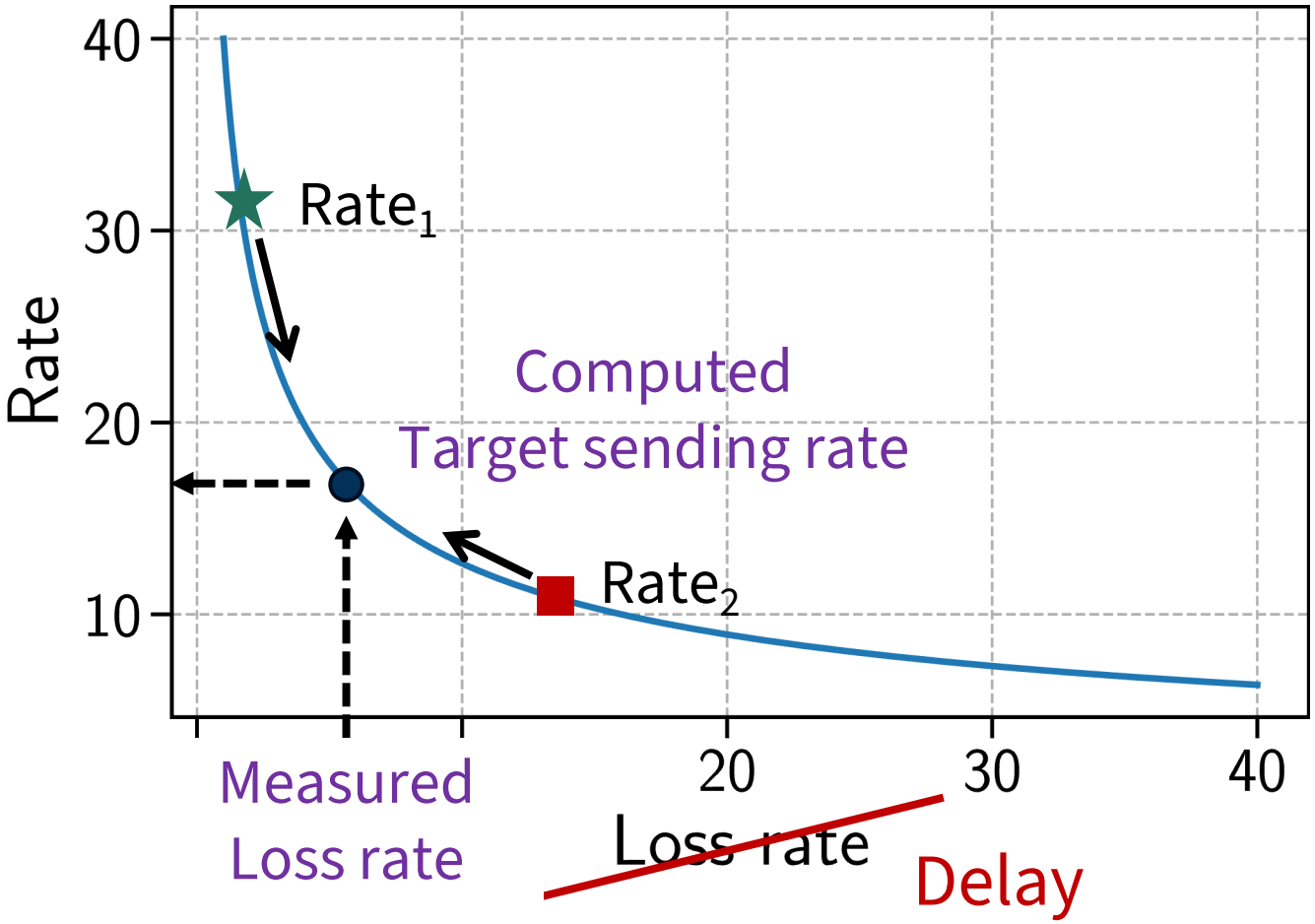


Reno (AIMD)
$Fair\ share = \frac{1}{\sqrt{loss\ rate}}$

Target rate

$$= \frac{1}{\sqrt{\text{observed loss rate}}}$$

1. CCAs coordinate by encoding info in congestion signals
2. Noise corrupts the coordination causing starvation



Copa/Vegas/FAST

~~Reno (AIMD)~~

$$Fair\ share = \frac{1}{\sqrt{loss\ rate}}$$

