Congestion Control In the Network

Brighten Godfrey
cs598pbg September 9 2010

Slides courtesy Ion Stoica with adaptation by Brighten

Today

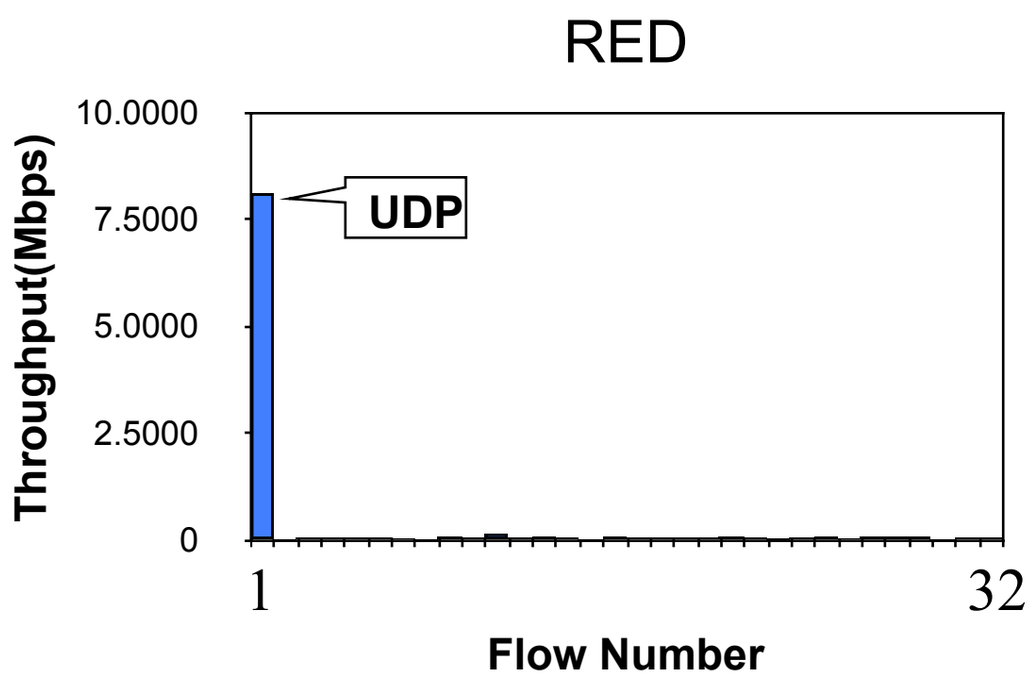
Fair queueing

XCP

Announcements

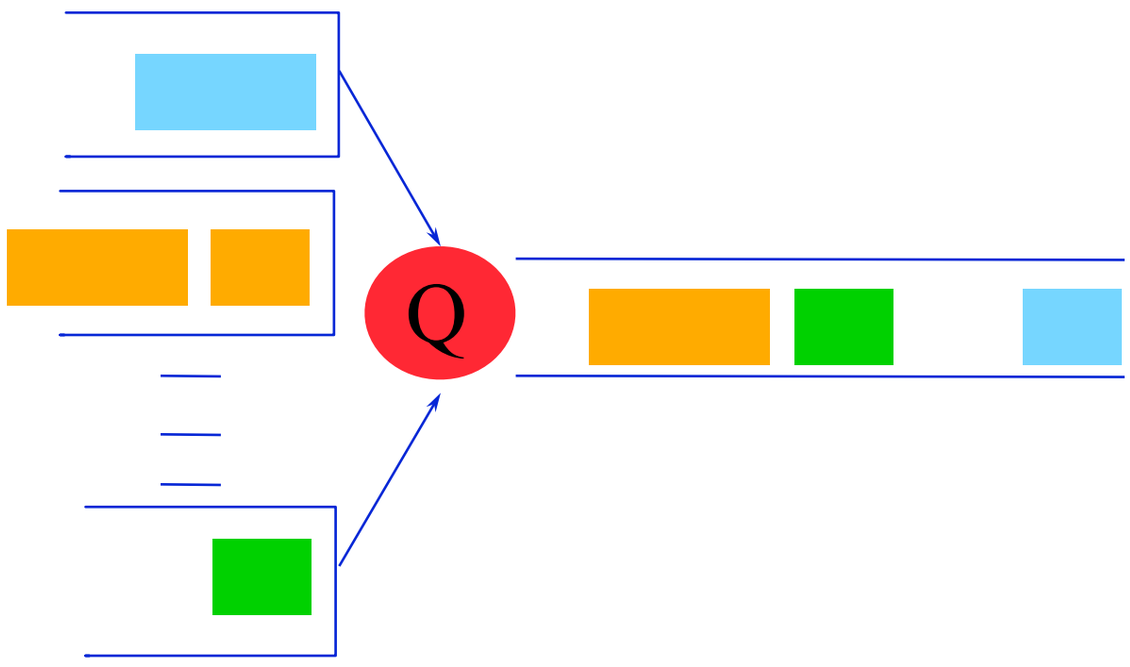
Problem: no isolation between flows

- No protection: if a flow misbehaves it will hurt the other flows
- Example: 1 UDP (10 Mbps) and 31 TCP's sharing a 10 Mbps link



A first solution

- Round-robin among different flows [Nagle ’87]
  - One queue per flow



Round-Robin Discussion

- Advantages: protection among flows
  - Misbehaving flows will not affect the performance of well-behaving flows
  - FIFO does not have such a property
- Disadvantages:
  - More complex than FIFO: per flow queue/state
  - Biased toward large packets – a flow receives service proportional to the number of packets (When is this bad?)

Fair Queueing (FQ) [DKS'89]

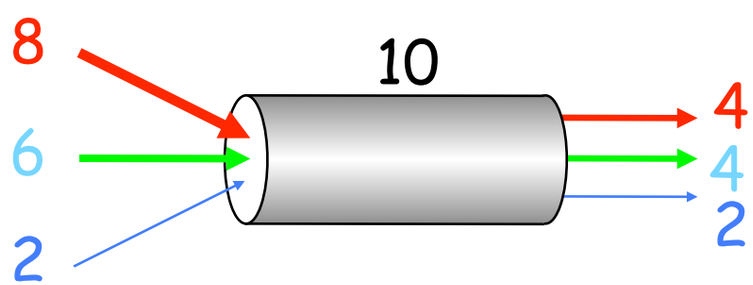
- Define a fluid flow system: a system in which flows are served bit-by-bit
  - i.e., bit-by-bit round robin

- Advantages
  - Each flow will receive exactly its max-min fair rate
  - ...and exactly its fair per-packet delay

Definition of fairness: Max-Min Fairness

- If link congested, compute *f* such that

$$\sum_i \min(r_i, f) = C$$



$f = 4$:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

Max-Min Fairness computation

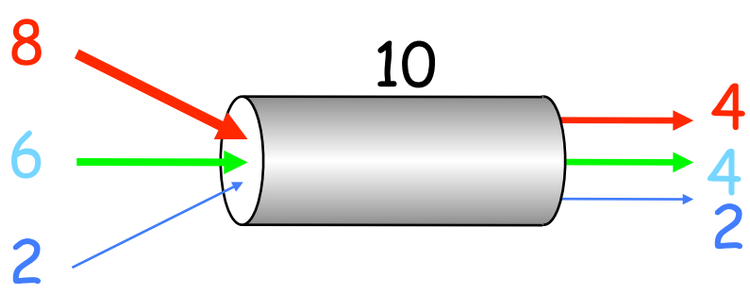
- Denote
  - C – link capacity
  - N – number of flows
  - r_i – arrival rate
- Max-min fair rate f computation:
  1. compute C/N
  2. if there are flows i such that $r_i <= C/N$, update C and N

$$C = C - \sum_{i\ s.t\ r_i \le C} r_i$$

  3. if no, $f = C/N$; terminate
  4. go to 1
- A flow can receive at most the fair rate, i.e., $\min(f, r_i)$

Example

- $C = 10$; $r_1 = 8$, $r_2 = 6$, $r_3 = 2$; $N = 3$
- $C/3 = 3.33 \rightarrow C = C - r3 = 8$; $N = 2$
- $C/2 = 4$; $f = 4$

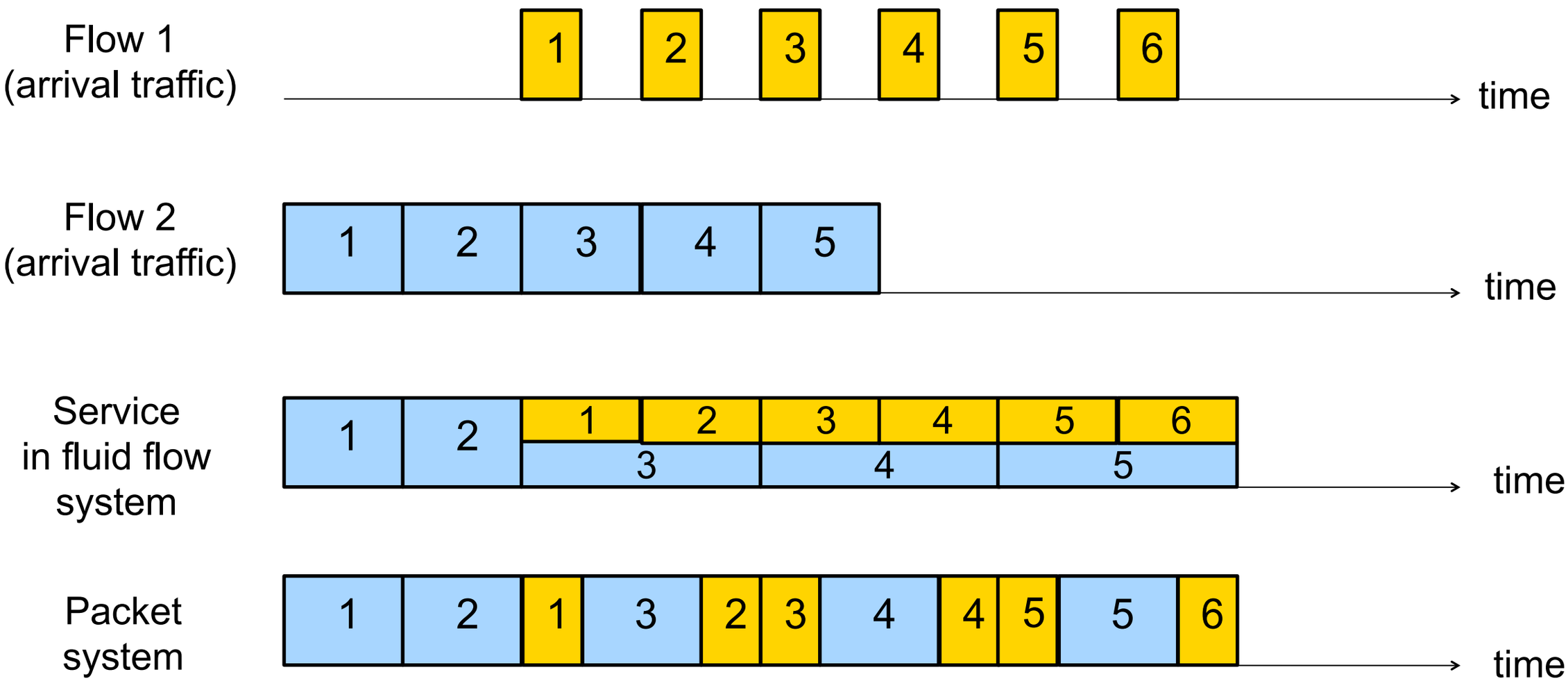


$f = 4$:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

Implementing Fair Queueing

- What we just saw was bit-by-bit round robin
- Can't do it – can't interrupt transfer of a packet (why not?)
- Idea: serve packets in the order in which they would have finished transmission in the fluid flow system
- Strong guarantees
  - Each flow will receive exactly its max-min fair rate (+/- one packet size)
  - ...and exactly its max-min fair per-packet delay (+/- one packet size) or better
    - What does that mean? Suppose you have full access to a link whose bandwidth is equal to your max-min fair rate. Then your delay with FQ will be your delay on that virtual link (or better), plus or minus one packet size.

