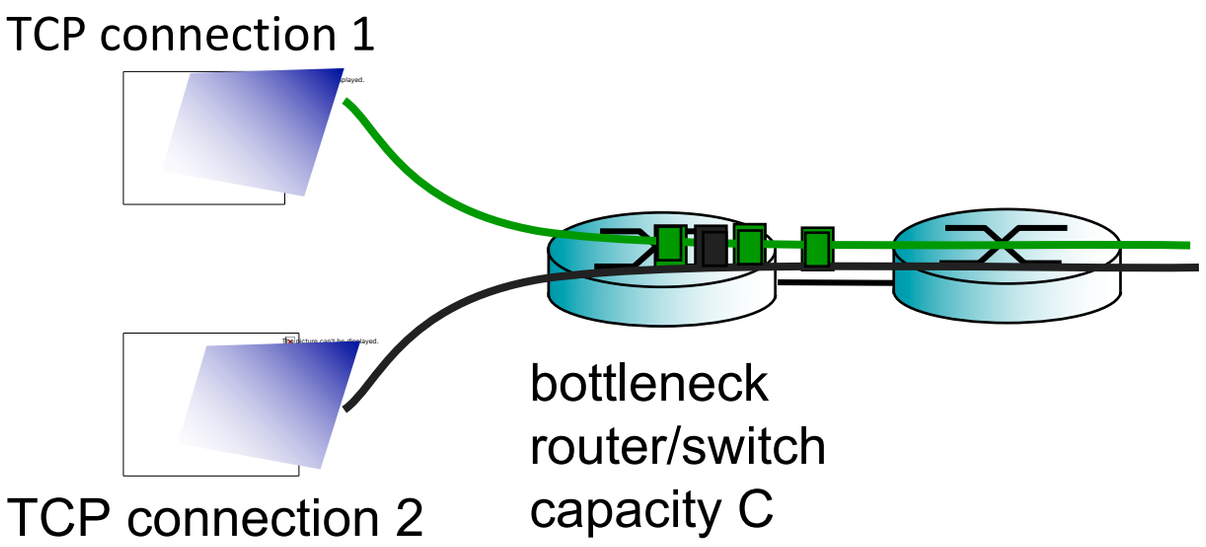
A Principled Look at the Utility of Feedback in Congestion Control

Mohit P. Tahiliani, Vishal Misra and K. K. Ramakrishnan

National Institute of Technology Karnataka, India,
Columbia University NY,
University of California, Riverside, CA

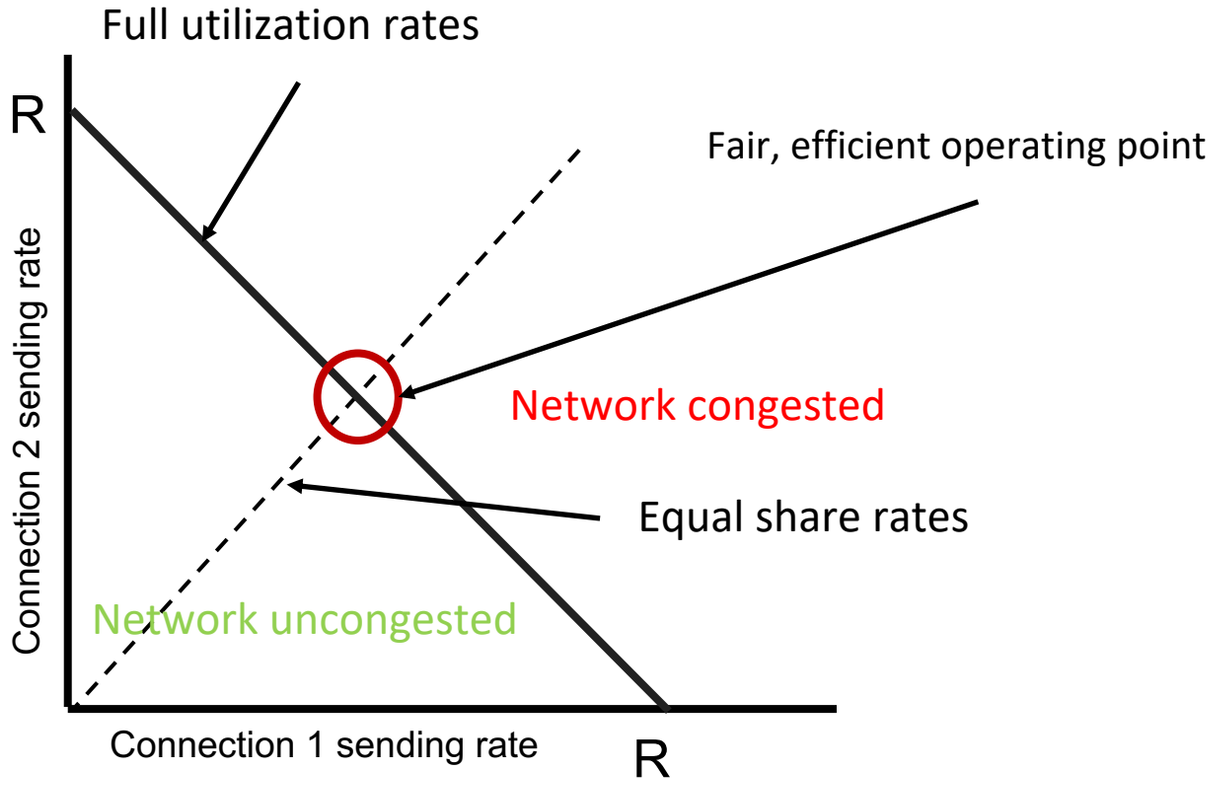
Distributed Congestion Control

- Distributed*:* Flows need to seek their available capacity without global knowledge, at least at fine time granularity
- Performance*:* Do not exceed capacity of shared resource in the network
- Fairness goal: if N (TCP) sessions share same bottleneck link of bandwidth C, each should have average rate of C/N
- Scalability goal: support large N, without global knowledge



Strike a balance between throughput and delay

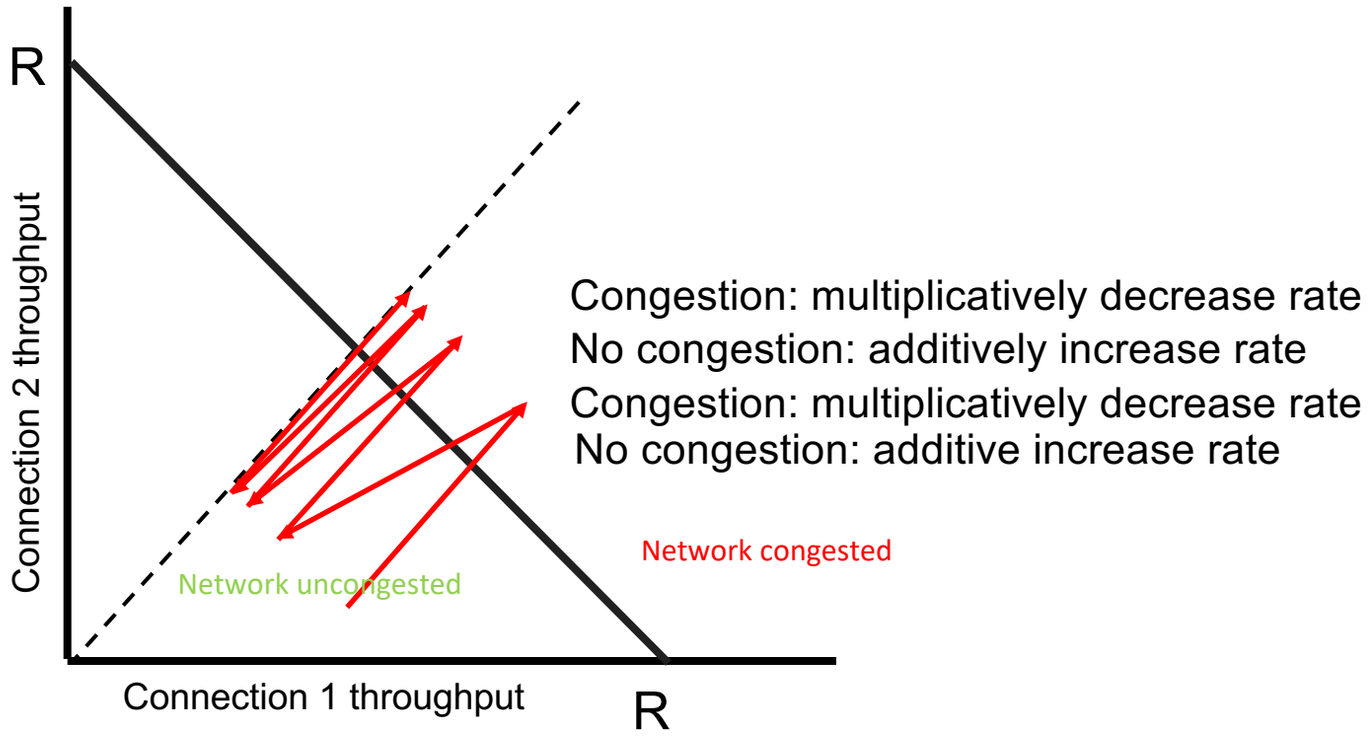
Classical analysis: Two flows sharing a bottleneck