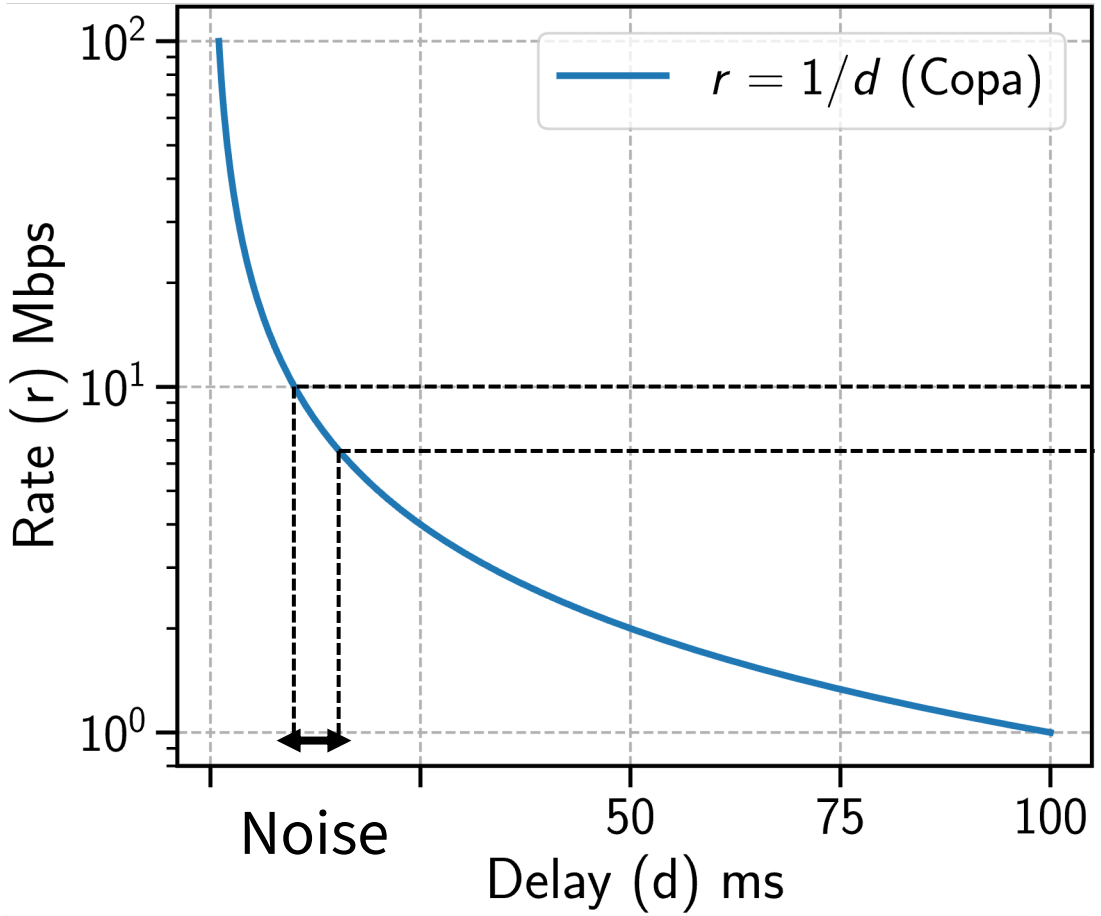
Delay

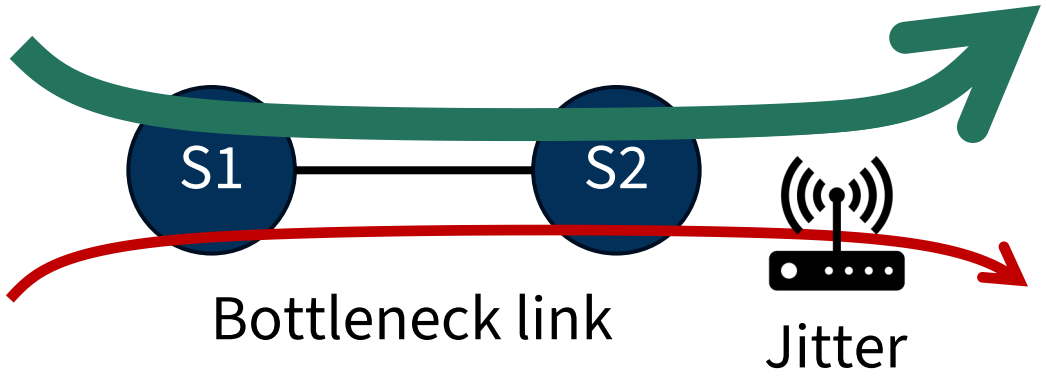
Target rate

$$= \frac{1}{\sqrt{\text{measured loss rate}}}$$

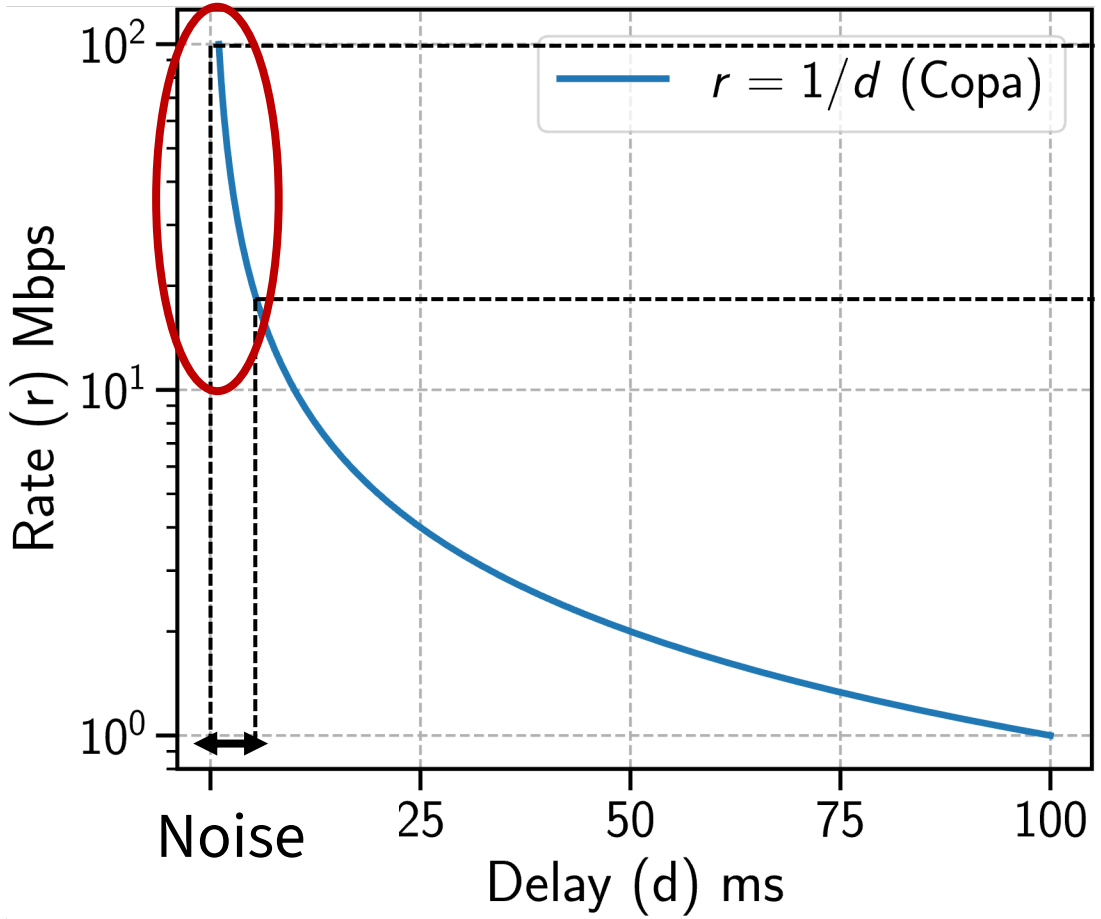
See Contracts [NINeS26] for more

1. CCAs coordinate by encoding info in congestion signals
2. Noise corrupts the coordination causing starvation





