Congestion Control Protocols: ECN or Delay or Both?

Vishal Misra
Columbia University (and Google*)
Presenting work of Monia Ghobadi, Jitu Padhye, KK Ramakrishnan, Mohit Tahiliani and Yibo Zhu

* This work has not be done at Google and does not represent Google's views

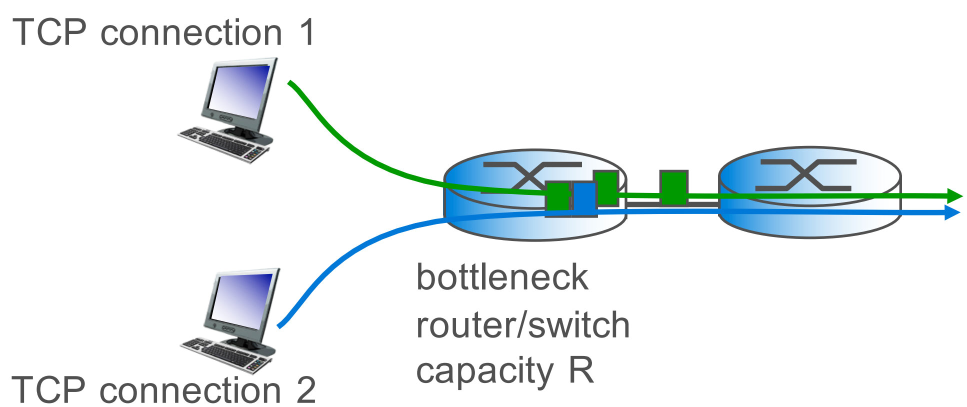
Talk overview

- Distributed Congestion Control: Goals and Approaches
- Background: Analysis of Delay vs ECN in context of RDMA*
  - Impossibility result related to delay based protocols
- Datacenter simulations for DCTCP and TCP (ECN based) and TCP BBR (delay based)
- Internet simulations for TCP with PI+ECN and TCP BBR
- Combining ECN and Delay: Virtual ECN

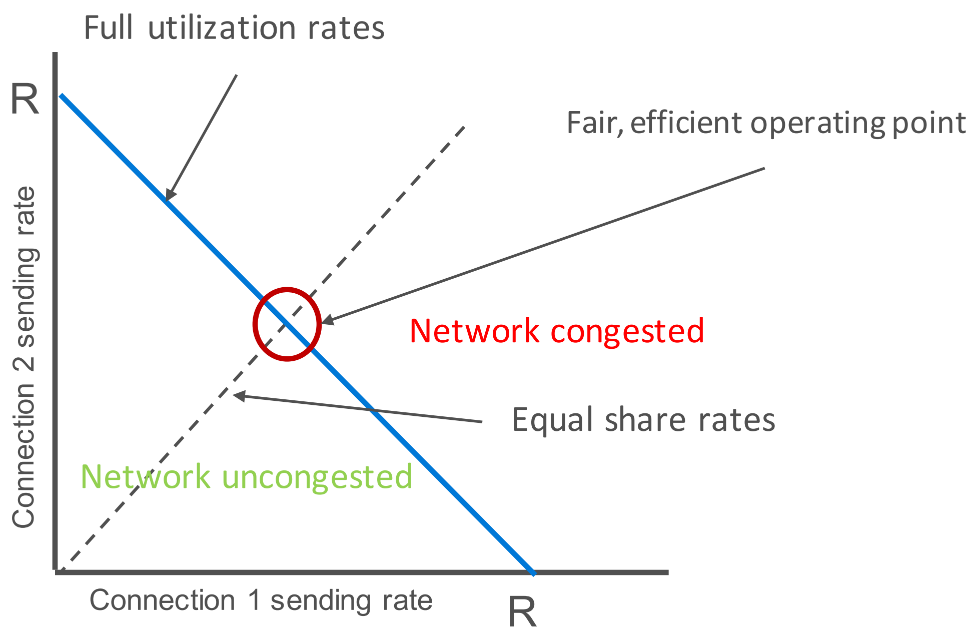
*Yibo Zhu, Monia Ghobadi, Vishal Misra and Jitendra Padhye, **ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY**, *Proceedings of 2016 ACM Conference on Emerging network experiment and technology (CoNEXT 2016)*, December, 2016.

Distributed Congestion Control

- *fairness goal:* if N TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/N
- *scalability goal:* rate adaptation should happen without global knowledge



Simple Scenario: Two flows sharing a bottleneck

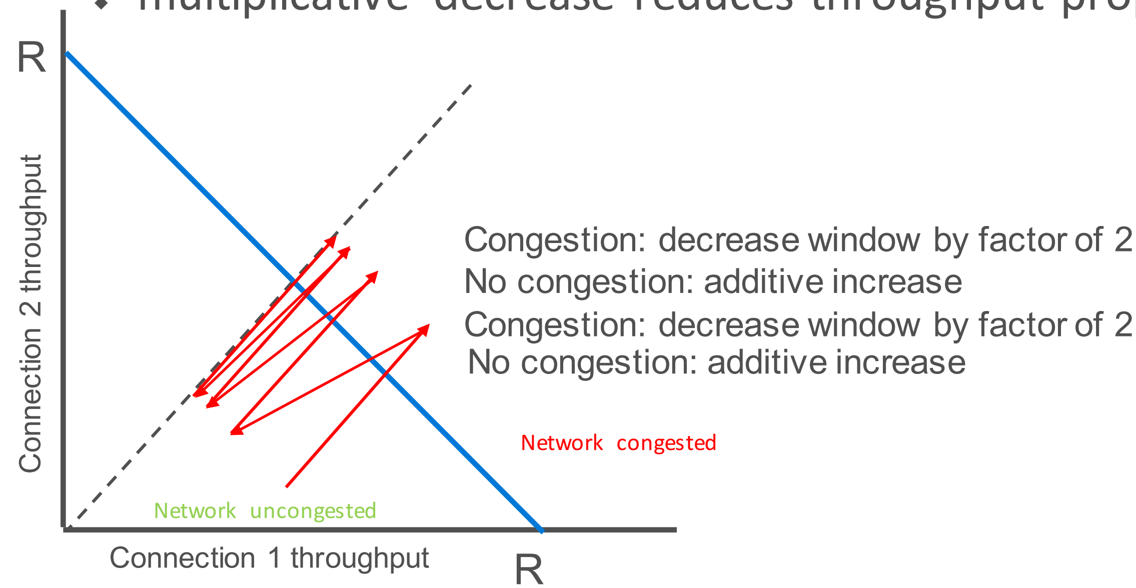


- Distributed Congestion control
  - Flows should detect whether the network is congested or not
  - Adjust their sending rates appropriately: increase if network is uncongested, decrease if it is congested
  - Full utilization rate operating points – how should flows react?
    - Come back to this question

AIMD Congestion Control

two competing sessions:

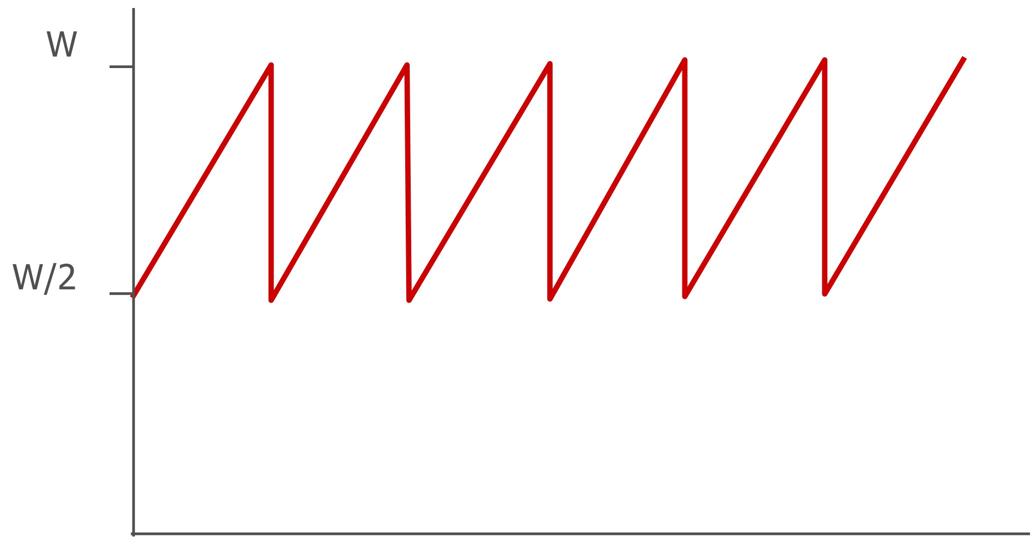
- additive increase gives slope of 1, as throughput increases
- multiplicative decrease reduces throughput proportionally



Chiu and Jain showed in full generality for N flows that AIMD converges to fair, efficient point

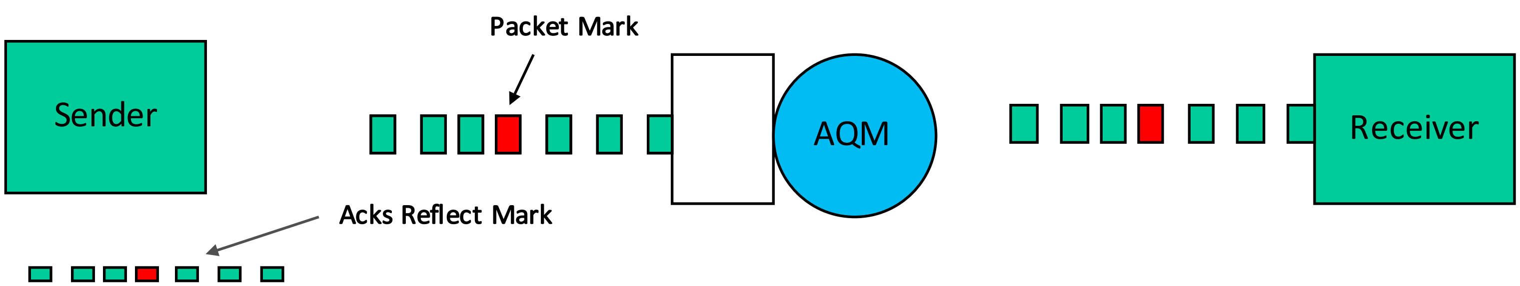
AIMD and TCP: loss as congestion indicator

- Variants of AIMD in TCP running since 1988 (Reno, NewReno, SACK..)
- Packet loss used as congestion indicator
- AI parameter: increase Window size by 1 every RTT, if no loss detected
- MD parameter: reduce Window size by ½ on detecting lost packet
- Leads to “saw tooth” behavior of TCP throughput
- Throughput of TCP ~ K/(RTT*sqrt(p)) where K is a constant, RTT is the round trip time and p is the loss probability



