Congestion Control in the Network

Brighten Godfrey

CS 538 Sept 17 2013

How TCP congestion control is broken

Limitations of TCP CC

Efficiency

Tends to fill queues (adding latency)

Slow to converge (for short flows or links with high bandwidth•delay product)

Loss ≠ congestion

May not fully utilize bandwidth

Limitations of TCP CC

Fairness

Unfair to large-RTT flows (less throughput)

Unfair to short flows if ssthresh starts small

Equal rates isn't necessarily "fair" or best

Vulnerable to selfish & malicious behavior

- TCP assumes everyone is running TCP!

Limitations of TCP CC

Fills queues: adds loss, latency

Slow to converge

Loss $\neq$ congestion

May not utilize full bandwidth

Unfair to large-RTT

Unfair to short flows

Is equal rates really “fair”?

Vulnerable to selfishness

Limitations of TCP CC

Fills queues: adds loss, latency

Slow to converge

Loss $\neq$ congestion

May not utilize full bandwidth

Unfair to large-RTT

Unfair to short flows

Hard to use only end-to-end information to find 'right' rate

Obvious solution: Get more info from network

Is equal rates really "fair"?

Vulnerable to selfishness

Limitations of TCP CC

Fills queues: adds loss, latency

Slow to converge

Loss ≠ congestion

May not utilize full bandwidth

Unfair to large-RTT

Unfair to short flows

Hard to use only end-to-end information to find 'right' rate

Obvious solution: Get more info from network

Is equal rates really "fair"?

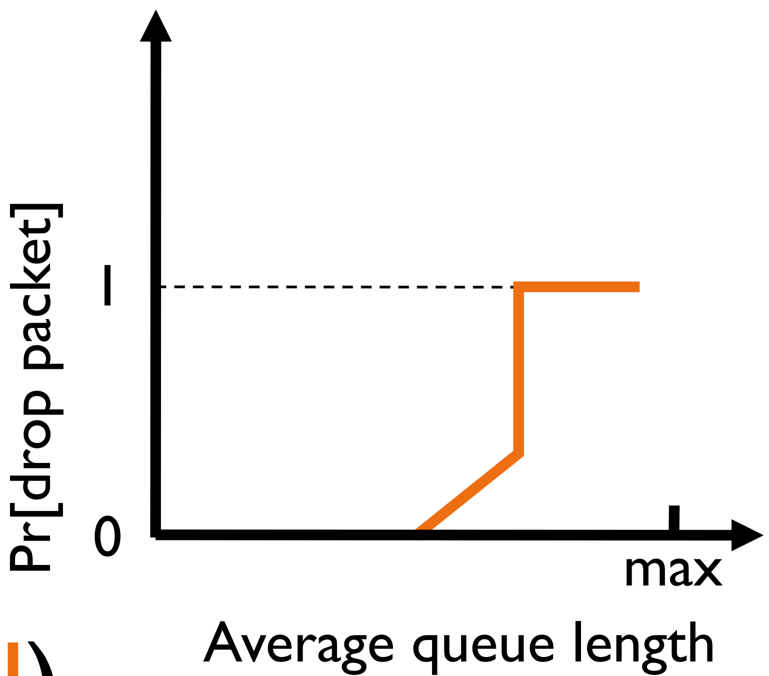
Vulnerable to selfishness

Incentive issues

Congestion control with help from the network

Getting better info from the net

Random early detection (RED)



- Drops more packets (randomly) as congestion increases
- Mechanism is entirely within routers

Explicit Congestion Notification (ECN)

- Mark bit in header instead of dropping

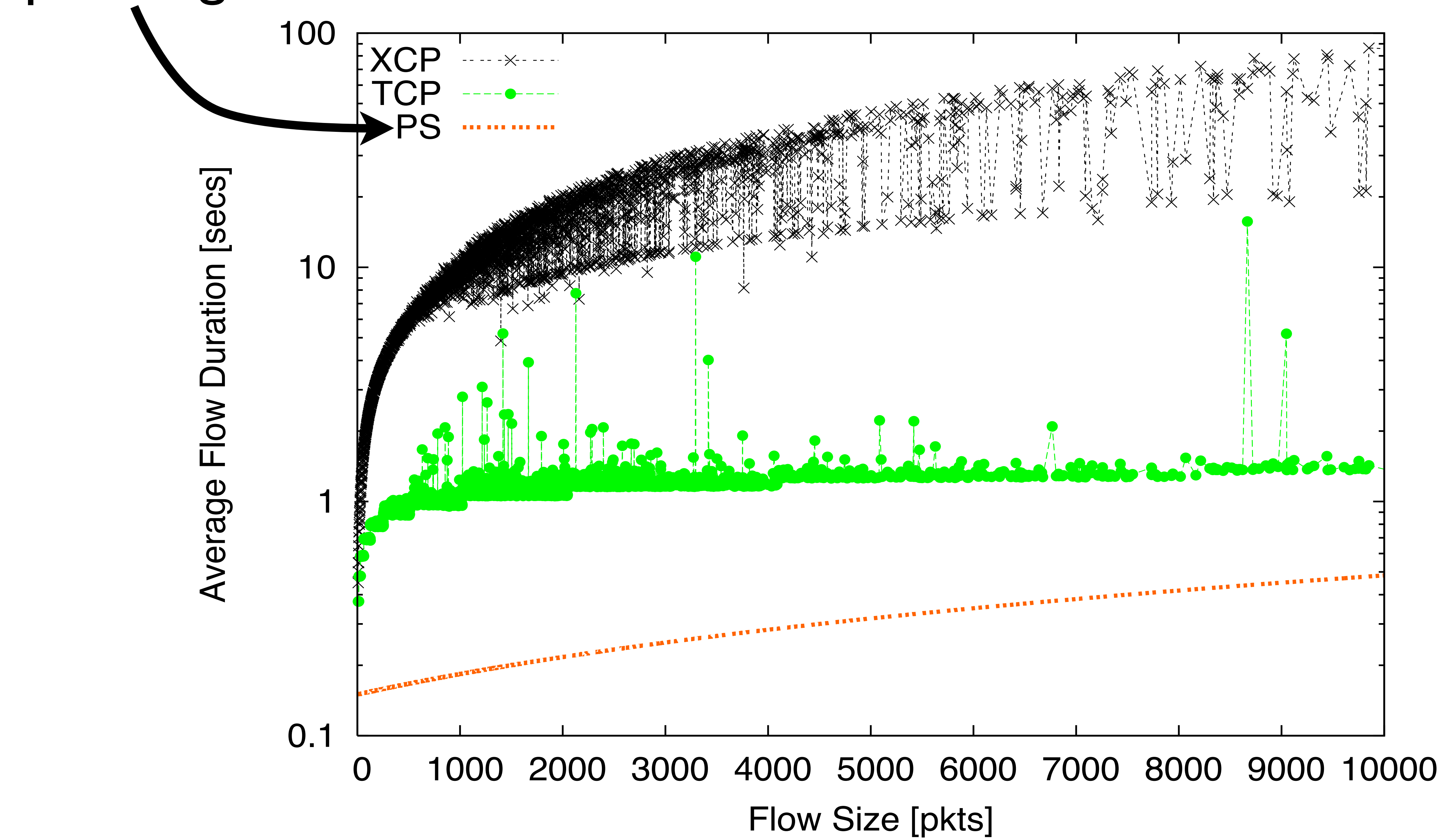
But what does the source really want?