1. CCAs coordinate by encoding info in congestion signals
2. Noise corrupts the coordination causing starvation



Error increases with capacity

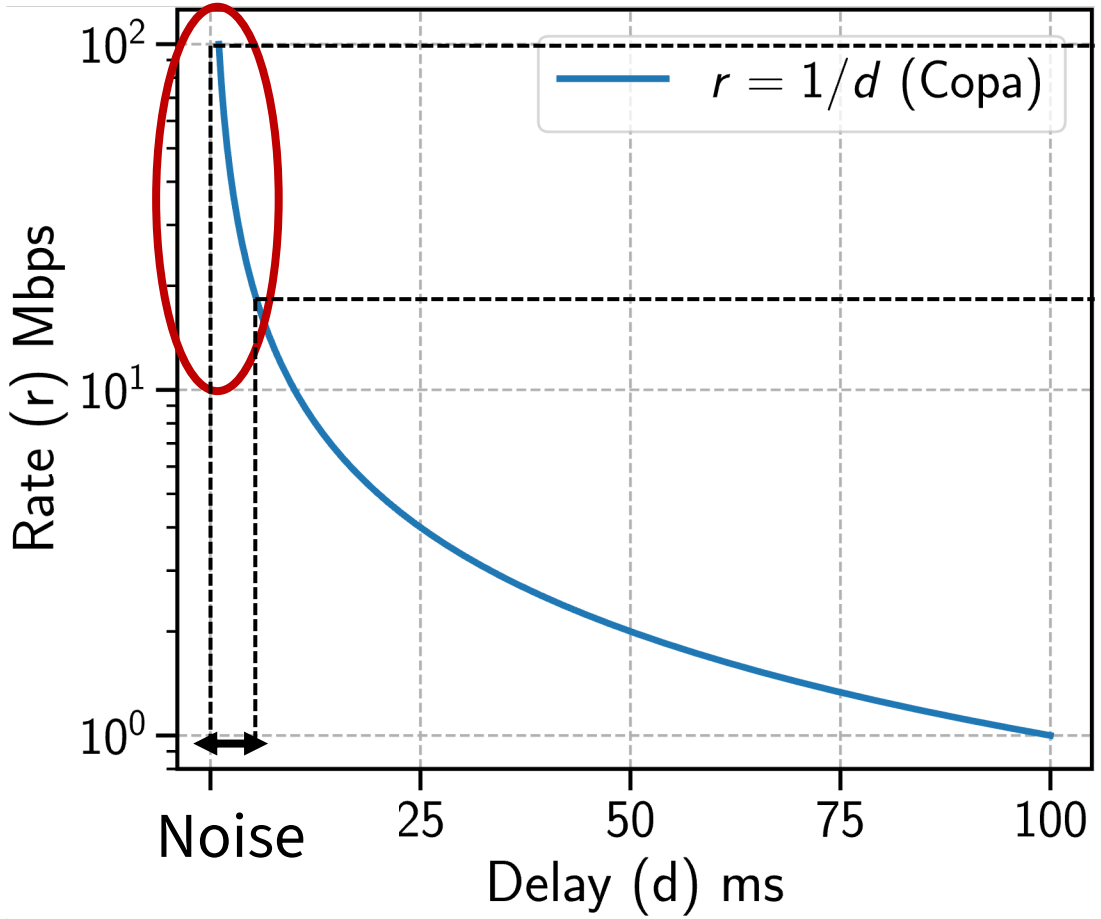
Asymptote on Y-axis

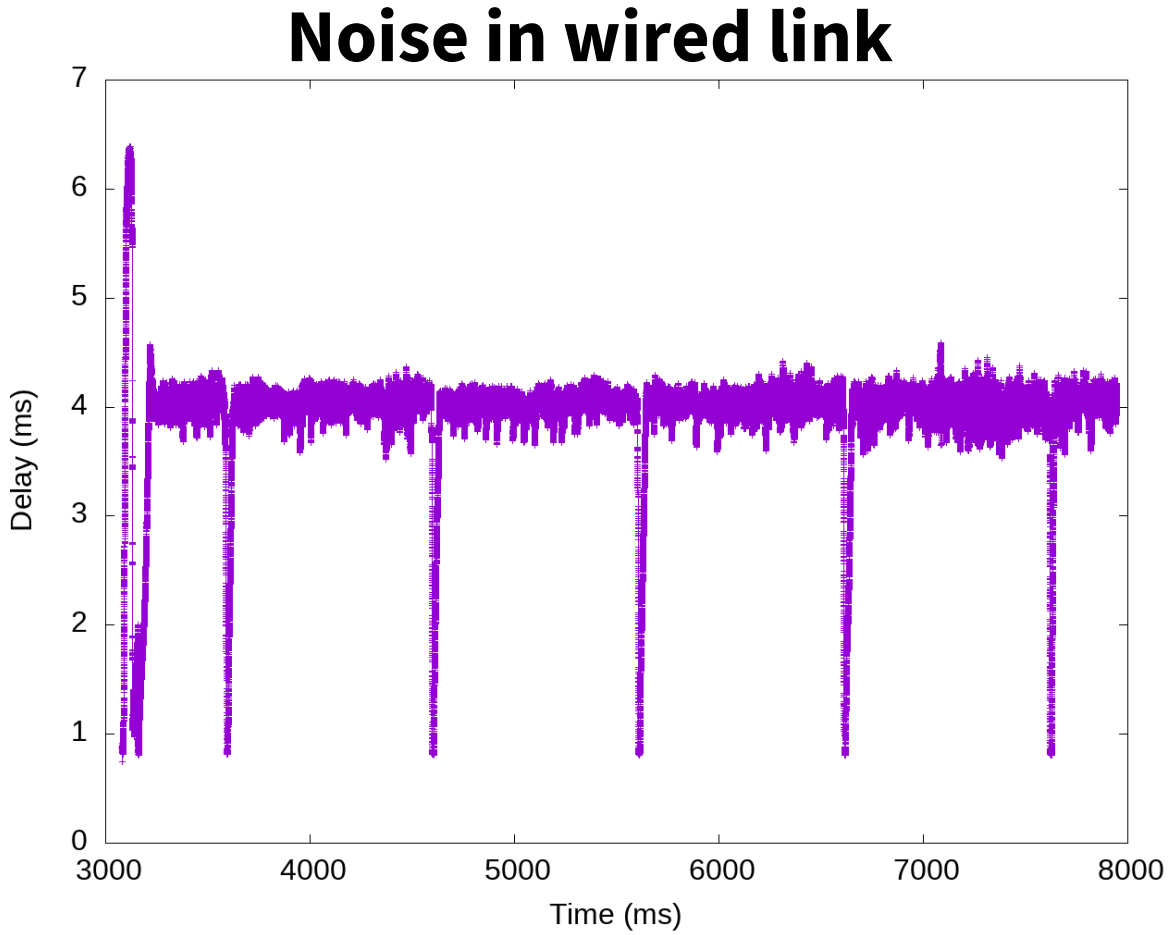
- **Small noise** creates
- **Arbitrarily large** error

Starvation!

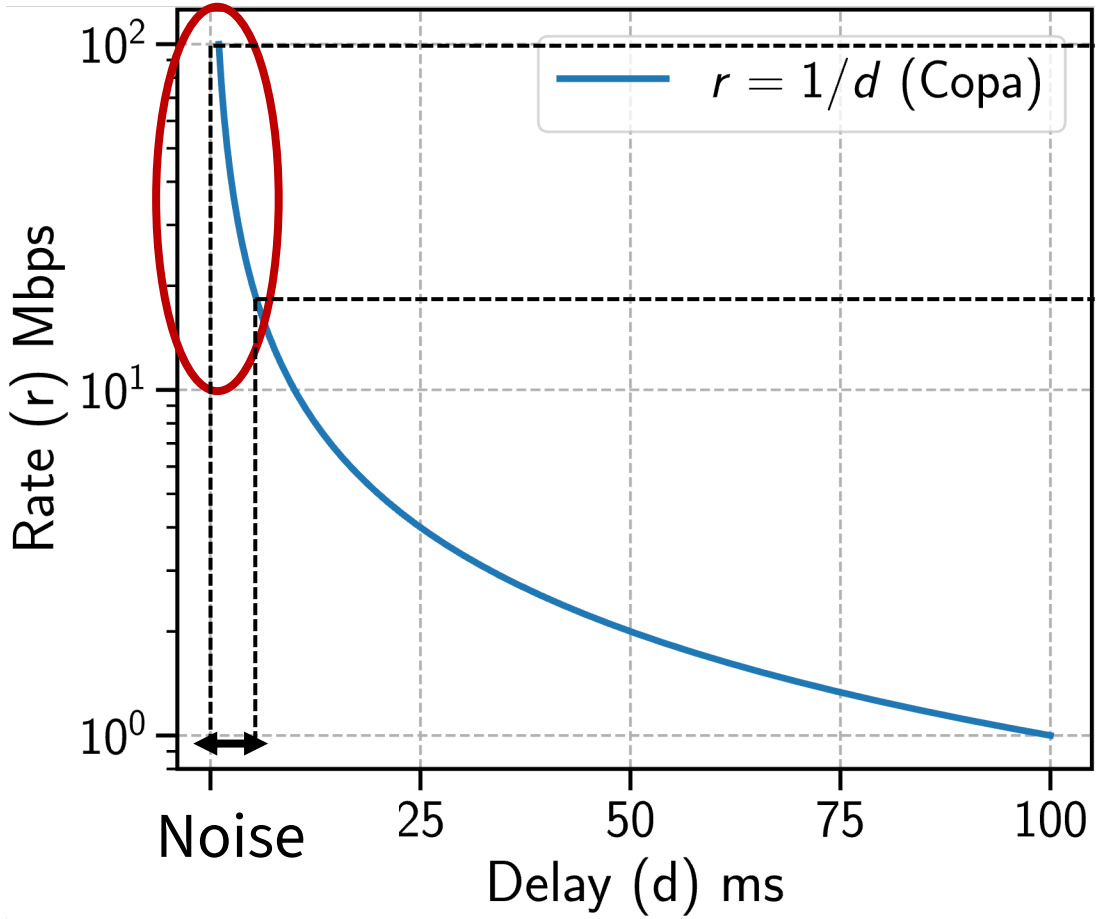
How can we avoid starvation?

