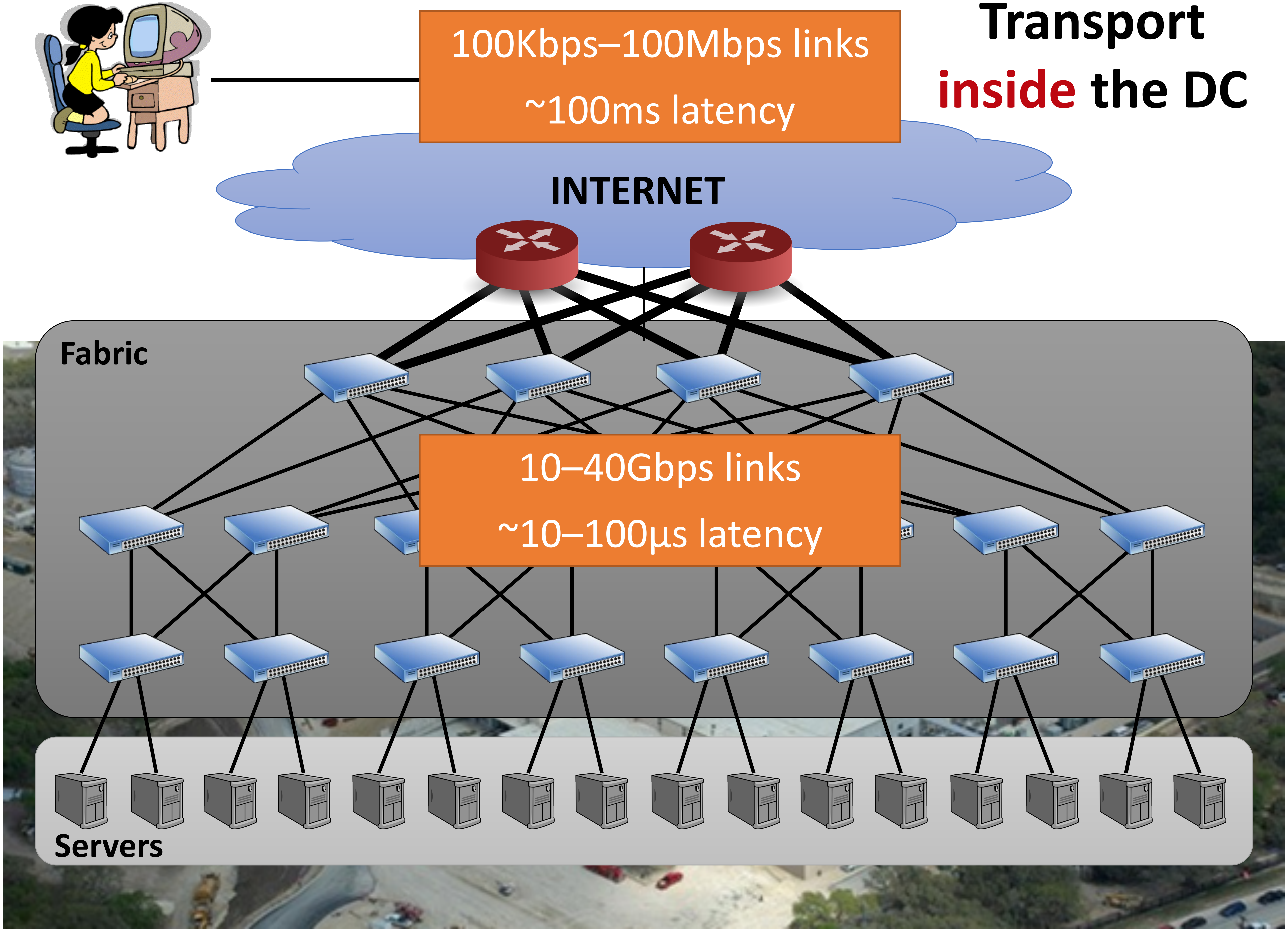
Congestion Control in Data Centers

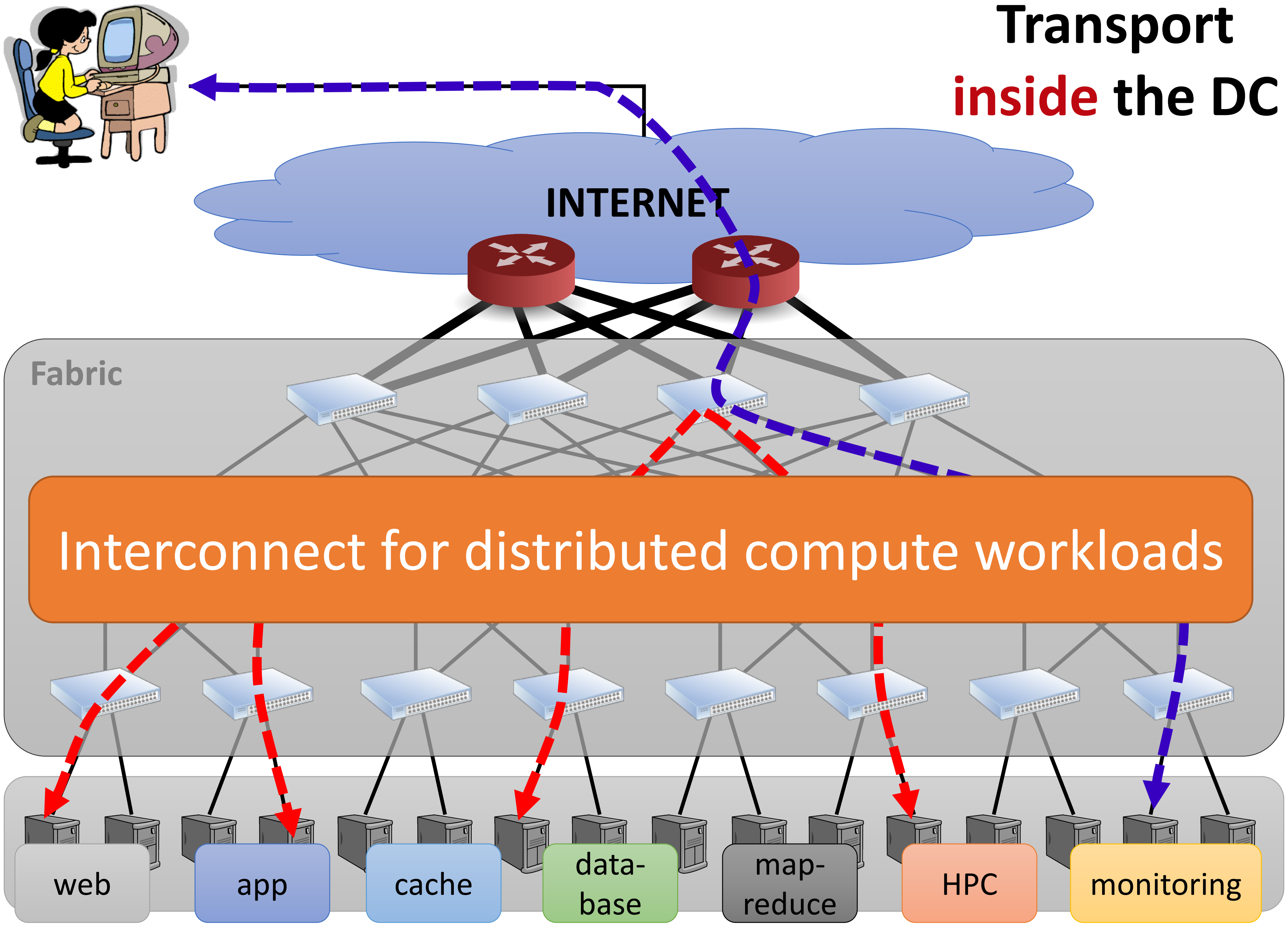
Lecture 16, Computer Networks (198:552)