- Just tell me the right rate, already!
- eXplicit Control Protocol (XCP)
- Rate Control Protocol (RCP)

Flows finish slowly

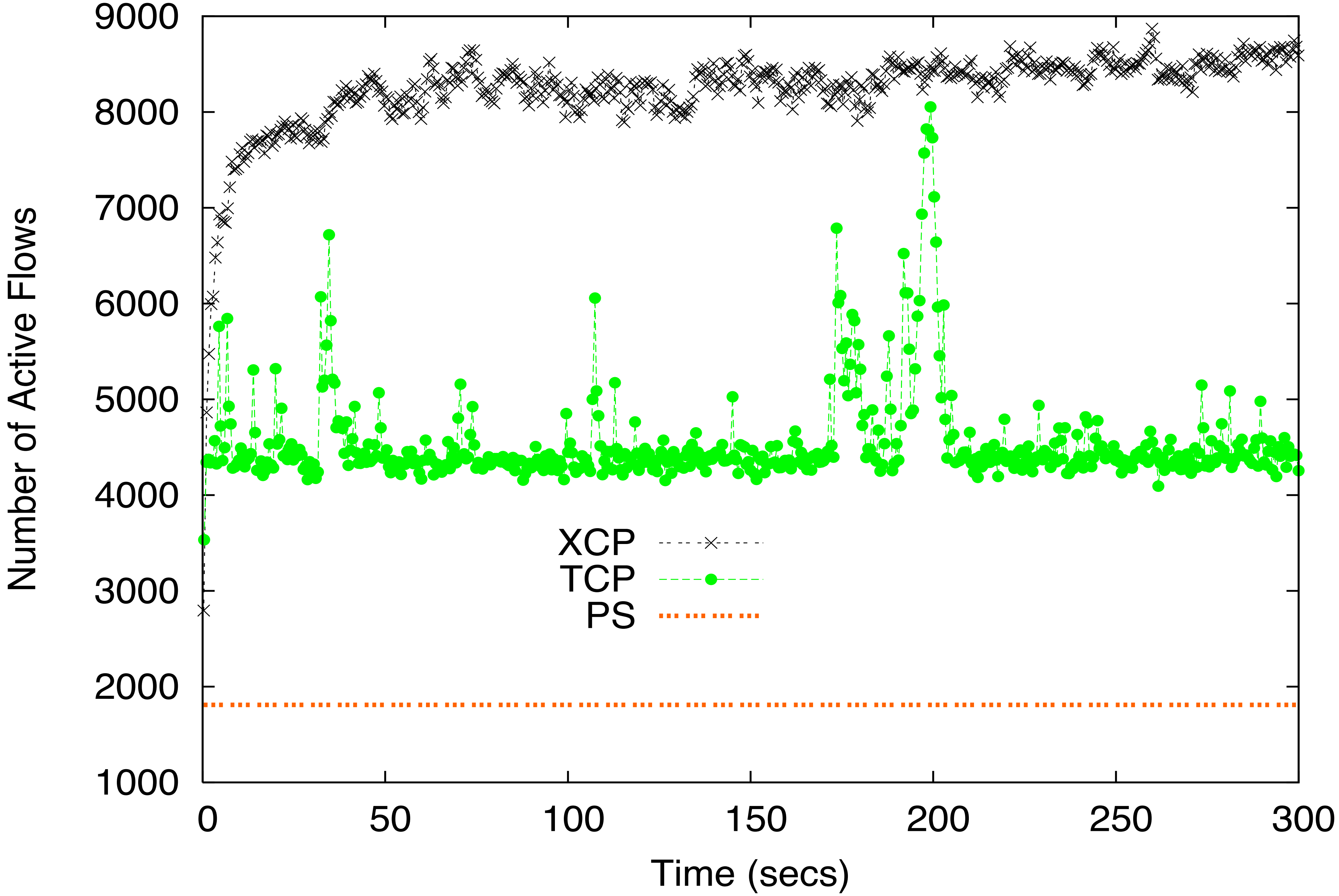
= fair queueing

[Dukkipati & McKeown '05]



Many flows waiting

[Dukkipati & McKeown '05]



RCP: finishing flows quickly

Rate Control Protocol [Dukkipati, Kobayashi, Zhang-Shen, McKeown, IWQoS 2005]

Router's algorithm:

- Compute fair per-flow rate R(t) at time t as whatever will fill up the link capacity (roughly)
- Tell end-hosts about this by putting the value in packets, and recompute every RTT

RCP rate computation

$$R(t) = R(t - d_0) + \frac{\alpha\big(C - y(t)\big) - \beta \frac{q(t)}{d_0}}{\hat{N}(t)}$$

old rate — $R(t - d_0)$

spare capacity — $C - y(t)$

queue size — $q(t)$

estimated # of flows — $\hat{N}(t)$

Simpler than XCP:

- rates instead of windows
- thus, feedback doesn't depend on a flow's RTT
- thus, same feedback to everyone

(How can you estimate # flows?)