Example


Flow 1
(arrival traffic)
1
2
3
4
5
6
time
Flow 2
(arrival traffic)
1
2
3
4
5
time
Service
in fluid flow
system
1
2
1
2
3
4
5
6
3
4
5
time
Packet
system
1
2
1
3
2
3
4
4
5
5
6
time

Guarantees

- Translating fluid to discrete packet model doesn't actually involve a lot of combinatorics.
- Theorem: each packet P will finish transmission at or before its finish time in fluid flow model.
  - assuming (for now) all packets are in queue at time 0
- Proof:
  - Suppose P's finish time is T in fluid model. Need to show P finishes by T in packet model.
  - In packet model, what packets come before P or are P? Packets that have finished by T in fluid model.
  - Total amount to send: ≤ RT bits (possibly less: some packets may still be in progress) where R is link rate
  - Packet model remains busy sending packets the entire time (because packets have all arrived at time 0), so these will be sent in time <= RT / R = T.

Guarantees

- Translating fluid to discrete packet model doesn't actually involve a lot of combinatorics.
- Theorem: each packet P will finish transmission at or before its finish time in fluid flow model.
  - assuming (for now) all packets are in queue at time 0
- So, why is the real guarantee (without assumption) only approximate (+/- one packet)?

Problem

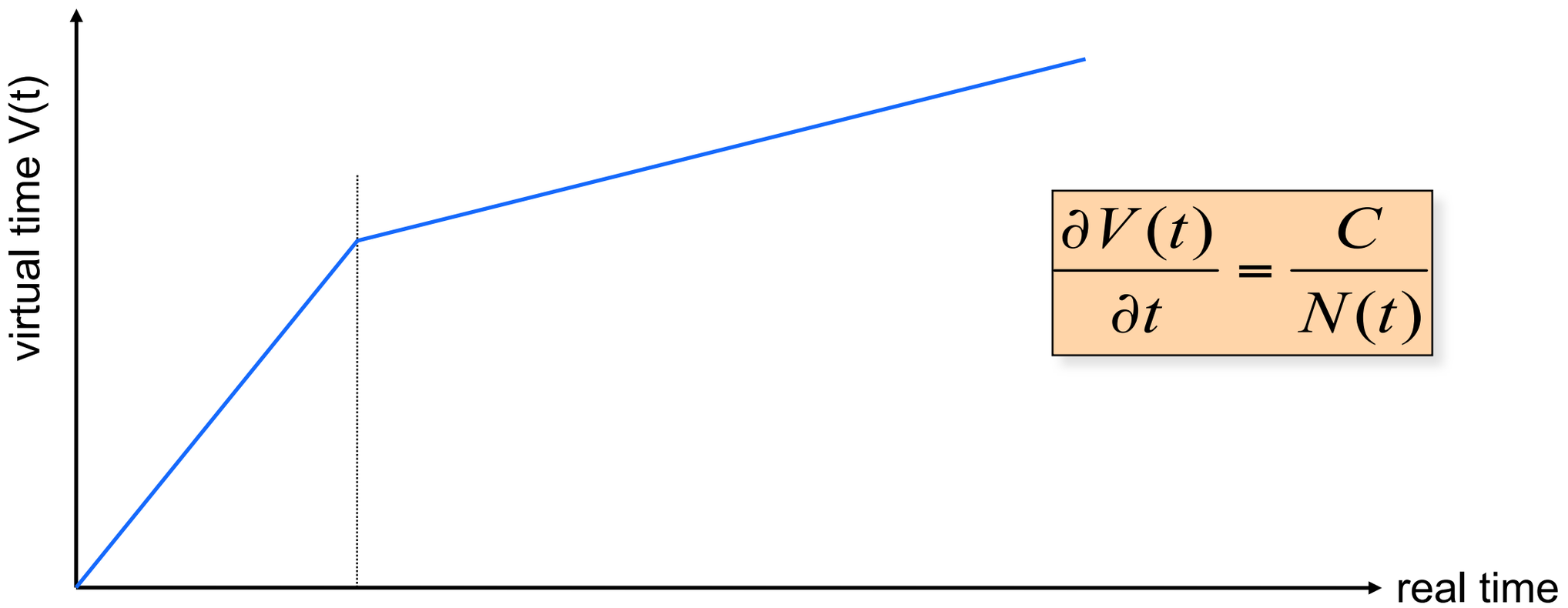
- Recall algorithm: “serve packets in the order in which they would have finished transmission in the fluid flow system”
- So, need to compute finish time of each packet in the fluid flow system
- ... but new packet arrival can change finish times of packets in the system (perhaps all packets!)
- Updating those times would be expensive

Solution: Virtual Time

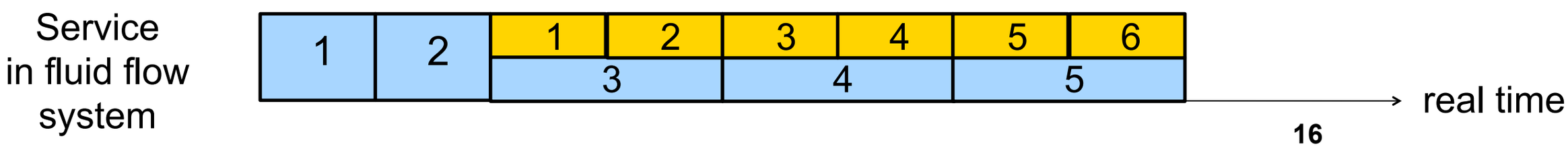
- Key Observation: while the finish times of packets may change when a new packet arrives, the order in which packets finish doesn't!
  - Only the order is important for scheduling
- Solution: instead of the packet finish time maintain the number of rounds needed to send the remaining bits of the packet (virtual finishing time)
  - Virtual finishing time doesn't change upon packet arrival
- System virtual time = index of the round in the bit-by-bit round robin scheme

System Virtual Time: *V*(t)

- Measure service, instead of time
- V(t) slope – rate at which every active flow receives service
  - *C* = link capacity
  - *N*(t) = number of active flows in fluid flow system at time t



$$\frac{\partial V(t)}{\partial t} = \frac{C}{N(t)}$$



Fair Queueing Implementation

- Define
  - F_i^k = virtual finishing time of packet k of flow i
  - a_i^k = arrival time of packet k of flow i
  - L_i^k = length of packet k of flow i

- Virtual finishing time of packet $k+1$ of flow i is

$$F_i^{k+1} = \max(V(a_i^k), F_i^k) + L_i^{k+1}$$

- Order packets by increasing virtual finishing time, and send them in that order

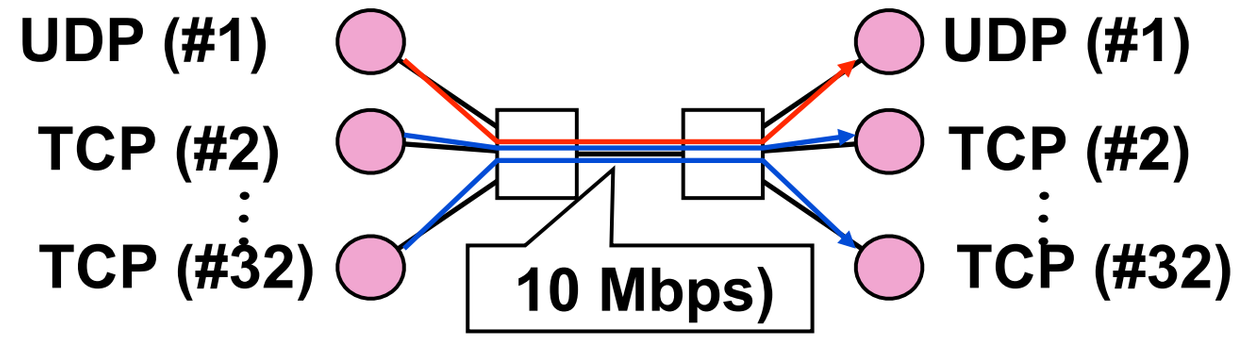
Weighted Fair Queueing (WFQ)

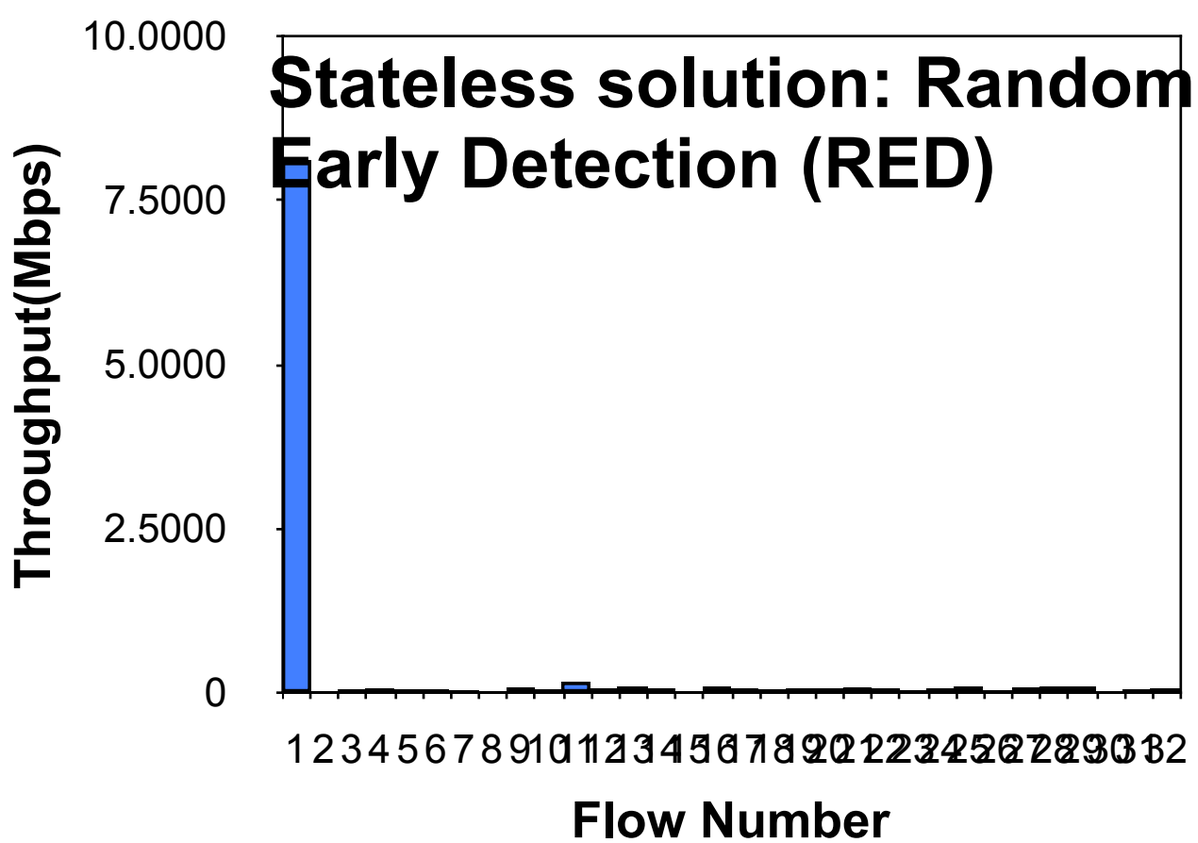
- What if we don't want exact fairness?
  - Maybe web traffic is more important than file sharing
- Assign weight w_i to each flow i
- And change virtual finishing time