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Transport inside the DC



Transport inside the DC



What’s different about DC transport?

- Network characteristics
  - Very high link speeds (Gb/s); very low latency (microseconds)
- Application characteristics
  - Large-scale distributed computation
- Challenging traffic patterns
  - Diverse mix of mice & elephants
  - Incast
- Cheap switches
  - Single-chip shared-memory devices; shallow buffers

Additional degrees of flexibility

- Flow priorities and deadlines
- Preemption and termination of flows
- Coordination with switches
- Packet header changes to propagate information

Data center workloads

- Mice and Elephants!

- Short messages
  (e.g., query, coordination)

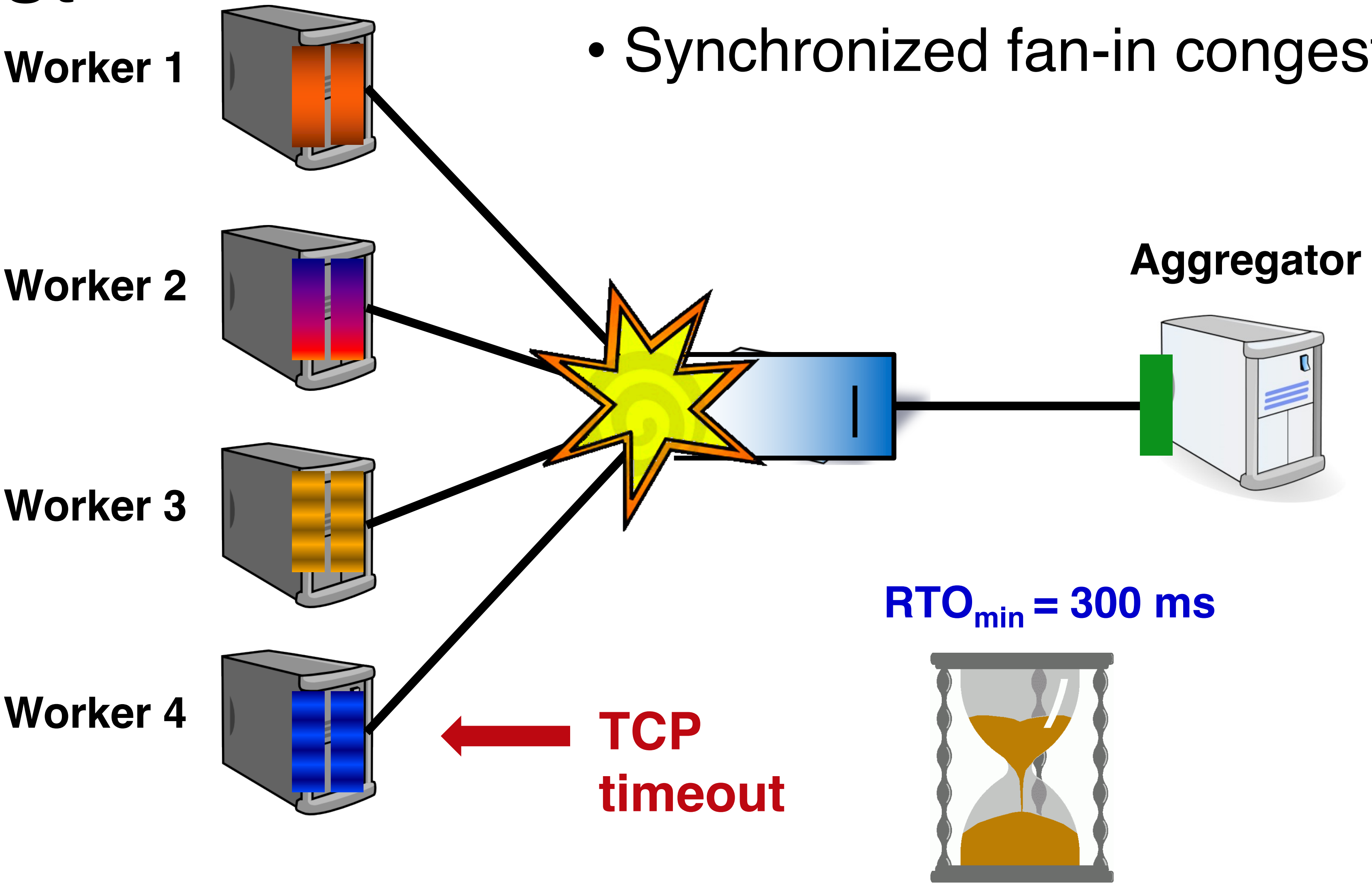
- Large flows
  (e.g., data update, backup)