ECN: A “drop in” replacement for loss

- Packet drops result in wasted capacity
- RTT*sqrt(p) relationship in throughput means for same fixed throughput (e.g. R/N sharing of some link), reducing RTT leads to increasing loss
- Instead, use routers/switches in the network to mark a bit to indicate congestion
- Requires “Active Queue Management” (AQM) on switches/routers
- Came out of DECbit work in the mid-80s (Ramakrishnan and Jain)
- Competing with DECbit was a delay based protocol with unpredictable and unfair behavior
  - (we will show why)



TCP Vegas: Delay as Congestion Indicator

- Uses measured delay as running indicator of congestion in network
- Uses Additive Increase Additive Decrease to adjust sending rates
- Able to successfully reduce the number of losses, however shown to be unfair with unpredictable performance
  - (we will show why)

Analyzing Delay vs ECN for RDMA

DCQCN and TIMELY: Congestion Control for ROCEv2

DCQCN (Microsoft)

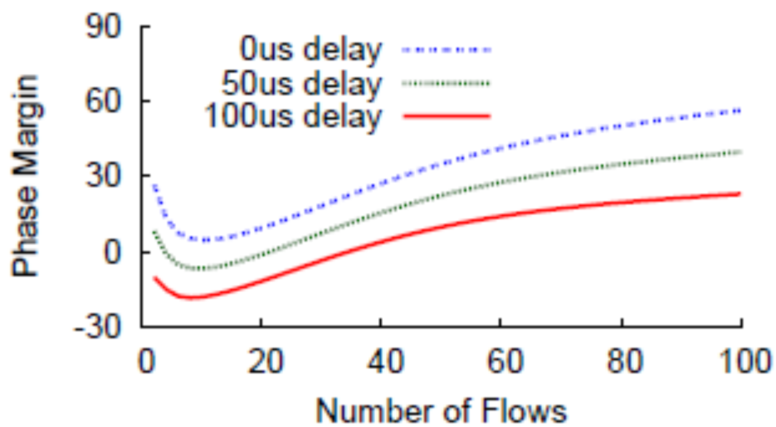
- Switch marks packets on detecting congestion (ECN)
- Receiver reflects marked packets via ACKs
- Sender adjusts rate using DCQCN algorithm
- Ongoing deployment in Microsoft Azure

TIMELY (Google)

- Switch plays no role (FIFO queue assumed)
- Receiver sends ACKs (once per burst)
- Sender estimates Delay, and responds to derivative.
- Variant deployed in Google*

DCQCN

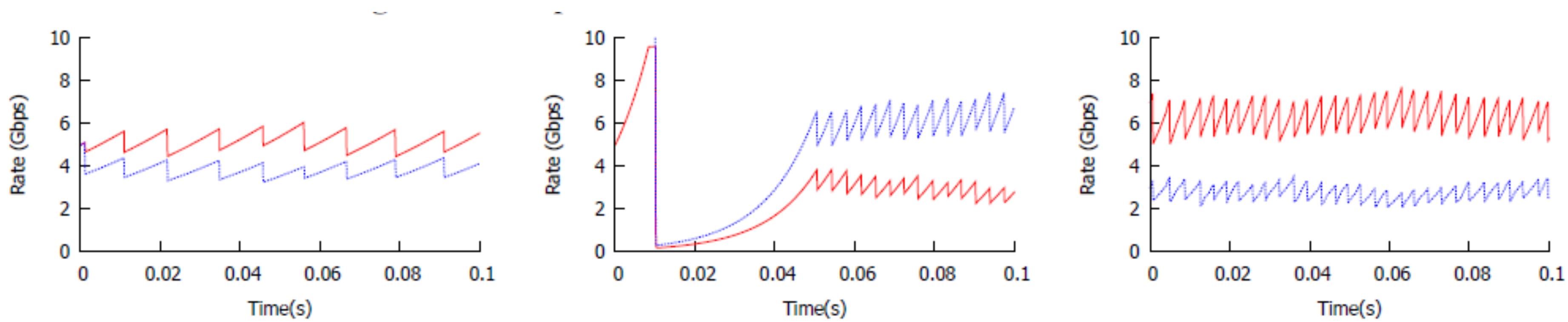
- DCQCN has a unique fixed point
- At the fixed point, all flows share the bottleneck equally
- Convergence is fairly rapid
- Relationship between stability and number of flows is non-monotonic



We don't have an intuitive explanation

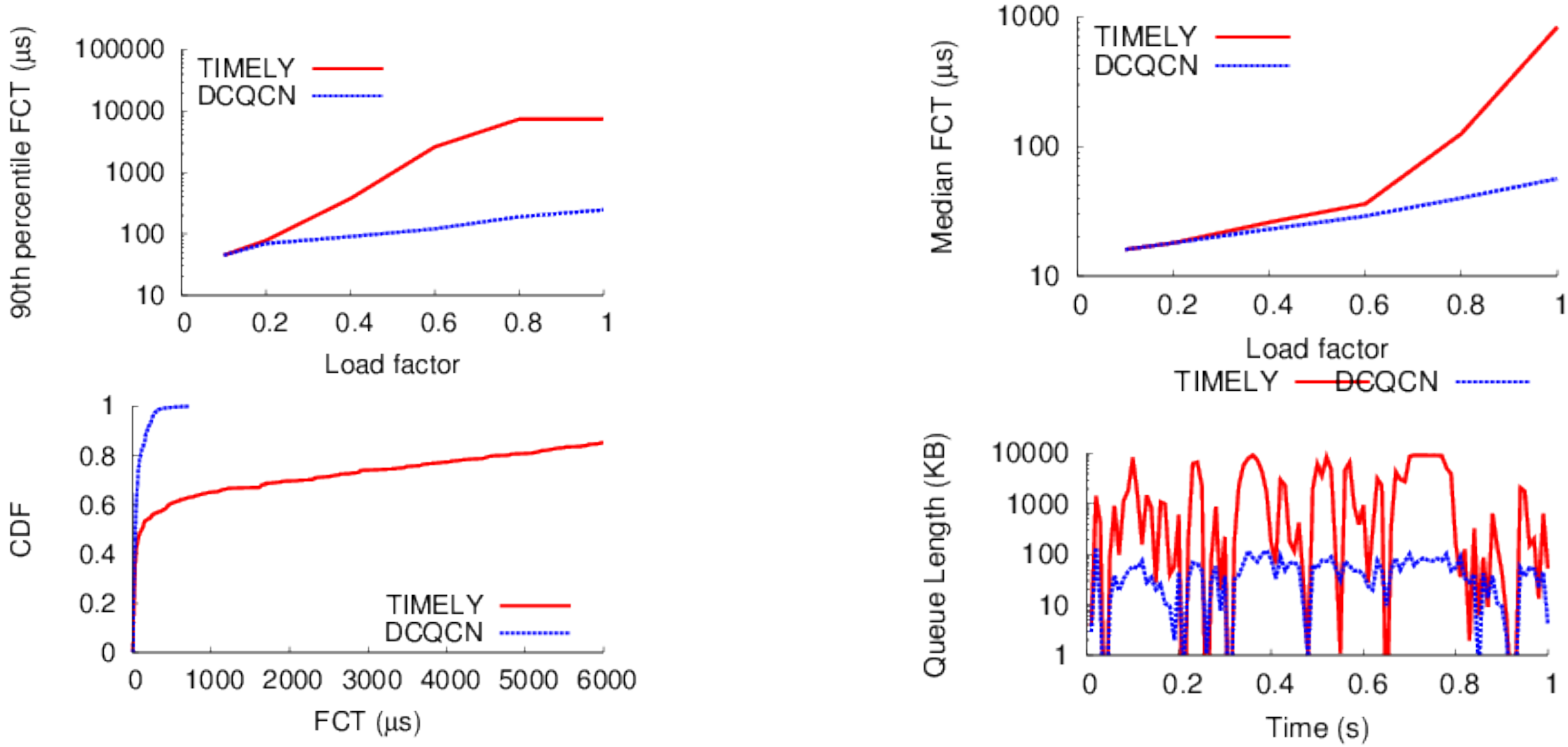
TIMELY

- Timely has no fixed point
  - changes rate in response to changes in latency (derivative)
  - Can stabilize at any point where sum of rates = bottleneck bandwidth



(a) Both flows start at time 0 at 5Gbps (b) Both start at 5Gbps, one starts 10ms late (c) One starts at 7Gbps, the other at 3Gbps

“Web Search” workload experiments



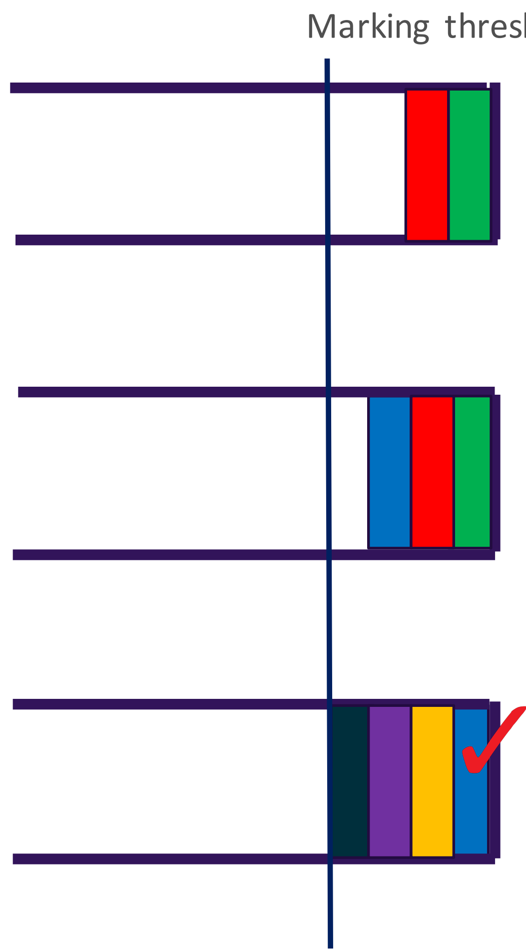
Why is TIMELY performing poorly?

- Reliance on delay differential
  - Can be fixed by making rate changes in response to absolute delay
- Feedback is delayed as queue builds up
- Can have fixed queue or fairness – but not both!
- ECN marking is resistant to feedback jitter

Why is TIMELY performing poorly?

