# Finishing Flows Quickly with Preemptive Scheduling


Chi-Yao Hong

Matthew Caesar

Brighten Godfrey

University of Illinois at Urbana-Champaign


(pptx version is available upon request)

# Goal

Low latency datacenter networks

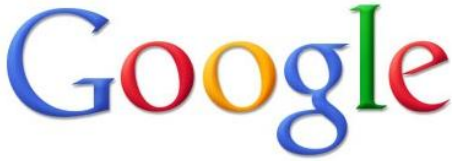
Why do we even care?


Revenue decreased by 1% of sales
for every 100 ms latency

[Speed matters; Greg Linden]


400 ms slowdown resulted in a traffic decrease of 9%

[Yslow 2.0; Stoyan Stefanov]




100 ms slowdown reduces # searches by 0.2-0.4%

[Speed matters for Google Web Search; Jake Brutlag]


Users with lowest 10% latency viewed 50% more pages than those with highest 10% latency

[The secret weapons of the AOL optimization team; Dave Artz]

mozilla Firefox

2.2 sec faster web response increases 60 million more Firefox install package downloads per year

[Firefox and Page Load Speed; Blake Cutler]


Users with 0-1 sec load time have 2x conversion rate of 1-2 sec

[Is page performance a factor of site conversion? And how big is it; Walmart Labs]

# Goal

Low latency datacenter networks


users really
respond to latency!

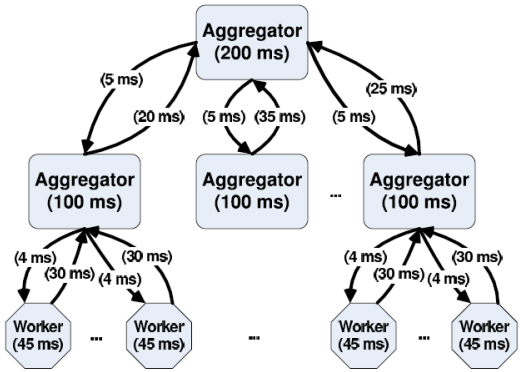
# Low latency datacenter networks

## Finish flows quickly

Users always want to get results earlier

## Meet flow deadline

User-facing web applications have latency target




Partition-aggregate structure:
web search, recommendation systems, MapReduce/Dryad, social networks

D3: [Wilson, Ballani, Karagiannis, Rowstron; SIGCOMM'11]

# Today's options for datacenter transport protocols

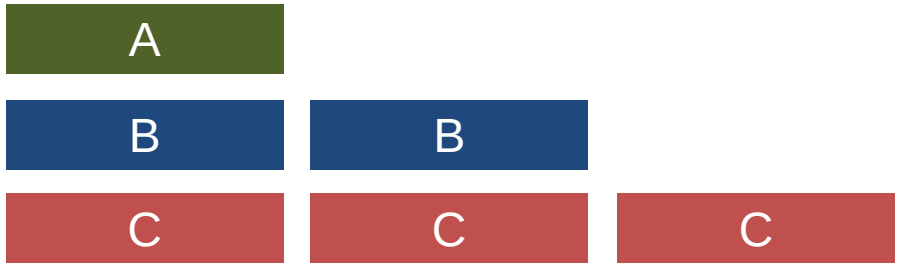
TCP ICTCP DCTCP XCP

| Feature | Help reduce flow completion time? |
|---|---|
| High throughput & utilization | YES |
| Small queues & drops | YES |
| Fairness | NO! |

In fact, fairness damages
flow completion time

# Example

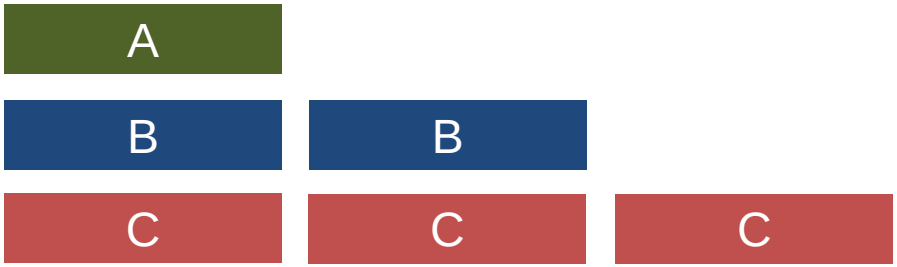
| | Size |
|---|---|
| Flow A | 1 |
| Flow B | 2 |
| Flow C | 3 |




arrive at the same time

share the same bottleneck link

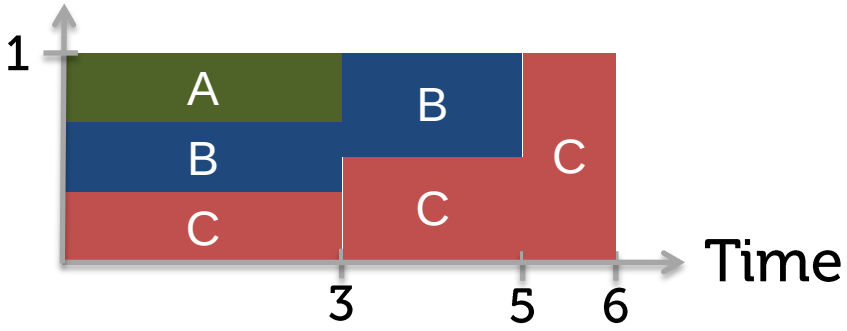
# Example: Fair sharing




Fair sharing:
3, 5, 6
mean: 4.67

far from optimal!