Low Latency

High Throughput

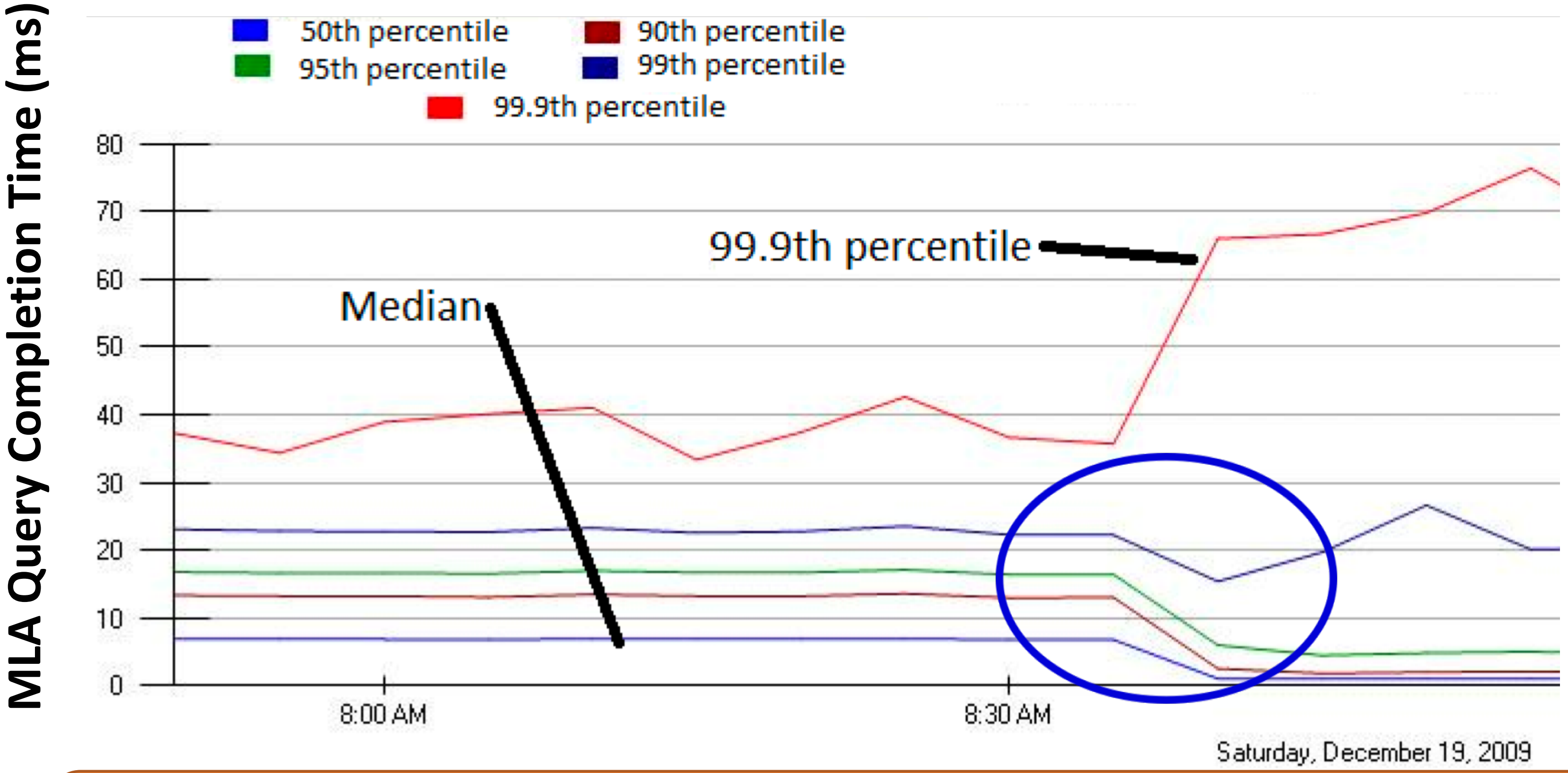
Incast

- Synchronized fan-in congestion



Vasudevan et al. (SIGCOMM'09)

Incast in Microsoft Bing



Jittering trades of median for high percentiles

DC transport requirements

1. **Low Latency**
   – Short messages, queries
2. **High Throughput**
   – Continuous data updates, backups
3. **High Burst Tolerance**
   – Incast

The challenge is to achieve these *together*

Data Center TCP

Mohammad Alizadeh et al., SIGCOMM'10

TCP widely used in the data center

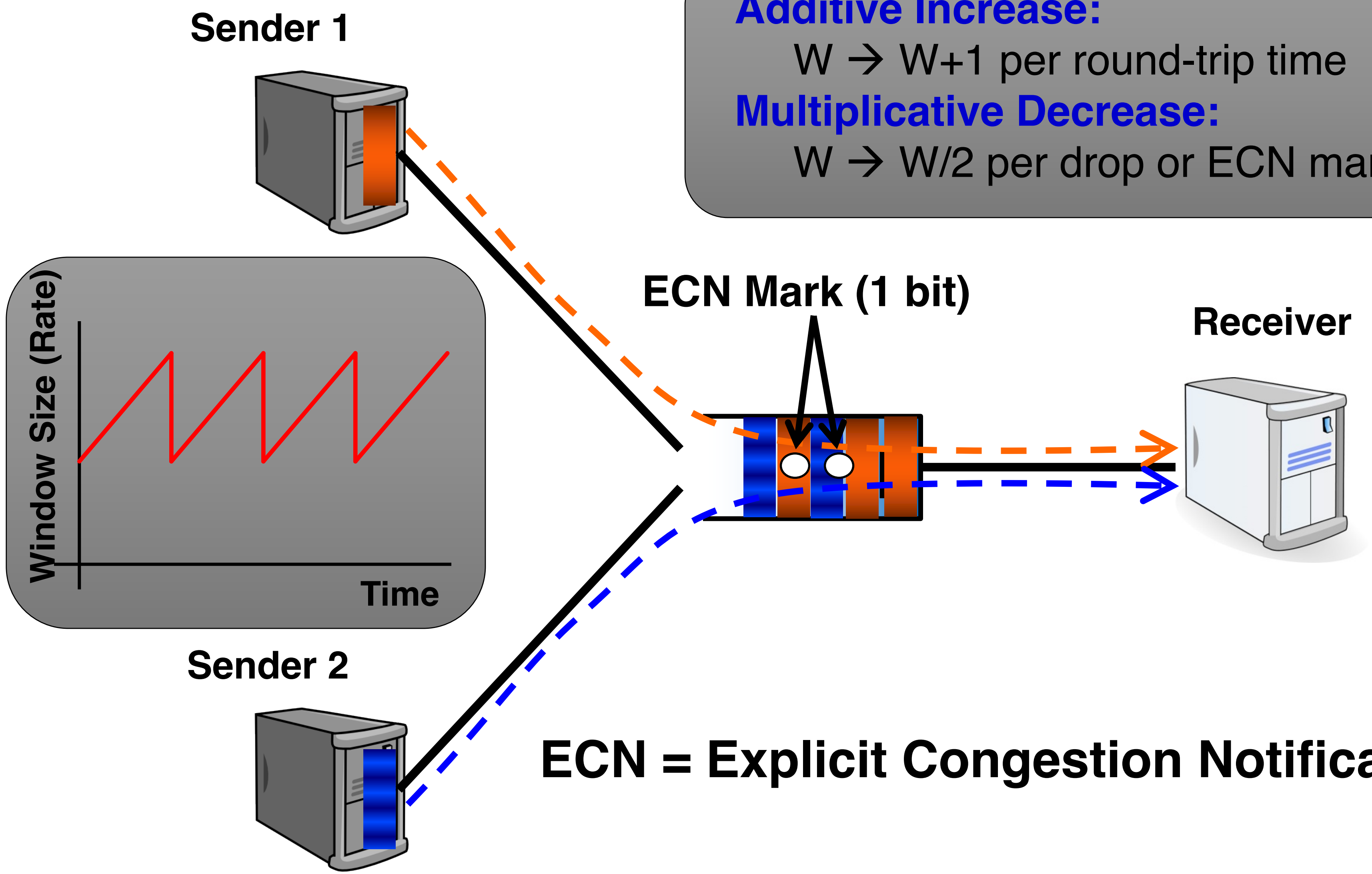
- Apps use familiar interfaces
  - TCP is deeply ingrained in the apps
  - ... And developers' minds