- Reliance on delay differential
  - Can be fixed by making rate changes in response to absolute delay
- Feedback is delayed as queue builds up
- Can have fixed queue or fairness – but not both!
- ECN marking is resistant to feedback jitter

What happens with ECN



T0, Q = 2 — Blue packet is about to arrive

T1, Q = 3 — Blue packet arrival complete

T2, Q = 4 — Blue packet ready to depart
… and is marked, reflecting state of queue at T2

What happens with delay

T0, Q = 2

Blue packet is about to arrive

T1, Q = 3

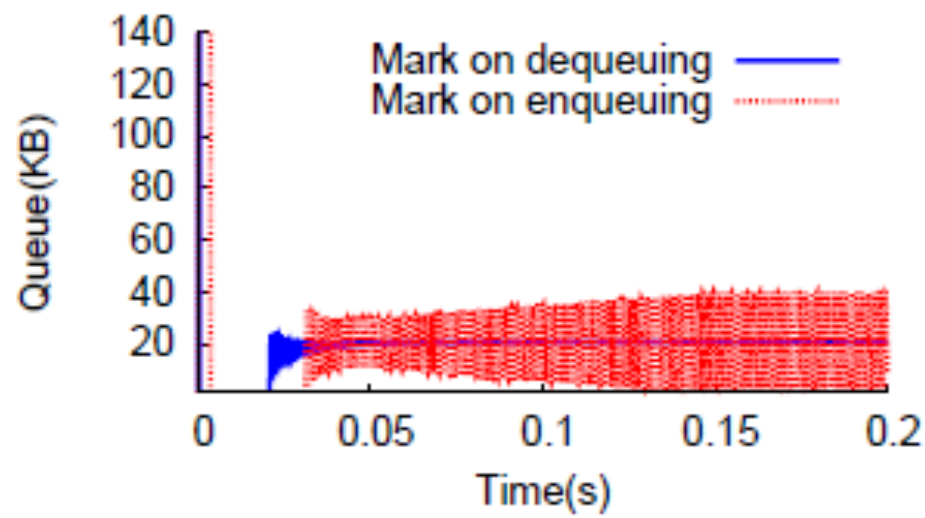
Blue packet arrival complete.
… timer starts

T2, Q = 4

Blue packet ready to depart
… and reflects state of queue at T0

In other words

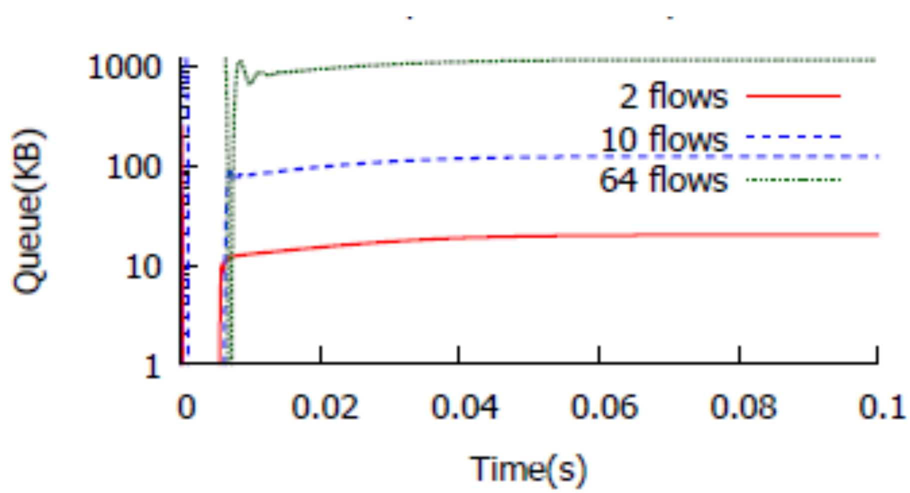
- Delay inherently reports "stale" information
- The staleness is affected by queue length!
  - Longer queue ➔ more stale feedback
- This can lead to instability



Why is TIMELY performing poorly?

- Reliance on delay differential
  - Can be fixed by making rate changes in response to absolute delay
- Feedback is delayed as queue builds up
- Can have fixed queue or fairness – but not both!
- ECN marking is resistant to feedback jitter

A problem with DCQCN (also TIMELY)



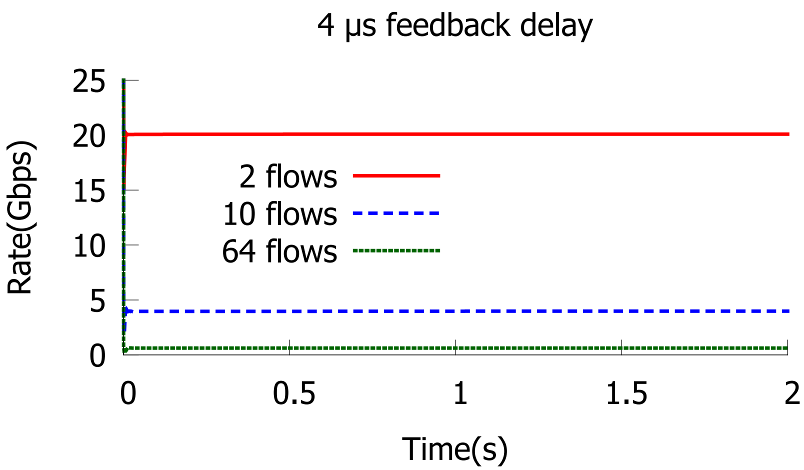
Bottleneck queue is a function of number of flows.

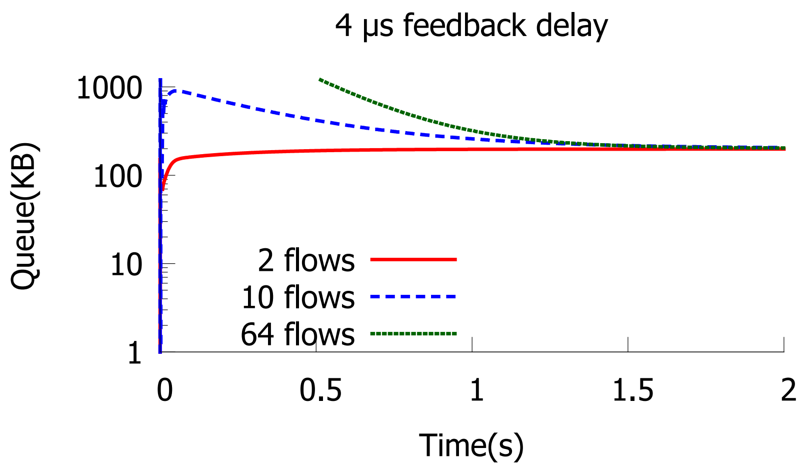
PI Controller

- Proportional Integral controller based on control theory
- "Integral" control drives an error signal down to 0
  - Set the error signal to (queue length – desired queue occupancy)
  - Guarantees the queue length to a fixed quantity
- Introduced for Active Queue Management (AQM) in network routers*
- Cisco's variant of PI (PIE) part of DOCSIS 3.1 standard to control bufferbloat in consumer cable modems

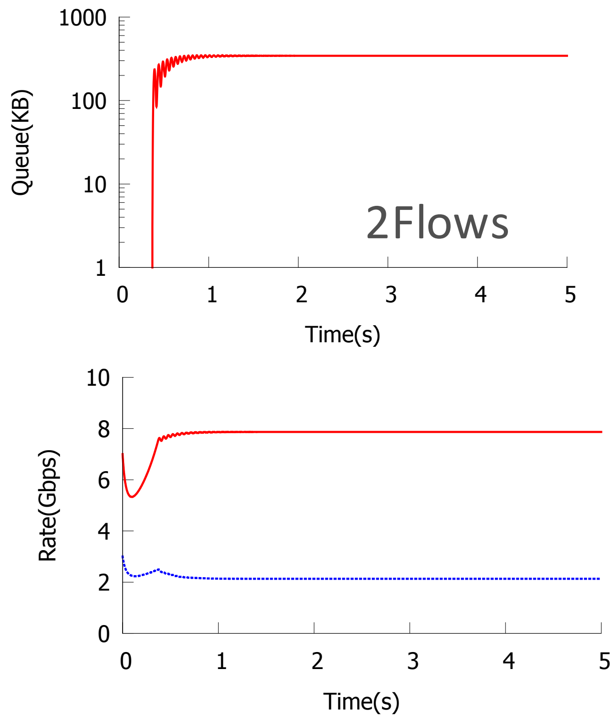
*C. V. Hollot, Vishal Misra, Don Towsley and Wei-Bo Gong, **On Designing Improved Controllers for AQM Routers Supporting TCP Flows**, *Proceedings of IEEE Infocom*, April, 2001.

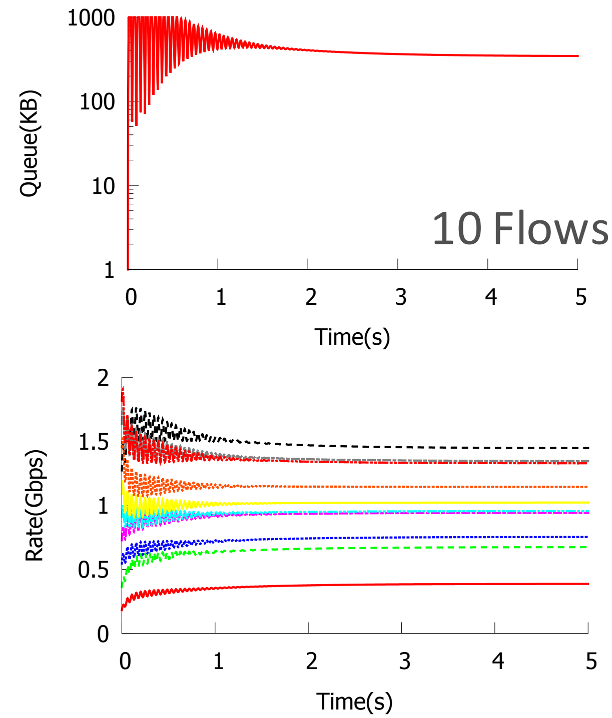
PI control at switch with DCQCN





PI Controller with TIMELY: lose fairness





Fundamental tradeoff result

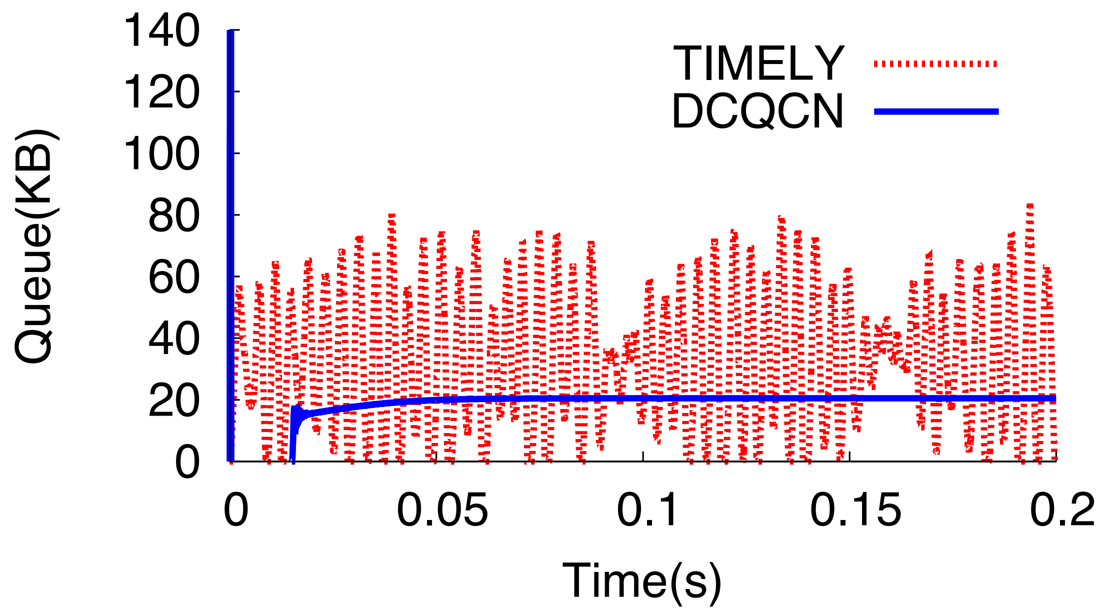
- THEOREM: (FAIRNESS/DELAY TRADEOFF). *For congestion control mechanisms that have steady state throughput of the kind* R = f(d,p)*, for some function* f*, delay* d *and feedback* p*, if the feedback is based on purely end to end delay measurements, you can either have fairness or a fixed delay, but 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 larger than number of equations -> infinite fixed points -> unfairness

Why is TIMELY performing poorly?