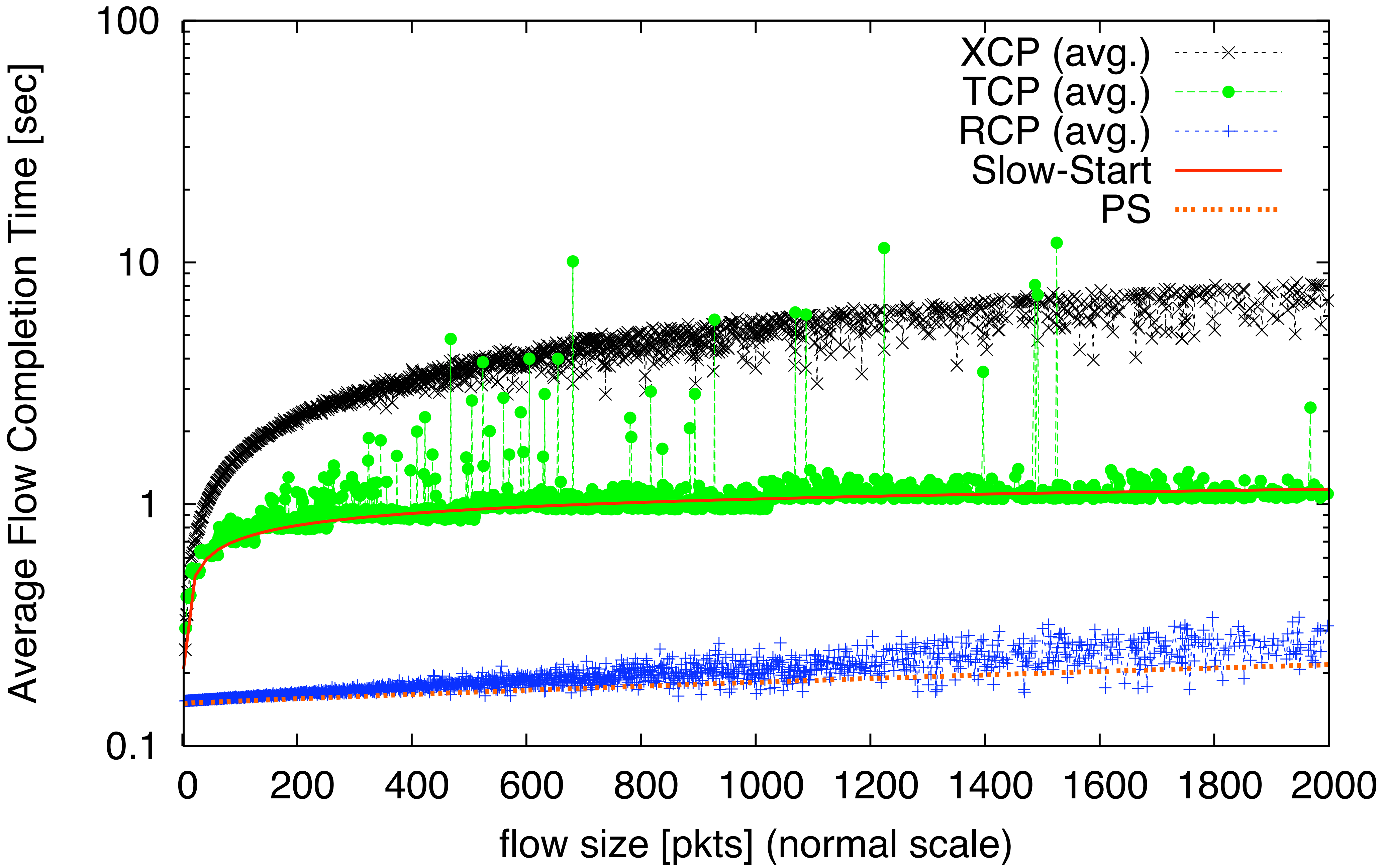
Estimating the number of flows

$$\hat{N}(t) = \frac{C}{R(t - d_0)}$$

If guess is wrong, what happens?

- Queue builds up; will reduce rate in next round
- Possibly this estimator could be improved