- However, TCP not really designed for data center environments
  - Complex to work around TCP problems
  - Ad-hoc, inefficient, often expensive solutions

Practical deployment is hard
→ keep it simple!

Review: TCP algorithm

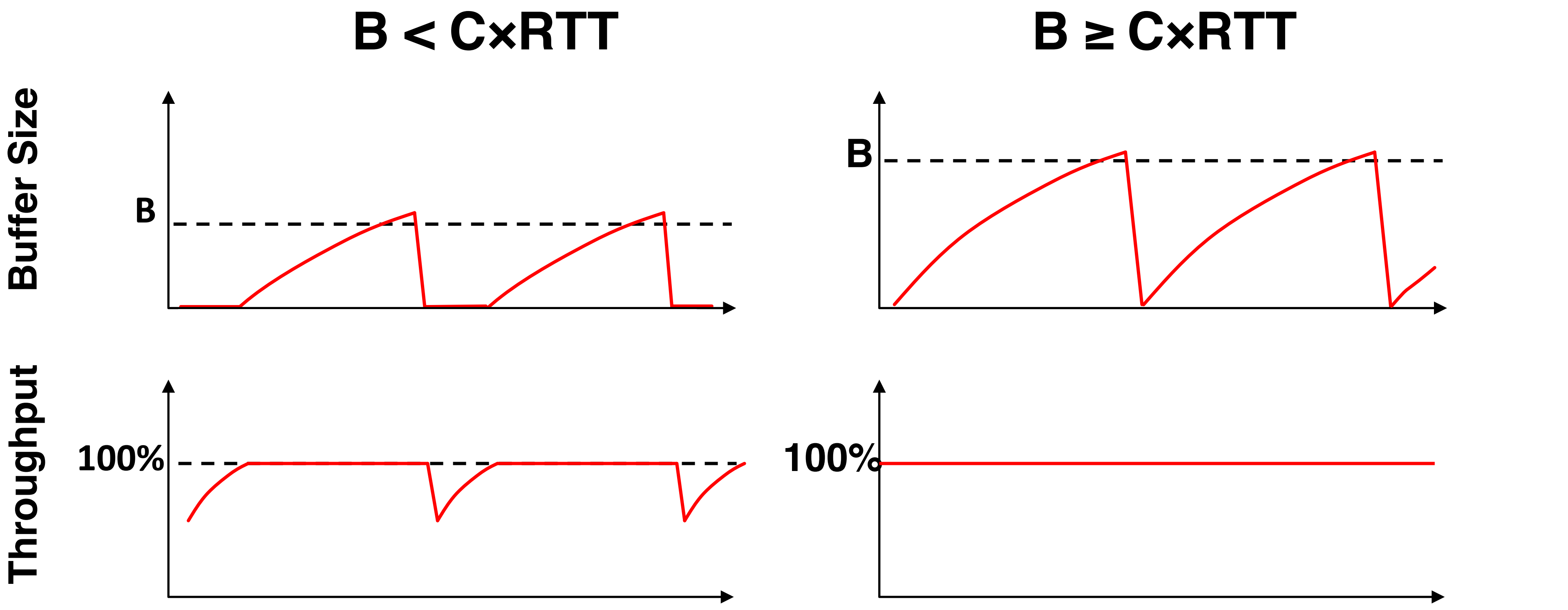
Additive Increase:
W → W+1 per round-trip time
Multiplicative Decrease:
W → W/2 per drop or ECN mark



ECN = Explicit Congestion Notification

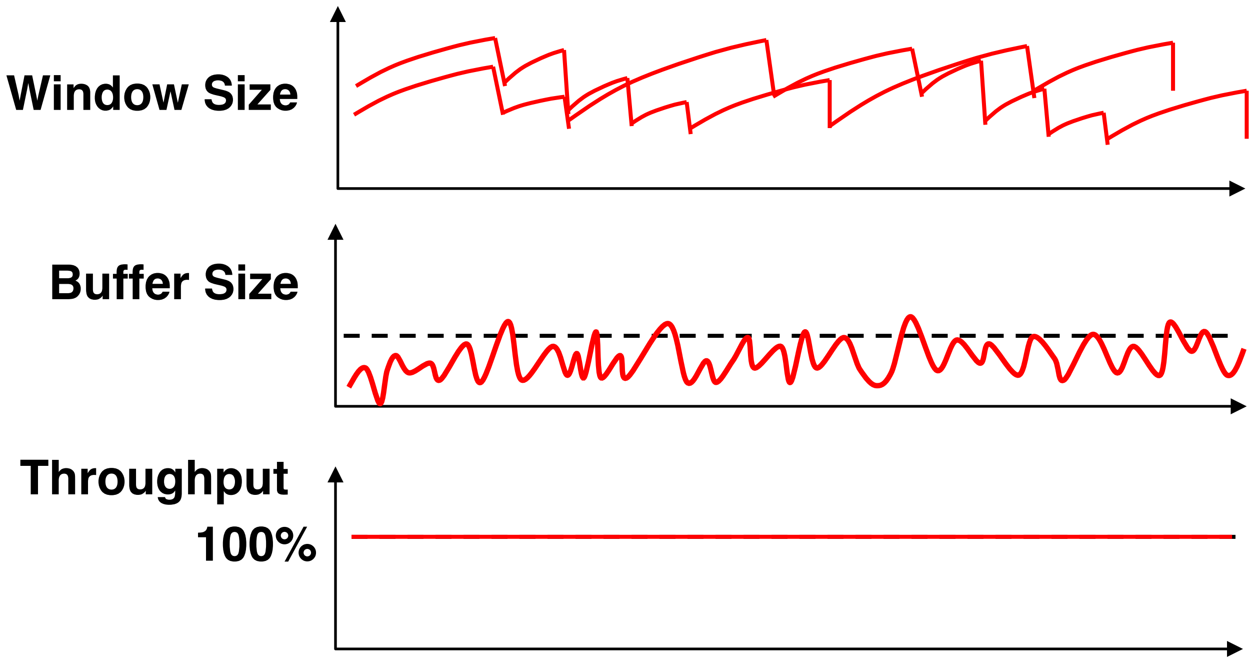
TCP buffer requirement

- Bandwidth-delay product rule of thumb:
  - A single flow needs **C×RTT** buffers for **100% Throughput.**



Reducing buffer requirements

- Appenzeller et al. (SIGCOMM '04):
  - Large # of flows: $C \times RTT/\sqrt{N}$ is enough.