Throughput



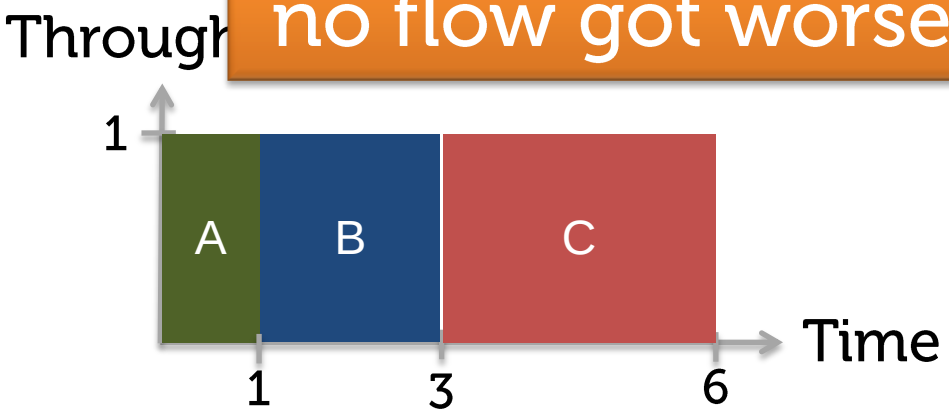

$$\text{mean flow completion time} = \frac{3+5+6}{3} = 4.67$$

# Example: Relaxing fairness constraint

## (A → B → C)


29% saving in mean

It's still fair:
no flow got worse!



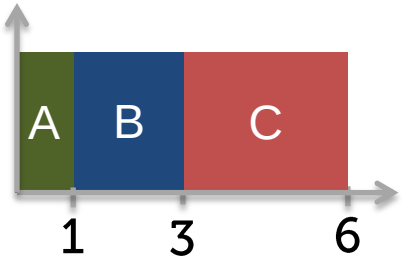

$$\text{mean flow completion time} = \frac{1+3+6}{3} = 3.33$$

Fair sharing:
3, 5, 6
mean: 4.67

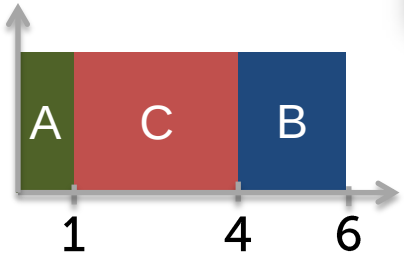
(A → B → C):
1, 3, 6
mean: 3.33

# Order matters

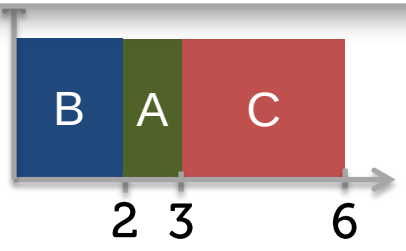
Fair sharing:
3, 5, 6
mean: 4.67



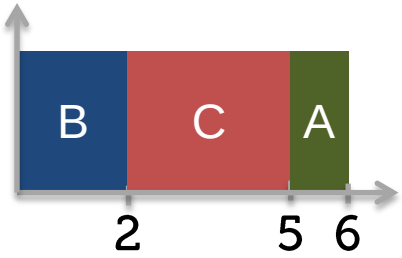

mean: 3.33



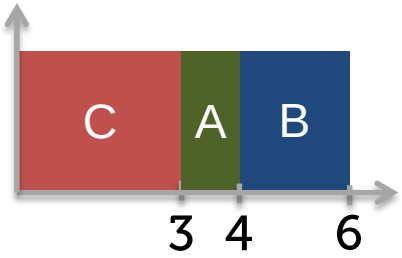

mean: 3.67



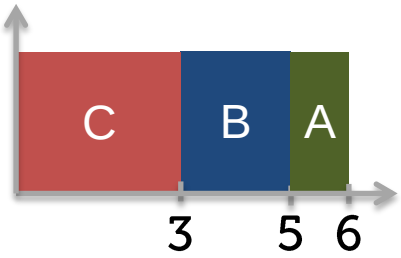

mean: 3.67




mean: 4.33




mean: 4.33




mean: 4.67

Relaxing fairness helps

Order matters!

# Our solution

Forget about fairness –
let's relax fairness constraints!

# PDQ

## Preemptive Distributed Quick flow scheduling

Pretty Damn Quick!

Chi-Yao Hong, Matthew Caesar, Brighten Godfrey University of Illinois at Urbana-Champaign