RCP finishes flows quickly



Enforcing fairness and isolation

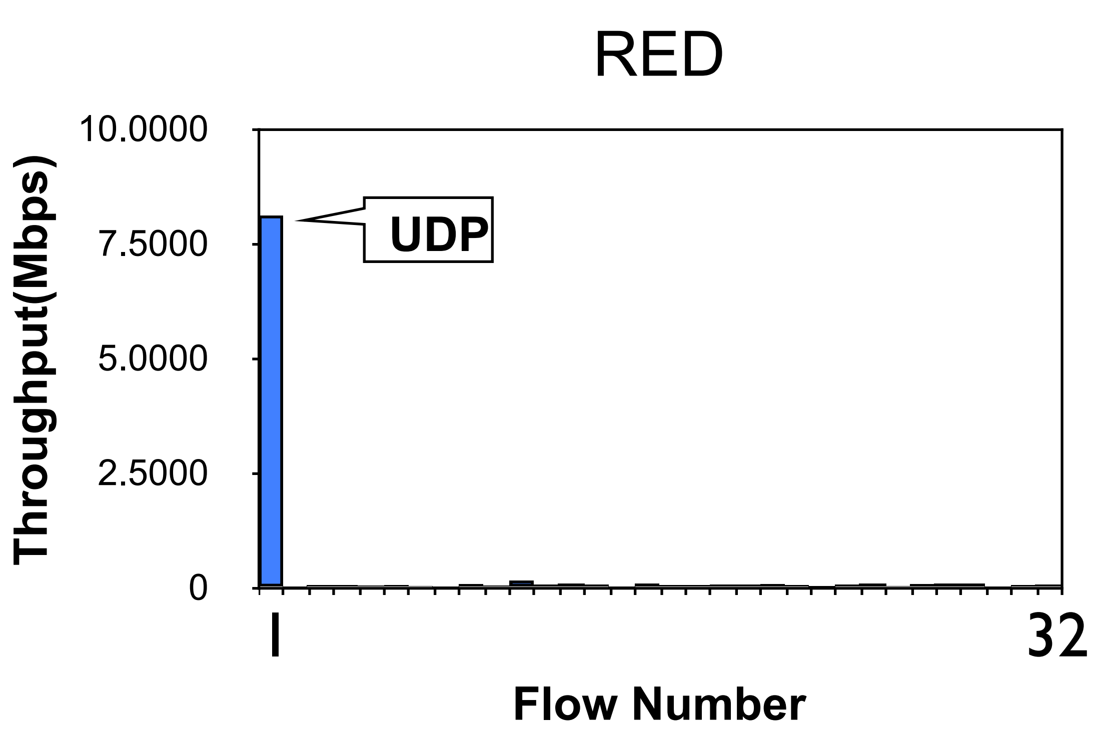
Based on slides by Ion Stoica

Problem: no isolation across flows

Assume router uses First In First Out (FIFO) queue

No protection: if a flow misbehaves it will hurt the other flows

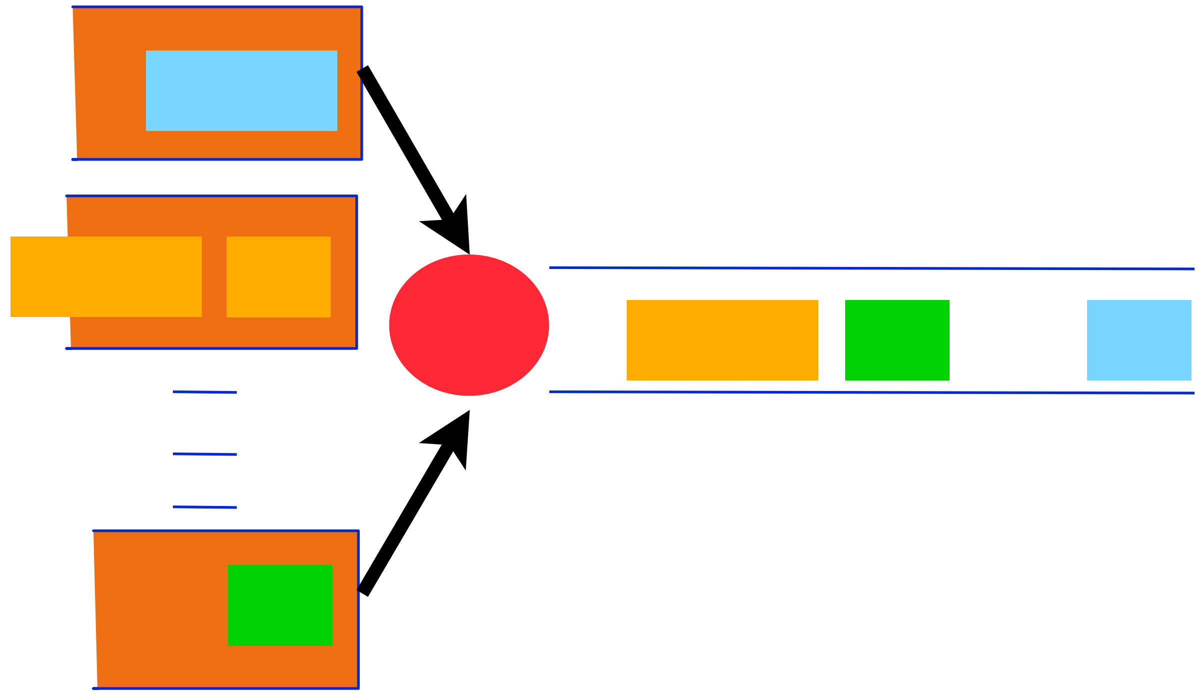
Example: 1 UDP (10 Mbps) and 31 TCP's sharing a 10 Mbps link



A first solution

Round robin among different flows [Nagle '87]

- One queue per flow
- while (1) { send one packet from each queue }



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

Fair Queueing (FQ) [DKS'89]

Define a fluid flow system: a system in which flows are served continuously

- essentially, bit-by-bit round robin

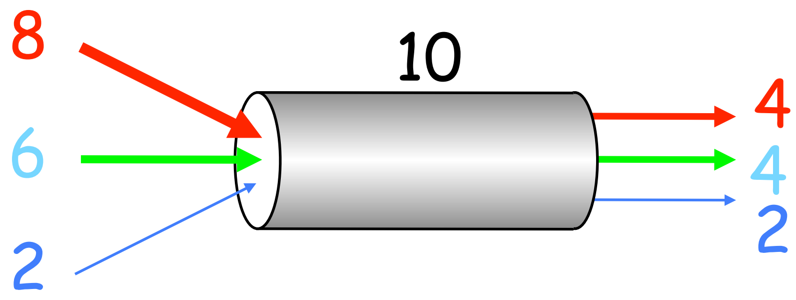
Advantages

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

Def'n 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

Implementing Fair Queueing

What we just saw was bit-by-bit round robin

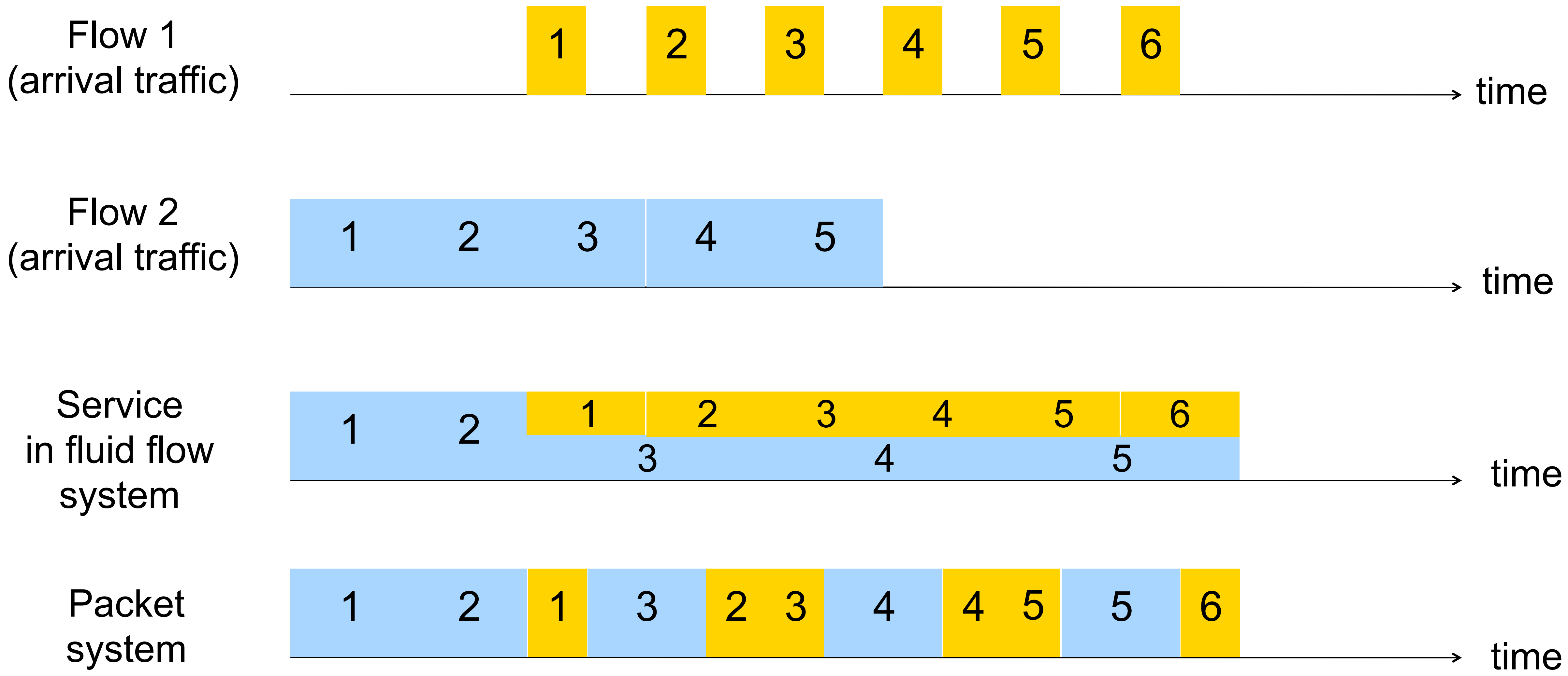
But can't interrupt transfer of a packet (why not?)