- Distributed Congestion control
  - ❖ Flows should detect whether the network is congested or not
  - ❖ Adjust their sending rates appropriately: increase if network is uncongested, decrease if it is congested
  - ❖ Operating at 'Full utilization rate'– how should flows react once they reach that point?
    - ○ Convergence to one point: efficient and fair?

AIMD Congestion Control

- Two competing sessions:
  - additive increase gives slope of 1, as throughput increases
  - multiplicative decrease reduces throughput proportionally



The work at DEC showed in full generality for N flows that AIMD converges to fair, efficient point

Why re-examine feedback for congestion control?

- Increasing heterogeneity in the network and flows:
- Managing congestion based on ambiguous signals difficult. Need to:
  - Infer where congestion is
  - For an end-point, infer if its flow is the cause
- E.g., Flows sharing a bottleneck and only feedback is end-to-end delay: how many flows are sharing is not known, even when converging to a fixed delay

Resolve 2 forms of ambiguity:

- What is the dimension of the space we are operating in? i.e., how many flows are sharing the bottleneck? What is N? (fairness ambiguity)
- How far away from the efficiency line are we? (congestion ambiguity)

When the system converges to an operating point – does it achieve both efficiency and fairness?

Two approaches to signal congestion

1. End-to-End, implicit signaling: use observed delay or loss to infer congestion
   a. inferring congestion via packet losses is wasteful
   b. using measured delay is **very** easy to deploy - needs only end-point implementation
   c. hardware/software advances mean **extremely** accurate measurements of delay available, especially in data centers
   d. accurate measurements imply high quality, multi-bit feedback is available for CC algorithms to operate upon
2. Network assisted explicit signaling: use AQM to mark ECN bits
   a. need network **and** end-point support. deployment is not easy
   b. ECN is a single bit feedback. does not offer the rich, multi-bit feedback of delay measurements
   c. configuring AQM at routers difficult, so only simple threshold based AQMs deployed

Implementation concerns aside, is there any reason to choose one over the other?

Fairness ambiguity

- **Theorem*: (Fairness/Delay Tradeoff).**

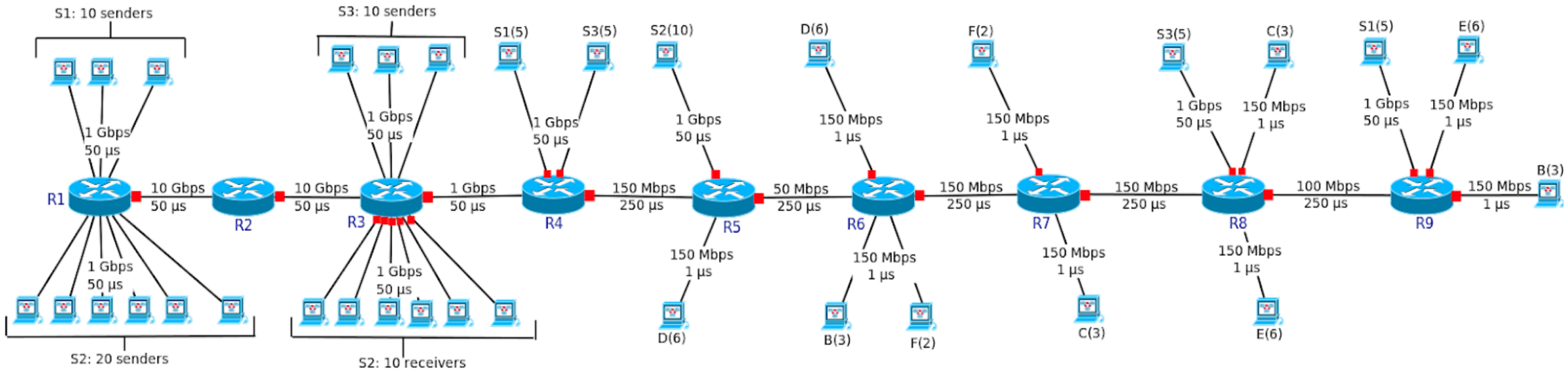
For congestion control mechanisms that have steady state throughput of the kind R = f(d,p), for some function f, delay d and feedback p, if the feedback is based on purely end to end delay measurements, you can either have fairness or a fixed delay, but not both simultaneously.

- **Proof idea:**

without in-network (ECN) feedback, flows don't have information on who else is sharing the bottleneck. Results in number of unknowns that are larger than number of equations -> infinite fixed points -> unfairness

*Yibo Zhu, Monia Ghobadi, Vishal Misra and Jitendra Padhye, **ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY**, *Proceedings of 2016 ACM Conference on Emerging network experiment and technology (CoNEXT 2016)*, December, 2016.

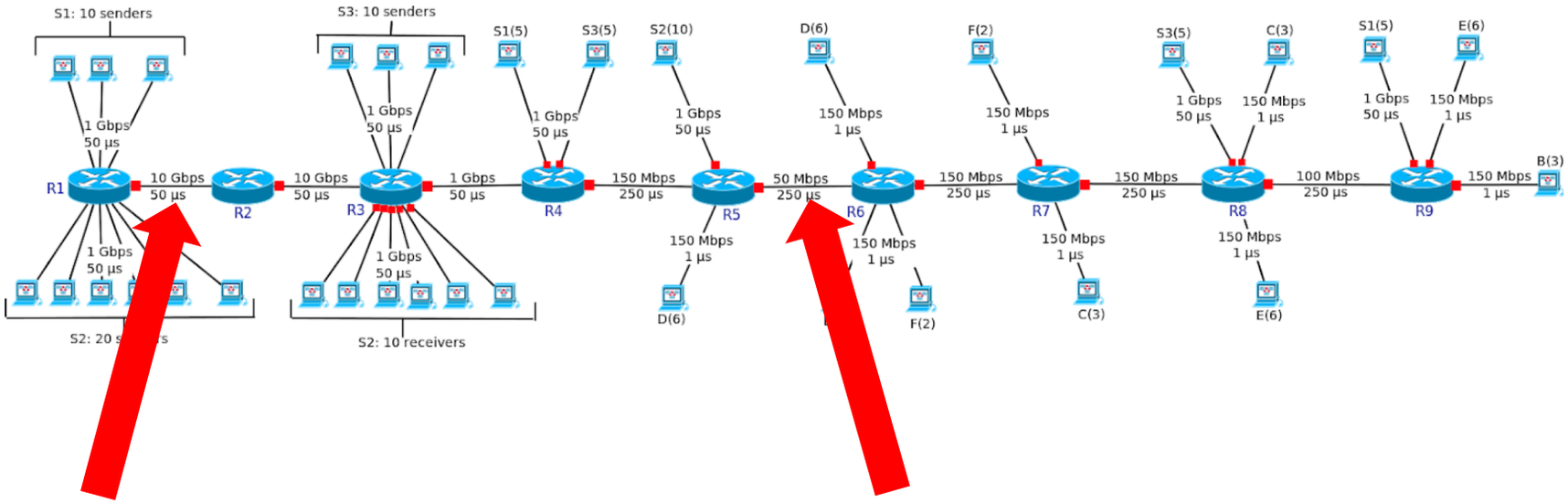
Congestion ambiguity



Hybrid topology (datacenter + WAN) with significant diversity in link capacities
10 Gbps to 50 Mbps. DC topology (in DCTCP) meets WAN (GFC-2)

