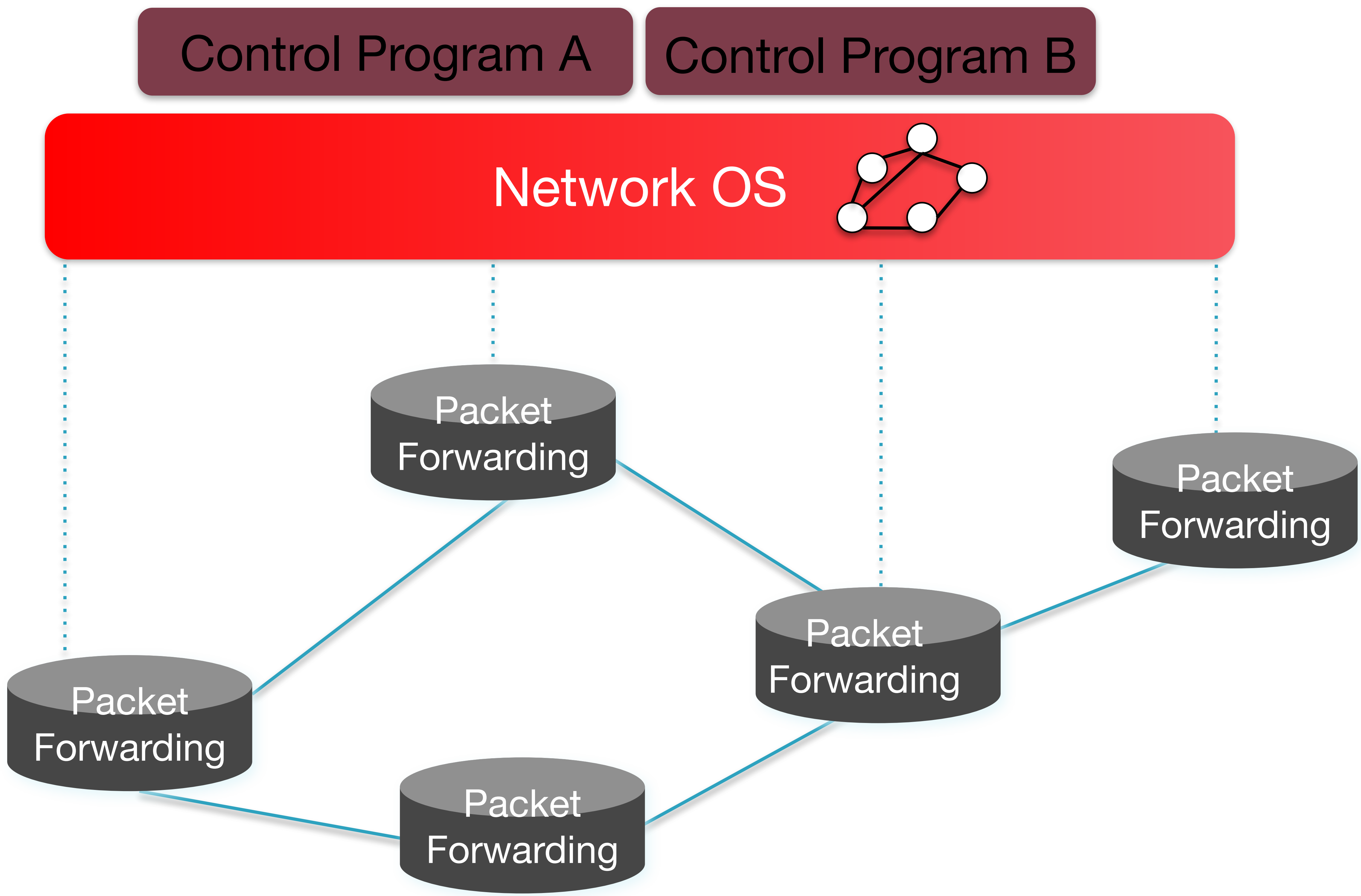
Software Defined Network (SDN)



Control Program

- Control program operates on view of network
  - **Input**: global network view (graph/database)
  - **Output**: configuration of each network device

- Control program is not a distributed system
  - Abstraction hides details of distributed state