# PDQ: Idea


Scheduling flows based on flow criticality

//

relative priority of flows;
transmission order

# PDQ: Two primitives

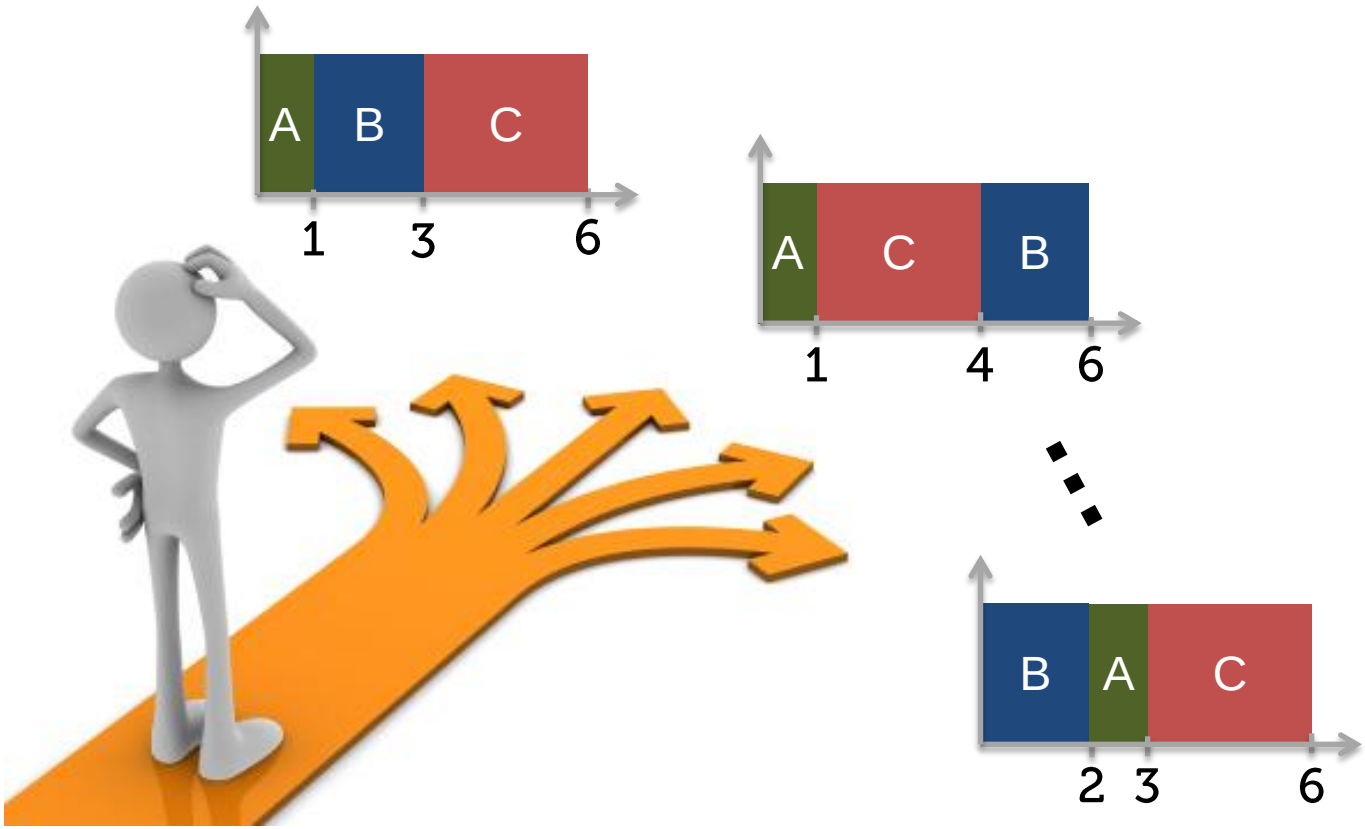
## Preemptive scheduling

Less-critical flows yield to critical flows


## Dynamic scheduling

Flow criticality may change over time

# How to choose flow criticality?



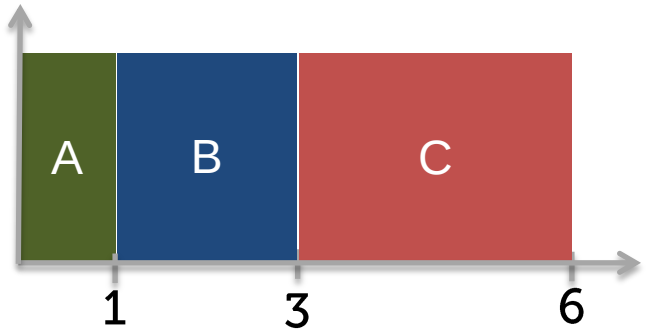
# How to choose flow criticality?

## scheduling discipline

choose

choose

choose



# PDQ's scheduling disciplines

## EDF (Earliest Deadline First)

Optimal for satisfying flow deadlines

## EDF + SJF

EDF if there's deadline. Give preference to deadline flows

## SJF (Shortest Job First)

Optimal for minimizing mean flow completion time

## Policy-based

Assignment that reflects business priority

**Any criticality-based scheduling**

OK, theoretically they are optimal
but in practice…

# Roadmap

## Challenges

- Computing schedule across large network
- Ensuring seamless flow switching

## Moving to actual datacenter settings

- Inaccurate flow information
- Fairness
- Scalability
- Failure Resilience
- Multipath
- …