Reducing buffer requirements

- Appenzeller et al. (SIGCOMM '04):
  - Large # of flows: $C \times RTT/\sqrt{N}$ is enough.

- Can't rely on stat-mux benefit in the DC
  - Measurements show typically **only 1-2 large flows** at each server

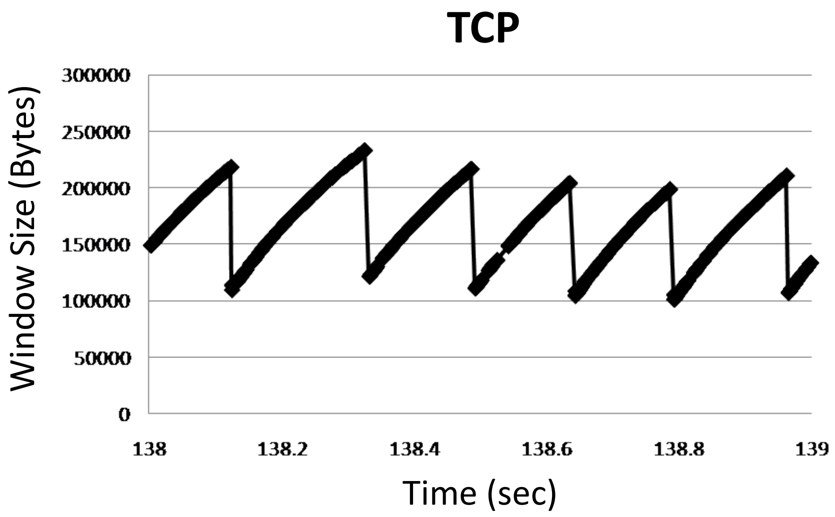
Key observation:
Low variance in sending rate → Small buffers suffice

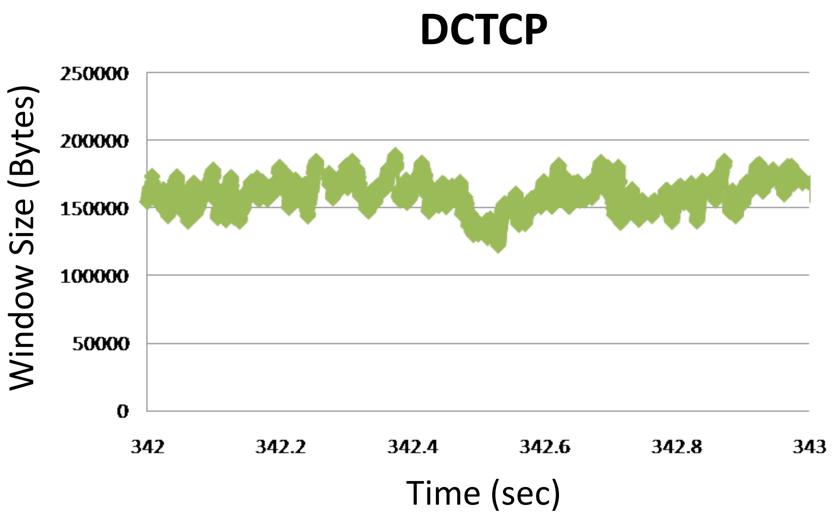
DCTCP: Main idea

- Extract multi-bit feedback from single-bit stream of ECN marks
  - Reduce window size based on **fraction** of marked packets

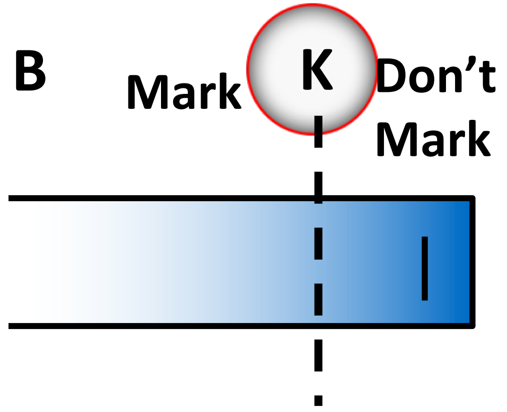
DCTCP: Main idea

ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%





DCTCP algorithm



Switch side:

- Mark packets when **Queue Length > K.**

Sender side:

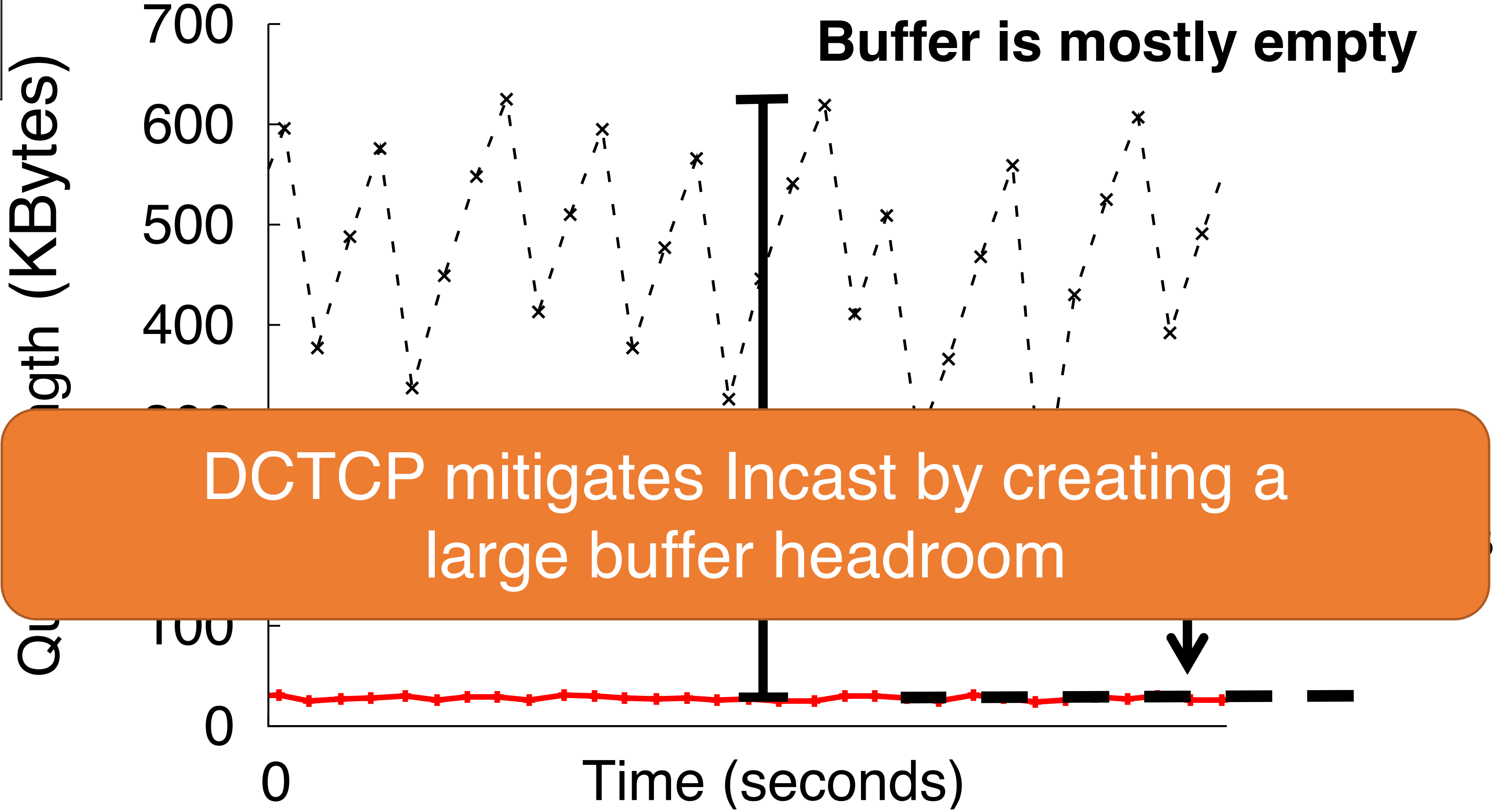
- Maintain running average of ***fraction*** of packets marked **(*α*)**.

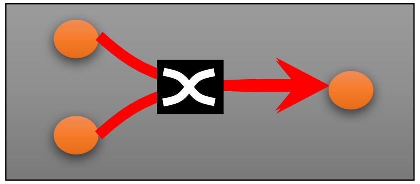
$$\text{each RTT}: \; F = \frac{\#\ \text{of marked ACKs}}{\text{Total}\ \#\ \text{of ACKs}} \quad \Rightarrow \quad \alpha \leftarrow (1-g)\alpha + gF$$

- **Adaptive window decreases:** $W \leftarrow (1 - \frac{\alpha}{2})W$
  - Note: decrease factor between 1 and 2.

DCTCP vs TCP

Experiment: 2 flows (Win 7 stack), Broadcom 1Gbps Switch





DCTCP mitigates Incast by creating a large buffer headroom

Why it works

1. Low Latency

- ✓ **Small buffer occupancies** → low queuing delay

2. High Throughput

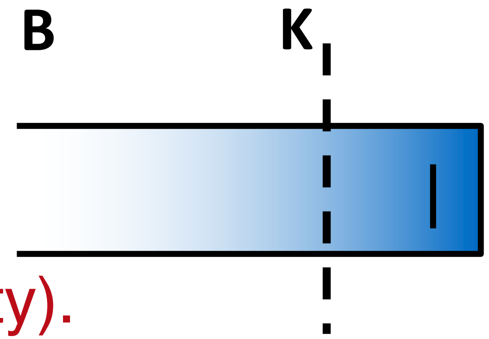
- ✓ **ECN averaging** → smooth rate adjustments, low variance

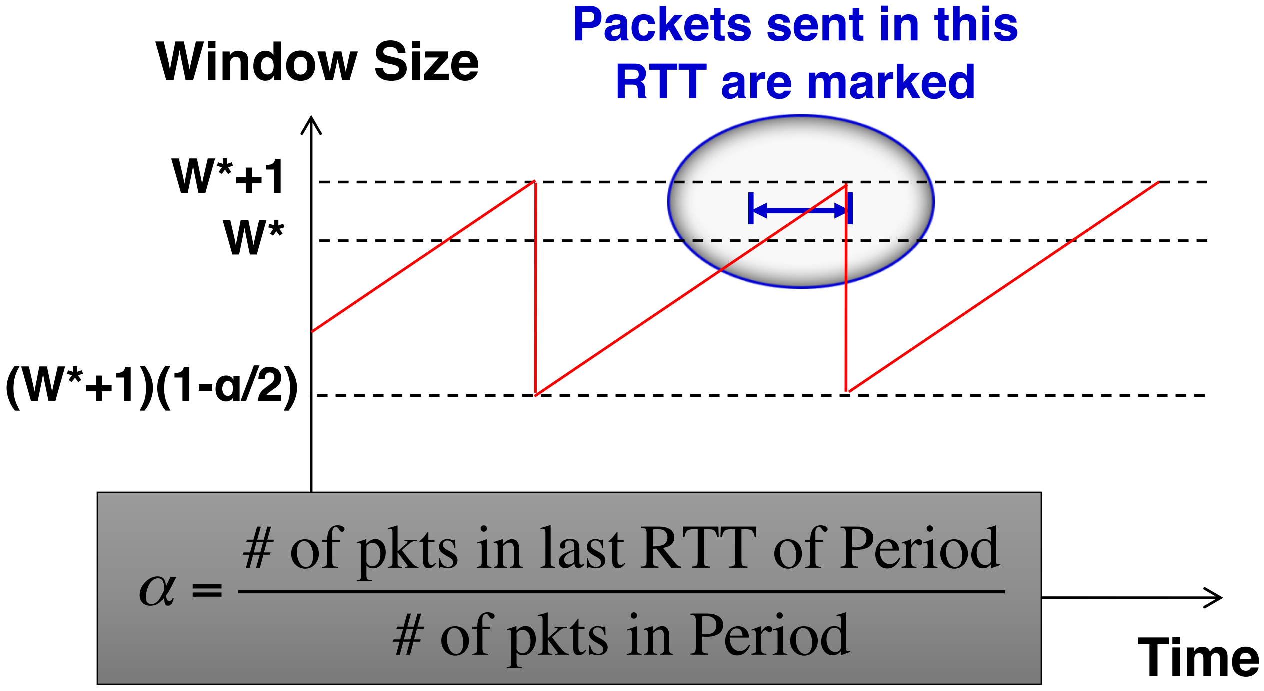
3. High Burst Tolerance

- ✓ **Large buffer headroom** → bursts fit
- ✓ **Aggressive marking** → sources react before packets are dropped