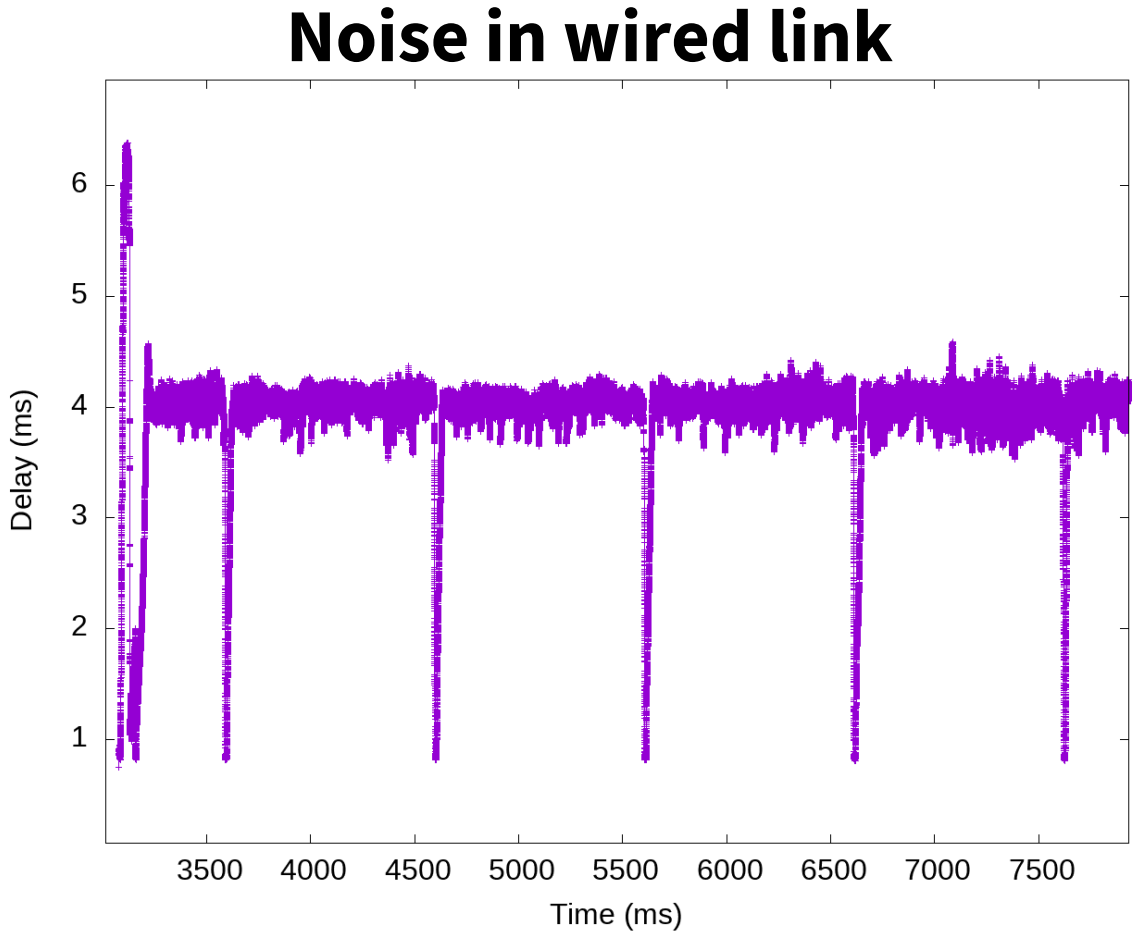
Strawman 1: Filter noise (min/mean/low-pass…) Hard to fully eliminate all noise



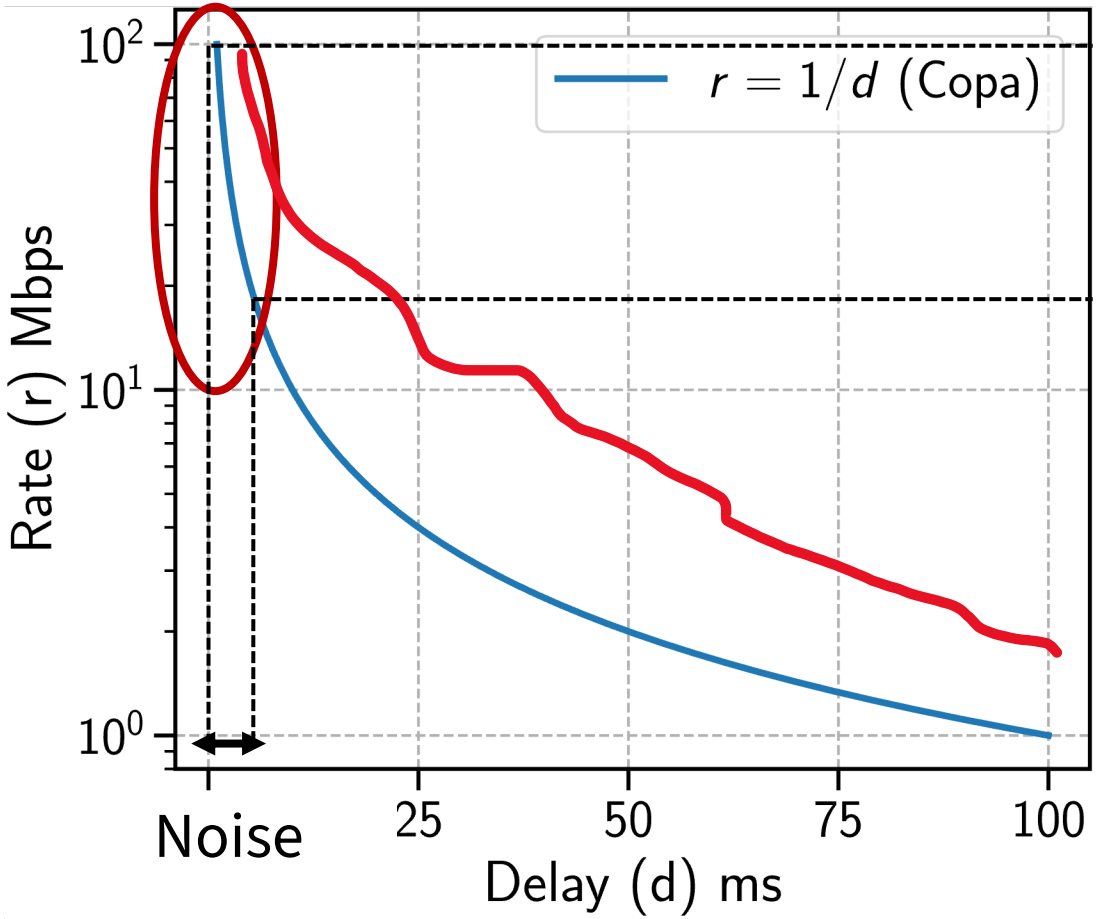


Strawman 1: Filter noise (min/mean/low-pass…) Hard to fully eliminate all noise

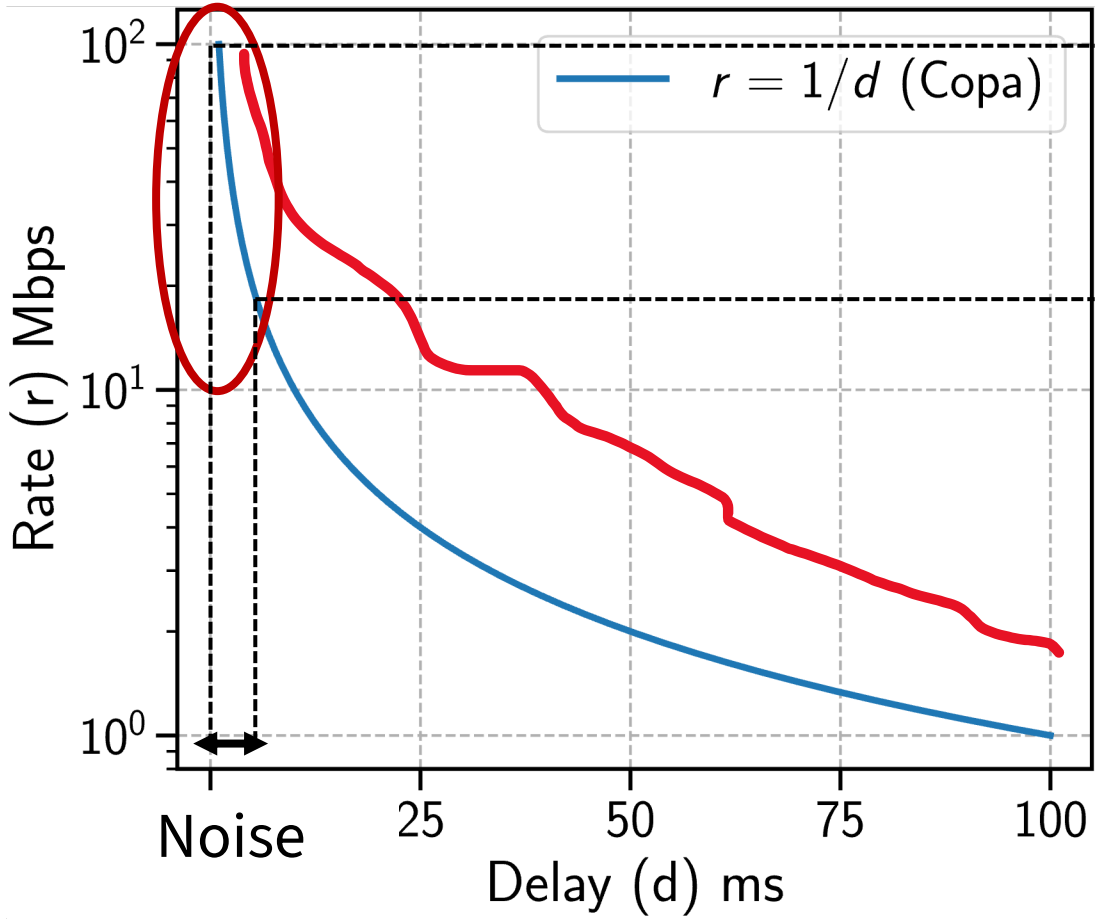




Strawman 2: Change the coordination function
All feasible rate-delay encodings have asymptote