Congestion ambiguity

End-to-End delay feedback cannot disambiguate congestion points



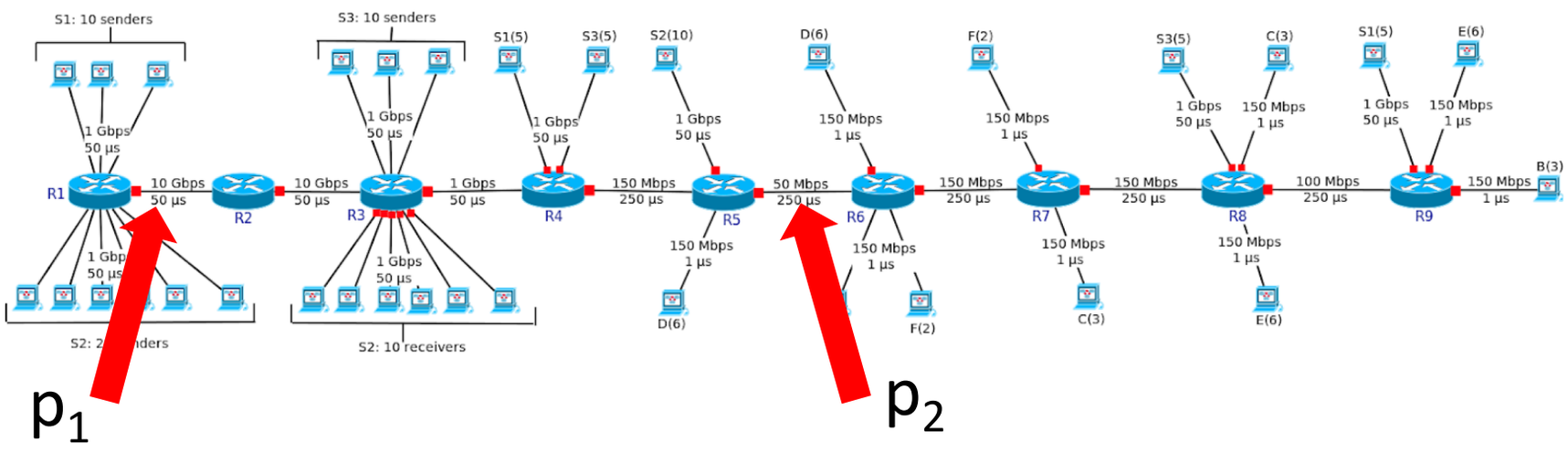
1 ms of increased delay -> 1200 packets in queue

1 ms of increased delay -> 6 packets in queue

Any increase in delay leads to same reaction, regardless of severity

(0+1)ms is interpreted same as (1+0)ms

Properly configured ECN - disambiguates



Aggregate feedback = $1-(1-p_1)*(1-p_2) = 1- 1+ p_1 + p_2 - p_1*p_2$

$\sim p_1 + p_2$ (if p_1 and p_2 are small)

- Extra 1ms delay @ 50 Mbps link: both p1 and p2 low; protocol makes a minor sending rate adjustment;
- Extra delay @ 10 Gbps link, then p1 high: protocol will cut sending rate by a lot.

Can in-network feedback help?

- Number of in-network ECN-style feedback introduced
  - ECN_a marks → probability; function of # flows sharing bottleneck, as in AQM (e.g., PI)
  - Protocols that adapt (increase/decrease) based on ECN marks converge to unique fixed point
  - ECN_t threshold on buffer size: marking is interpreted as level of congestion, but is also dependent on window of measurement
  - Protocols that adapt (decrease) based on level of congestion inferred by # marks may reach an efficient operating point, but can settle on a point that does not guarantee fairness

A note on the PI AQM

- PI computes marking probability p(t) every update interval as

$$p(t+1) = p(t)+a*(q(t)-q_{ref})+b*(q(t-1)- q_{ref})$$

where q_{ref} is some threshold, a and b are controller parameters.

- It converges/stabilizes when $q(t) = q(t-1) = q_{ref,}$ i.e. when the buffer has *cleared* ($q(t) = q_{ref}$) and rates have *matched* ($q(t) = q(t-1)$)

PI feedback **contains rate mismatch** information

Key takeaway

Accuracy of measurement doesn’t help if the quantity measured is ambiguous

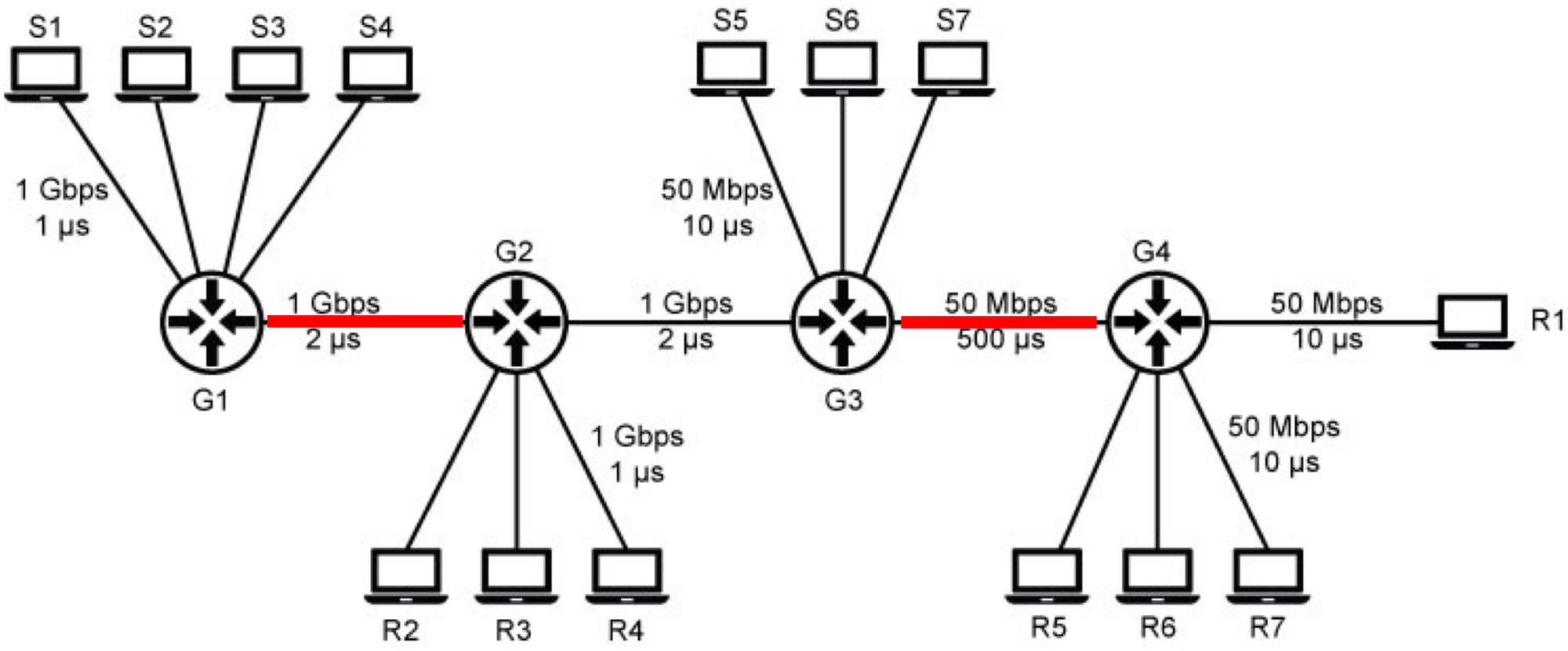
Single bit, crude ECN feedback can contain **more information**

Emulation Results

Model and Evaluation Framework

- Testbed of emulated data sources, sinks and routers (network namespaces)
- Link: A pair of ‘veth’ interfaces make up a link
- On every interface:
  - Hierarchical Token Bucket queue discipline - to set the TX bandwidth.
  - A NetEm (Network Emulator) qdisc to artificially delay the packets that are being transmitted.
- On an interface of a bottleneck link, packets to be transmitted are redirected to an AQM qdisc on an IFB (Intermediate Function Block) device.
- Stats collection using Flexible Network Tester (Flent)
- BBRv2 model: v2alpha branch at https://github.com/google/bbr (July 2019)
- CUBIC model: default in the Linux kernel (Ubuntu 5.2.0-rc3+)