Setting parameters: A bit of analysis

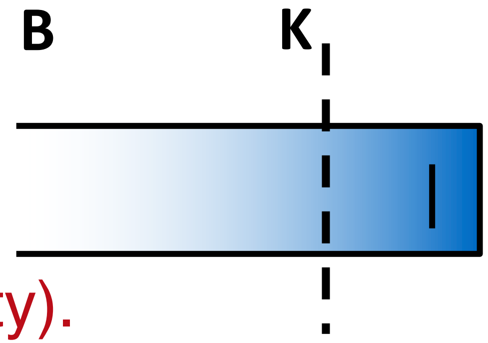
- How much buffering does DCTCP need for 100% throughput?
  - ➢ Need to quantify queue size oscillations (Stability).





$$\alpha = \frac{\text{\# of pkts in last RTT of Period}}{\text{\# of pkts in Period}}$$

Setting parameters: A bit of analysis

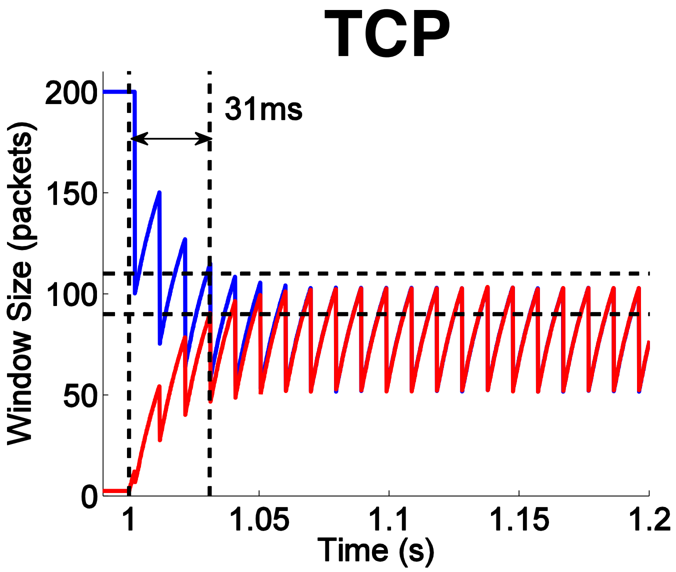


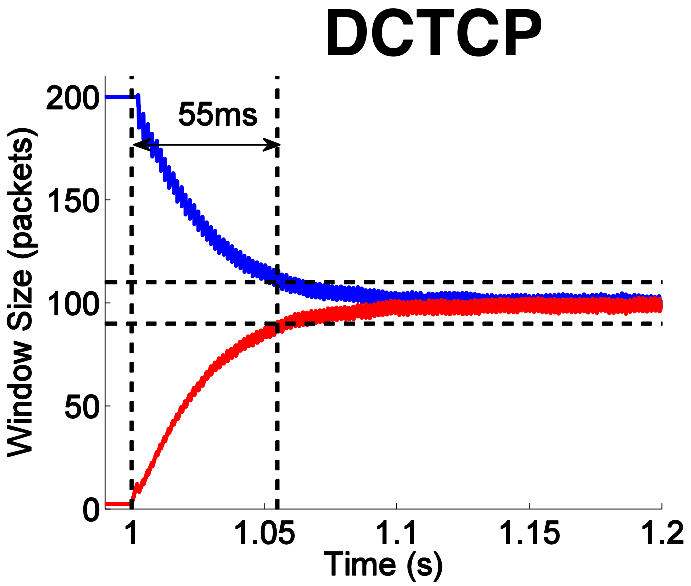
- How small can queues be without loss of throughput?
  - ➢ Need to quantify queue size oscillations (Stability).