- Reliance on delay differential
  - Can be fixed by making rate changes in response to absolute delay
- Feedback is delayed as queue builds up
- Can have fixed queue or fairness – but not both!
- ECN marking is resistant to feedback jitter

Impact of reverse path delay

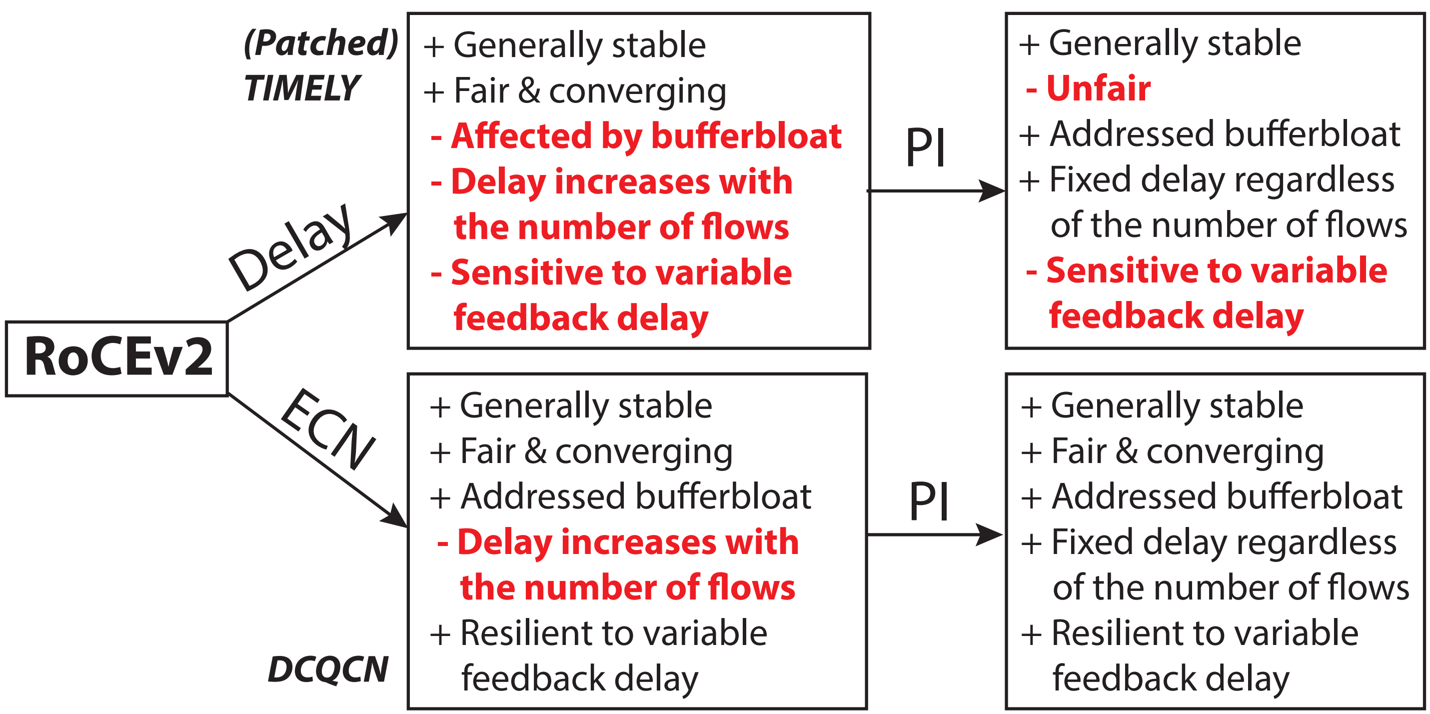


ECN is more resistant as feedback signal is only *delayed*

With Delay, the feedback signal is both delayed and *distorted*

Delay:ECN::AM:FM

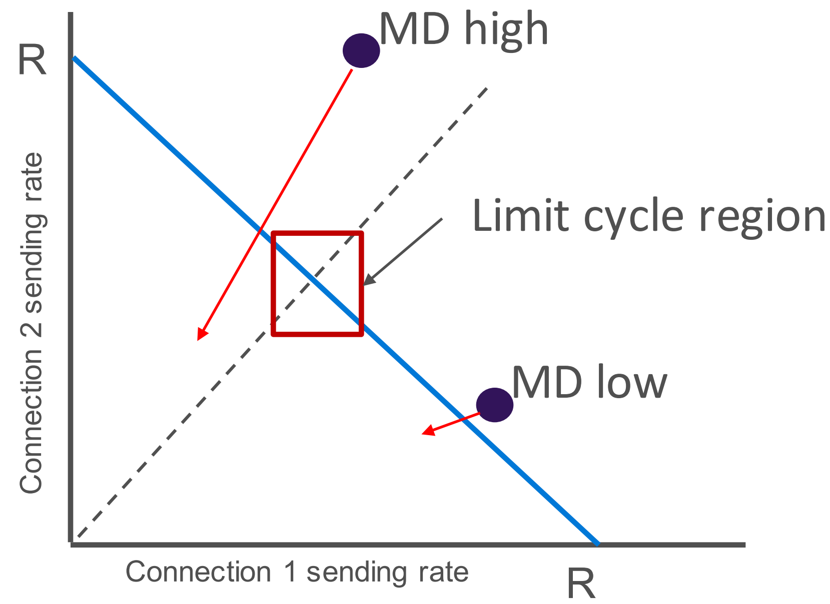
Conclusion: ECN appears better



Back to Delay vs ECN for TCP

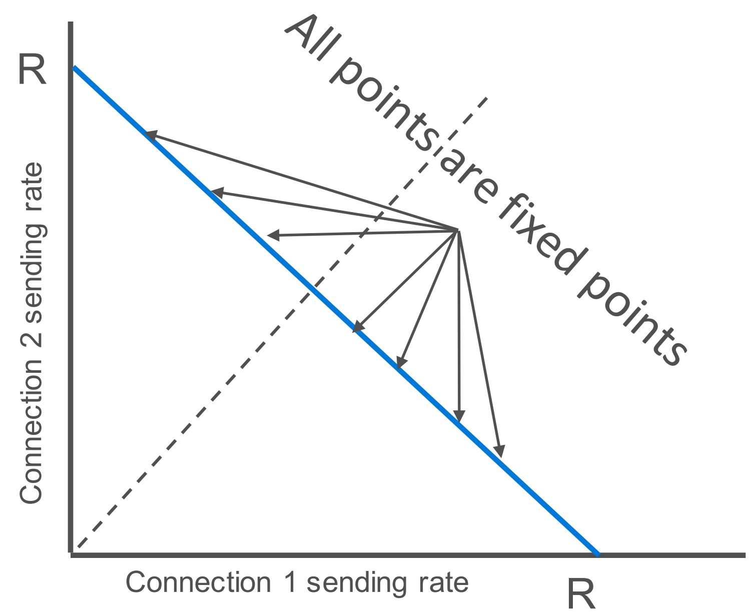
DCTCP: Data Center TCP

- TCP adapted for datacenter environments
- Uses ECN for congestion indication
  - Simple threshold based AQM: mark all packets when queue > threshold
  - Flows use fraction of marked packets as indicator of level of congestion
  - MD factor adjusted according to inferred level of congestion: results in smoother behavior
  - Doesn’t have a fixed point, but limit cycles around an “operating point”
  - MD adjustment considers distance from efficiency line, not fairness line: could lead to slow convergence to fairness



BBR: Delay based TCP for Internet

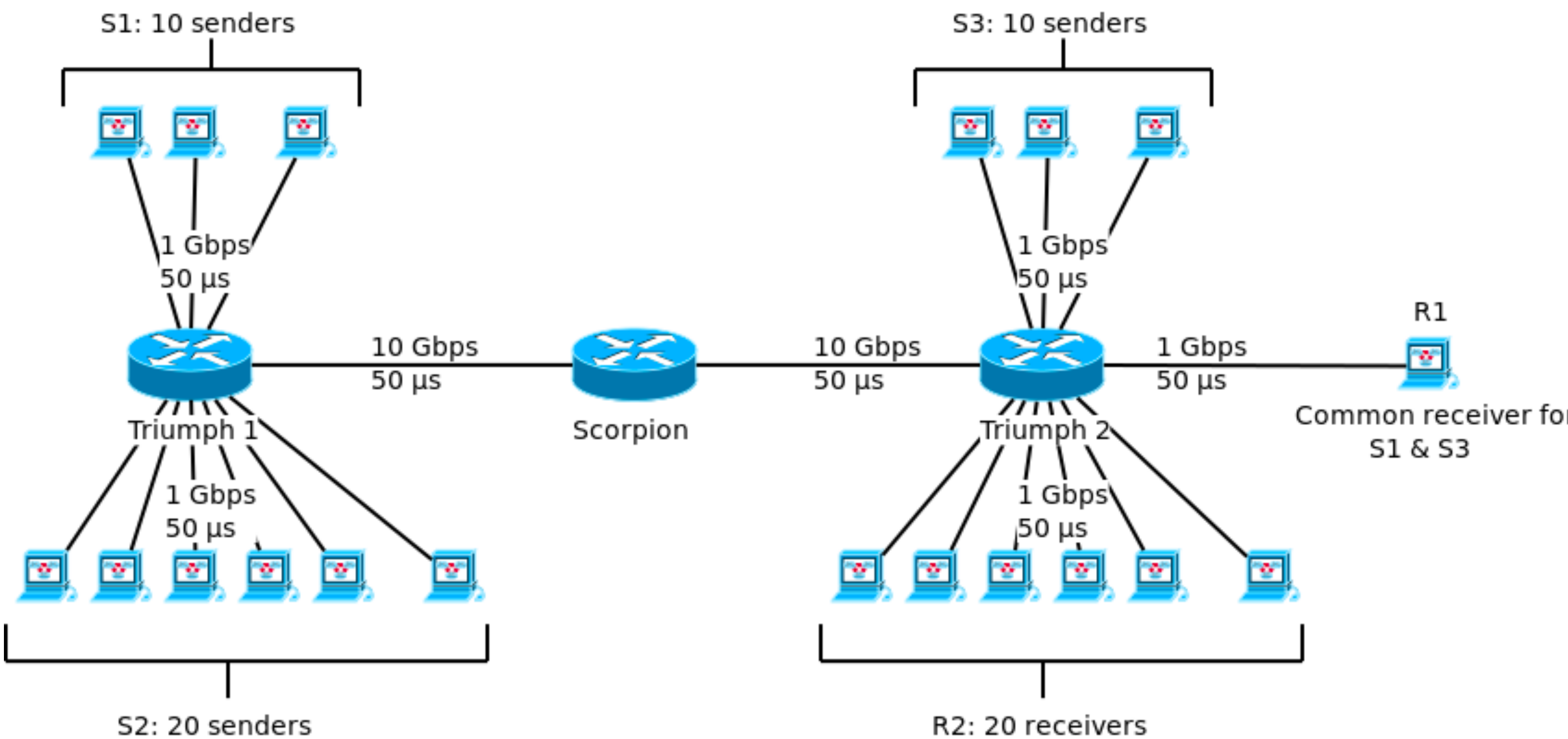
- Recent variant of TCP congestion control introduced by Google
- Like Vegas, uses delay measurements to estimate network state congested/uncongested
- Aims to get 0 queueing delay at the bottleneck router
- Has no unique “fixed point”: flows converge to a rate where they believe bottleneck queue = 0 -> infinite fixed points.
- Every 10 seconds, flows probe and “release capacity” allowing other flows to claim it
- Adjustment cannot be classified under any classical AIMD or AIAD or MIMD scheme
- Infinite fixed points -> widespread unfairness reported