Idea: serve packets in the order in which they would have finished transmission in the fluid flow system

Strong guarantees: same as having a virtual link of the max-min fair capacity. Each flow gets:

- Exactly its max-min fair rate (+/- one packet size)
- Exactly its max-min fair per-packet delay (+/- one packet size) or better

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

Problem

Recall: "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 existing packets (perhaps all)!

Updating those times would be expensive

Solution: virtual time

Solution: Virtual Time

Key Observation: finish times may change when a new packet arrives, but the finish order doesn't

- Only the order is important for scheduling

Solution: maintain the number of rounds needed to send the remaining bits of the packet

- New packet arrival doesn't change # remaining rounds
- *Does* change rounds executed per unit time, but that's ok

System virtual time = index of the final round in the bit-by-bit round robin scheme

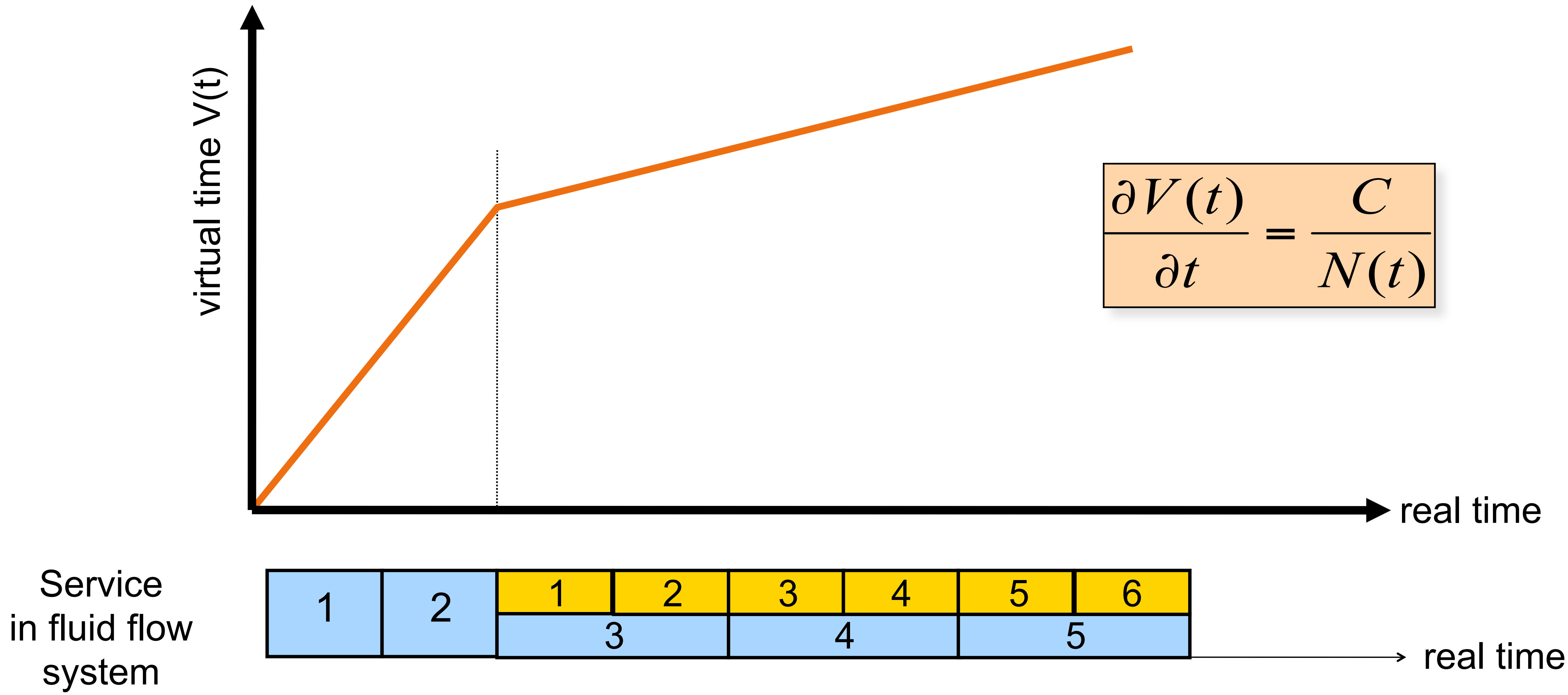
System Virtual Time: *V*(t)

Measure service, instead of time

Slope of V(t) = 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 to

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

FQ summary

FQ does not eliminate congestion; it just manages the congestion

Provides isolation between flows

- complete isolation?

Still need both end-host and router-based congestion control

- End-host congestion control to adapt rate
- Router congestion control to protect/isolate

Rethinking “fairness”: Congestion pricing

“*The Internet routes money; packets are just a side effect.*”

– Unknown, via Dave Clark

What is "fair"?

Flow rate equality!