Key Insight: Split fair-share: capacity & flow count
Encode less information ⇒ bound impact of noise



"Fair share"

= Capacity / Flow count

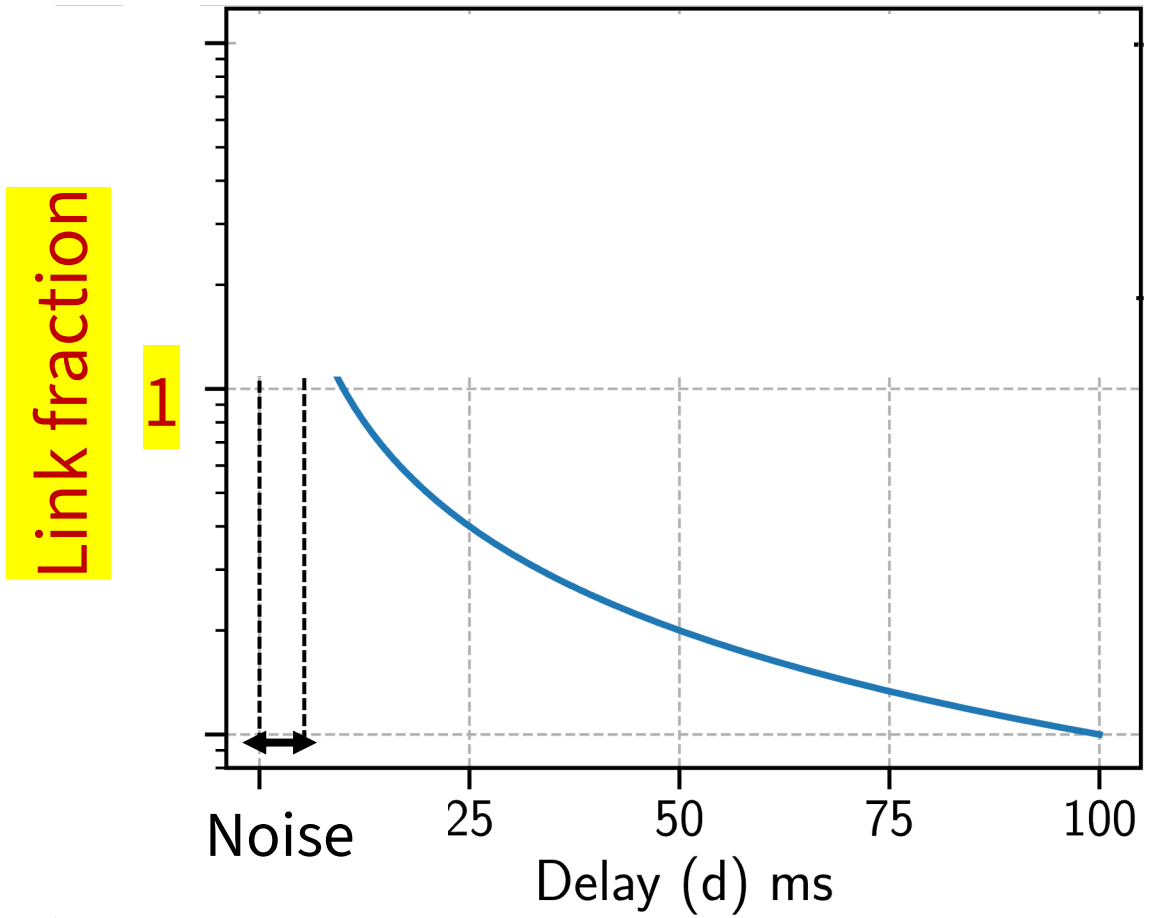
= Capacity * Fair link fraction

❷ Can probe directly

❶ Requires coordination

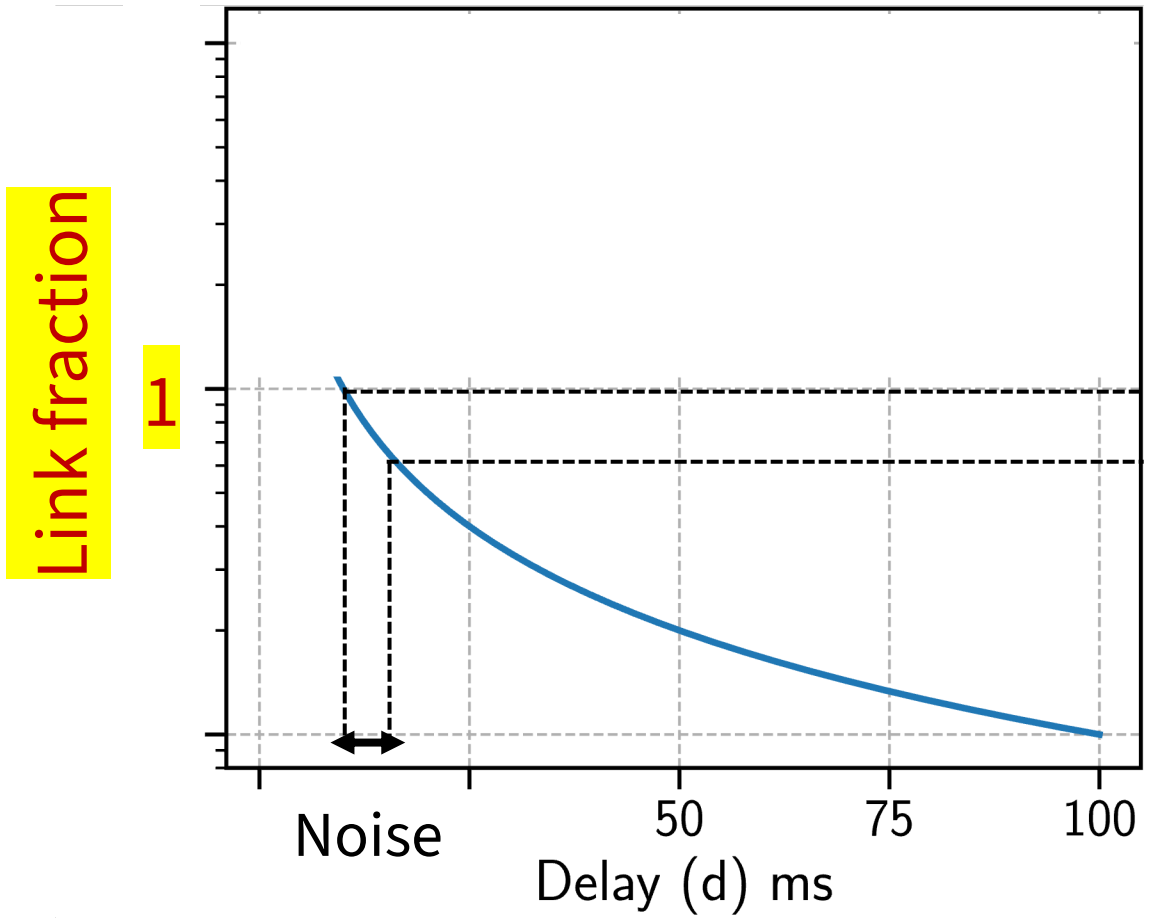
❶ Coordinate “fair link fraction”

Link fraction ≤ 1. No asymptote. No starvation!