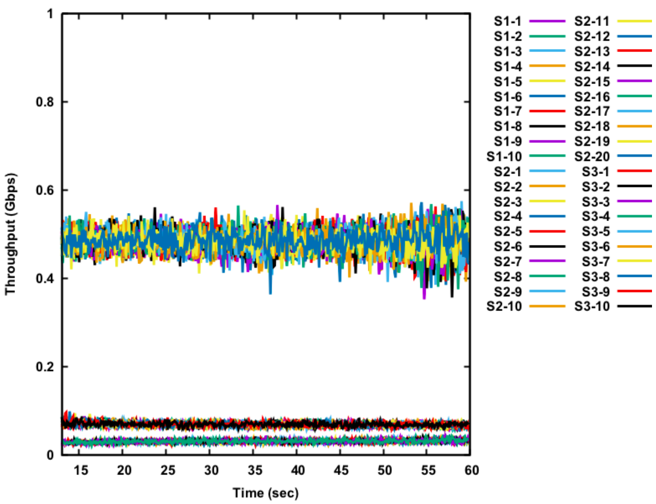
Data Center simulations

- Simulations performed in ns-3 for 3 variants of TCP
  - DCTCP (ECN based, targeted for data centers)
  - TCP NewReno with PI+ECN (regular TCP, multiplicative decrease adjusted to 7/8)
  - TCP BBR (end to end delay based TCP)
    - BBR is not designed for the data center, but is picked as a representative delay based TCP

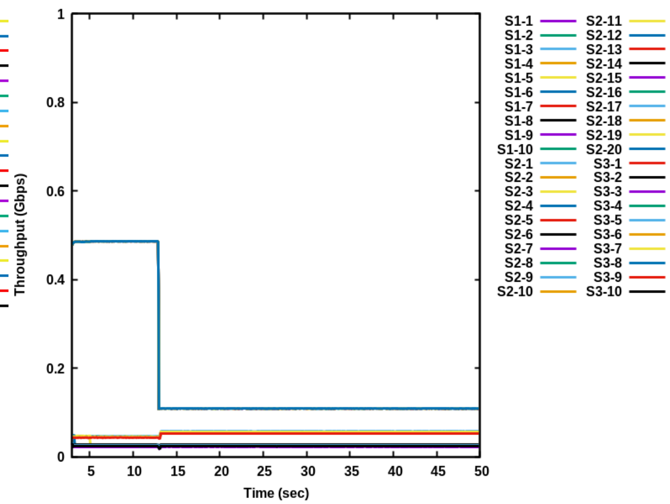
Simulation Topology (same as DCTCP paper)



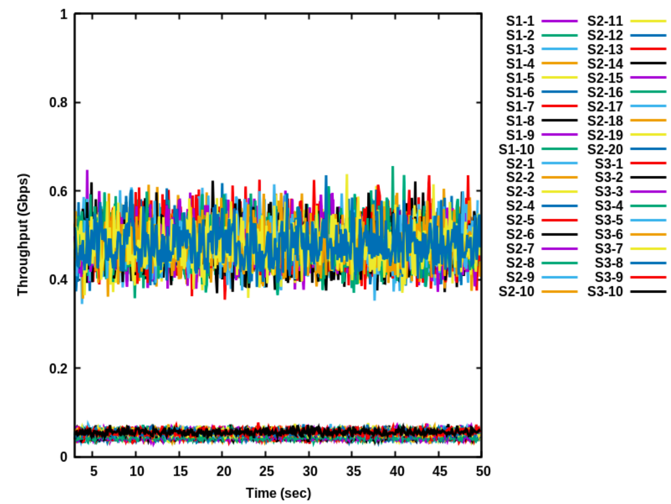
Flow Throughputs



DCTCP

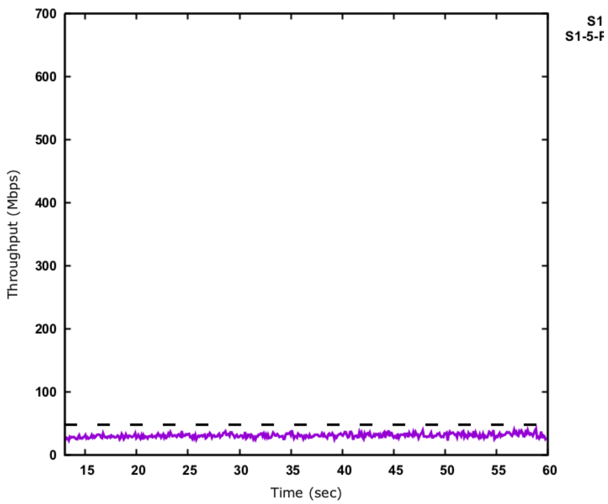


BBR

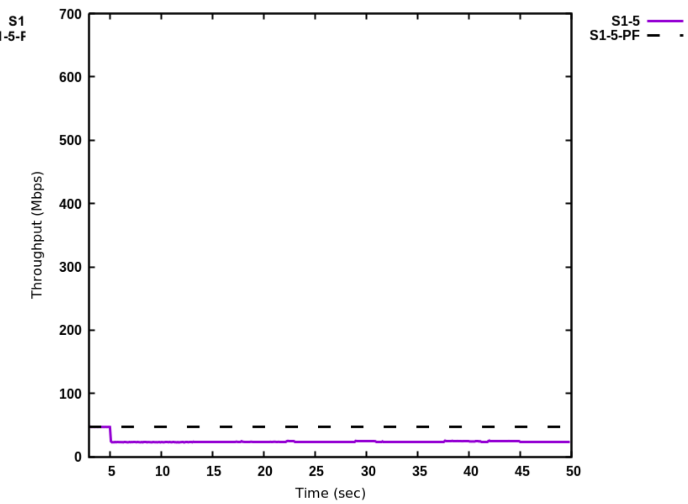


PI+ECN

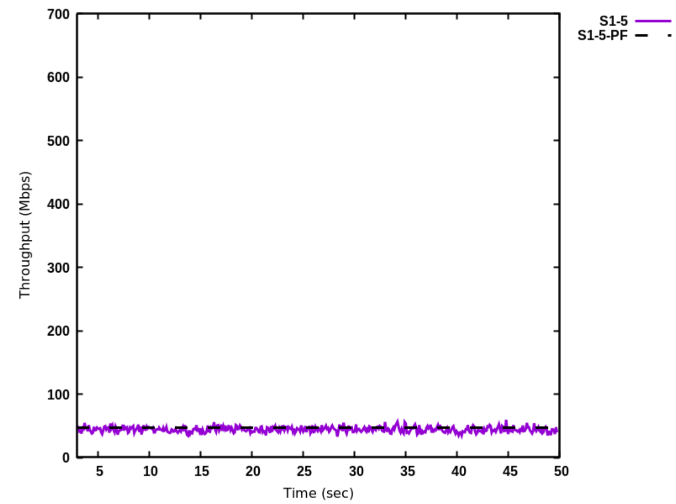
Fairness



DCTCP



BBR



PI+ECN

Aggregate Metrics

Utilization

	PI + ECN		BBR		DCTCP	
	Gbps	**%**	**Gbps**	**%**	**Gbps**	**%**
Triumph 1	10.00	100	4.09	40.9	10.00	100
Scorpion	10.00	100	4.09	40.9	10.00	100
Triumph 2	1.00	100	0.66	66.0	1.00	100

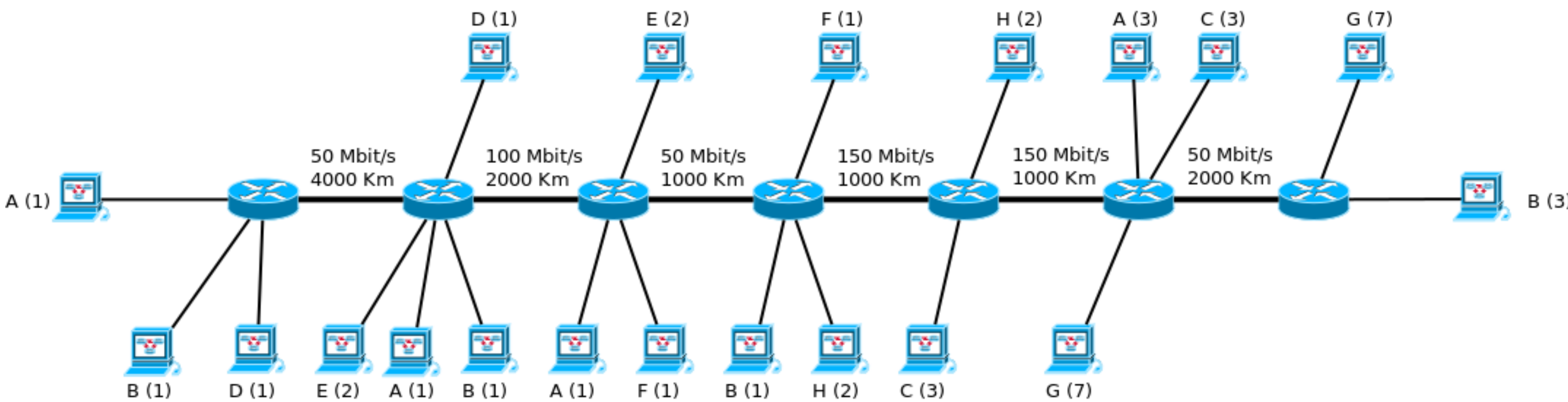
Drops/Marks

Switches	**PI + ECN**	**BBR**	**DCTCP**
Triumph 1	270095 marks	0	6480390 marks
Scorpion	0 marks	0	0 marks
Triumph 2	192474 marks	0	776279 marks