# Challenge: Scheduling requires centralized computations

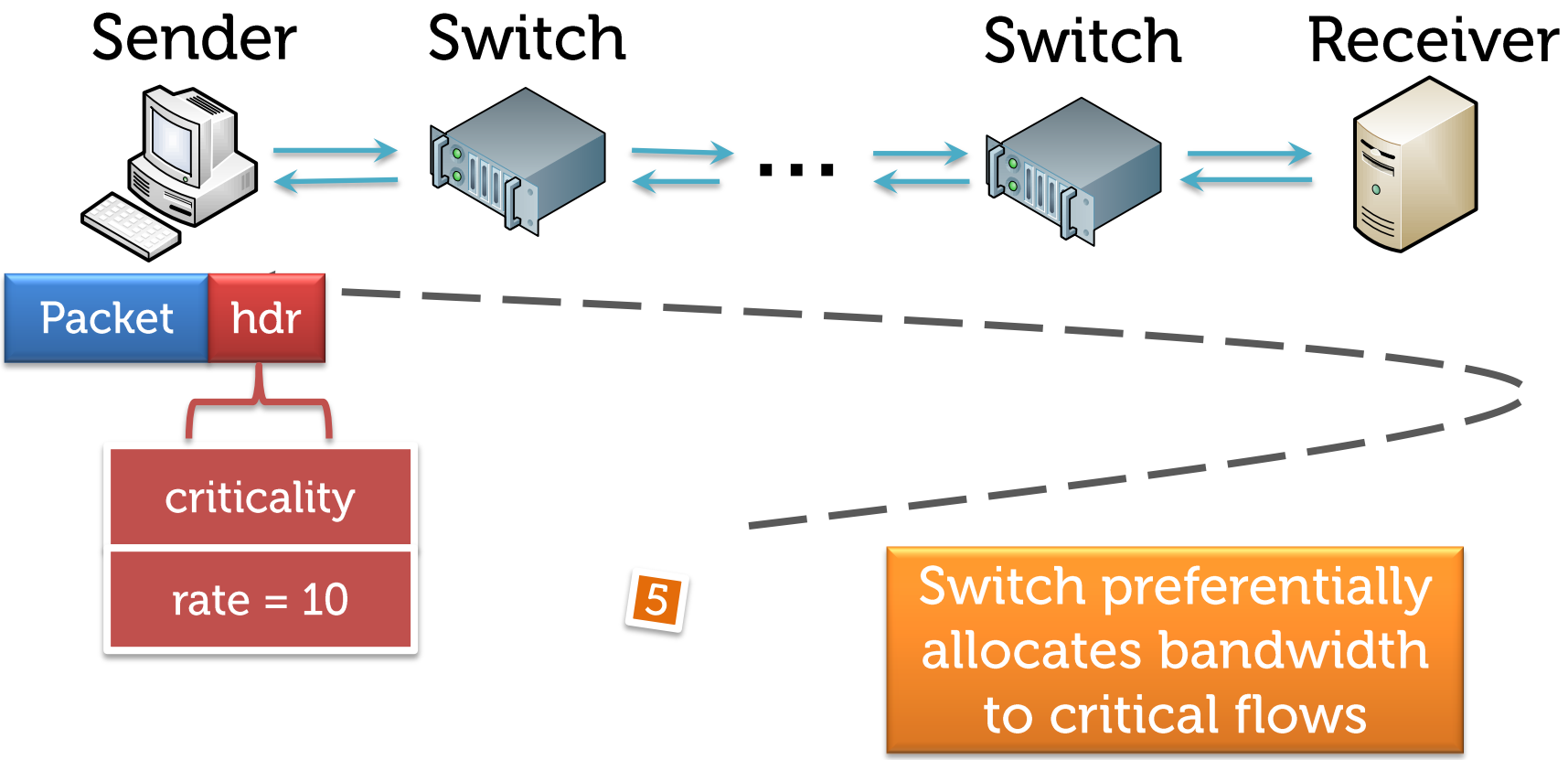
Centralized coordinator fails to scale

- Single point of failure
- Congestive hot-spot
- High flow initialization overhead *esp. for short flows*

Fully distributed solution

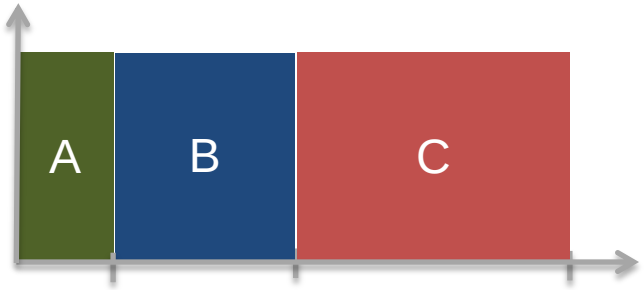
- PDQ switches collaboratively control flow schedule by tagging packet headers
- No deadlock
- Bounded convergence time
- Use only FIFO tail-drop queue
- Some computation required at the switch per flow