Hybrid DC-WAN topology



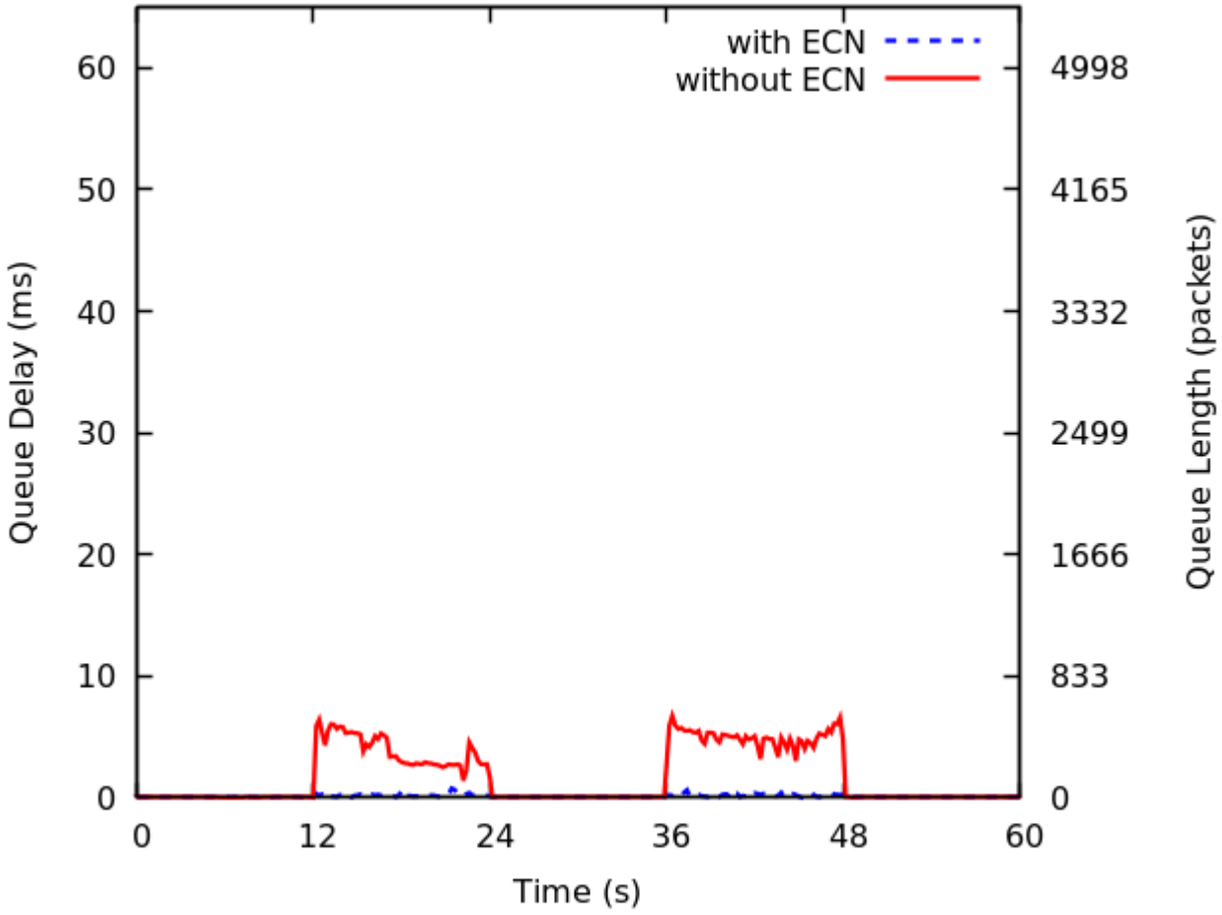
S1: long lasting TCP flow crossing both bottlenecks

S2 - S7: Short lived TCP flows, having ON and OFF duration of 12 seconds each

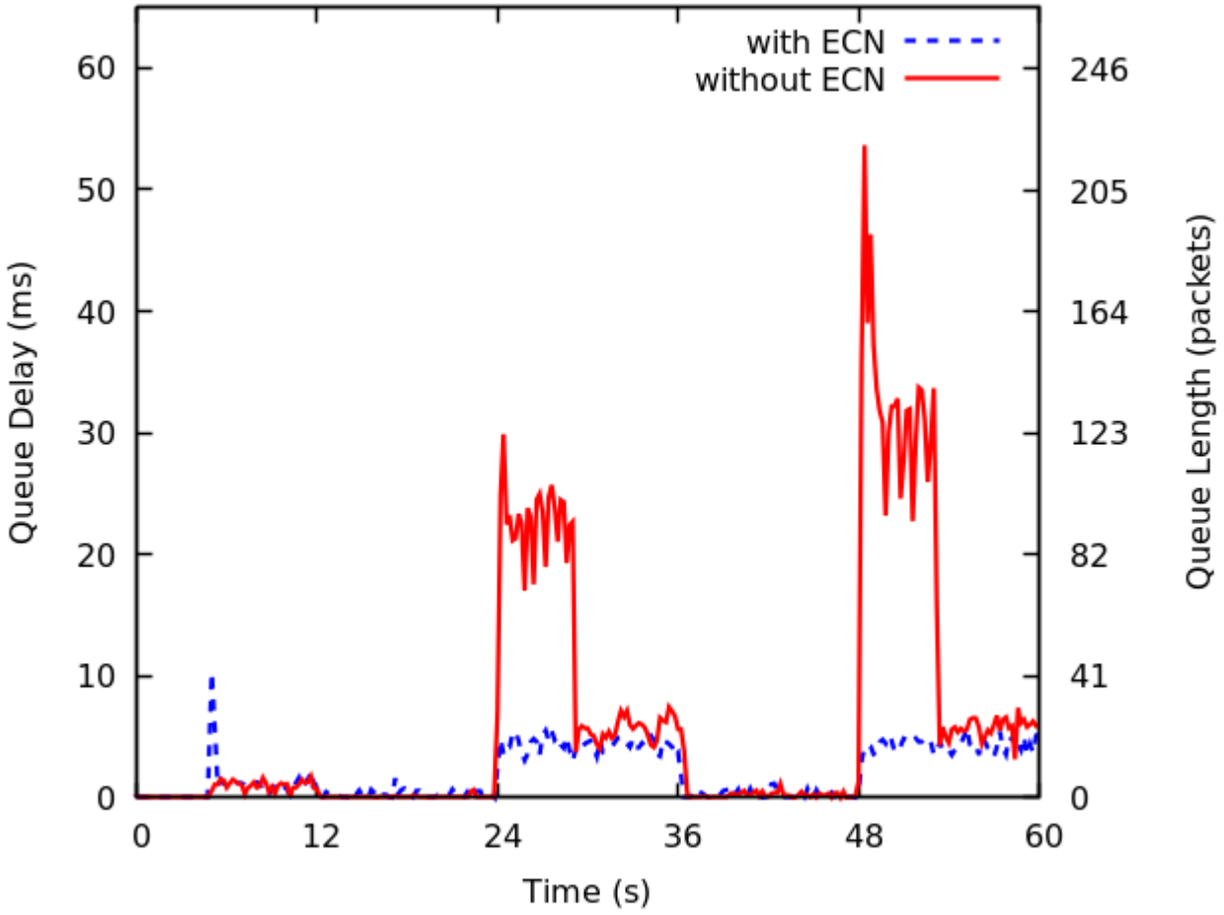
Comparison of Alternatives

- BBRv2 (adapted to not have ECN_t threshold based marking) ➔ pure end-end scheme without in-network feedback
- BBRv2 (with ~DCTCP style ECN_t threshold based marking)
- TCP-CUBIC with ECN_a - AQM based marking, using PI controller.
  - Supporting an AQM style ECN with BBRv2 more complex – still examining needed changes
- One bottleneck link in DCN (G1-G2); another in WAN (G3-G4)
- Reference queue length: 20 packets
- One flow (S1) is active throughout the simulation period
- 3 flows (S2-S4) in the DCN are active from 12-24 sec., and 36-48 sec.
- 3 flows (S5-S7) in the WAN are active from 24-36 sec., and 48-60 sec.