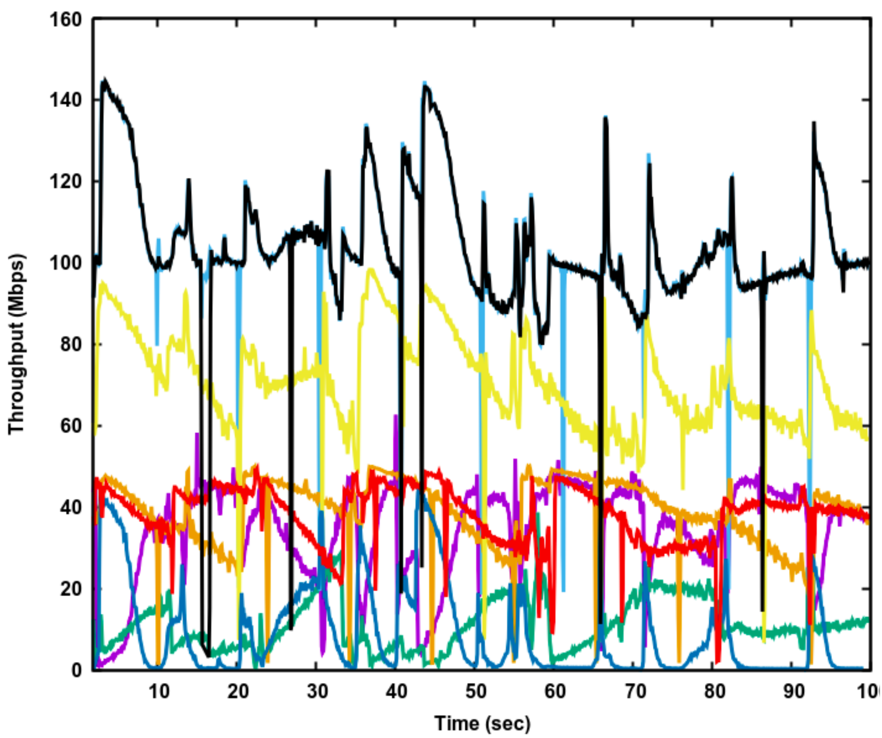
Internet simulations

- Simulations performed in ns-3 for 2 variants of TCP
  - TCP NewReno with PI+ECN
  - TCP BBR

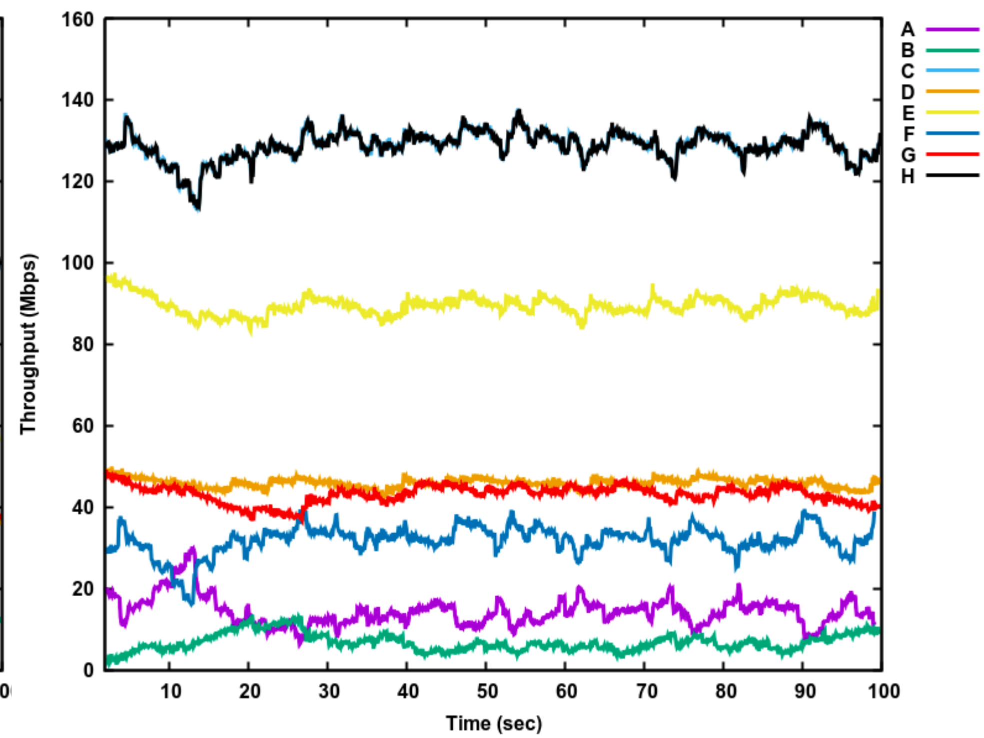
Simulation Topology (GFC-2)