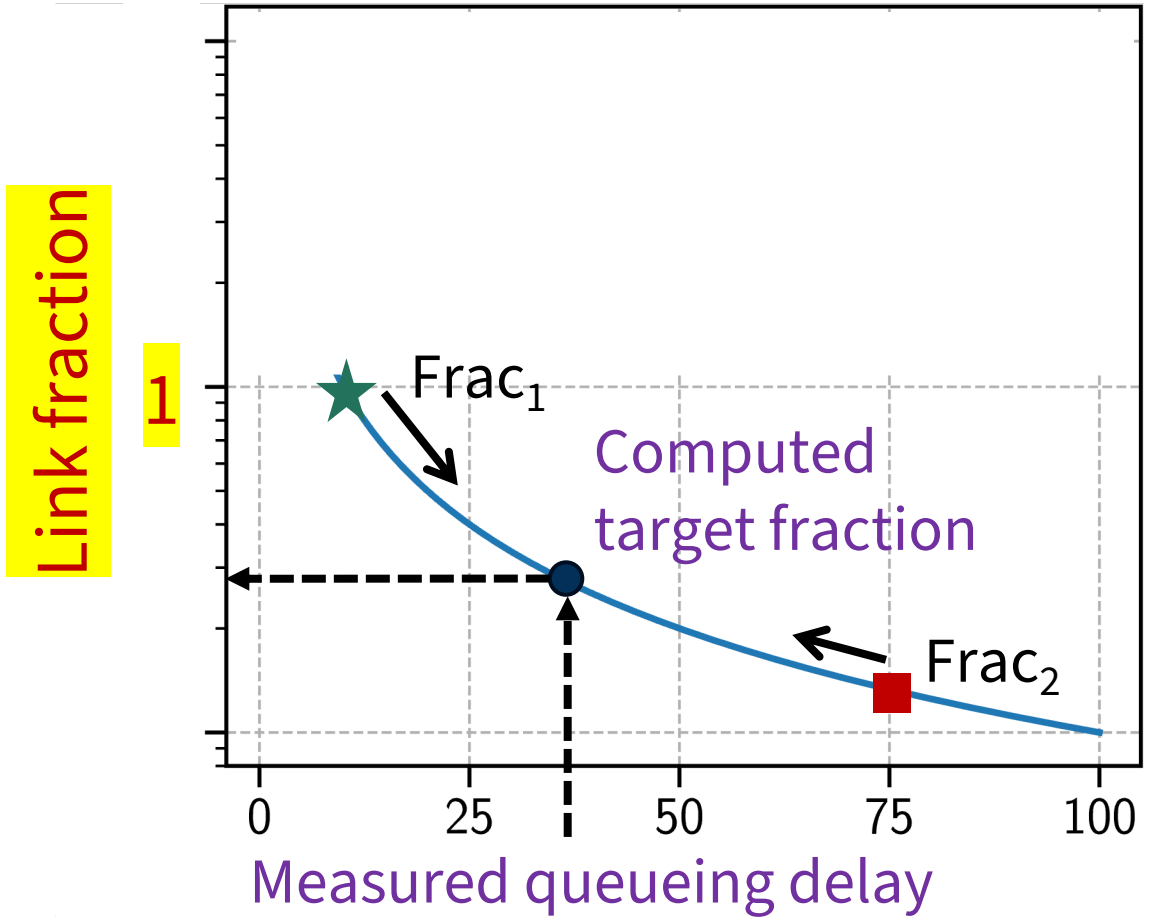
❶ Coordinate “fair link fraction”

Link fraction ≤ 1. No asymptote. No starvation!



❶ Coordinate "fair link fraction"

Link fraction ≤ 1. No asymptote. No starvation!



1. $Frac_i$ = Fraction of link consumed by $flow_i$

   = $Rate_i$ / Estimated capacity

2. Encoding of link fraction into delays:

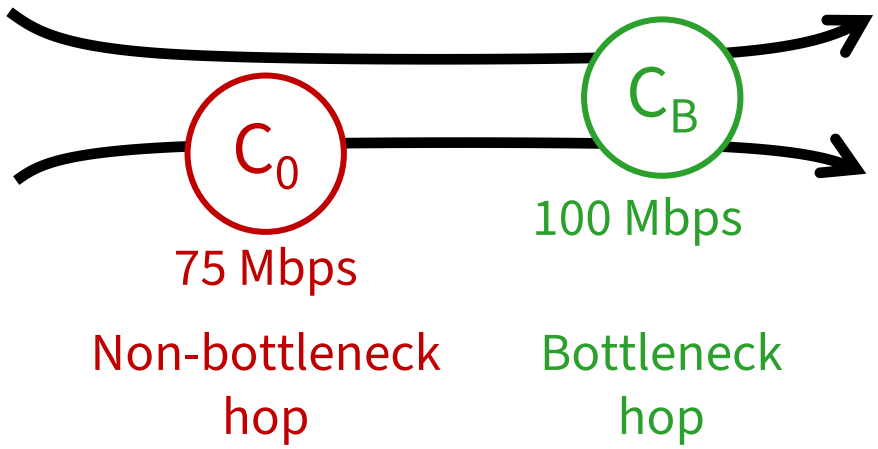
   Target frac = min(1, 1/delay)