Queue Occupancy: BBRv2 with and without ECN



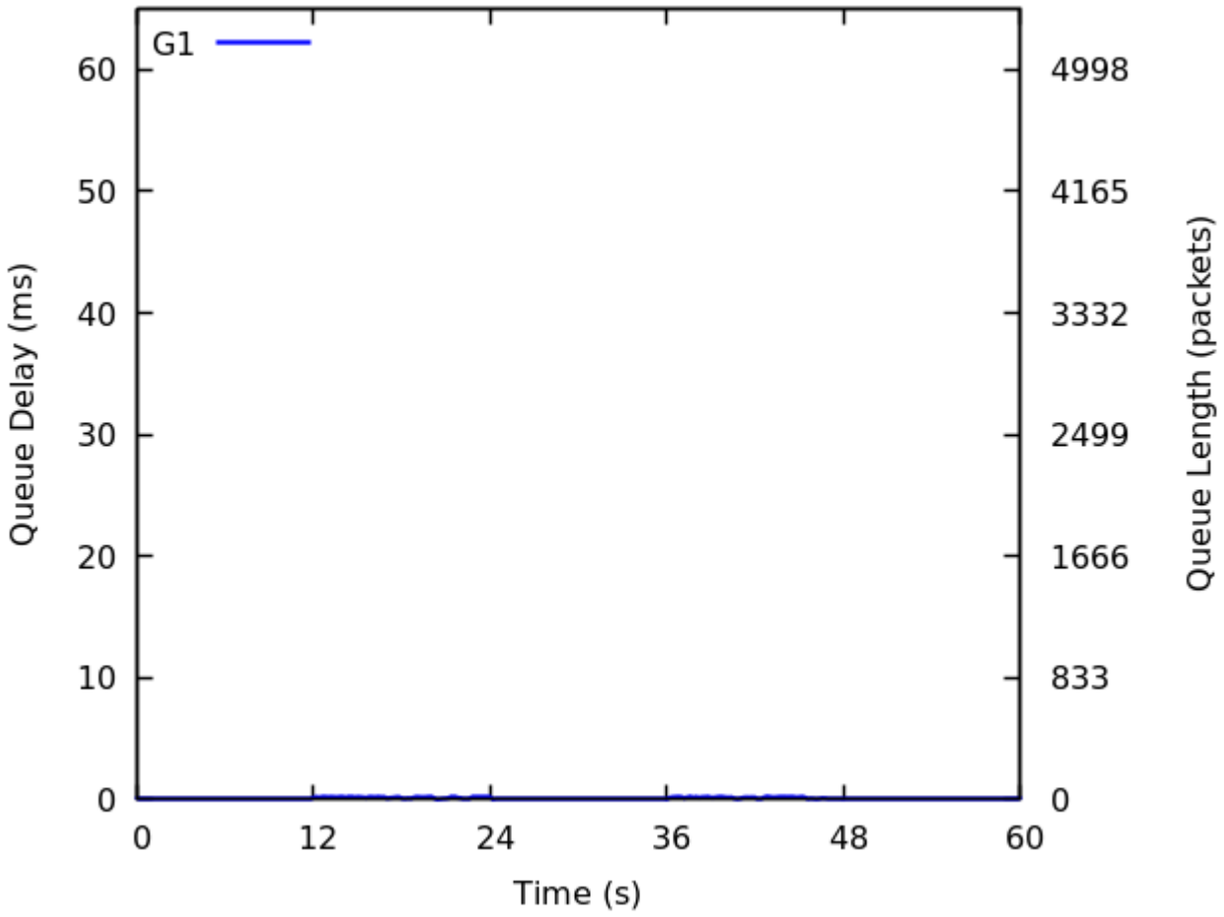
Router G1



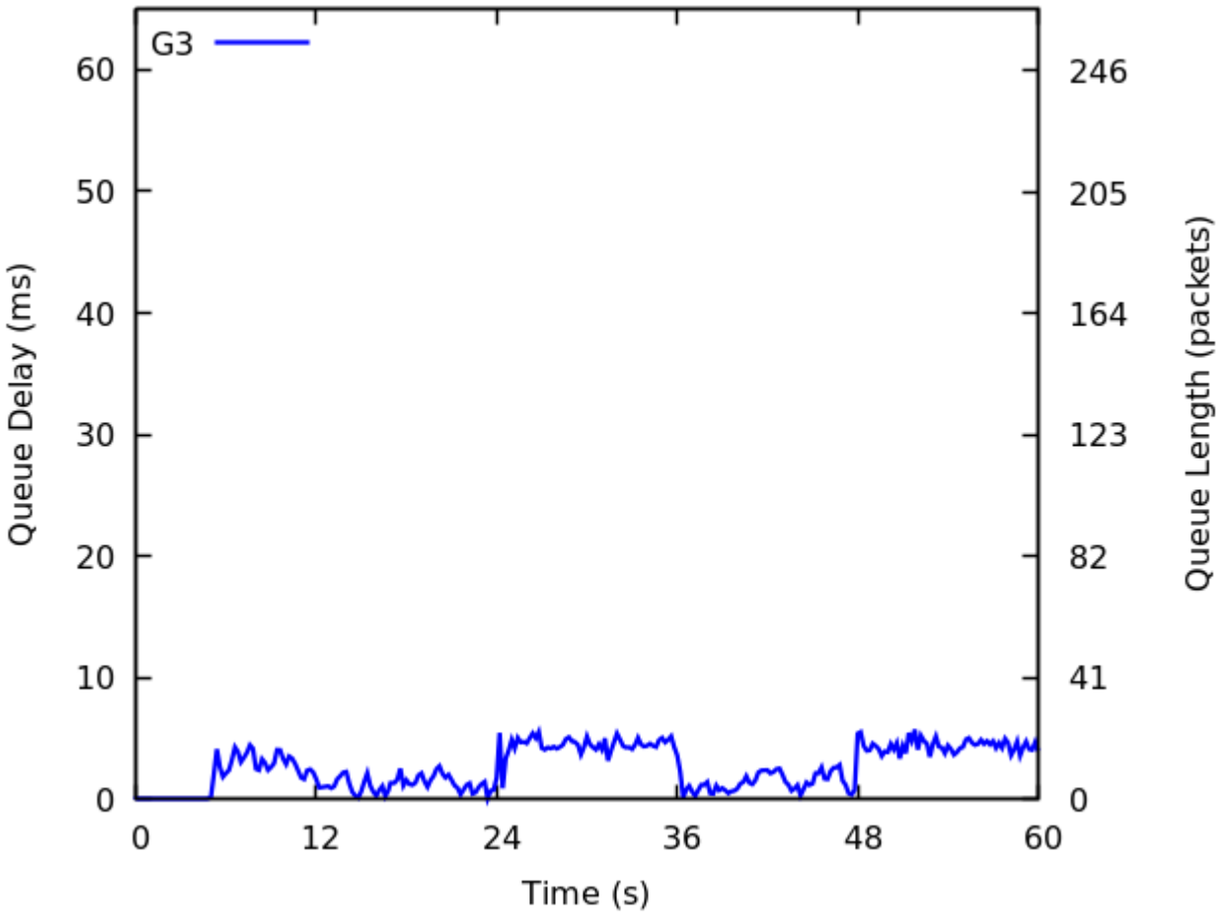
Router G3

- BBRv2 without ECN – More difficult to control the queue size
- Even BBRv2 - once ECN is introduced, provides better control on the queue