# PDQ: Fully distributed design

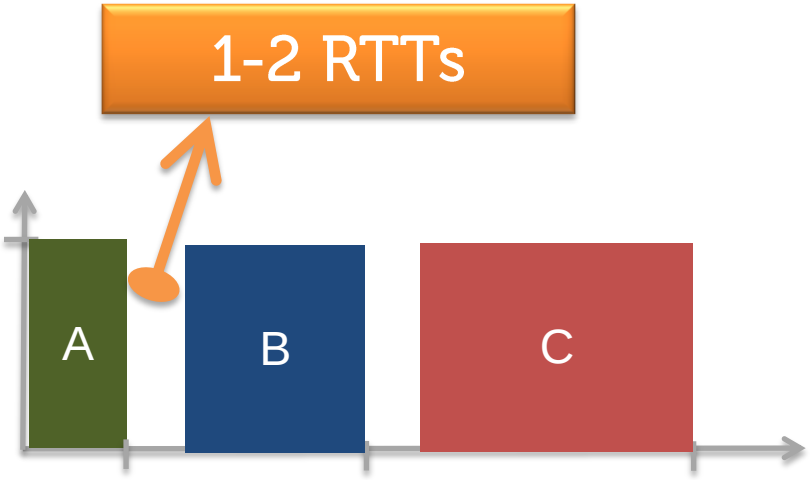


# Challenge:
# Low utilization during flow switching

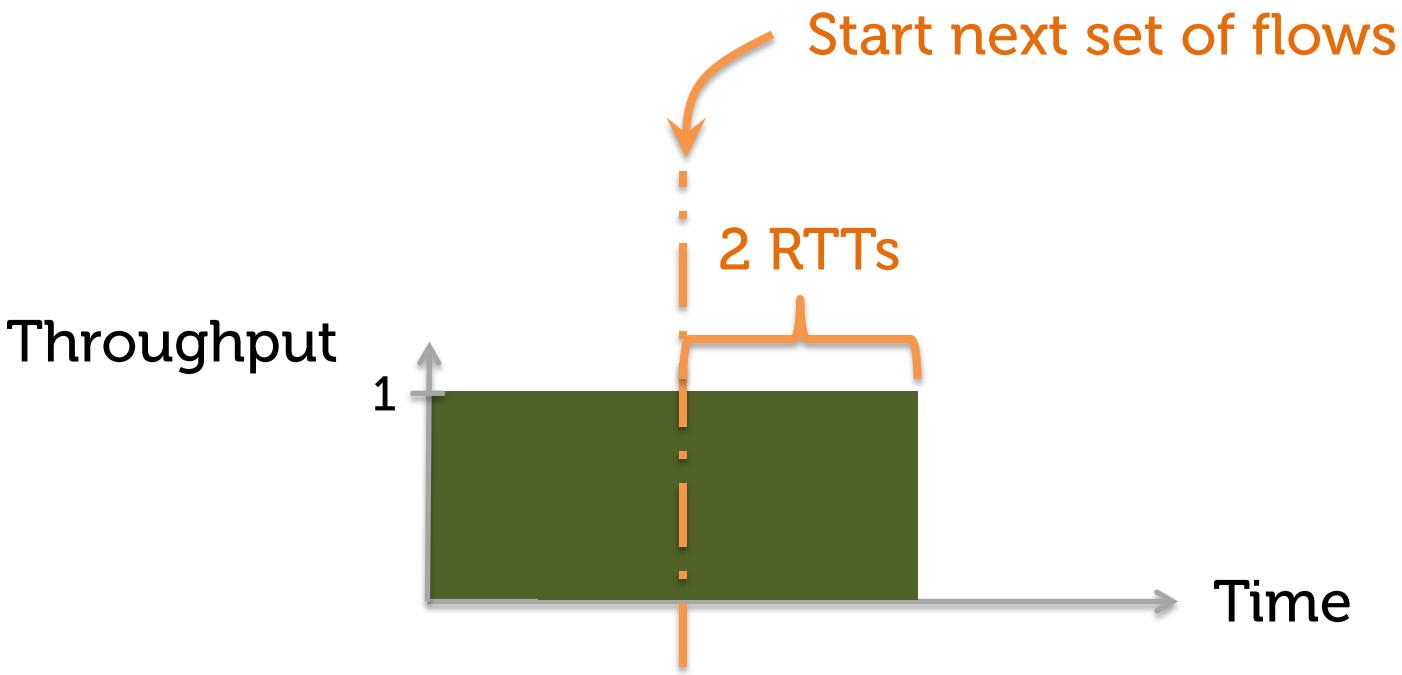
Goal:




Practice:

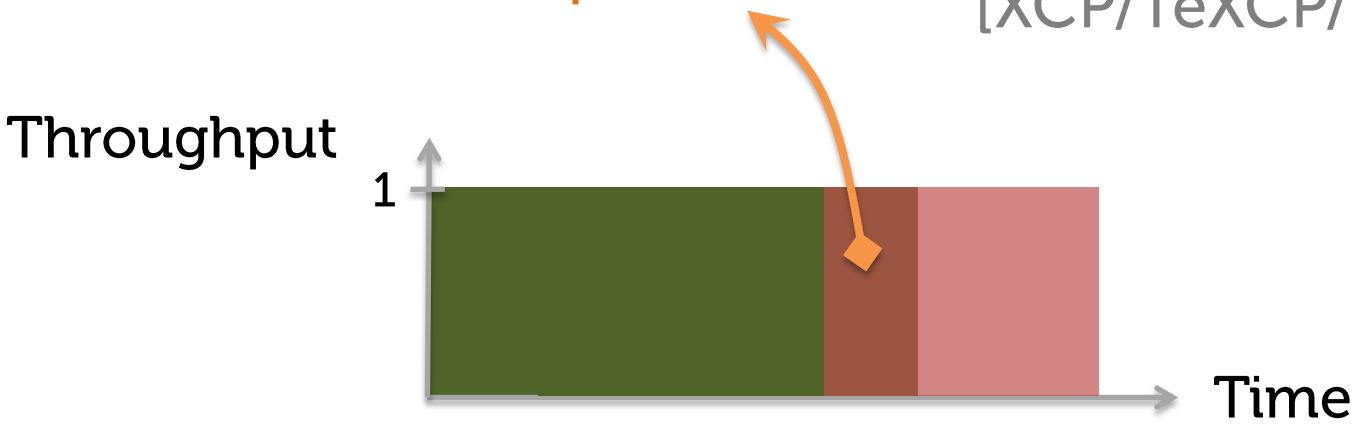


# Early Start: Seamless flow switching

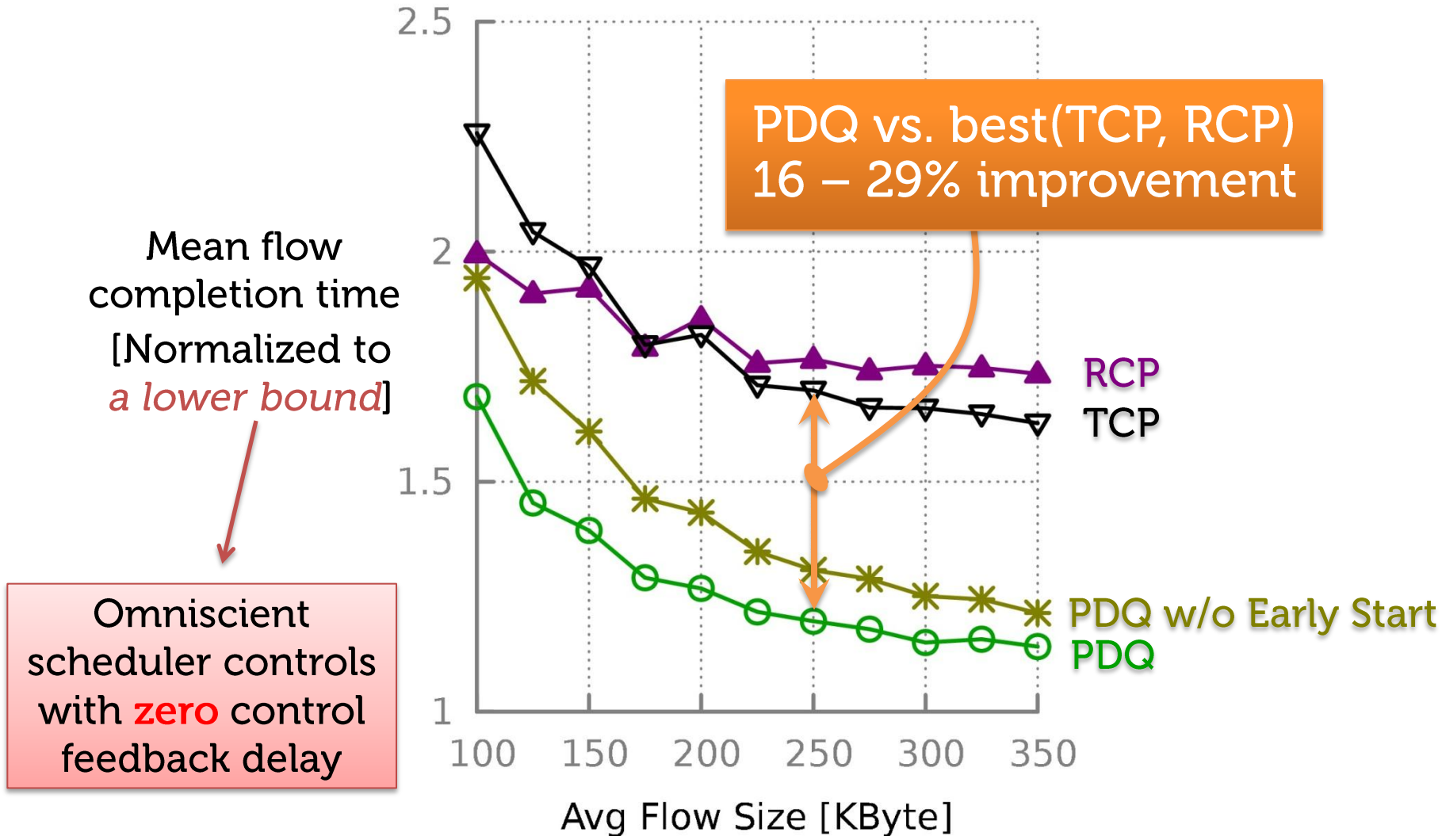


# Early Start: Seamless flow switching

Solution:
rate controller at switches
[XCP/TeXCP/D3]