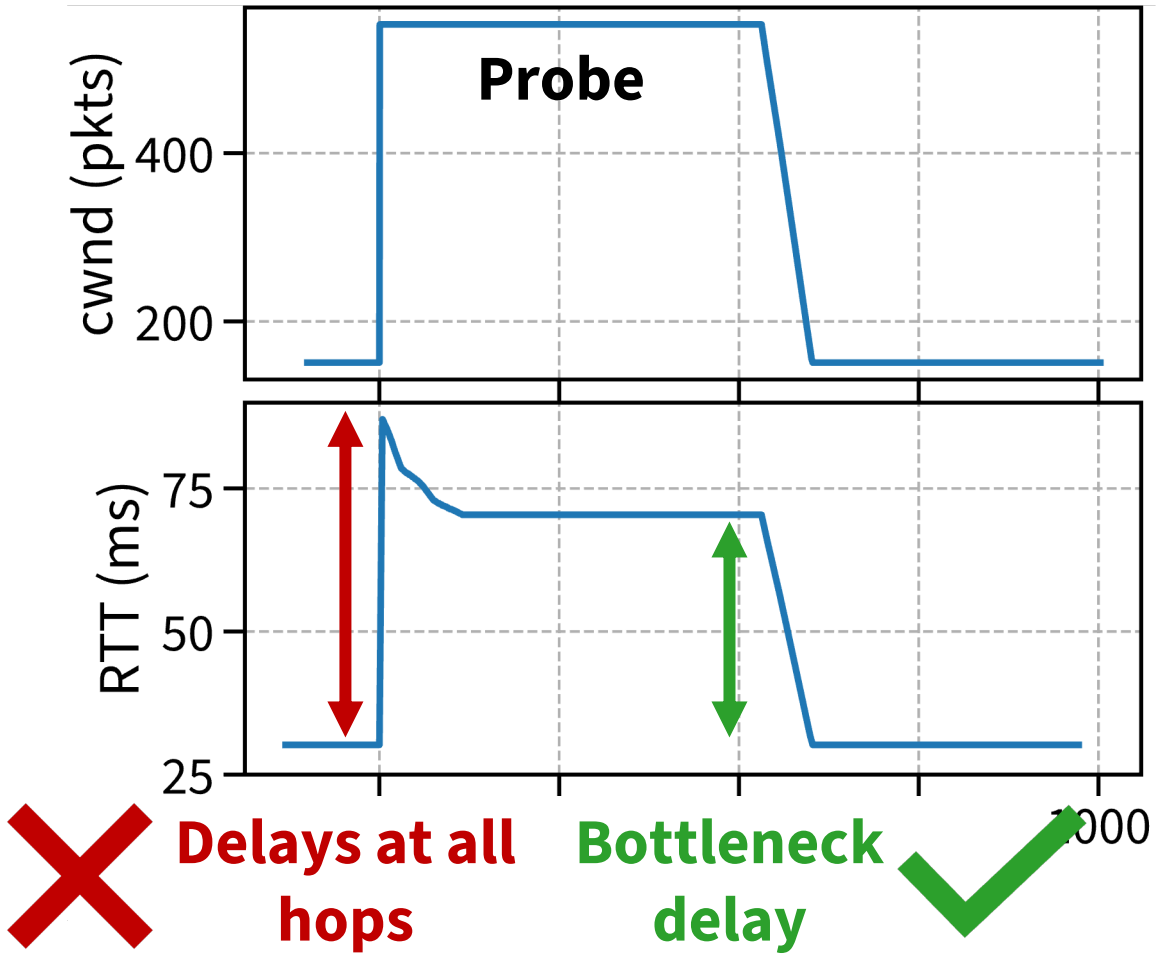
❷ Probe for link capacity

2a. Measure sustained change in delay by probe

Packet pairs, packet trains, …

$$\text{Capacity estimate} = \frac{\Delta\,\text{cwnd}}{\Delta\text{delay}}\ \frac{\text{bytes}}{\text{second}}$$





❷ Probe for link capacity

2b. Capacity probes need to be large enough

Want Δ delay >> noise

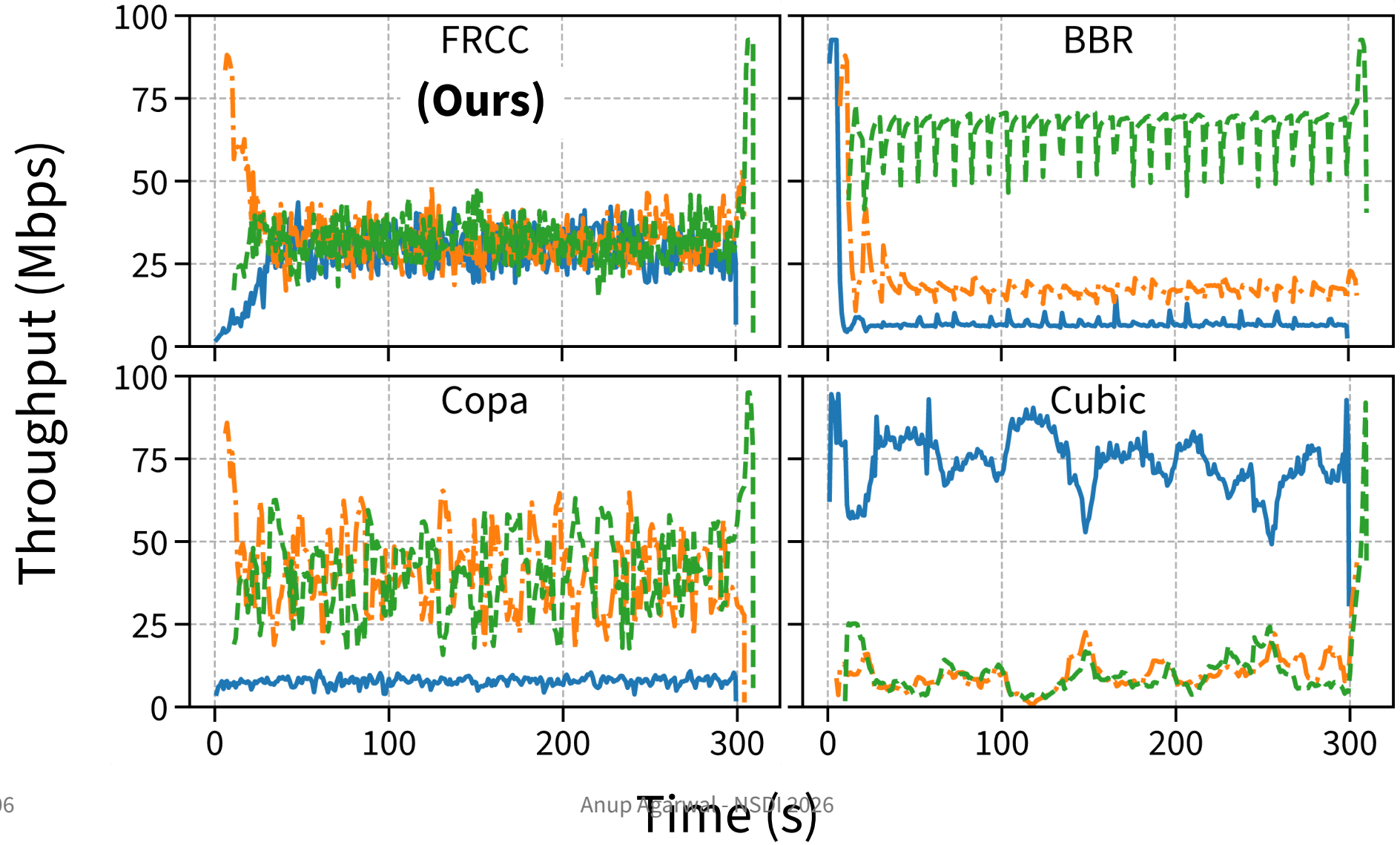
Δ cwnd = 0.25 * cwnd (small fish in big sea)

Δ cwnd = Link capacity * const

= **Flow count** * sending rate * const

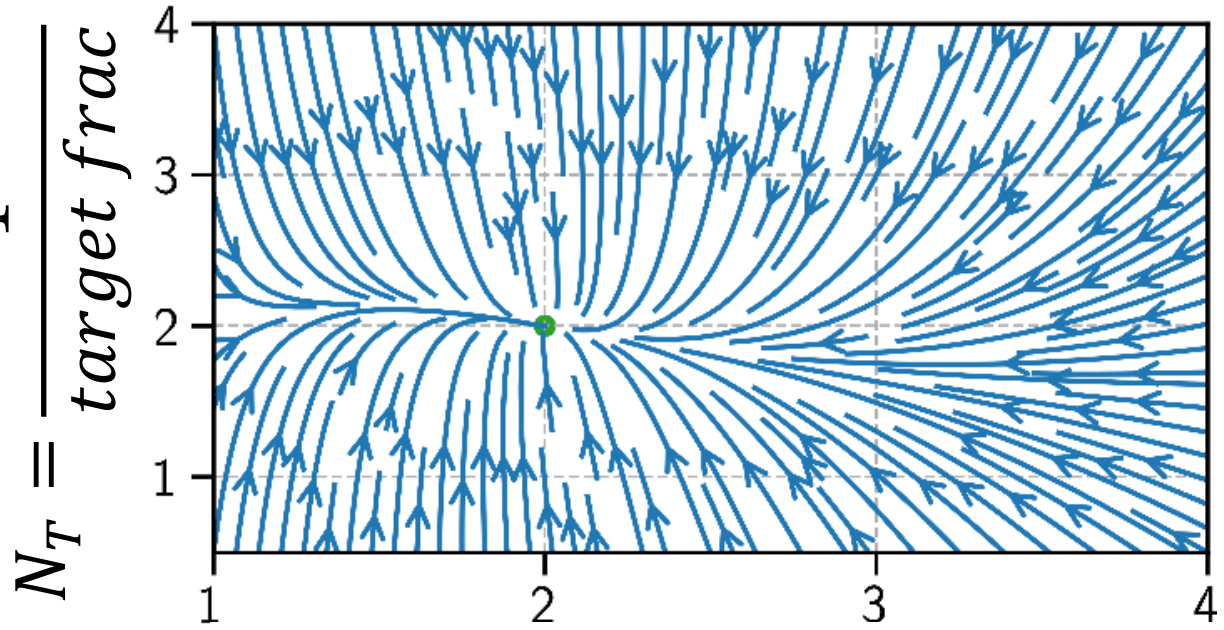
= **(1/Target fraction)** * sending rate * const