Forwarding Abstraction

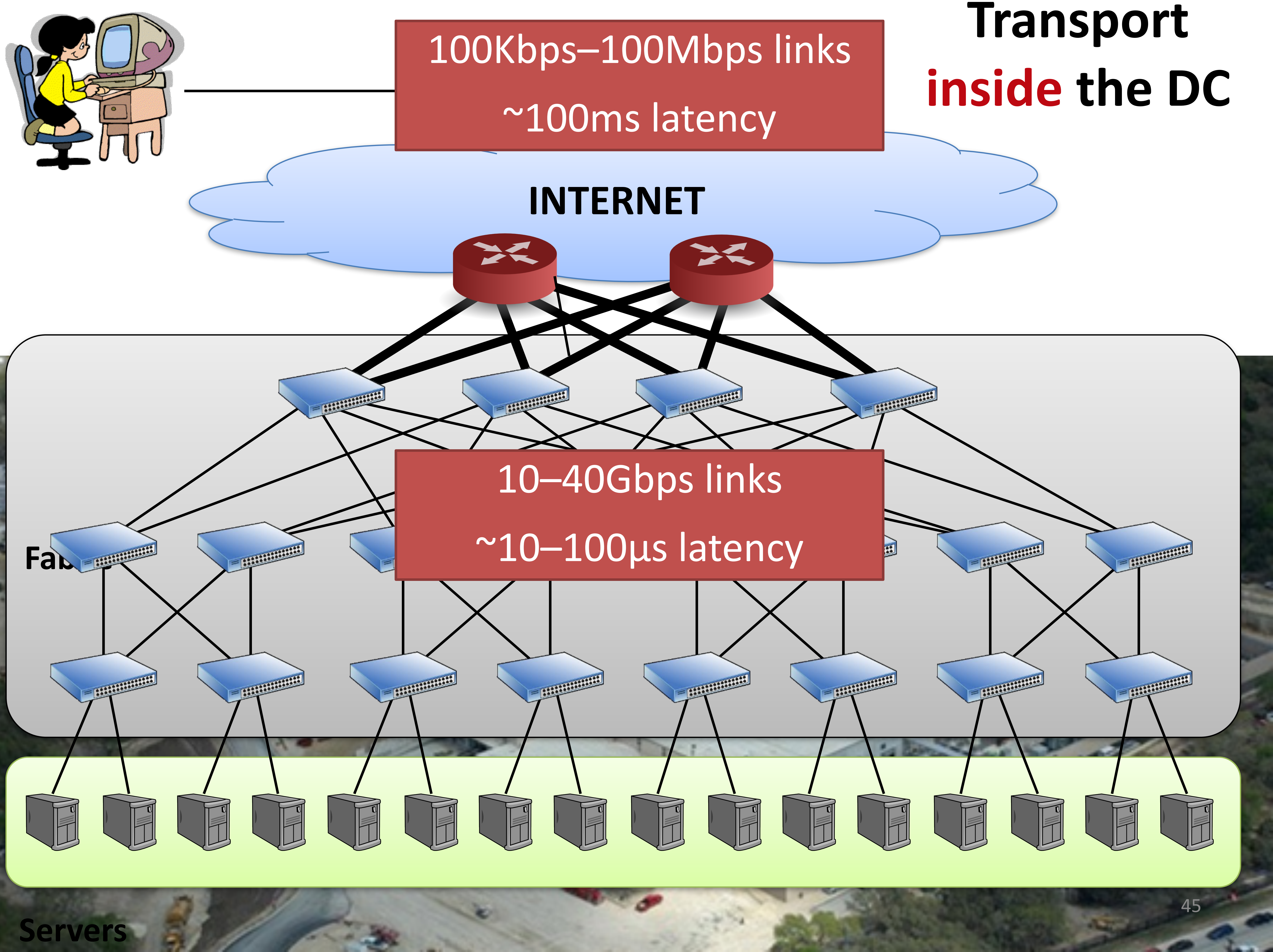
Purpose: Abstract away forwarding hardware

- Flexible
  - Behavior specified by control plane
  - Built from basic set of forwarding primitives
- Minimal
  - Streamlined for speed and low-power
  - Control program not vendor-specific