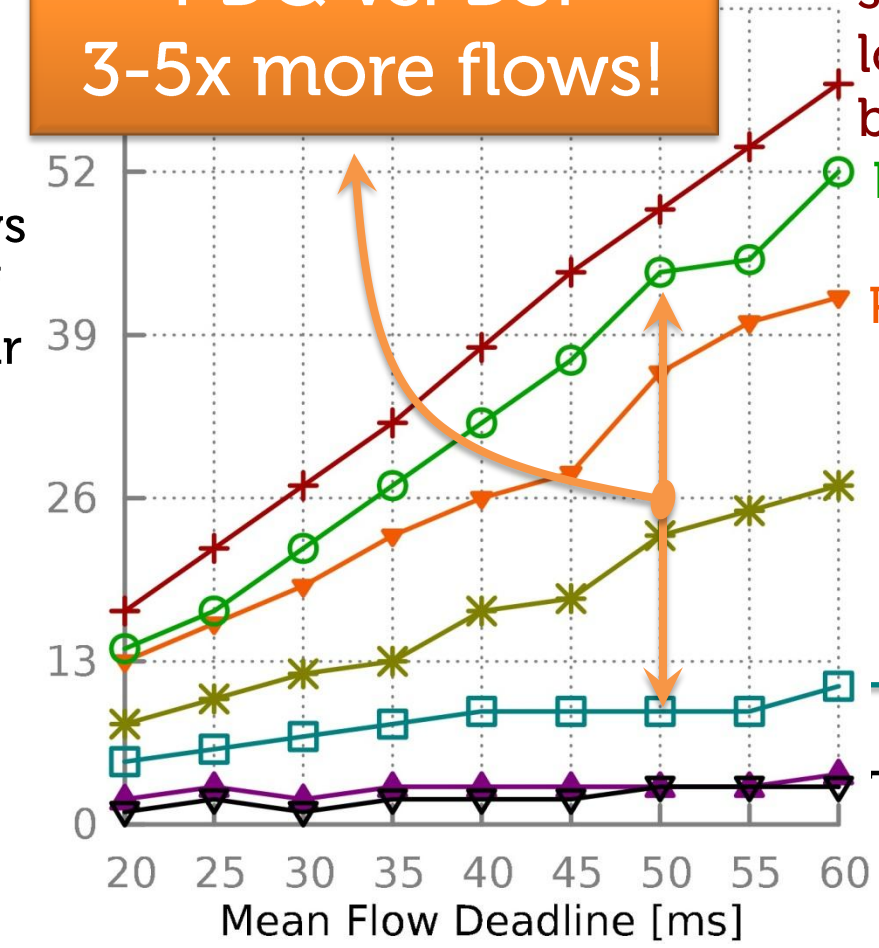


# Mean flow completion time: PDQ vs. Alternatives



# Number of "supported" deadline flows: PDQ vs. Alternatives

PDQ vs. D3:
3-5x more flows!




[Upper bound]: Omniscient scheduler (zero feedback loop delay); EDF + DP-based admission control

PDQ w/ Early Termination

PDQ

PDQ w/o Early Start

D3: [Wilson, Ballani, Karagiannis, Rowstron; SIGCOMM'11]

TCP & RCP


works well in a "nice" environment, but…

- Topology and Scalability: works well with scale?
- Sending Pattern: works well beyond query aggregation?
- Traffic Pattern: works well in real datacenter workloads?
- Resilience to error: what if packet gets lost or flow information is inaccurate?
- Multipath: does PDQ benefit from multipath forwarding?
- ✔ Flow size estimation: What if estimation fails?
- ✔ Fairness: can long flows starve?

**In all of the tested scenarios, PDQ provides significant advantages**

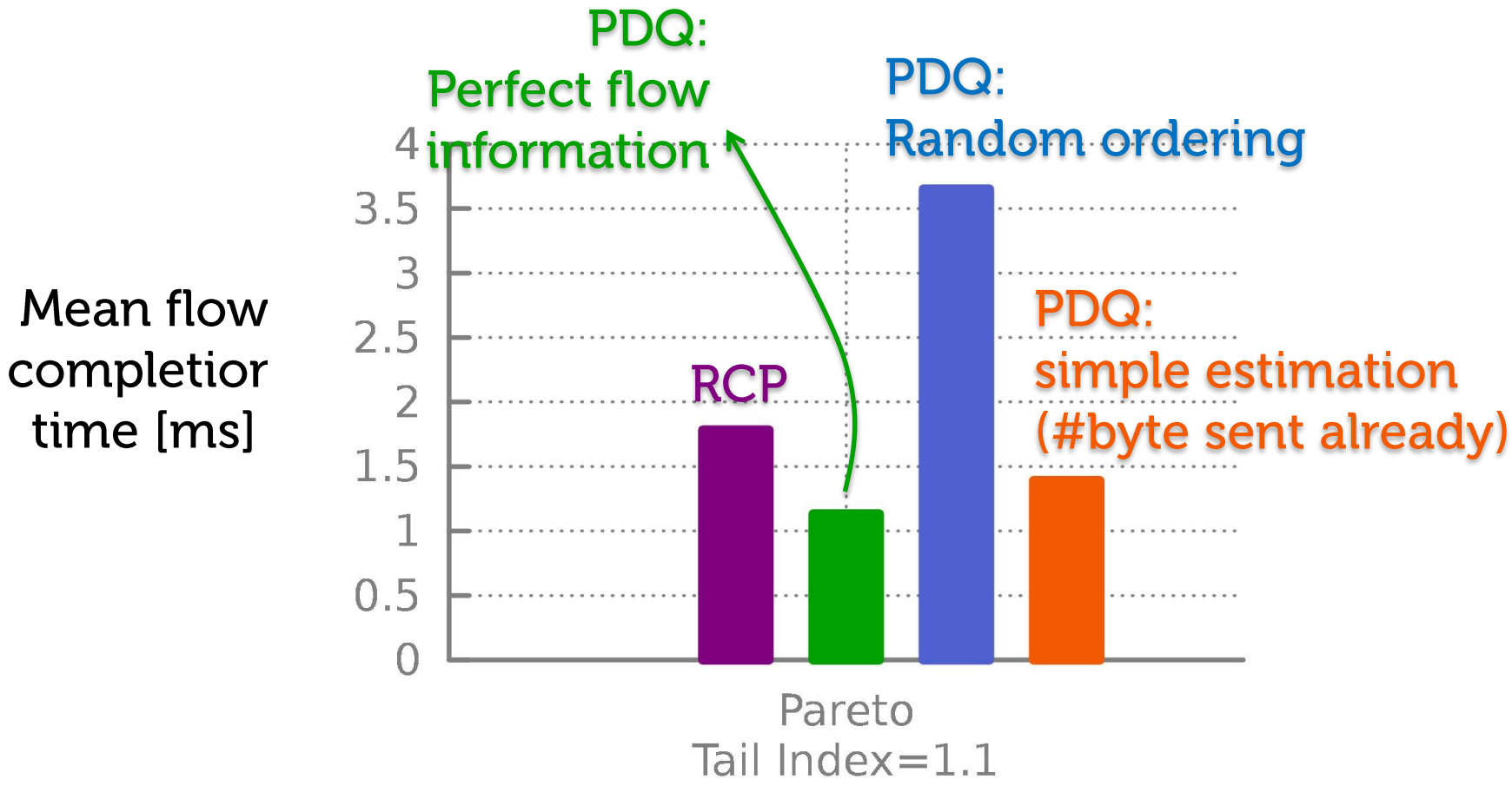
# Flow size information

For many data center applications, flow size can be precisely known at flow initialization time