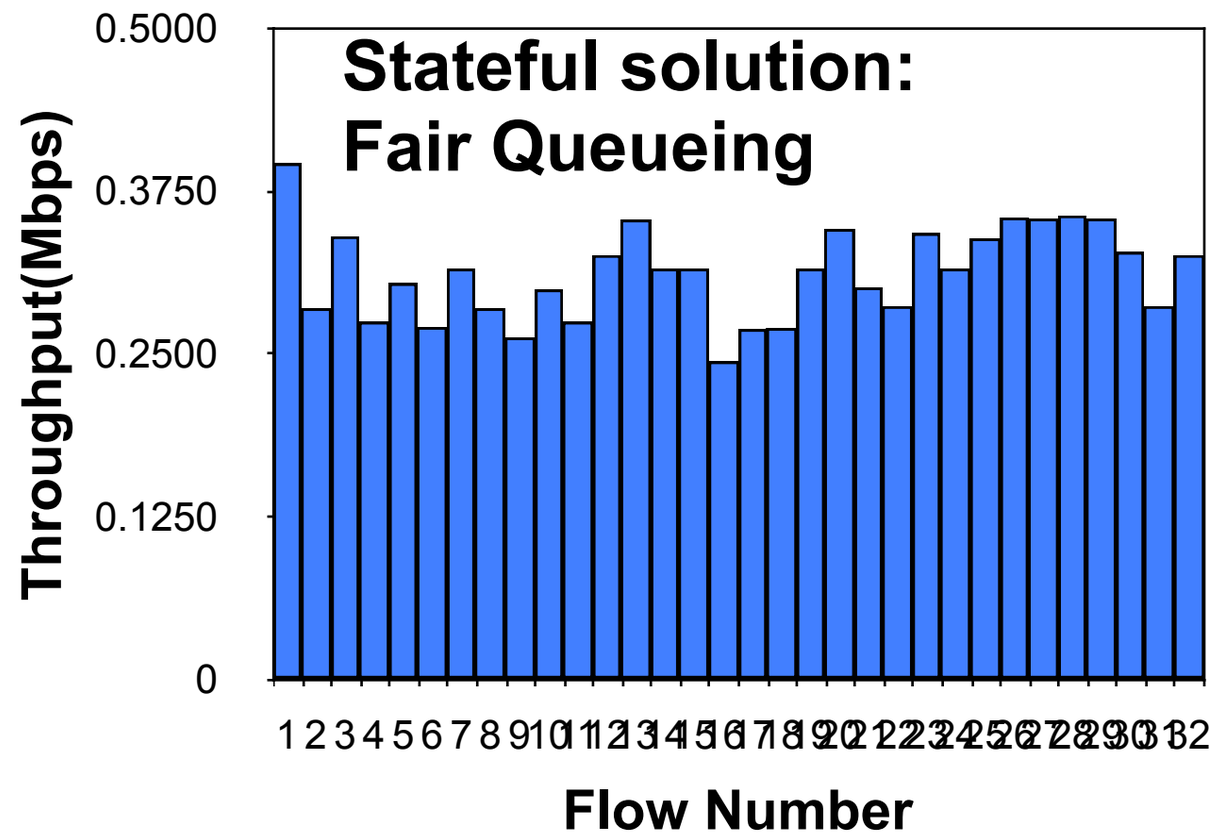
$$F_i^{k+1} = \max(V(a_i^k), F_i^k) + \frac{L_i^{k+1}}{w_i}$$

Simulation Example

- 1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps link







Summary

- FQ does not eliminate congestion; it just manages the congestion
- You need both end-host congestion control and router support for congestion control
  - End-host congestion control to adapt rate
  - Router congestion control to protect/isolate
- Don’t forget buffer management: you still need to drop in case of congestion. Which packets would you drop in FQ?
  - One possibility: packet from the longest queue

Today

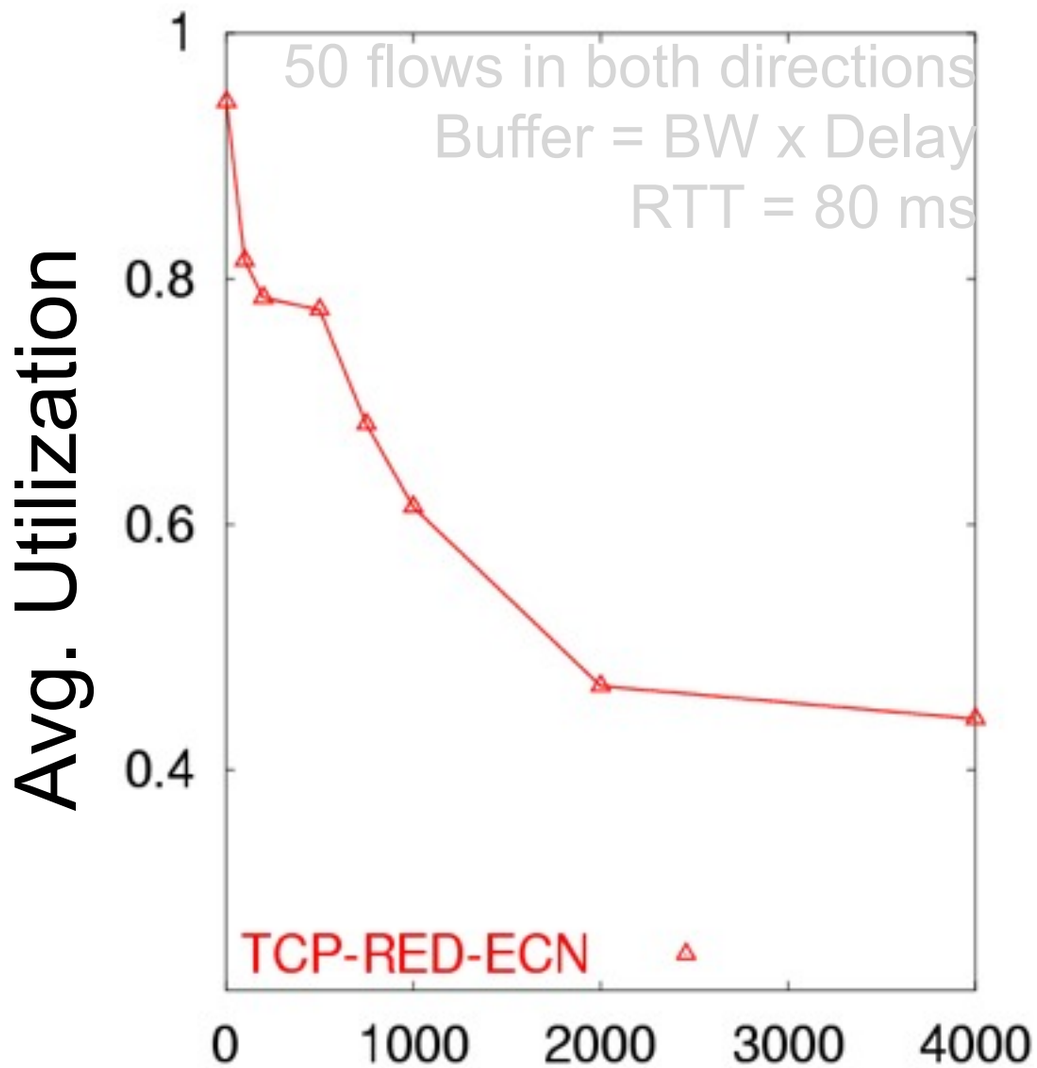
Fair queueing

XCP

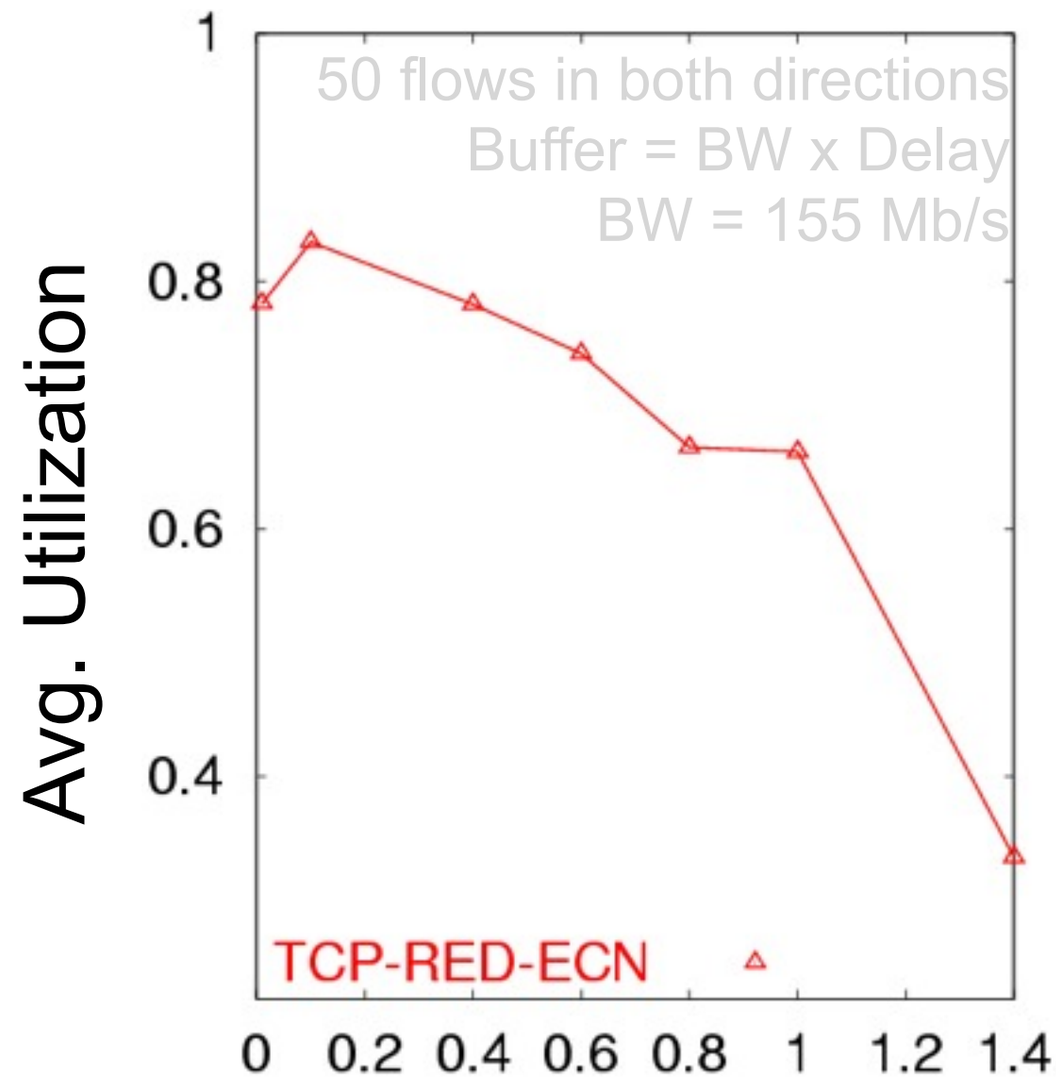
Announcements

TCP congestion control performs poorly as bandwidth or delay increases

Shown analytically in [Low01] and via simulations



Bottleneck Bandwidth (Mb/s)