K > (1/7) C x RTT ⟷ **for TCP: K > C x RTT**

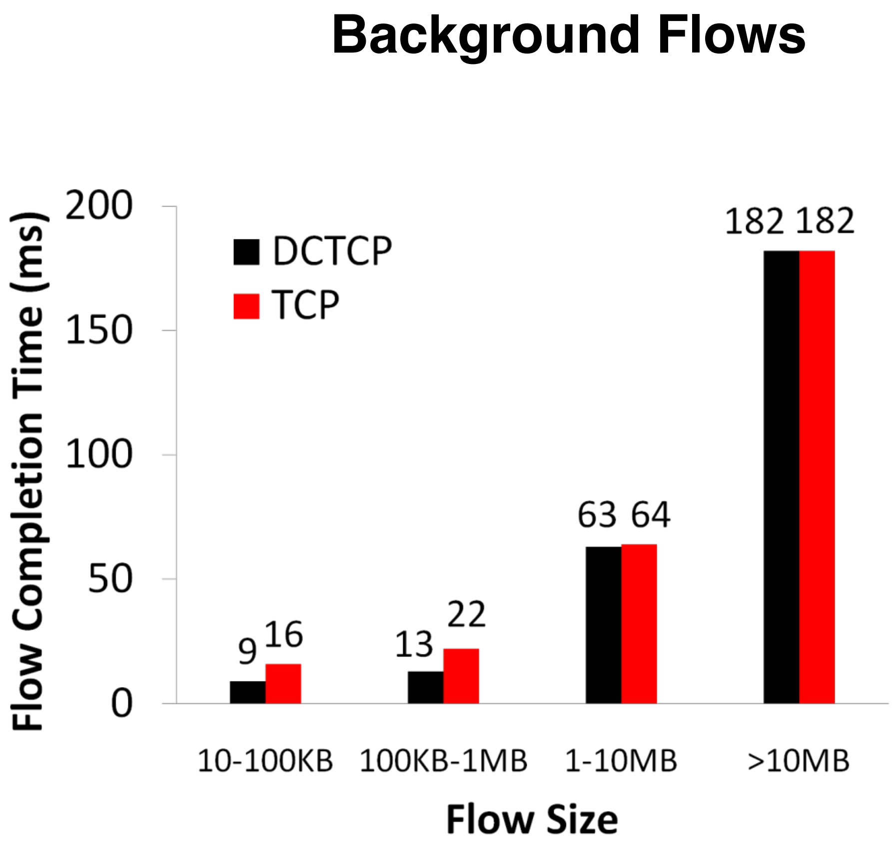
Convergence time

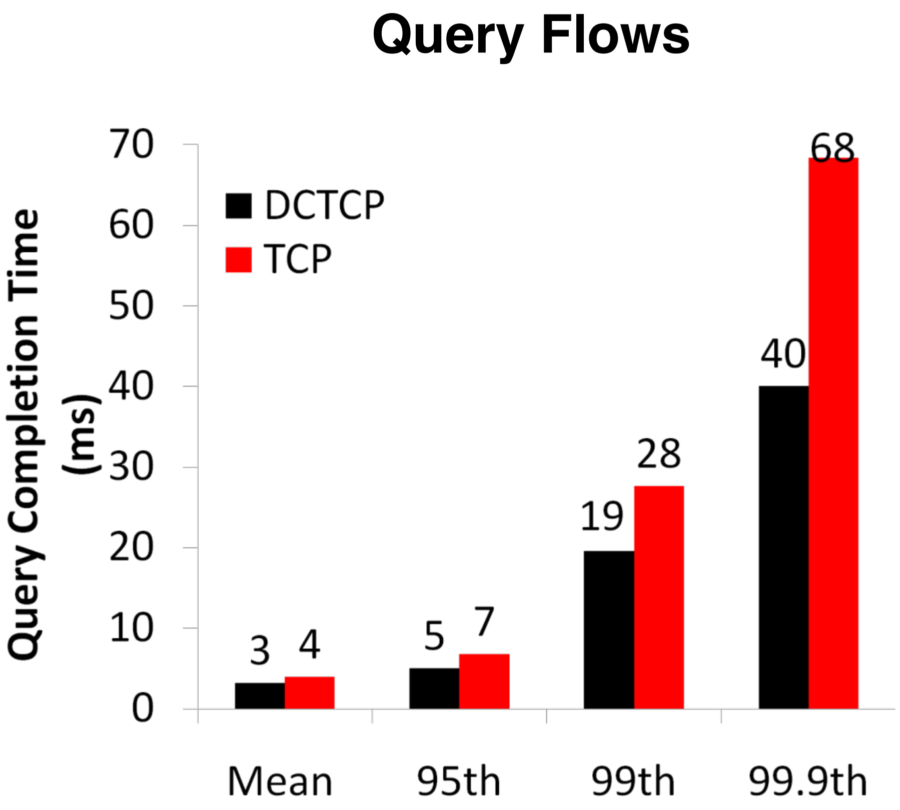
- DCTCP takes at most ~40% more **RTTs** than TCP
  - “Analysis of DCTCP”, SIGMETRICS 2011
- **Intuition:** DCTCP makes smaller adjustments than TCP, but makes them much more frequently



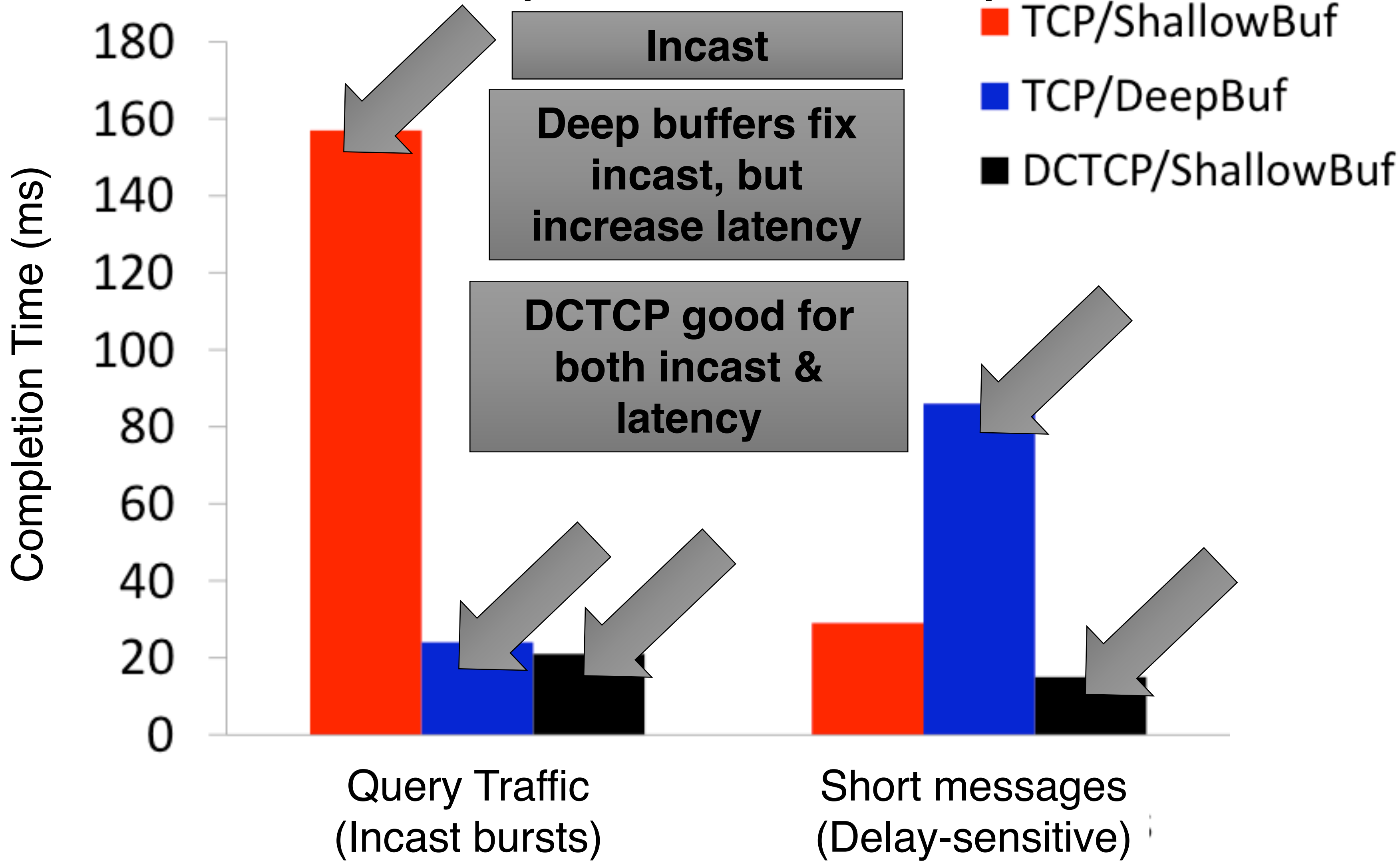


Bing benchmark (baseline)





Bing benchmark (scaled 10x)



Discussion

- Between throughput, delay, and convergence time, what metrics are you willing to give up? Why?
- Are there other factors that may determine choice of K and B besides loss of throughput and max queue size?
- How would you improve on DCTCP?
- How could you add on flow prioritization over DCTCP?

Acknowledgment

- Slides heavily adapted from material by Mohammad Alizadeh