Queue Occupancy: TCP CUBIC with PI+ECN



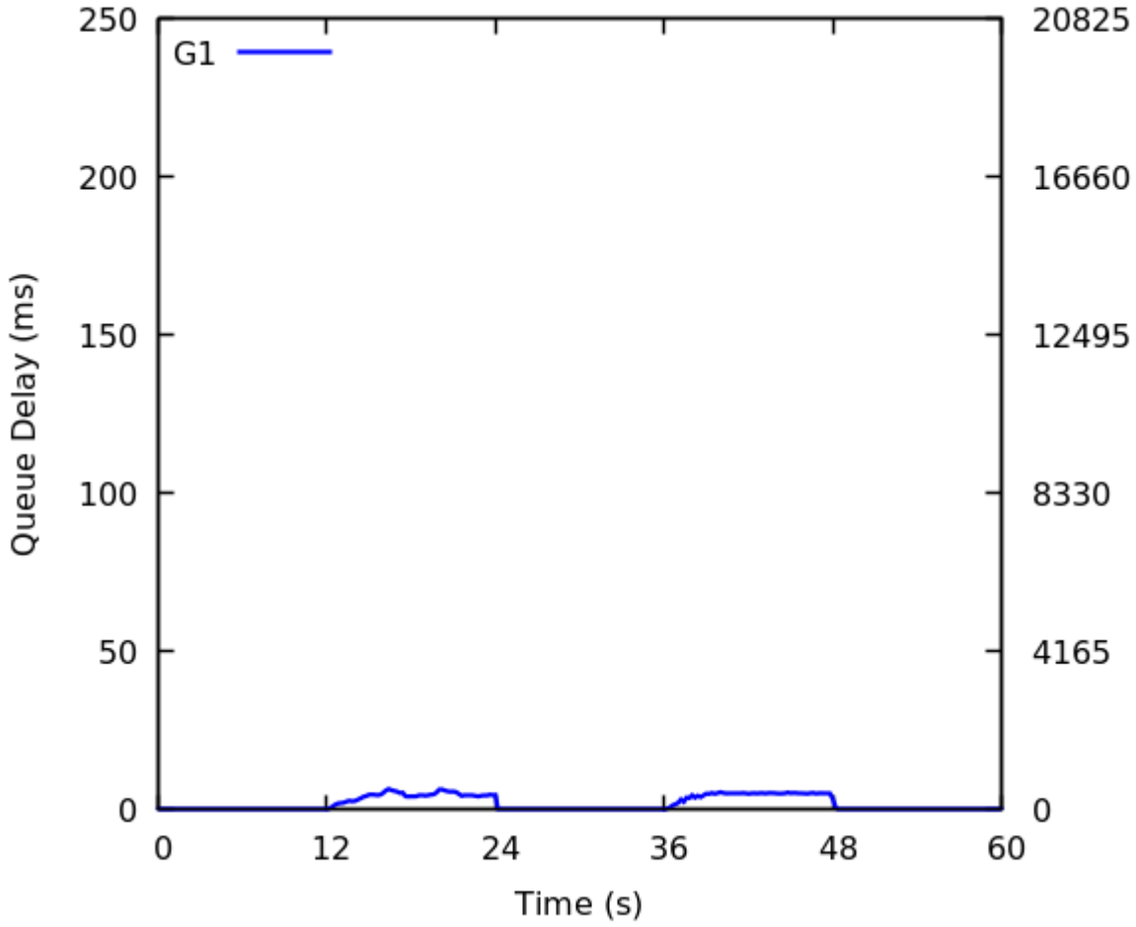
Router G1



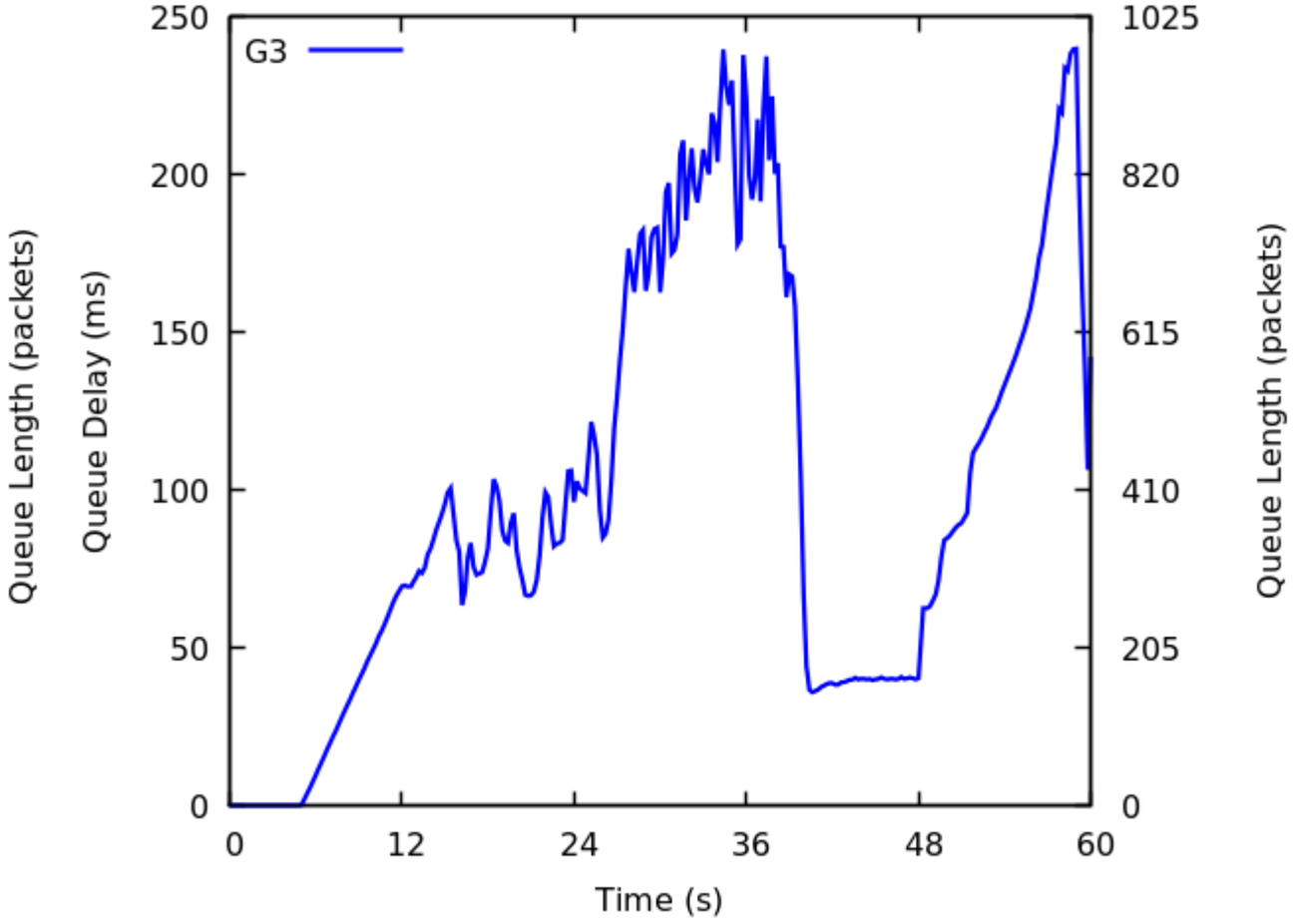
Router G3

- With CUBIC including PI+ECN – Able to control the queue occupancy
  - Queue remains close to (just below) the set point of 20 packets.

Queue Occupancy: Default CUBIC (without PI+ECN)