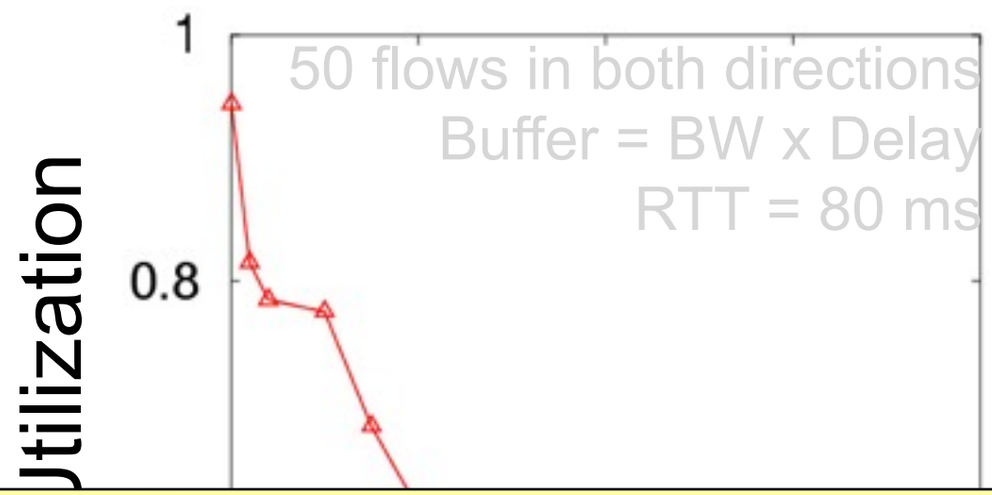


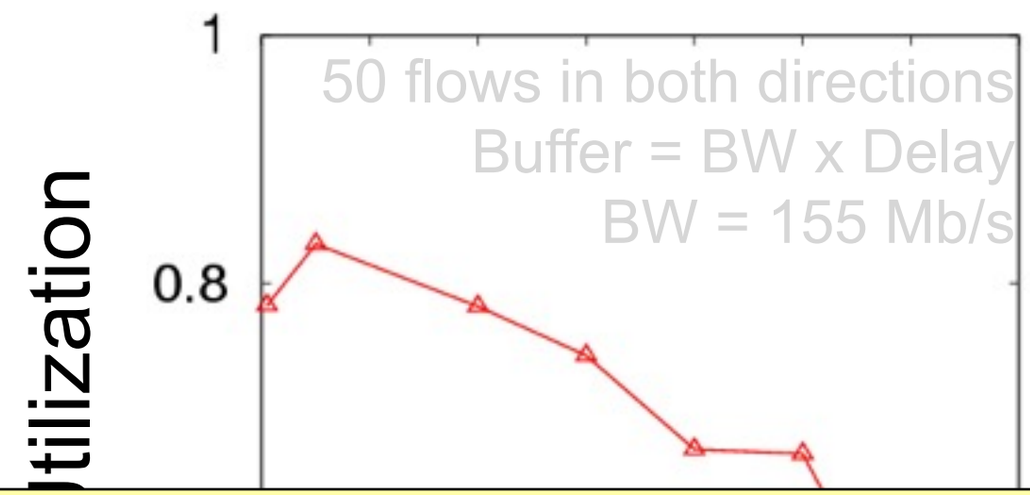
Round Trip Delay (sec)

Why?

TCP congestion control performs poorly as bandwidth or delay increases

Shown analytically in [Low01] and via simulations





Because TCP lacks fast response

- Spare bandwidth is available ⇒ TCP increases by 1 pkt/RTT even if spare bandwidth is huge
- When a TCP starts, it increases exponentially ⇒ Too many drops ⇒ Flows ramp up by 1 pkt/RTT, taking forever to grab the large bandwidth

Approach

- Routers inform sources
  - Don't have to guess about right rate to send
  - Can give different feedback to different senders
- Decouple congestion control from fairness

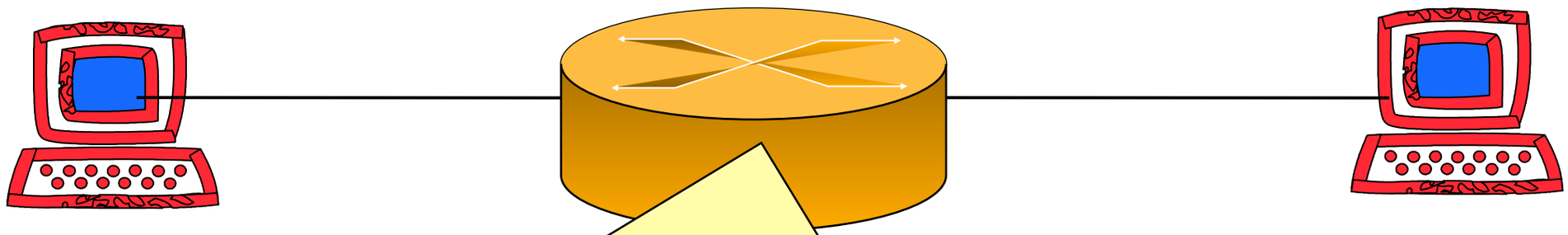
High Utilization;
Small Queues;
Few Drops

Bandwidth
Allocation
Policy

Characteristics of XCP Solution

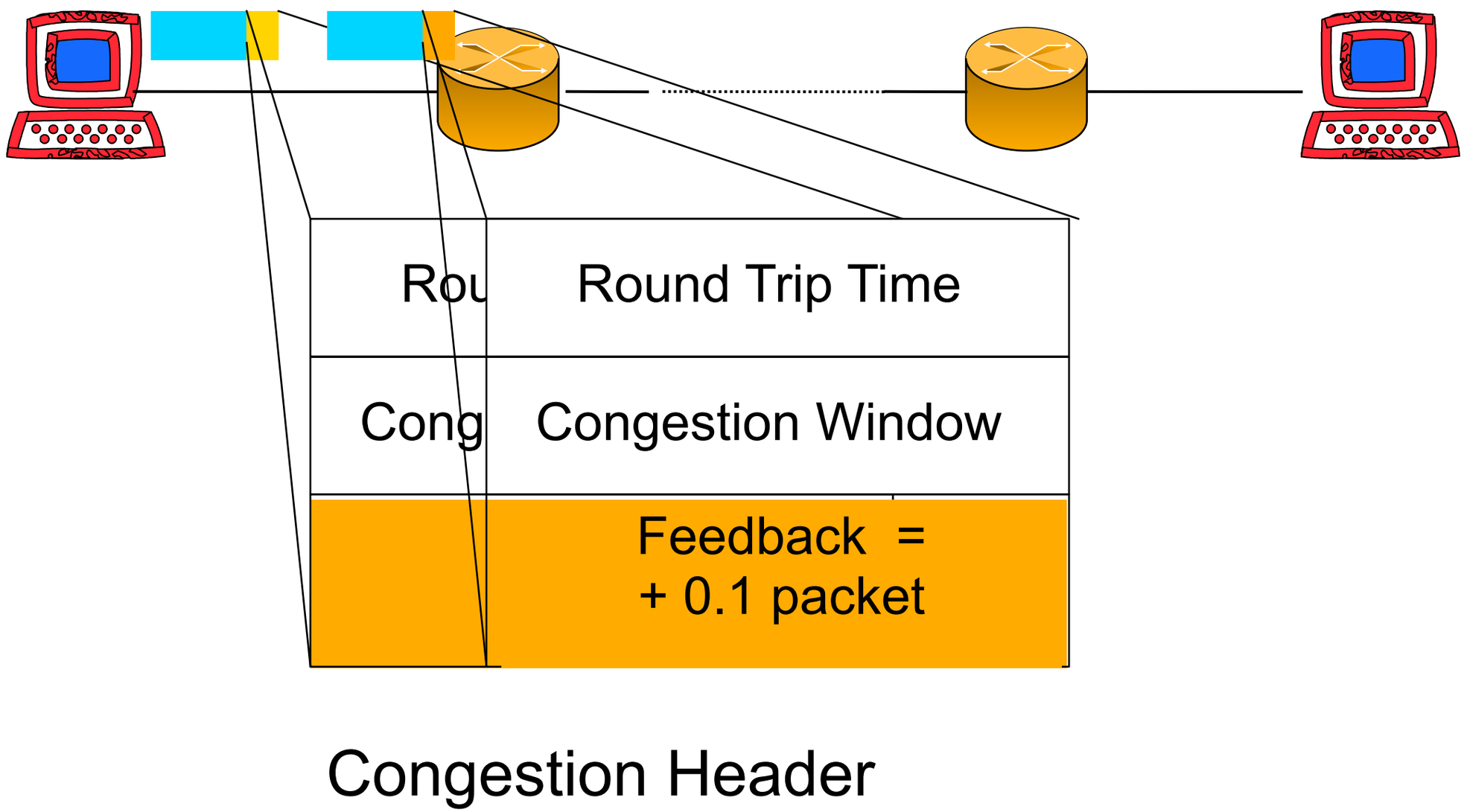
1. Improved Congestion Control (in high bandwidth-delay & conventional environments):
   - Small queues
   - Almost no drops
2. Improved Fairness
3. Scalable (no per-flow state)
4. Flexible bandwidth allocation: min-max fairness, proportional fairness, differential bandwidth allocation,…