Throughputs

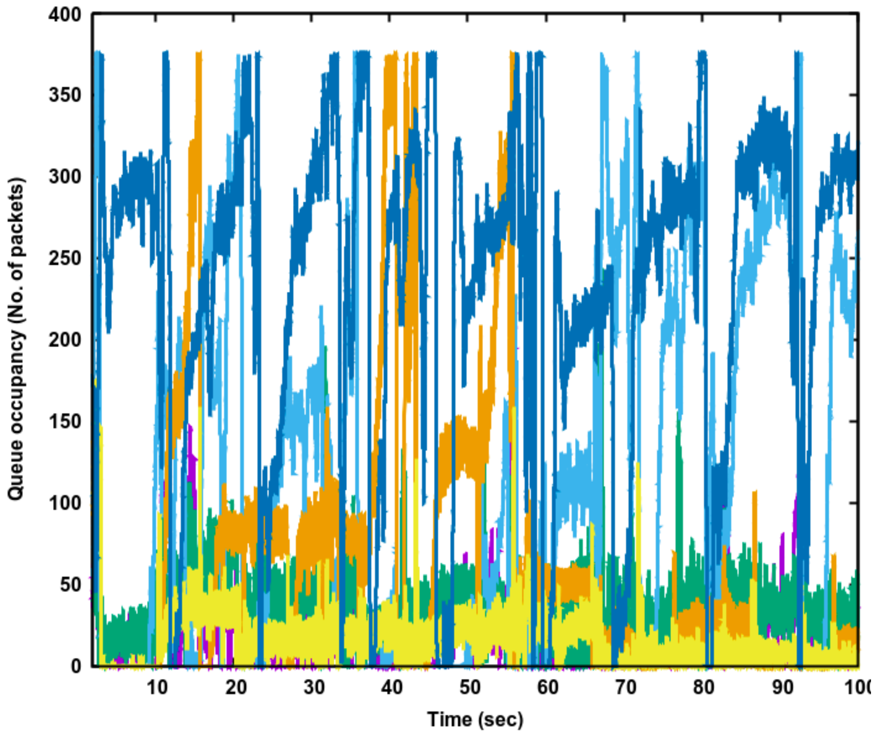


BBR

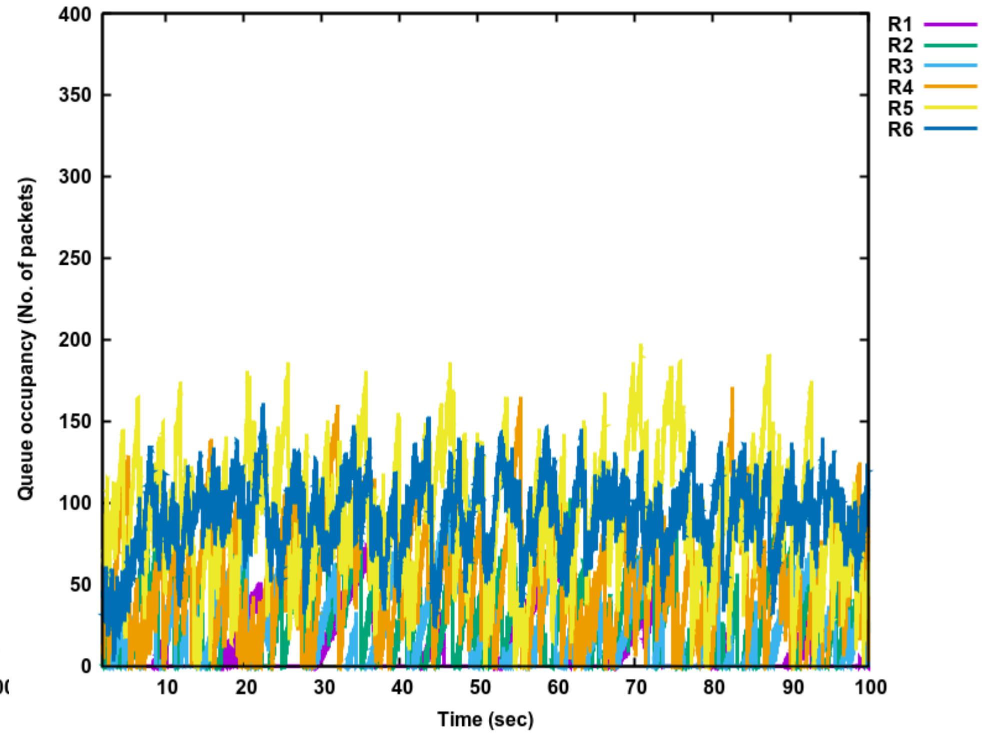


PI+ECN

Queue lengths

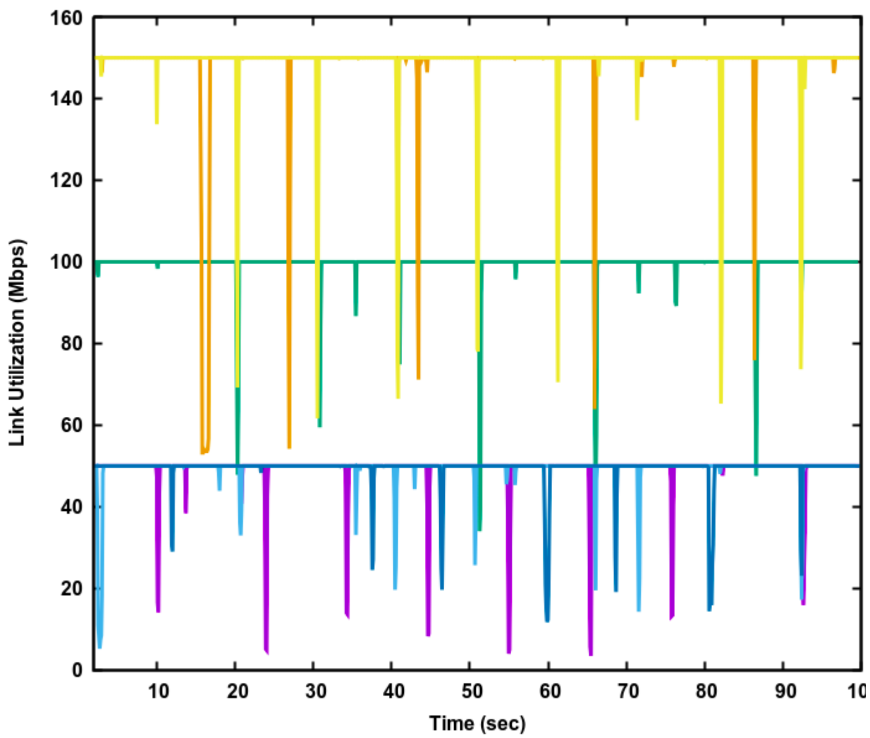


BBR

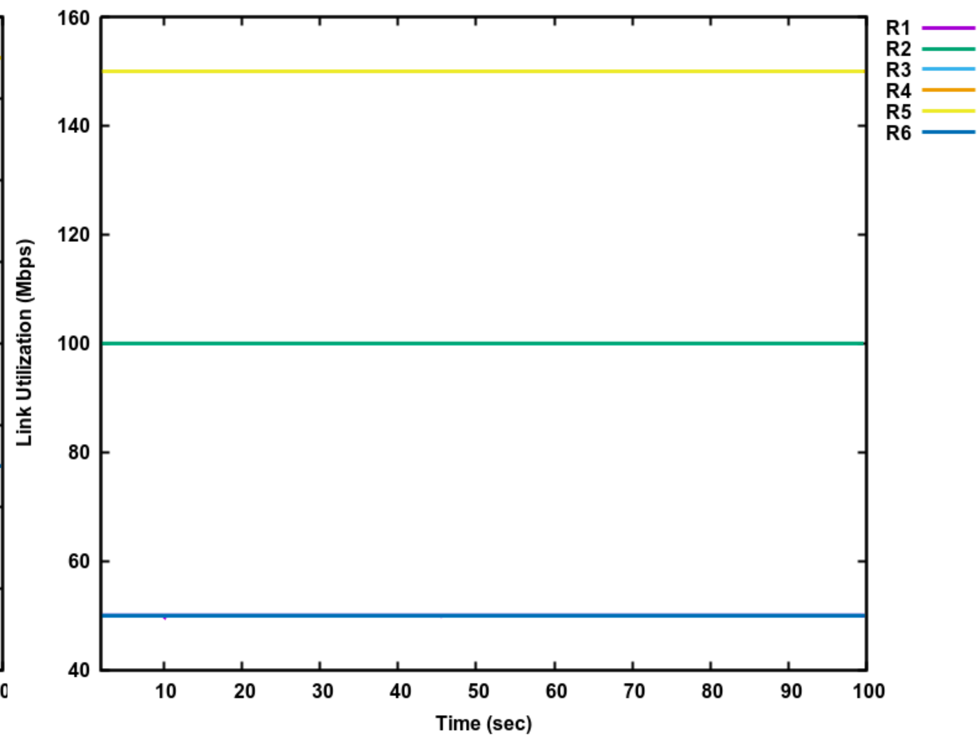


PI+ECN

Link Utilizations

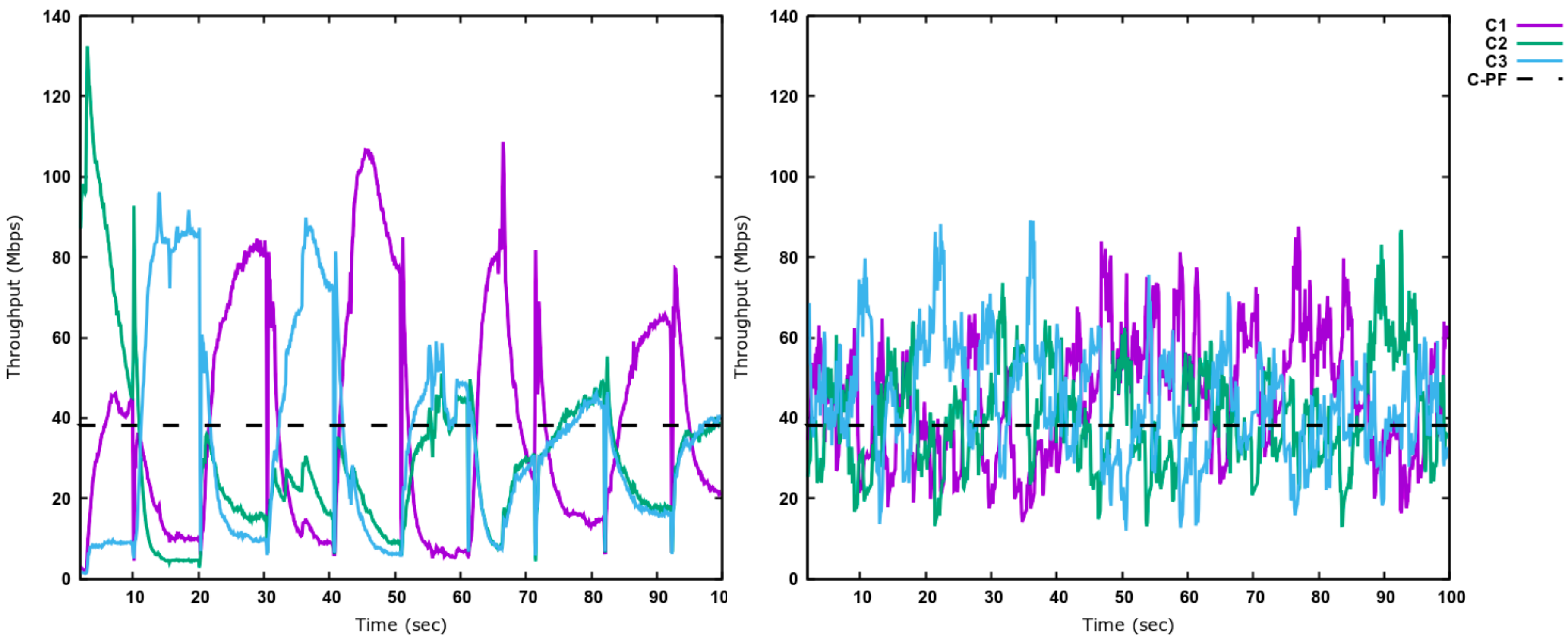


BBR



PI+ECN

Fairness



BBR

PI+ECN

Aggregate metrics

Utilization

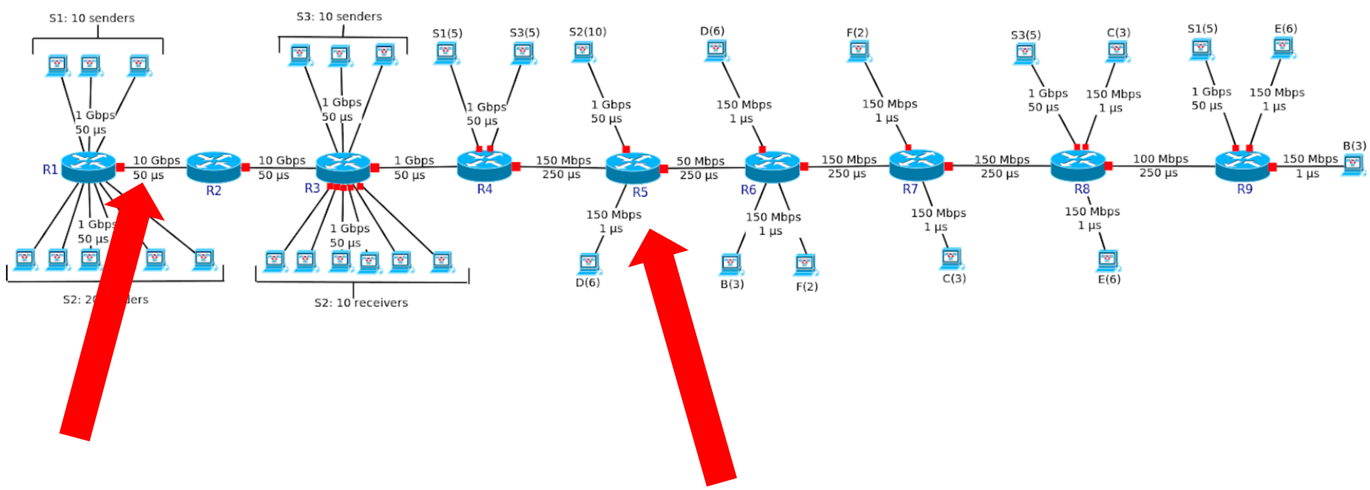
	PI + ECN		BBR	
	Mbps	**%**	**Mbps**	**%**
L1	50.00	100	49.14	98.28
L2	100.00	100	99.35	99.35
L3	50.00	100	49.31	98.62
L4	150.00	100	148.24	98.83
L5	150.00	100	148.76	99.17
L6	50.00	100	49.31	98.62

Drops/Marks

	NewReno + PI + ECN	BBR
R1	8 marks	0
R2	57 marks	0
R3	42 marks	636 drops
R4	96 marks	1623 drops
R5	182 marks	0
R6	343 marks	3417 drops