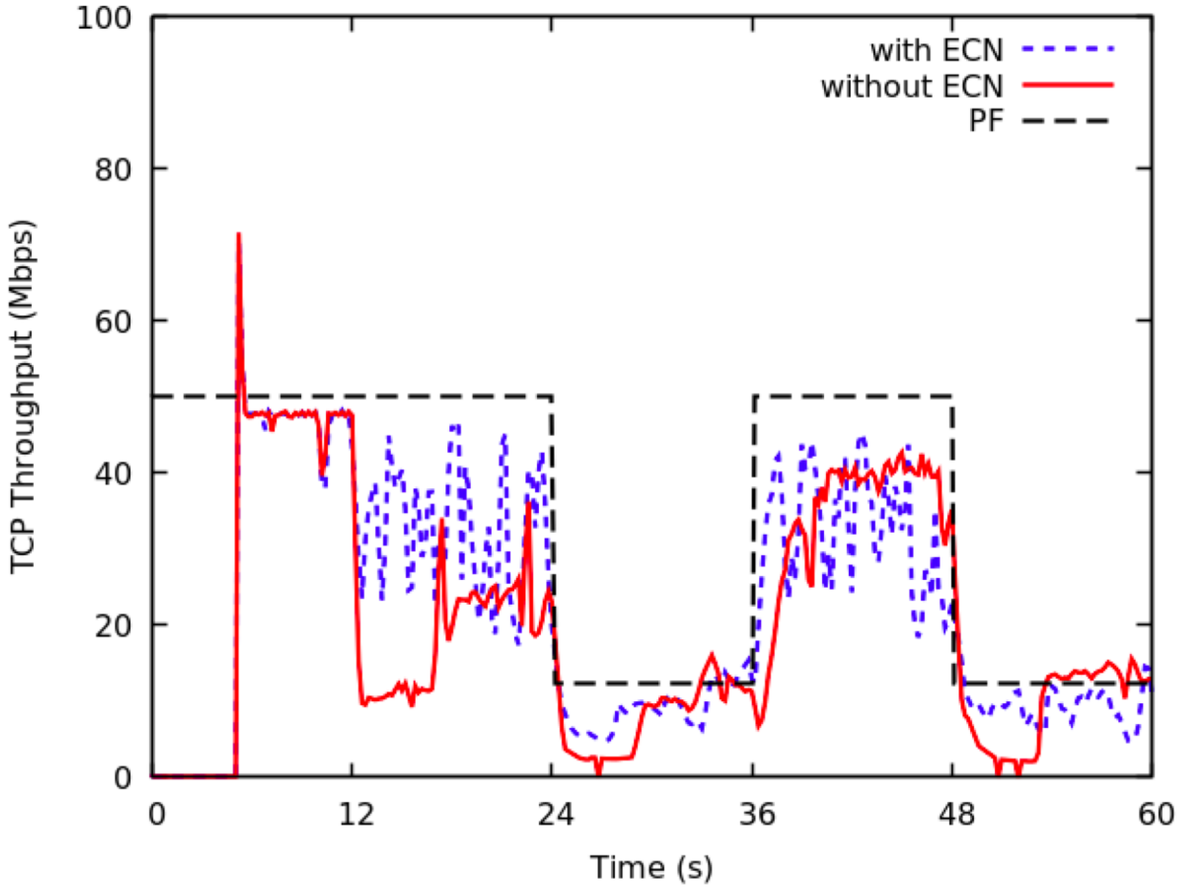
Router G1



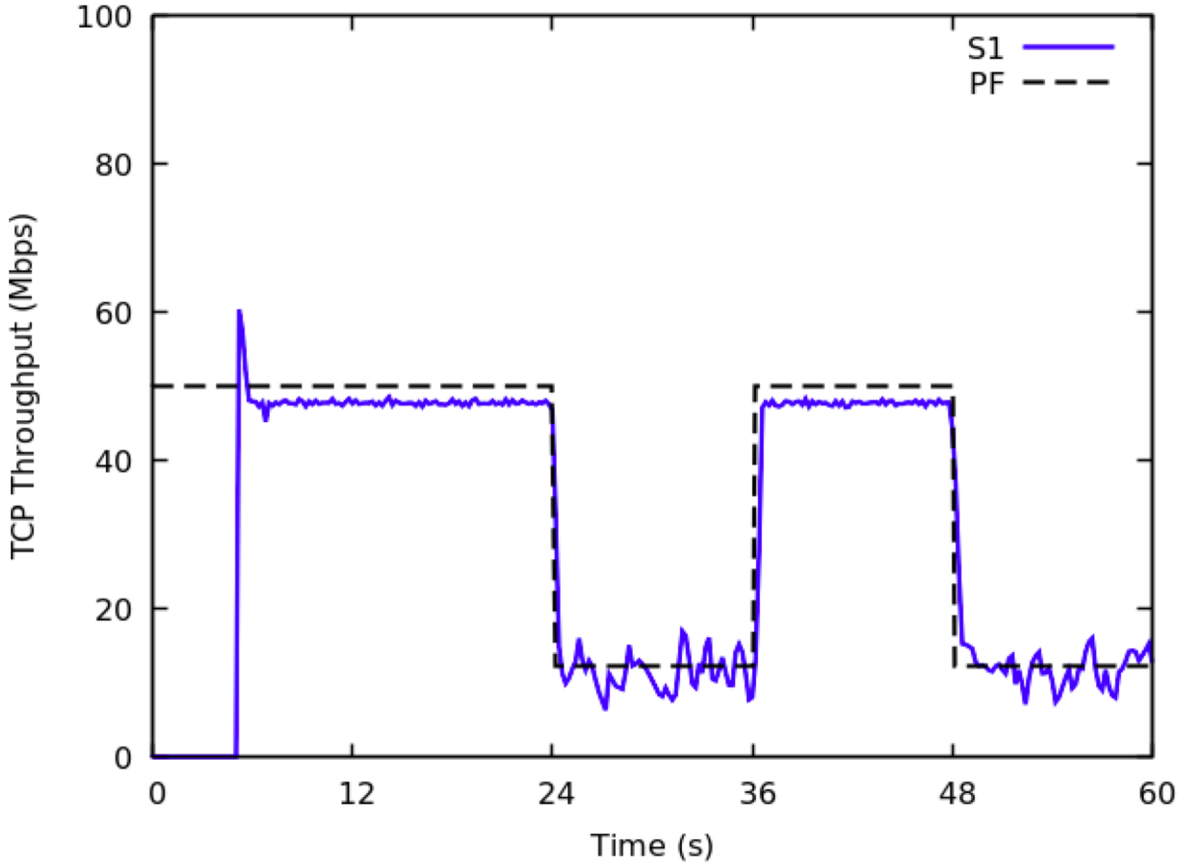
Router G3

- With default CUBIC based on packet loss – Queues build to capacity (1000 pkts)
  - Queue occupancy is based on the buffer capacity.

Throughput: Flow S1 (long lasting TCP flow)



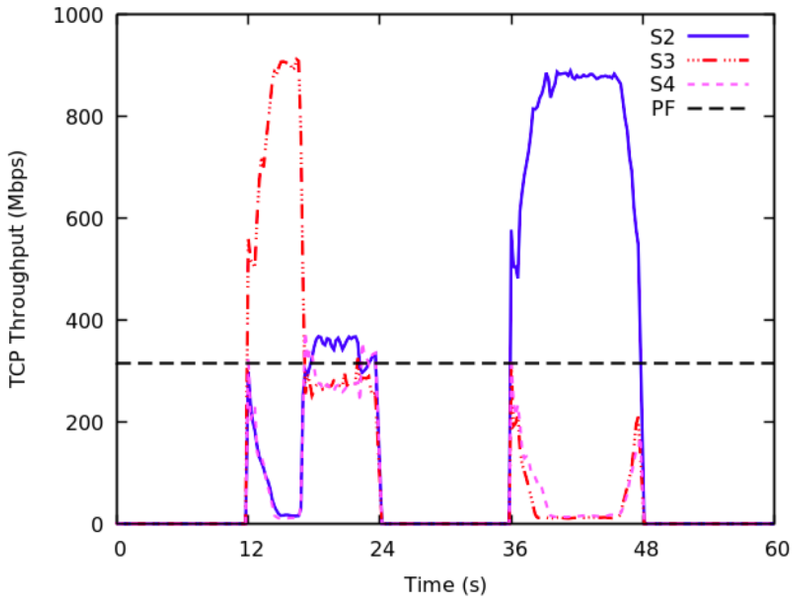
BBRv2 with and without ECN



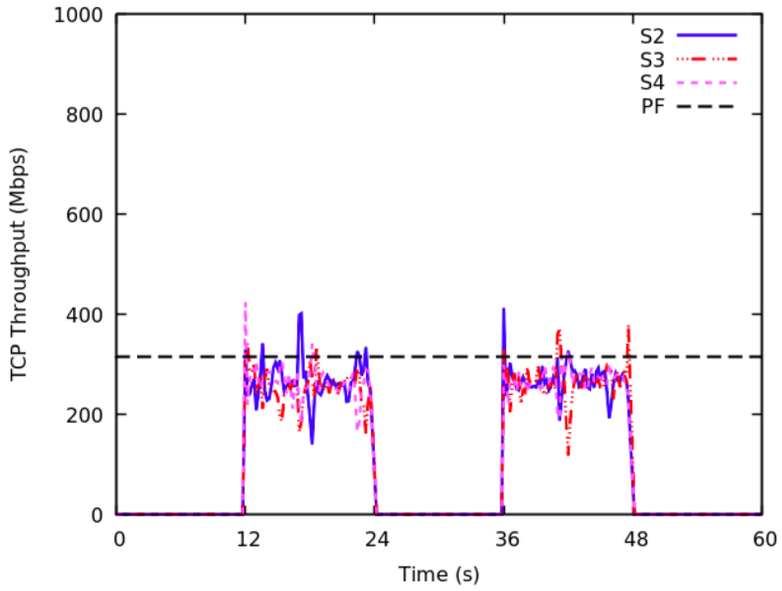
CUBIC with PI+ECN

- BBRv2 without ECN or with ECN : long lasting flow (S1) impacted by congestion in DCN, even though it is bottlenecked elsewhere
- CUBIC w/ PI+ECN has correct throughput ➔ matches proportional fair share