XCP: An eXplicit Control Protocol

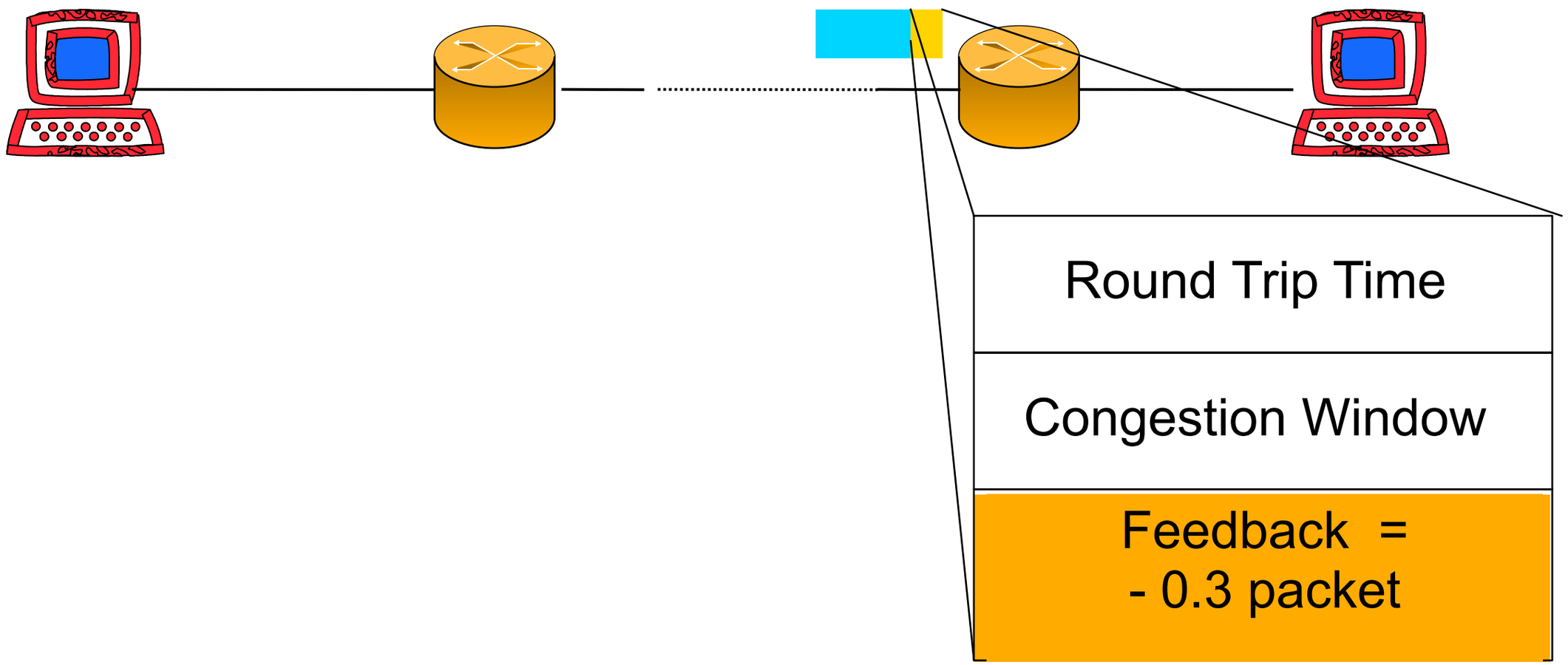


1. Congestion Controller
2. Fairness Controller

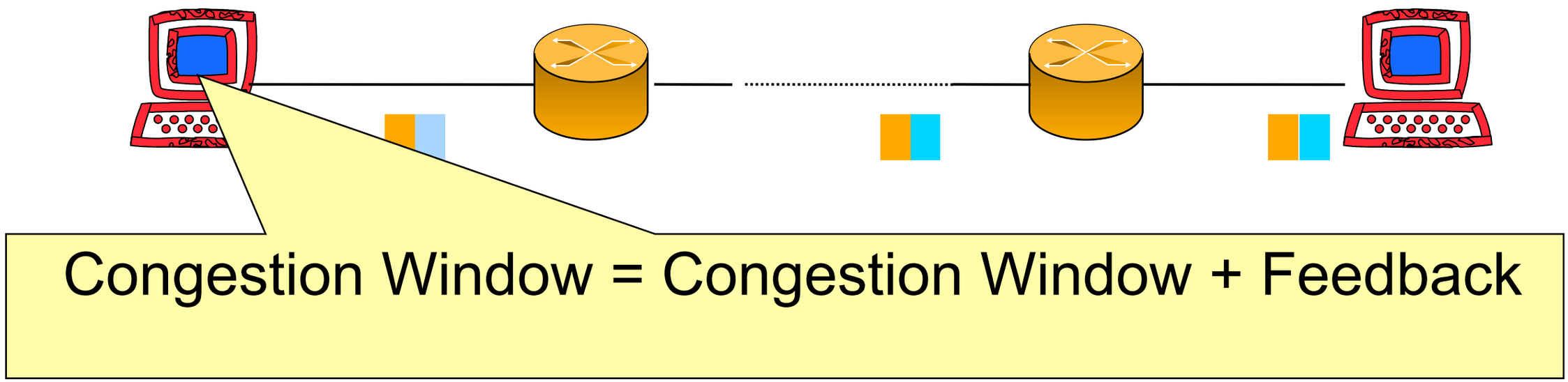
How does XCP Work?



How does XCP Work?



How does XCP Work?



XCP extends ECN and CSFQ (Core Stateless Fair Queueing)

Routers compute feedback without any per-flow state

How Does an XCP Router Compute the Feedback?

Congestion Controller

Goal: Matches input traffic to link capacity & drains the queue

Looks at aggregate traffic & queue

Algorithm:

Aggregate traffic changes by Δ

$\Delta \sim$ Spare Bandwidth

$\Delta \sim$ - Queue Size

So, $\Delta = \alpha\, d_{avg}\, Spare - \beta\, Queue$

Fairness Controller

Goal: Divides Δ between flows to converge to fairness

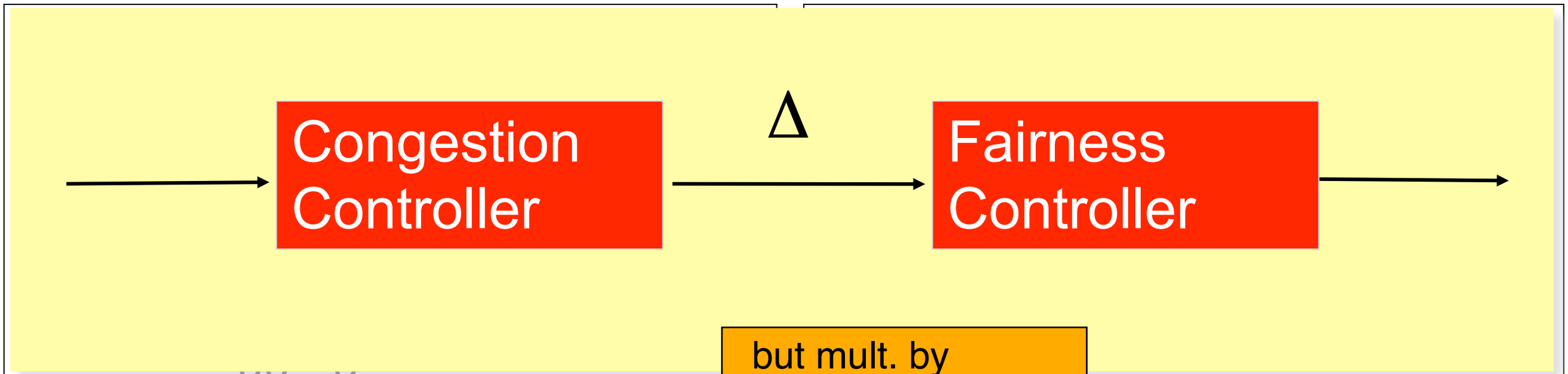
Looks at a flow's state in Congestion Header

Algorithm:

If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rates

How Does an XCP Router Compute the Feedback?



queue

MIMD

but mult. by available capacity

Hea

AIMD

<u>Algorithm:</u>

Aggregate traffic changes by Δ

Δ ~ Spare Bandwidth

Δ ~ - Queue Size

So, $\Delta = \alpha\, d_{avg}\, Spare - \beta\, Queue$

<u>Algorithm:</u>

If Δ > 0 ⇒ Divide Δ equally between flows

If Δ < 0 ⇒ Divide Δ between flows proportionally to their current rates

Getting the devil out of the details …

Congestion Controller

$\Delta = \alpha\, d_{avg}\, Spare - \beta\, Queue$

Theorem: System converges to optimal utilization (i.e., stable) for any link bandwidth, delay, number of sources if:

$$0 < \alpha < \frac{\pi}{4\sqrt{2}} \quad and \quad \beta = \alpha^2\sqrt{2}$$

No Parameter Tuning

Fairness Controller

Algorithm:

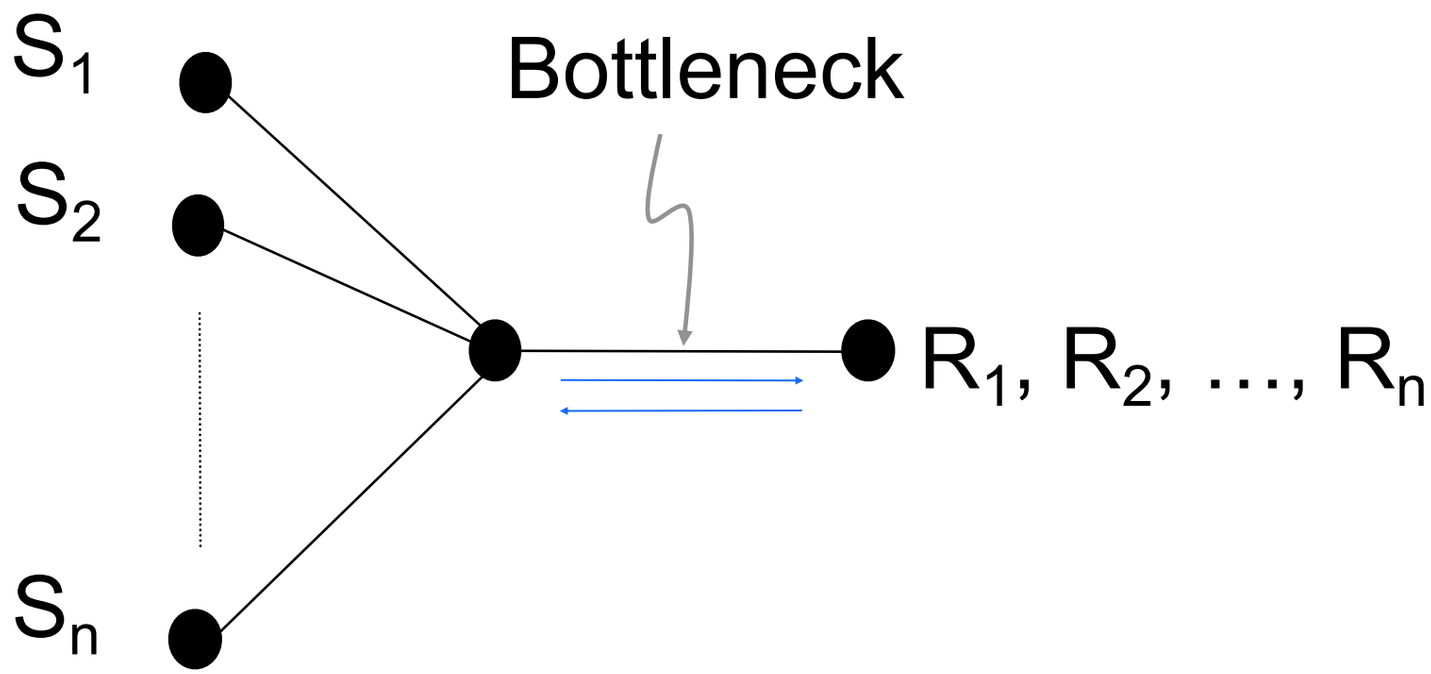
If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rates

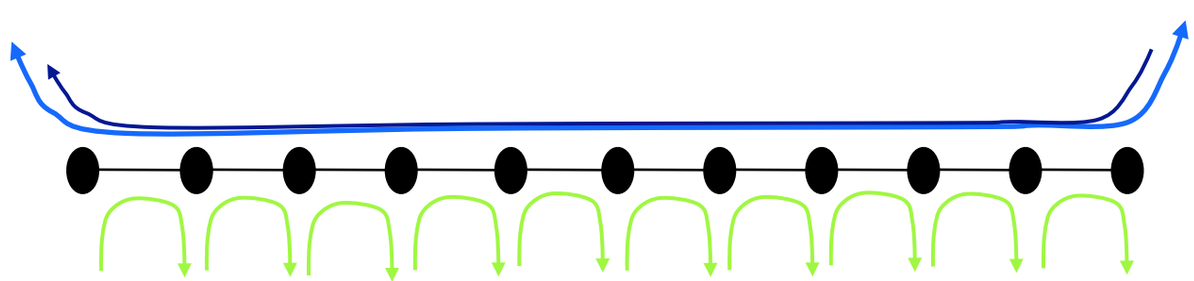
$$p_i = \frac{h + \max(\phi, 0)}{d \cdot \sum \frac{rtt_i \cdot s_i}{cwnd_i}} \cdot \frac{rtt_i^2 \cdot s_i}{cwnd_i}$$