Easily circumvented

Doesn't even optimize for any metric of interest

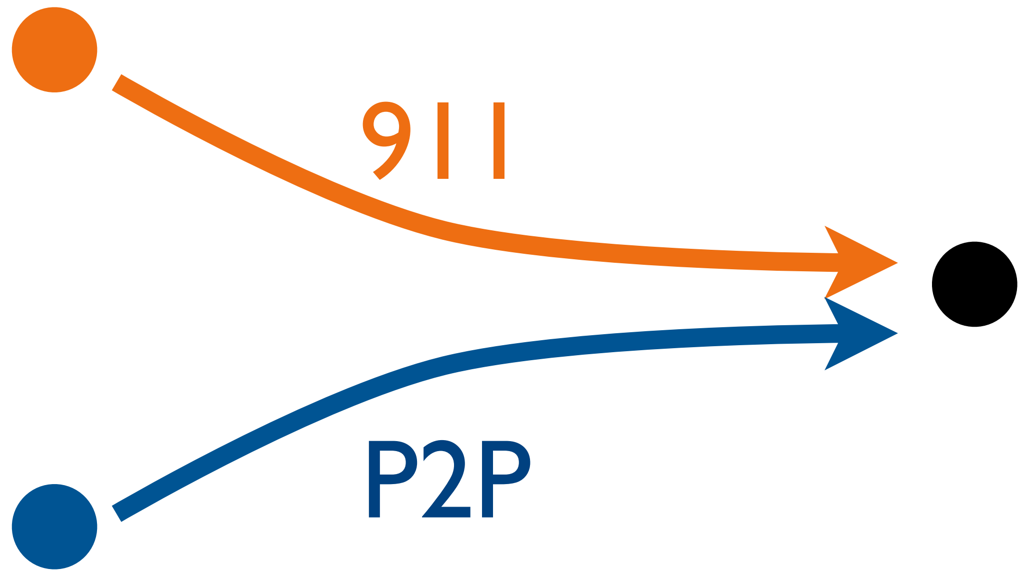


Fig. 1: Poppycock.

Fairness for real life resources

Plentiful: use as much as you want

- air
- advisor’s grant money

Scarce: pay for what you want

- price set by market
- result (under assumptions): socially optimal allocation

Fig. 2: Invisible hand of the market.

Briscoe's main points

Flow rate fairness (FRF) is not useful

Cost fairness is useful

Fig 3: Briscoe.

Flow rate fairness is hard to enforce

Cost fairness is feasible to enforce

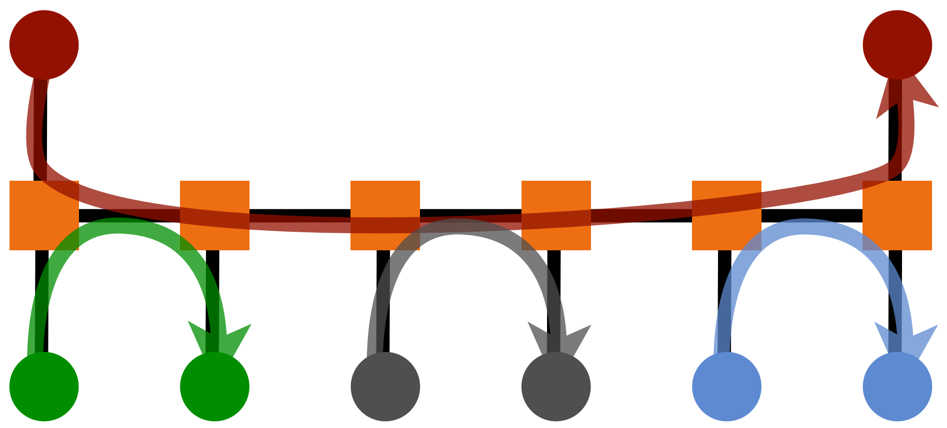
FRF not useful

Doesn't equalize benefits

- e.g., SMS message vs. a packet of a video stream

Doesn't equalize costs

- e.g., "parking lot" network: long flow causes significant congestion but is given equal rate by fair queueing



Therefore, doesn't equalize cost or benefit

FRF not useful

Myopic: no notion of fairness across time

In summary, FRF does not optimize utility

- except for strange definitions of utility...

So, even cooperating entities should not use it!

Cost fairness is useful

Economic entities pay for the costs they incur

- This is "fair" (in a real-world sense), not "equal"—and that's fine

In other words, networks charge packets for the congestion they cause

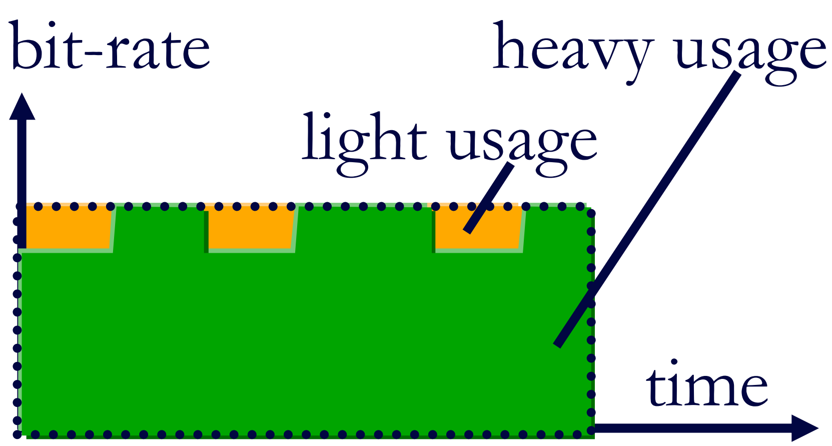
- Can networks lie about congestion?
- Yes. So it's really a market price, not exactly congestion

Result: senders want to maximize utility

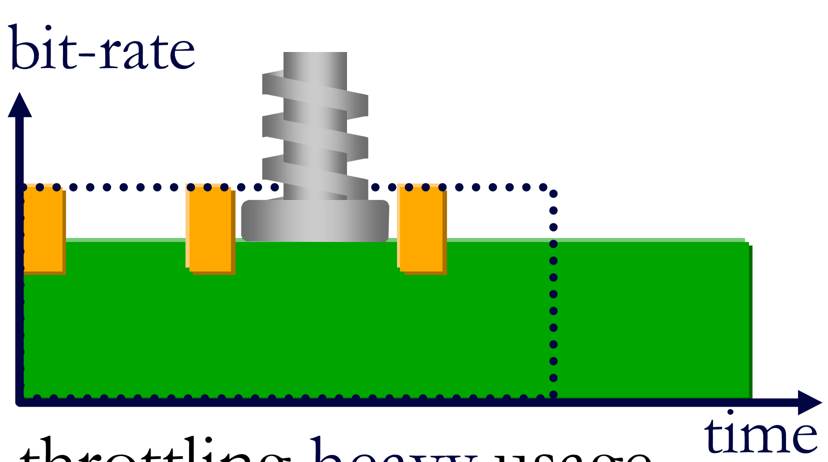
- Will balance benefit with cost (utility = benefit – cost)