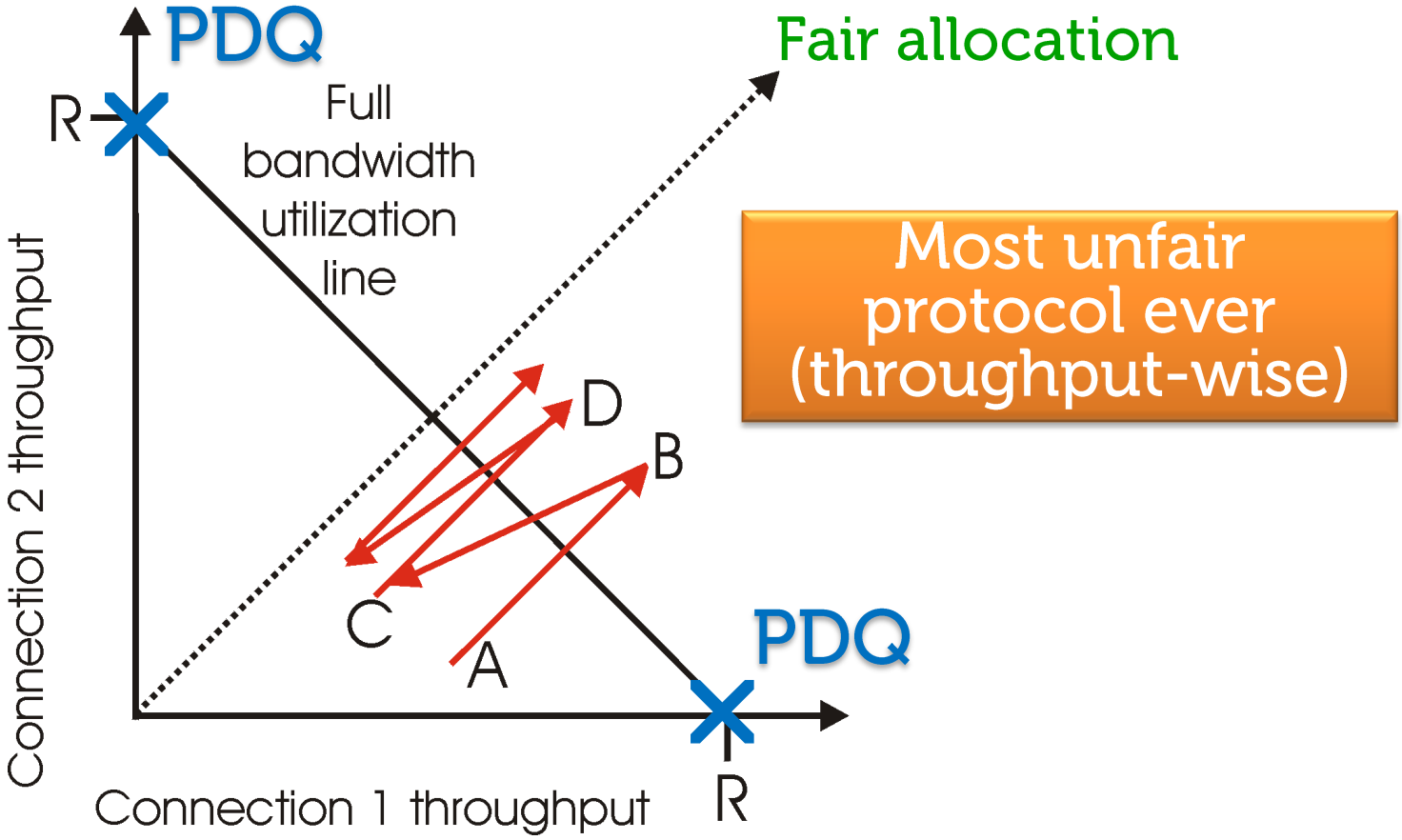
[Wilson, Ballani, Karagiannis, Rowston; SIGCOMM'11]

Otherwise…

# Simple flow size estimation works well



# Fairness




[Chiu and Jain; Computer Networks and ISDN Systems, 1989]

# ≥99% of jobs complete faster under SJF than under fair sharing

[Bansal, Harchol-Balter; SIGMETRICS'01]

Assumption: heavy-tailed flow distribution

# For datacenter topologies

We found that 85-95% of flows
complete faster under PDQ than under RCP

Topologies: Fat-tree, BCube, Jellyfish

# Shorten flow completion time tail

Aging: Increase criticality
based on waiting time

# Conclusion

## Finish flows faster

Improved mean FCT by 16-30% over TCP, RCP and $D^3$

## Satisfy flow deadline

3x as many flows as $D^3$ while meeting flow deadlines