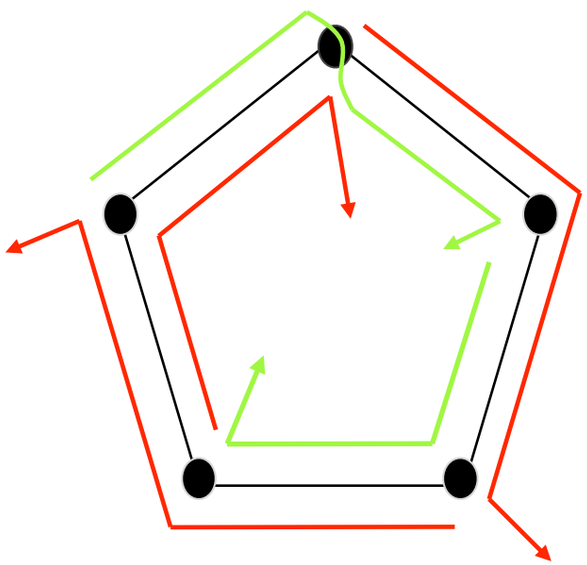
No Per-Flow State

Subset of Results



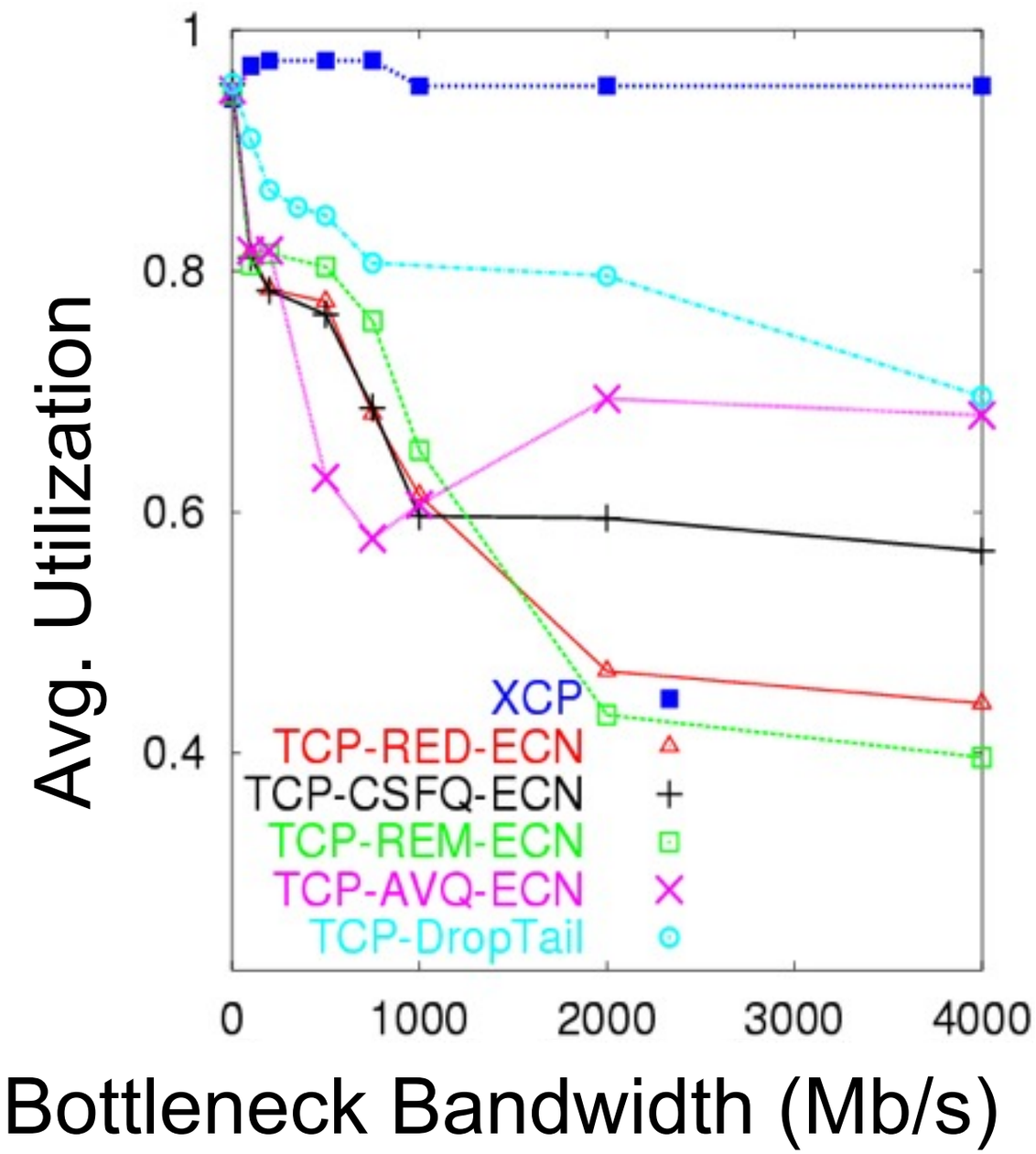
Similar behavior over:



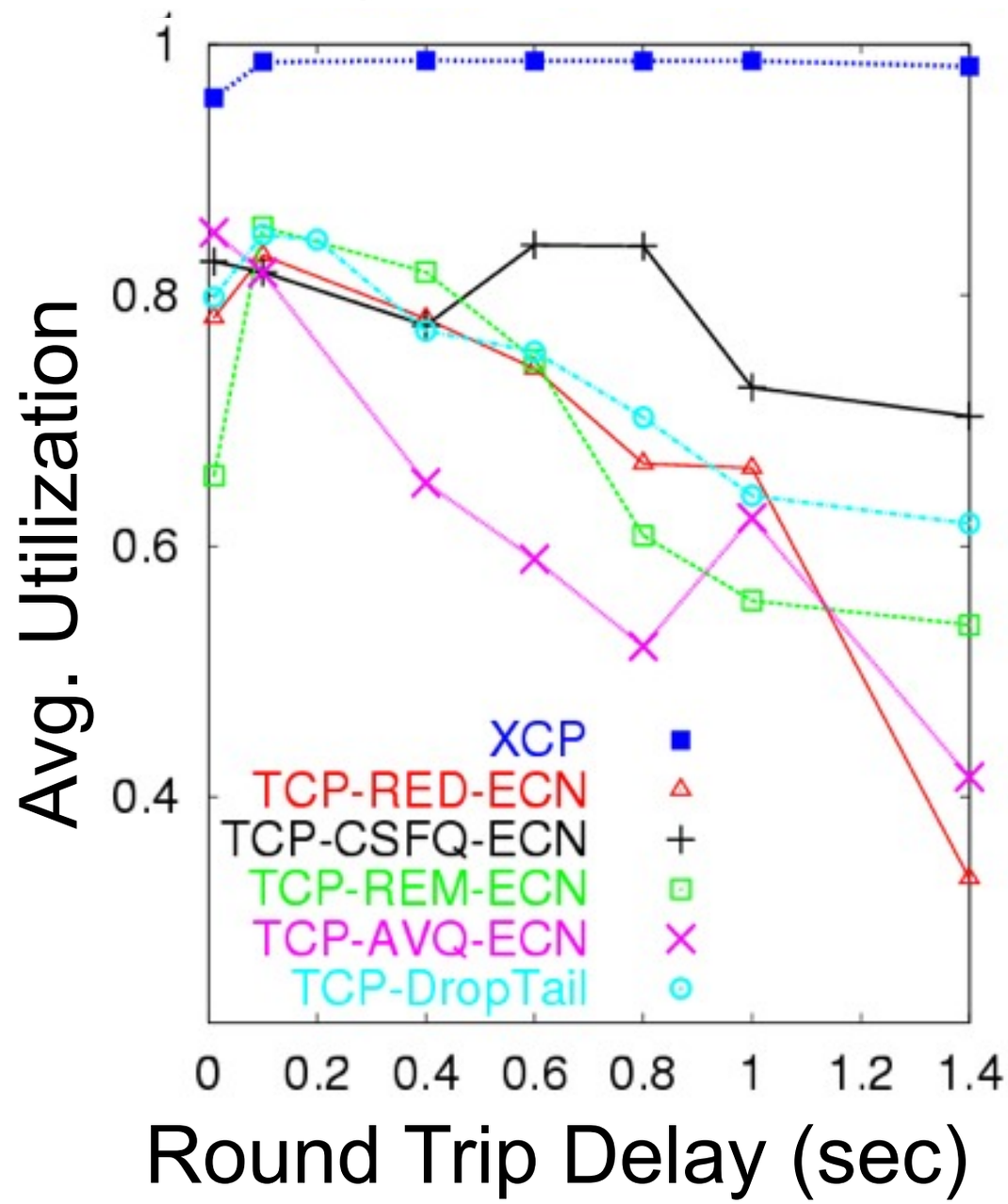


XCP Remains Efficient as Bandwidth or Delay Increases

Utilization as a function of Bandwidth

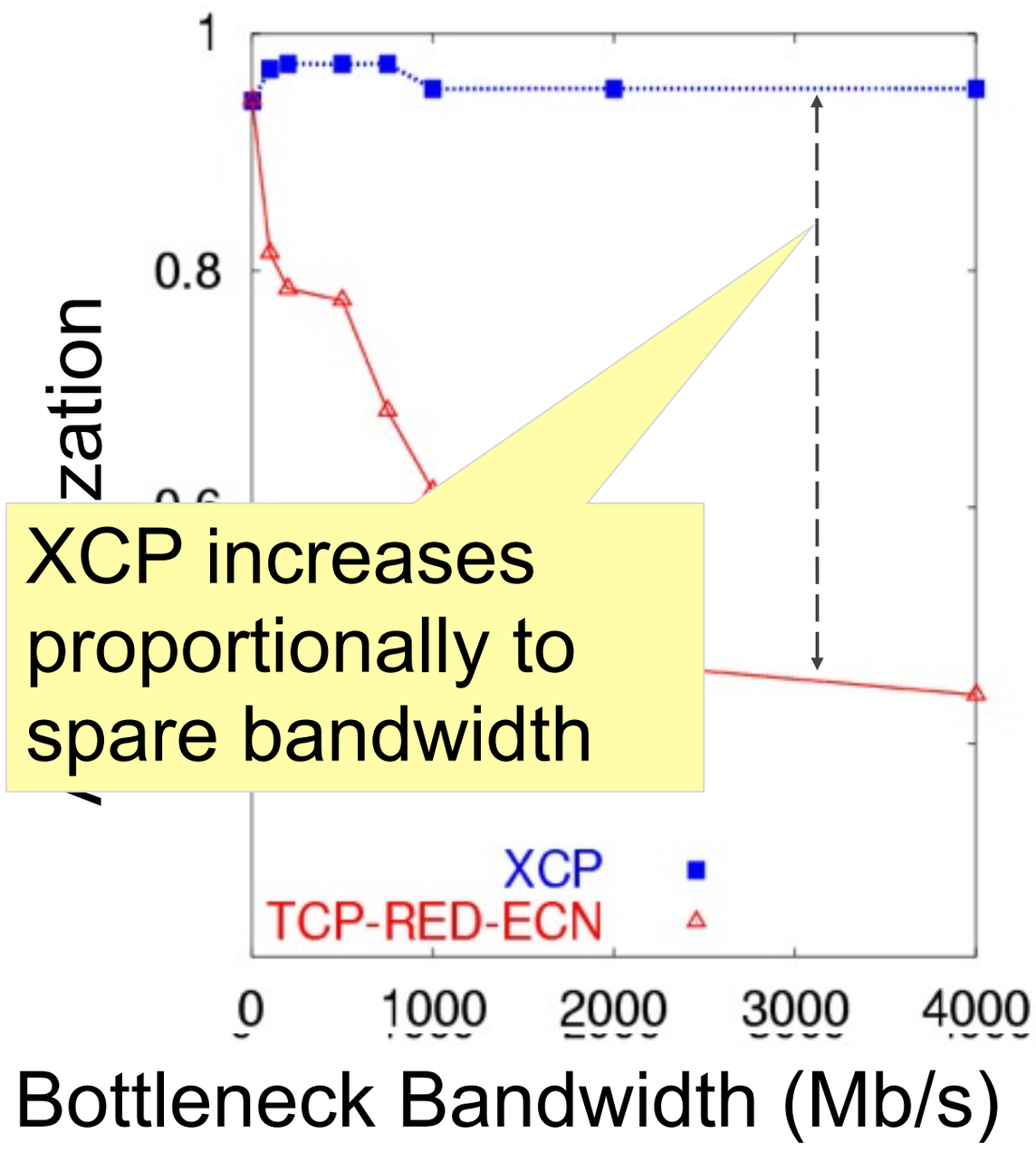


Utilization as a function of Delay



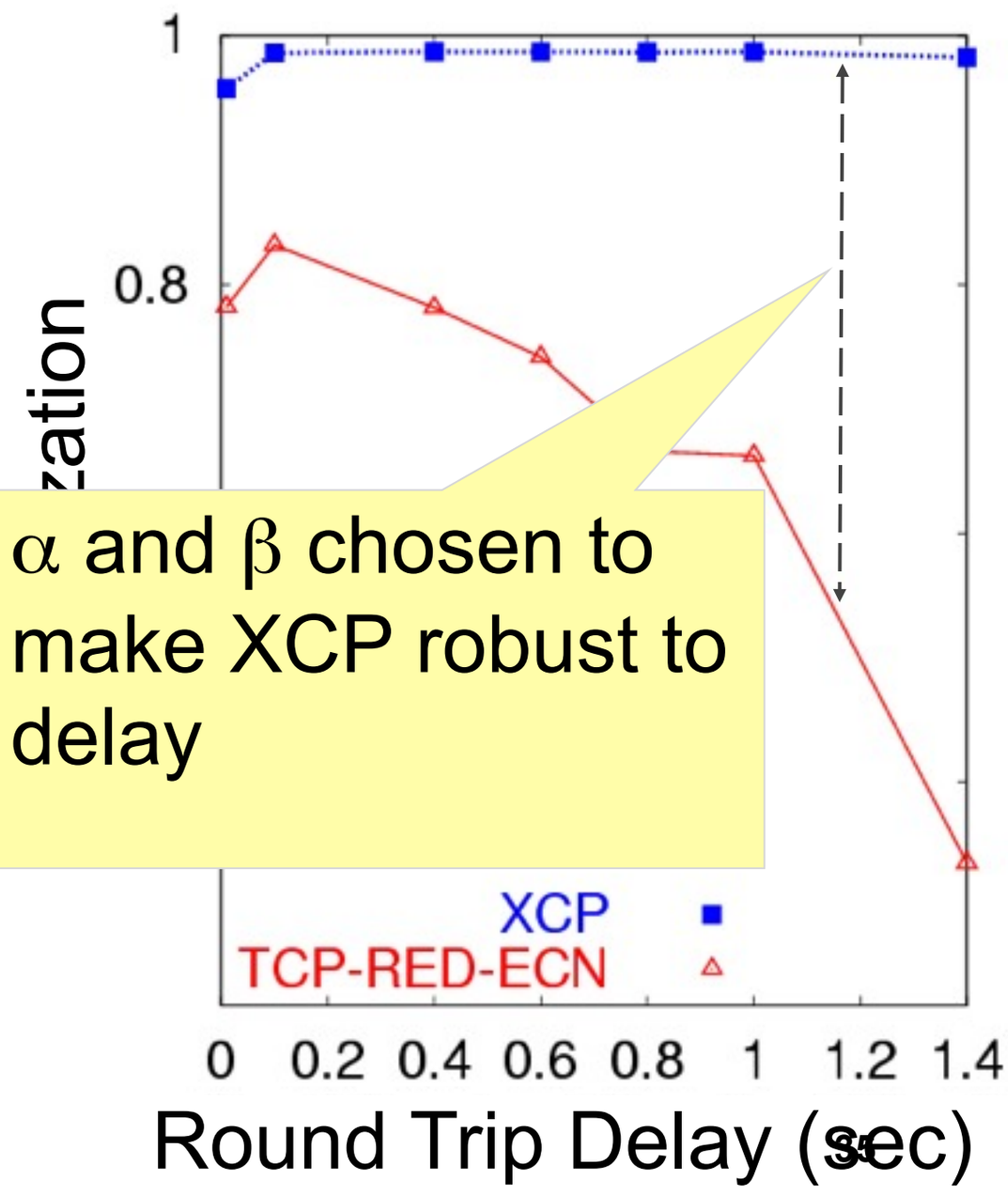
XCP Remains Efficient as Bandwidth or Delay Increases

Utilization as a function of Bandwidth



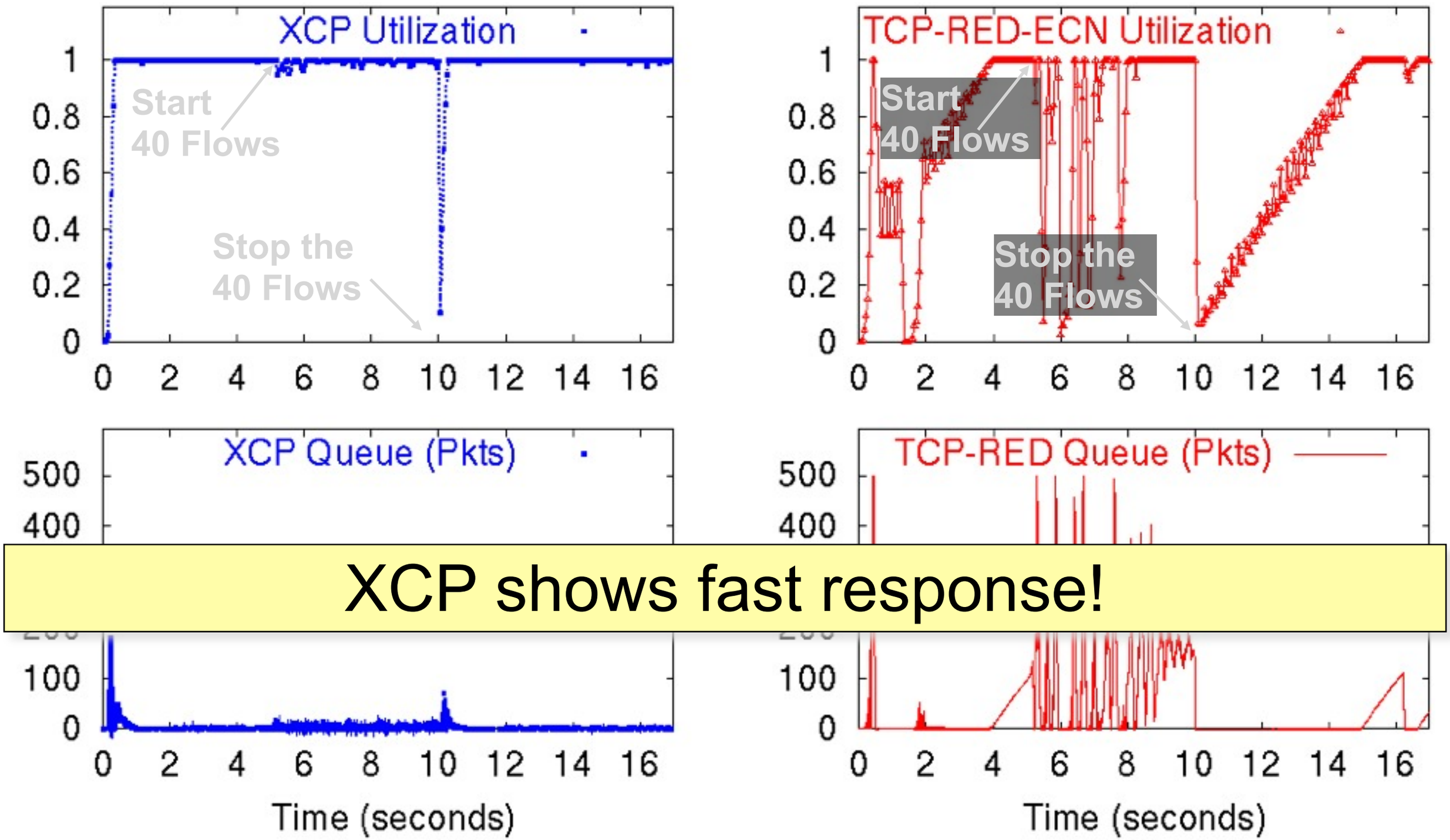
XCP increases proportionally to spare bandwidth