Example: light & heavy traffic

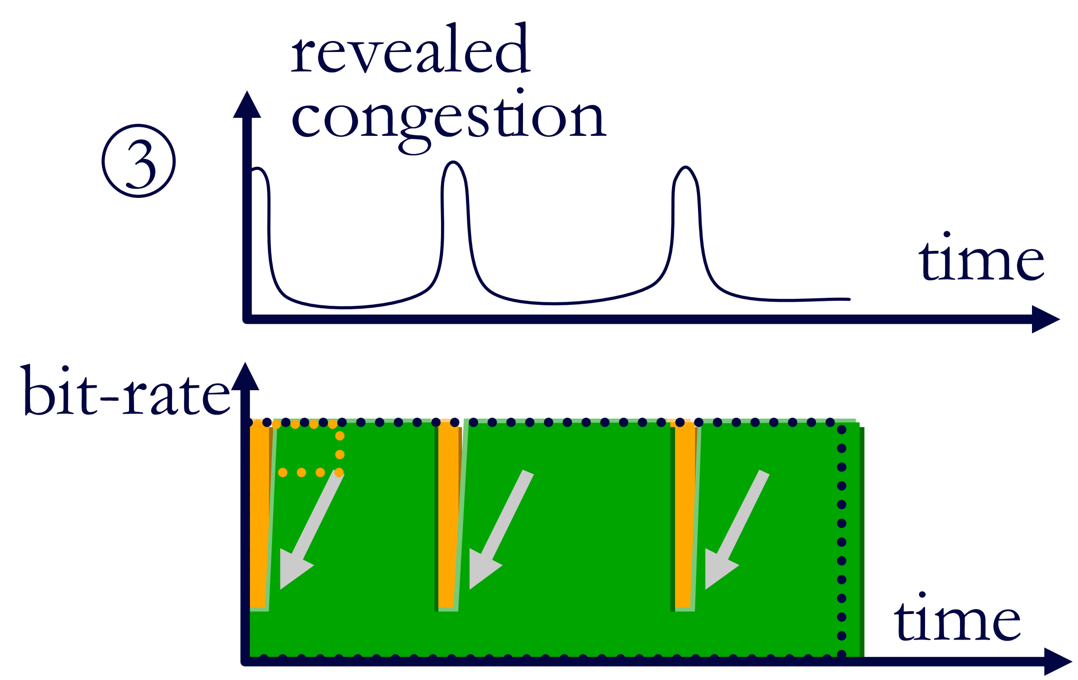
[Briscoe 2009]



'unfair' TCP sharing



throttling heavy usage



weighted TCP sharing

Key point: Benefit per bit is high for light flow and low for heavy flow.

CF is provably useful

Frank Kelly 1997: Cost fairness maximizes aggregate utility

i.e.: any different outcome results in suboptimal utility

Why won't anyone
listen to Kelly? Hello??! ... where
did everybody go?

Kelly's model (one congested link)

Each user i has utility $U_i(r_i)$ for rate r_i

Each user i pays p_i for access to link (its own choice)

Link sets price per unit bandwidth: $p = (\text{Sum } p_j) / C$

- thus, $r_i = p_i / p = C\, p_i / (\text{Sum } p_j)$

Theorem: assuming U_i concave, strictly increasing, and continuously differentiable, then

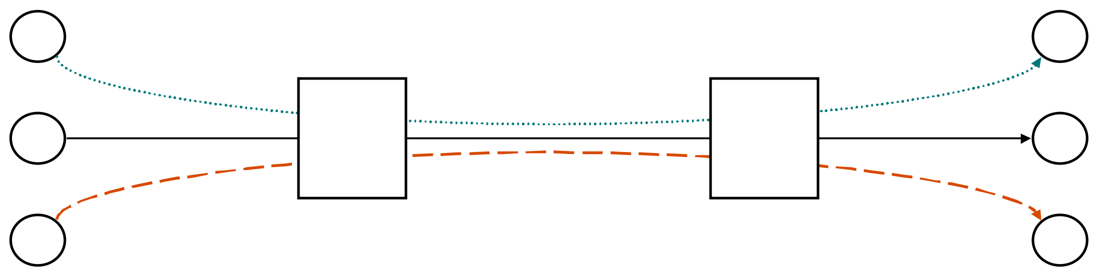
- A competitive equilibrium exists: setting of p_is in which no user can improve their utility given current price
- This equilibrium maximizes Sum $U_i(r_i)$

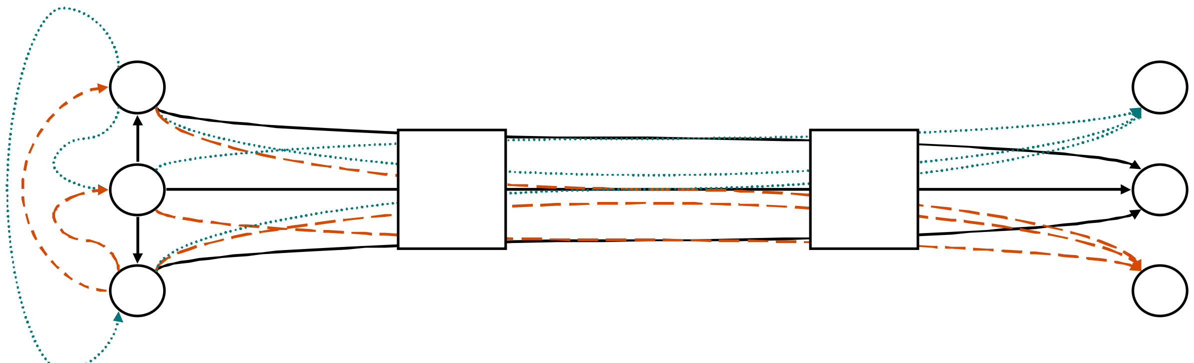
FRF is hard to enforce

Run your flow longer

Create more flows (similar to sybil attack)

- Multiple TCP connections between same source/ destination (web browsers)
- Spoof source IP / MAC address
- Multiple flows to other destinations (BitTorrent)