FRCC: first CCA to ensure fairness under noise



[Case 1. No noise] FRCC achieves fairness

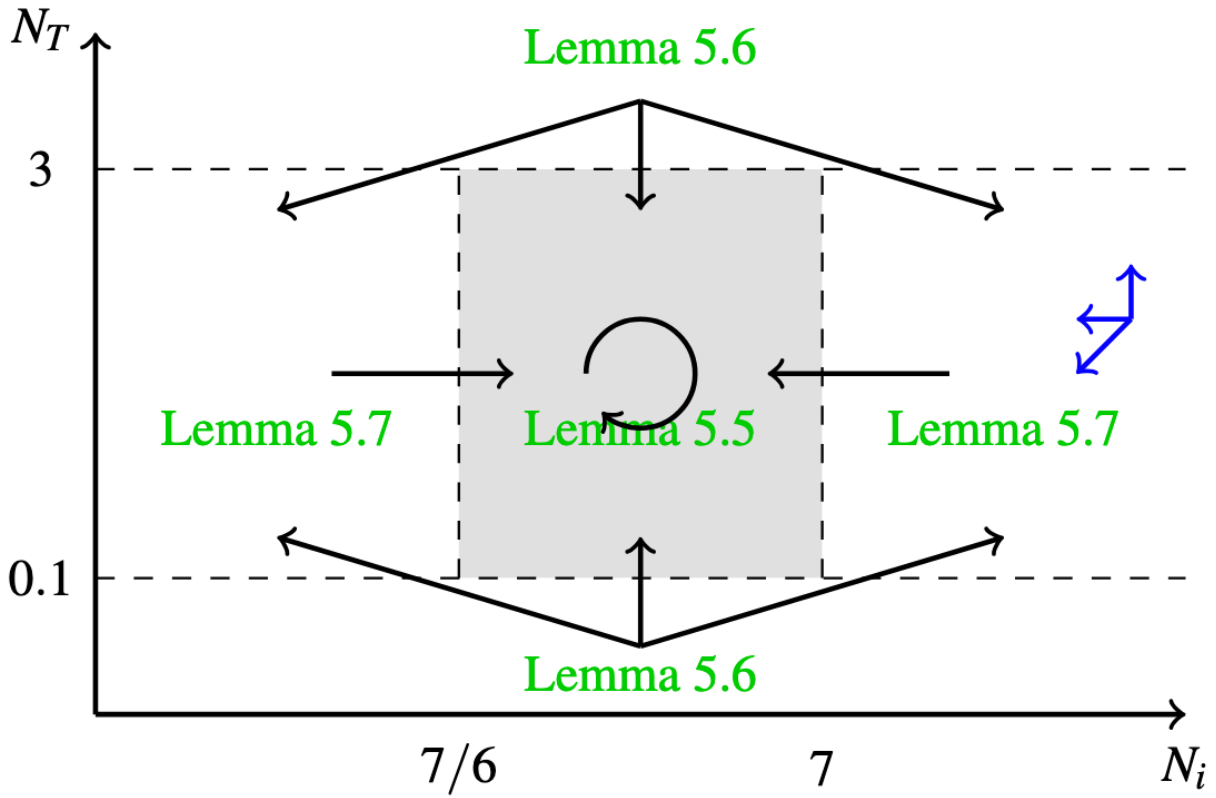
Theorem: (Dynamical system) N_i, N_T converge to the ground truth flow count.



$$N_i = \frac{1}{frac_i}$$

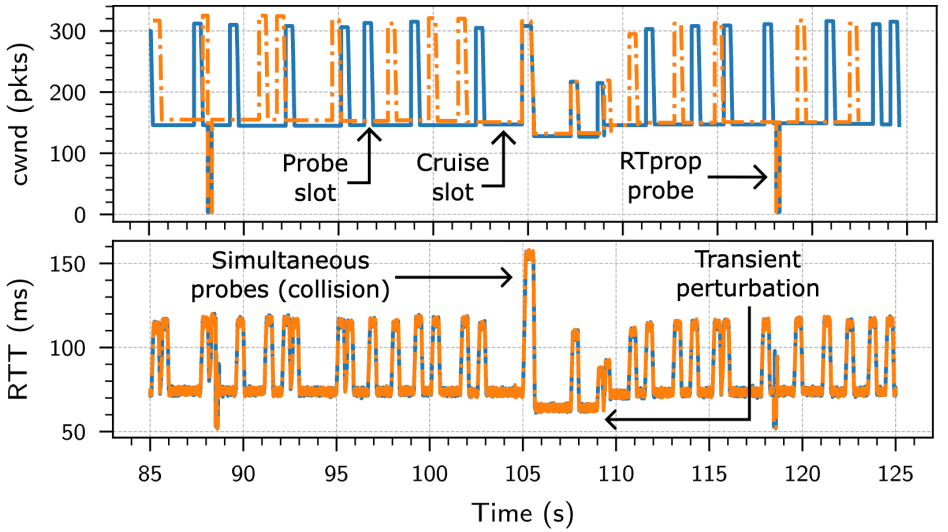
[Case 2. With noise] FRCC achieves fairness

Theorem: (Using Z3) For all noise patterns, N_i, N_T converge towards ground truth flow count and remain near it.



More in the paper

1. **Manage simultaneous probes**
2. **Flows with vastly different RTTs**
3. **Multiple bottlenecks in a path**
4. **…**



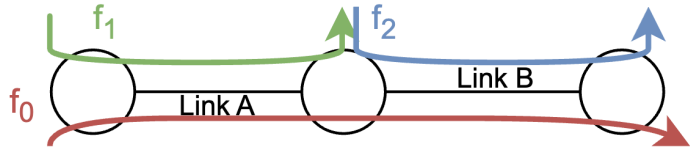


Figure 17: Parking lot topology. Flow f_0 gets observations from both hops, f_1, f_2 only see a single hop.

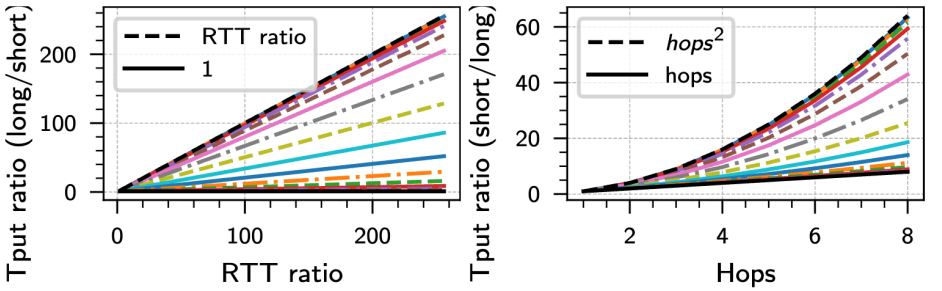


Figure 18: Throughput ratios at the fixed point with different RTprops (**Left**), and parking lot topology (**Right**). The different lines correspond to different choices of probe size ($\gamma\mathcal{D}$) relative to the RTprop of the short flow (with $\gamma\mathcal{D}/R$ ranging from 2^{-7} to 2^{8}). Large probe size yields better (lower) throughput ratios.

FRCC: Fair and robust congestion control

1. **Directly coordinating fair shares susceptible to starvation**
2. **Embrace noise in control loops**
3. **Decouple fair shares into**
    1. Flow count (coordinate)
    2. Link capacity (probe independently)

Contact: 108anup@gmail.com

Code: https://github.com/108anup/frcc

Thank you!

Backup slides

Limitations & Future work

- **Limitations:**
  - Capacity probes are too long/large (shallow buffers)
  - Convergence time

- **Future work:**
  - Improve capacity estimation
    - INT based datacenter CCAs (e.g., Poseidon/HPCC) may use separate signaling for bottleneck capacity and congestion (flow count)
  - Conservative tuning
  - Overprovision the Internet
  - Fair queueing

Timeseries of FRCC