Wait there's more…

- End-to-End delay feedback cannot disambiguate congestion points



1 ms of queue buildup -> extreme congestion

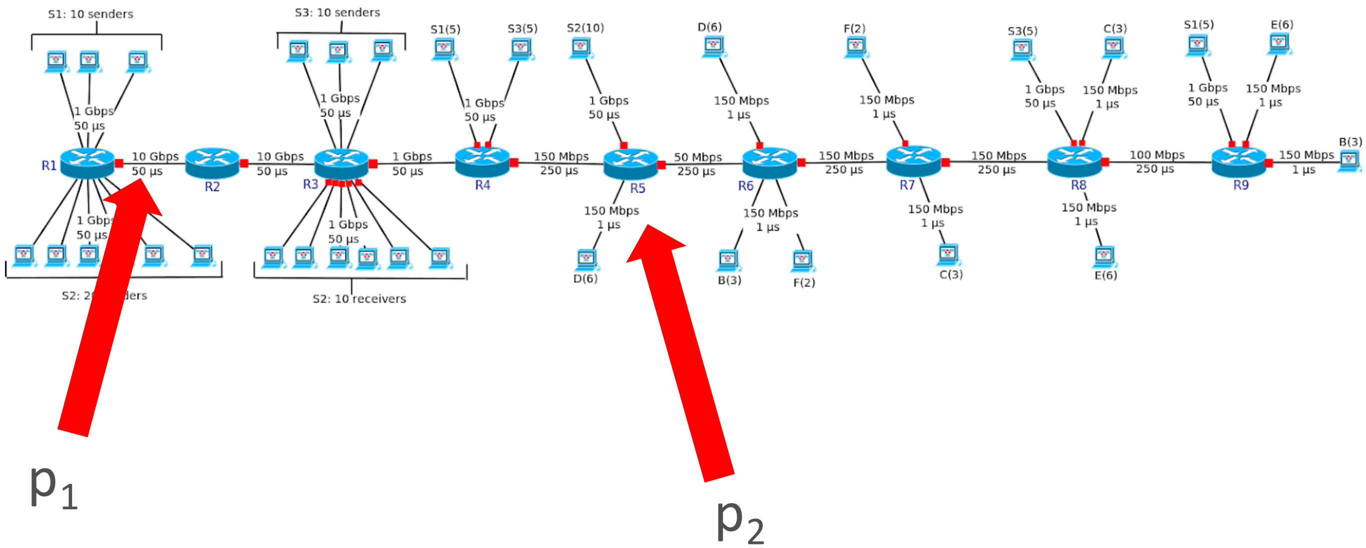
1 ms of queue buildup -> noise

- 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

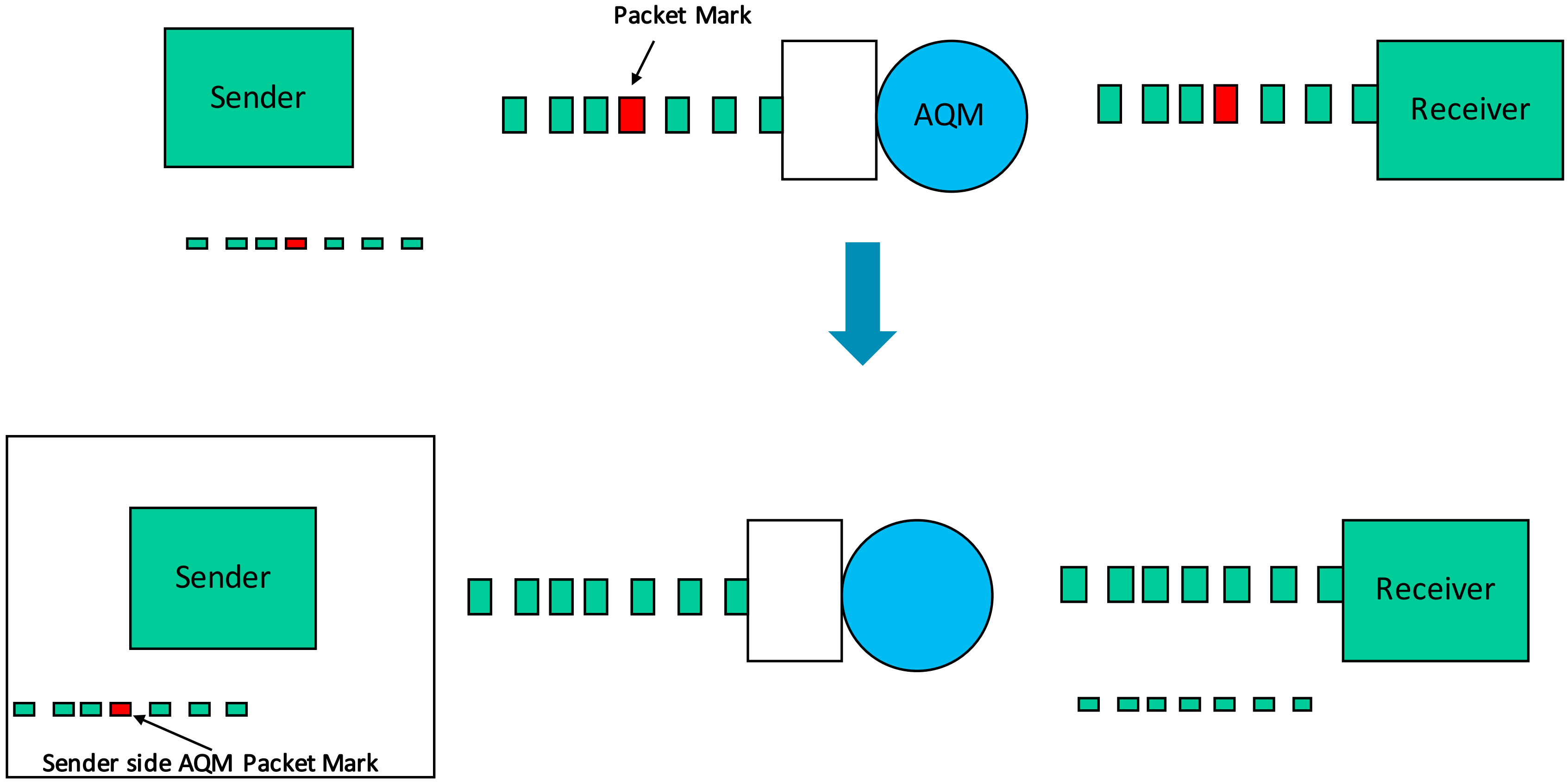
- Aggregate feedback $= 1-(1-p_1)*(1-p_2) = 1- 1+ p_1 + p_2 - p_1*p_2$

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

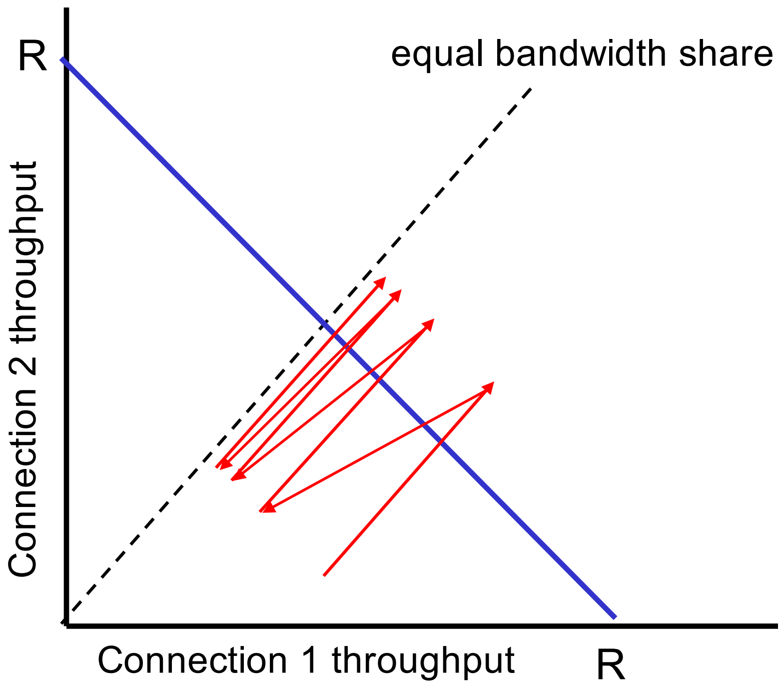


ECN AUR Delay
Can we combine the two?

Can we combine ECN and delay?



What about Fairness and the impossibility result?



AIMD to the rescue: probes all fixed points frequently and regardless of starting state, converges to the fair one

Results from PERT paper*

PERT does AIMD -> fair fixed points

Vegas does AIAD -> unfair fixed points

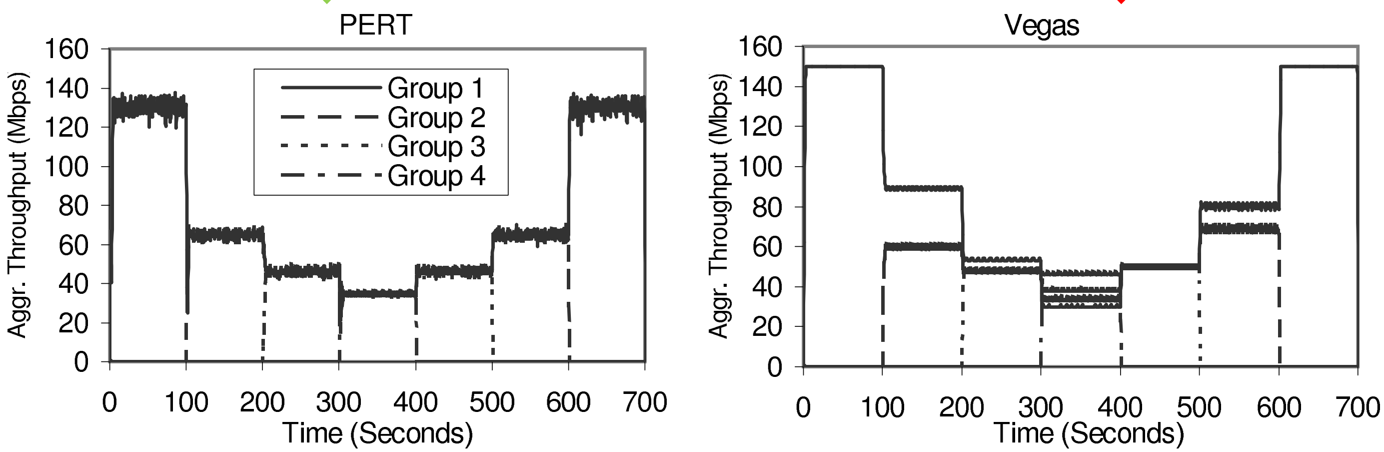


Figure 12: Response to sudden changes in responsive traffic.

*__Emulating AQM from End Hosts__ Sigcomm 2007, Bhandarkar, Reddy et al.

Virtual ECN implementation and simulations

- Ongoing work shows the idea works both in implementation and simulations
- Benefits of Virtual ECN approach
  - PI controller integral action provides multi-bit feedback
  - Noise filtered out by the controller, direct delay response leads to sudden fluctuations
  - Probabilistic response leads to stable aggregate behavior, removes synchronization

Takeaways

- Network (Router/Switch) based AQM leads to a unique fixed point
  - Then you have relative freedom as to how you go to that fixed point - AIMD with fixed parameters, MIMD, AIMD with dynamic parameters etc
- For end-to-end delay based congestion control AIMD with fixed parameters only way to reach fair fixed point and fixed delay
  - Even with AIMD, different flows with different notions of delay->congestion map will reach different fixed points
- For BBR style congestion control flows reach different operating points, all else being same
  - BBR with multiple bottlenecks doesn’t perform well
  - Probing with BBR needs to happen at RTT time scales, not a fixed constant (10 seconds)
  - AIMD style probing needed

Key abstraction to reason about all Congestion Control