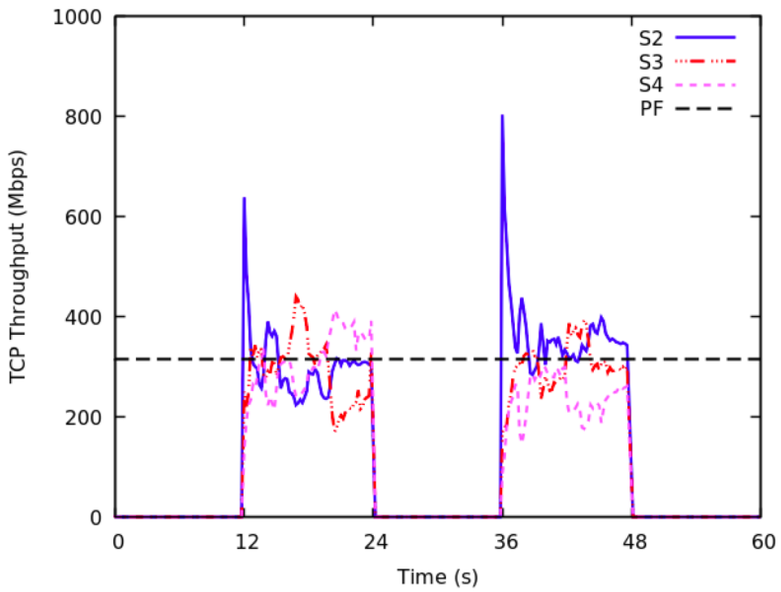
Throughput: Flow S2-S4 (short lived DC flows)



BBRv2 without ECN



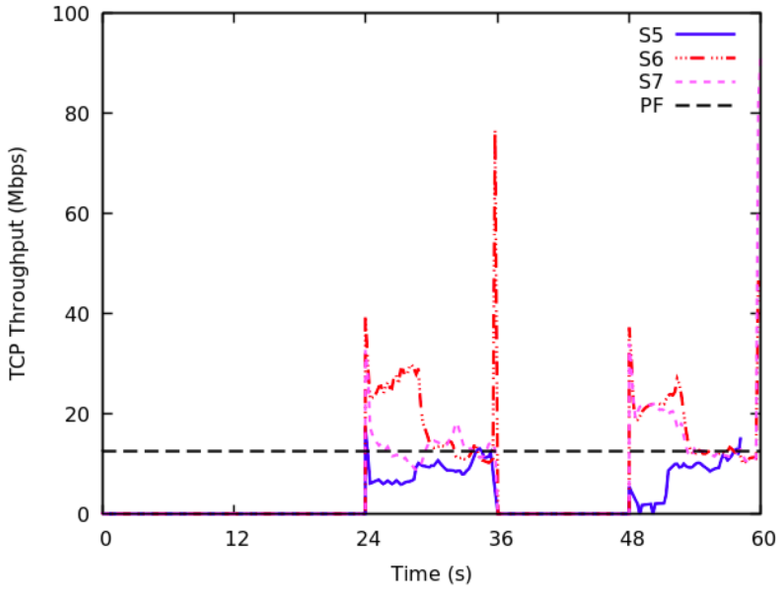
BBRv2 with ECN



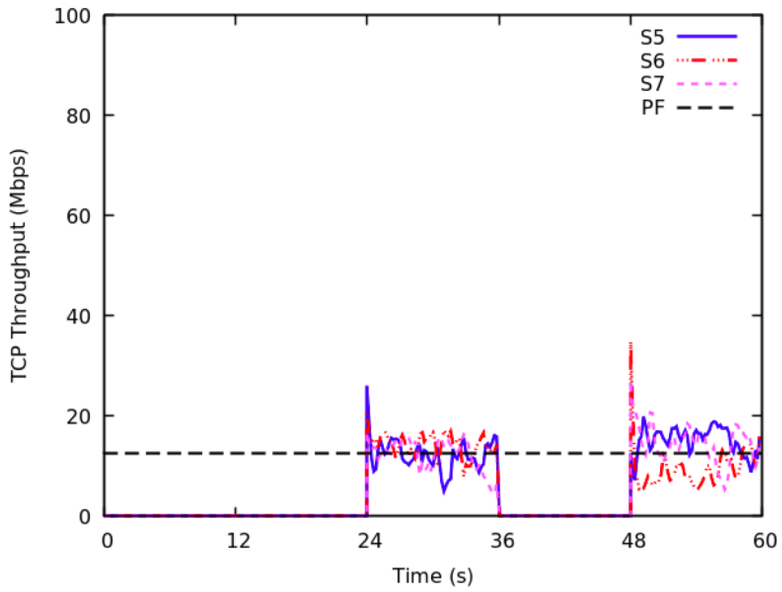
CUBIC with PI+ECN

- BBRv2 without ECN – S2-S4: throughput of different flows settle at different levels
- BBRv2 with ECN much better behaved tput ➔ matches proportional fair share
- CUBIC with PI+ECN also reasonable; over-correction due to timer implementation constraints in Linux

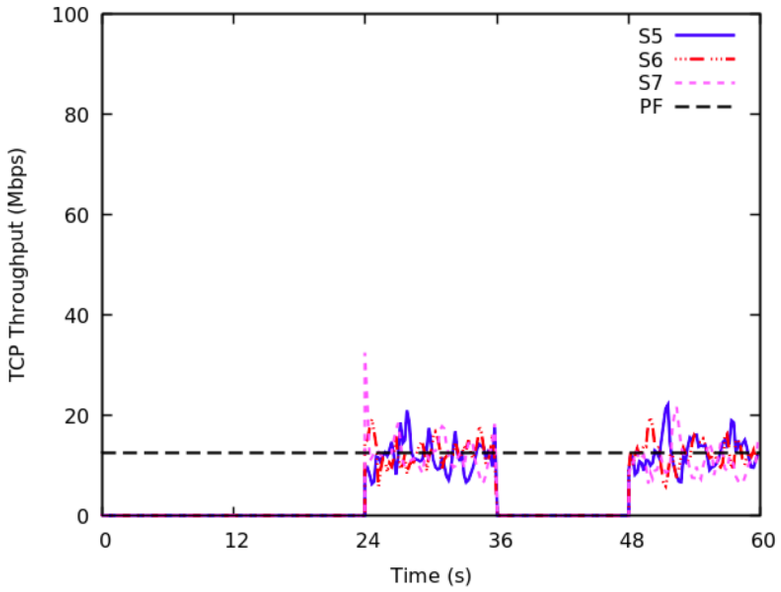
Throughput: Flow S5-S7 (short lived WAN flows)



BBRv2 without ECN



BBRv2 with ECN



CUBIC with PI+ECN

- BBRv2 without ECN – S5-S7: throughput of different flows settle at different levels
- BBRv2 with ECN much better behaved tput ➔ matches proportional fair share
- CUBIC with PI+ECN also reasonable; similar for ECN_t and ECN_a

Summary

- Implicit congestion feedback was easier to deploy
  - But explicit feedback is becoming almost as easy - ubiquitous capability in routers (hardware)
- Implicit feedback has fundamental limitations in dealing with ambiguity
  - Algorithmic or parameter tuning is unlikely to overcome
- With a good control algorithm and protocol, we can decouple the issues of buffer occupancy from buffer sizing
  - Provide buffers to deal with bursts; without concerns of latency