Cost fairness is enforceable

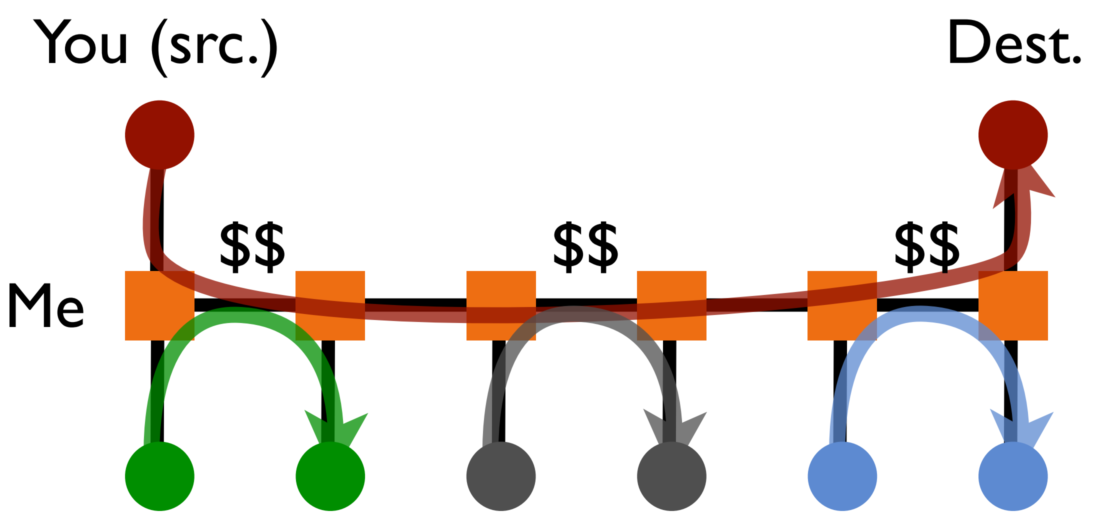
You send me a packet; I handle delivery and charge you for it

How much do I charge?

- Depends on cost on entire remainder of path!

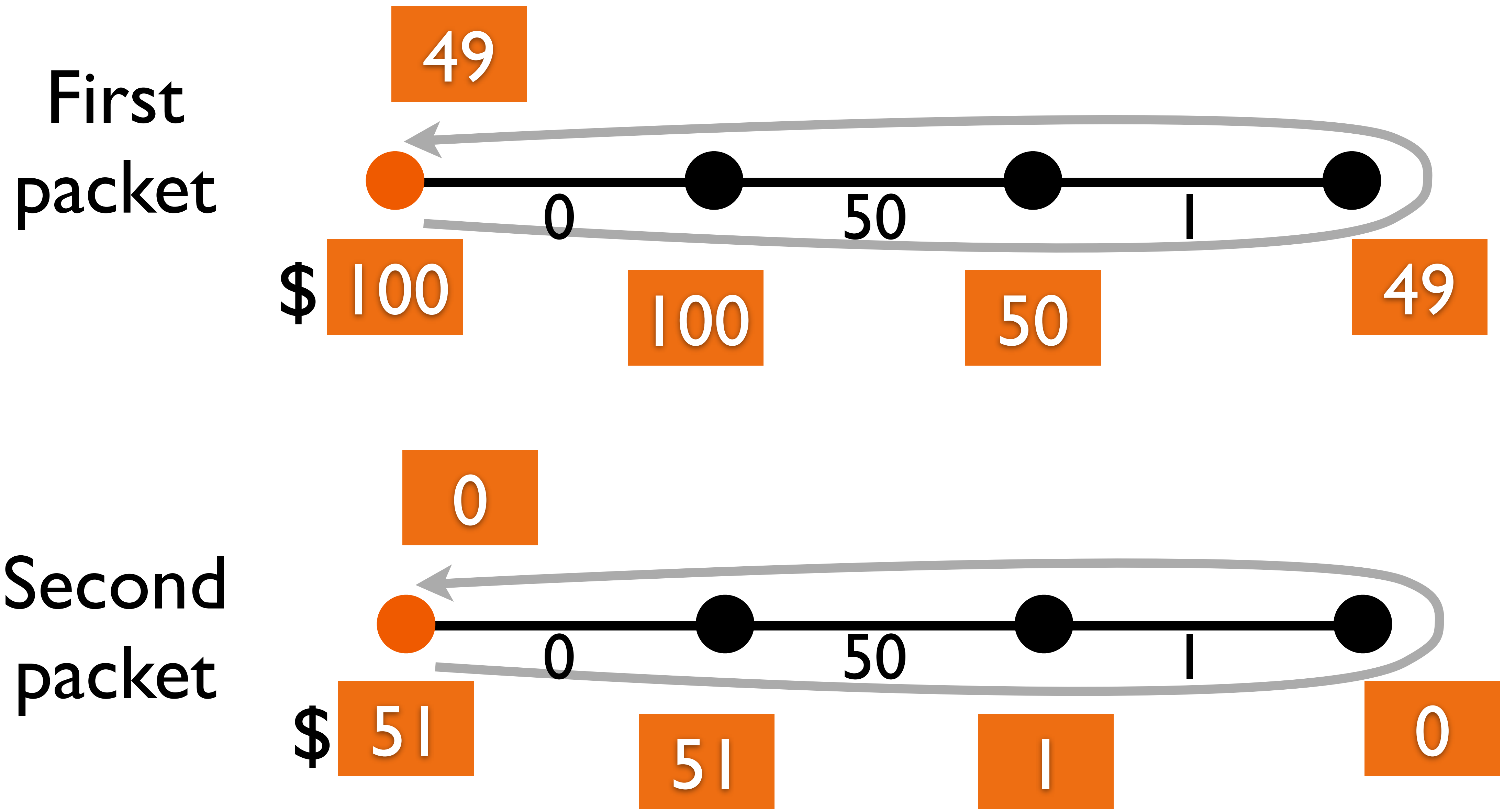
Not the only way of arranging payments, but it is convenient

- payments are between neighbors that already have an economic relationship



Mechanism: Re-Feedback

Key property: every hop knows total congestion along downstream path



Not necessarily about $$

Previous explanation was in terms of money, but doesn't have to directly involve money

- Re-feedback is a mechanism
- Doesn't imply a particular way of implementing congestion pricing

Possible variants of congestion pricing

- pay per packet?
- monthly allowance?
- only at edges?
- between all ISPs?

Discussion: What if...

Host running a persistent "light" job is interrupted by heavy flows congesting the net?

Host is compromised? (botnet) Who pays?

If we want cost fairness, is Weighted Fair Queueing useless?

- No: provides mechanism to isolate flows, virtualize links
- e.g., could use congestion pricing to set WFQ's weights

Conclusion (Briscoe style!)

"It just isn't realistic to create a system the size of the Internet and define fairness within the system without reference to fairness outside the system."

Cost fairness optimizes aggregate utility and is feasible to enforce

Flow rate fairness does not optimize utility and is not feasible to enforce

- Cease publication on the topic and stop teaching it in undergraduate courses

Announcements

Announcements

Assignment 1 was due 2pm today

- Accepting late submissions (-15%) till 2pm Wed
- No credit thereafter

A bunch more project ideas

- To be released late tonight (see Piazza)

Next reading: Forwarding hardware