Utilization as a function of Delay



α and β chosen to make XCP robust to delay

XCP Shows Faster Response than TCP



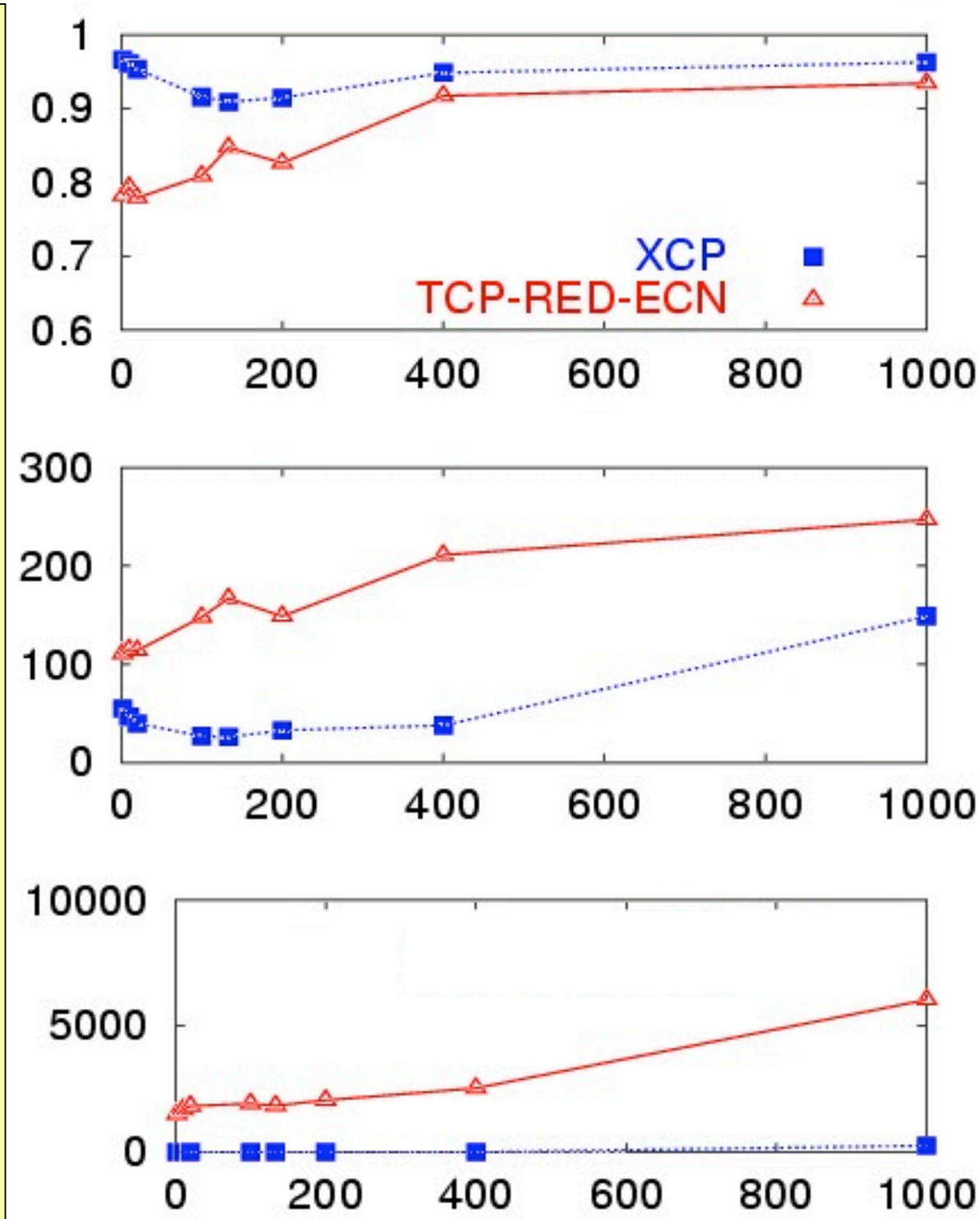
XCP shows fast response!

XCP Deals Well with Short Web-Like Flows

Average Utilization

Average Queue

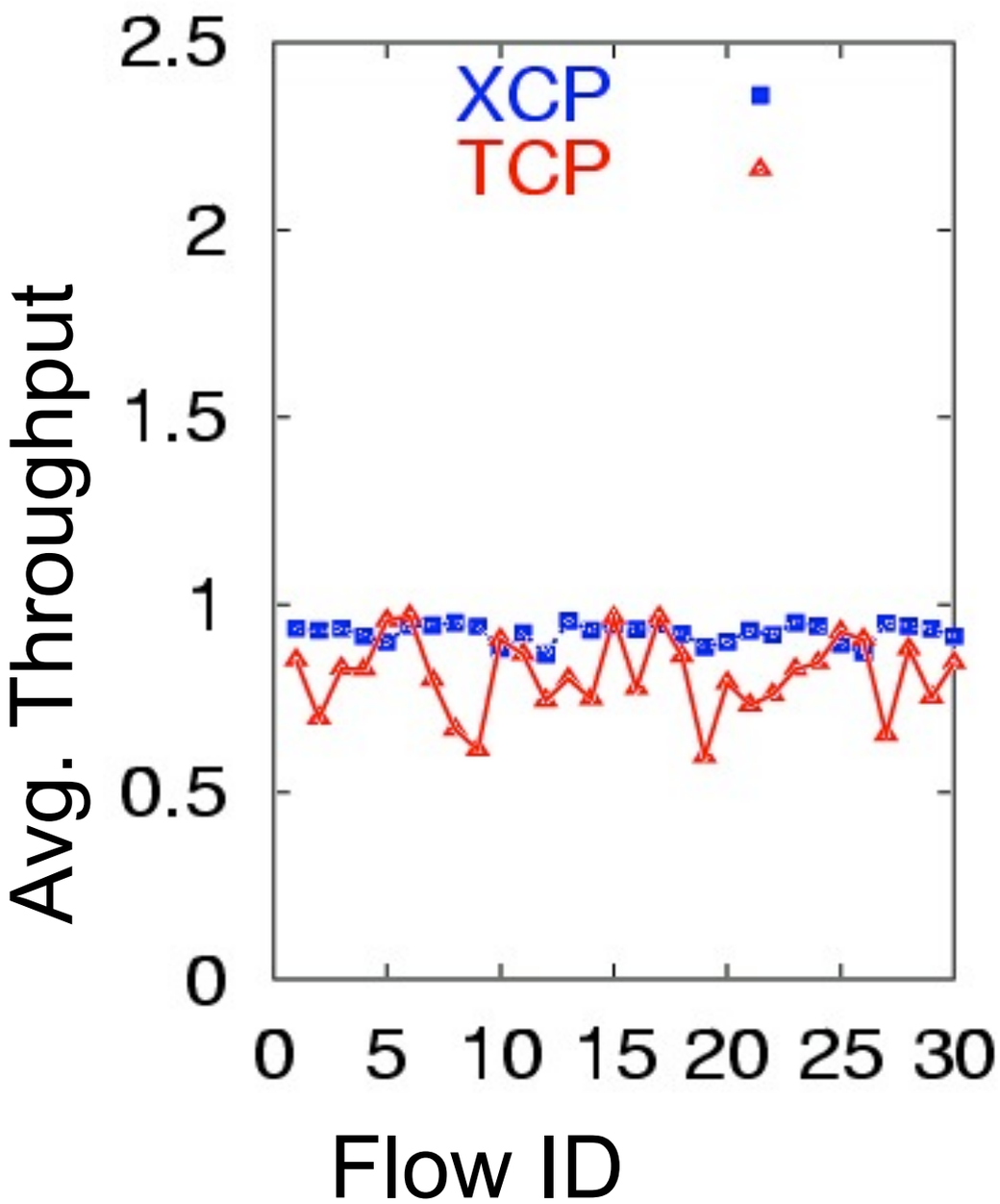
Drops



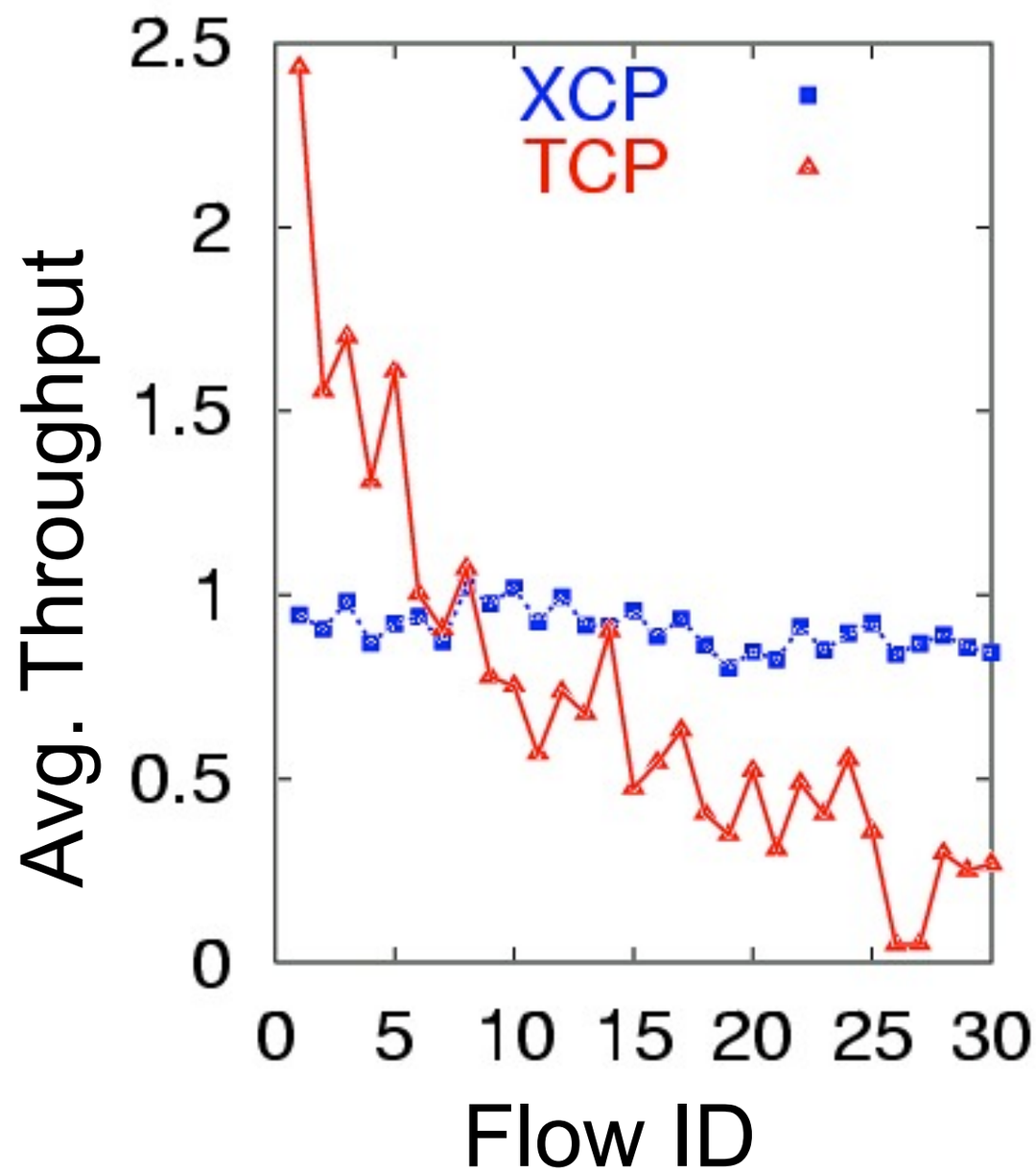
Arrivals of Short Flows/sec

XCP is Fairer than TCP

Same RTT



Different RTT



(RTTs are 40 ms to 330 ms)

What have we fixed so far?

TCP Problem	Solution
Fills queues: adds loss, latency	RED (partial), XCP
Slow to converge	XCP (except short flows)
Loss ≠ congestion	XCP
May not utilize full bandwidth	XCP
Unfair to large-RTT	XCP
Unfair to short flows	?
Is equal rates really "fair"?	?
Vulnerable to selfishness	XCP & Fair Queueing (partial)

Today

Fair queueing

XCP

Announcements

Office hours

- Tuesdays after class: is it good for you?

Project proposals

- Project proposals due 11:59 p.m. Tuesday
  - Submit via email to Brighten
  - 1/2 page, plaintext
- Describe:
  - the problem you plan to address
  - what will be your first steps
  - what is the most closely related work, and why it has not addressed your problem
  - if there are multiple people on your project team, who they are and how you plan to partition the work among the team.

Project proposals

- Come talk to me before your proposal
  - Available until 6 pm today after class
  - Or we can make an appointment

Also on Tuesday...

- First opportunity to present a paper! One of:
  - Dukkipati & McKeown. "Why Flow-Completion Time is the Right metric for Congestion Control and why this means we need new algorithms", CCR 2006
  - Alizadeh et al. "Data Center TCP (DCTCP)", SIGCOMM 2010
- If you would like to present one of these two, email me by Friday morning. FCFS.