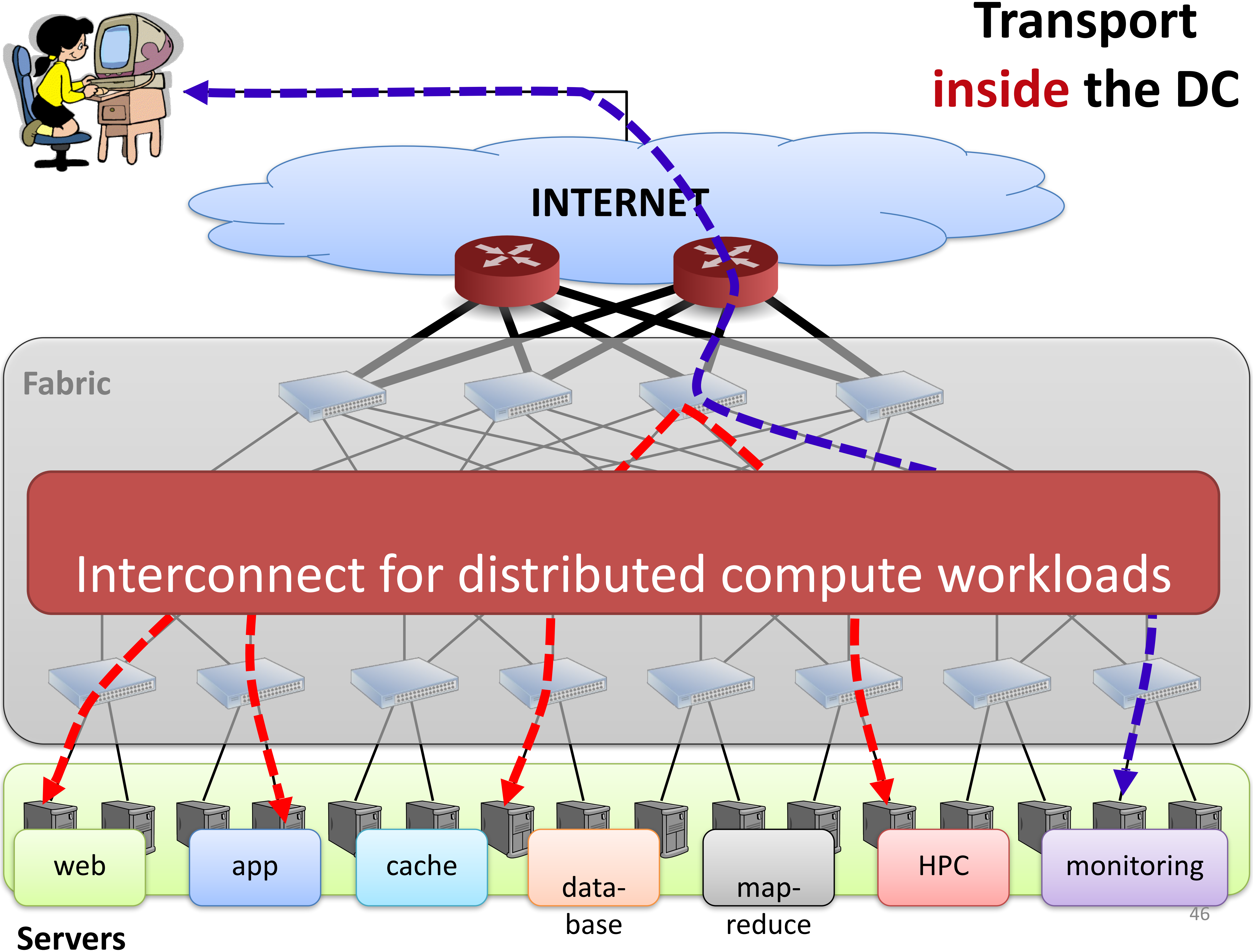
- OpenFlow is an example of such an abstraction

Congestion control in the data center

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

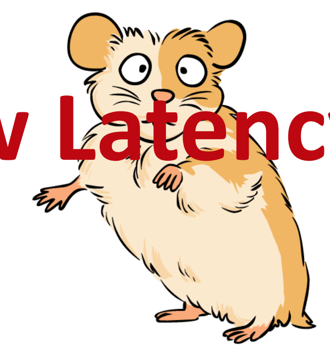
Challenging traffic patterns
- Diverse mix of mice & elephants
- Incast

Cheap switches
- Single-chip shared-memory devices; shallow buffers

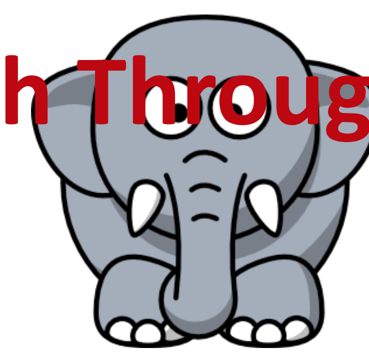
Data Center Workloads

Mice & Elephants

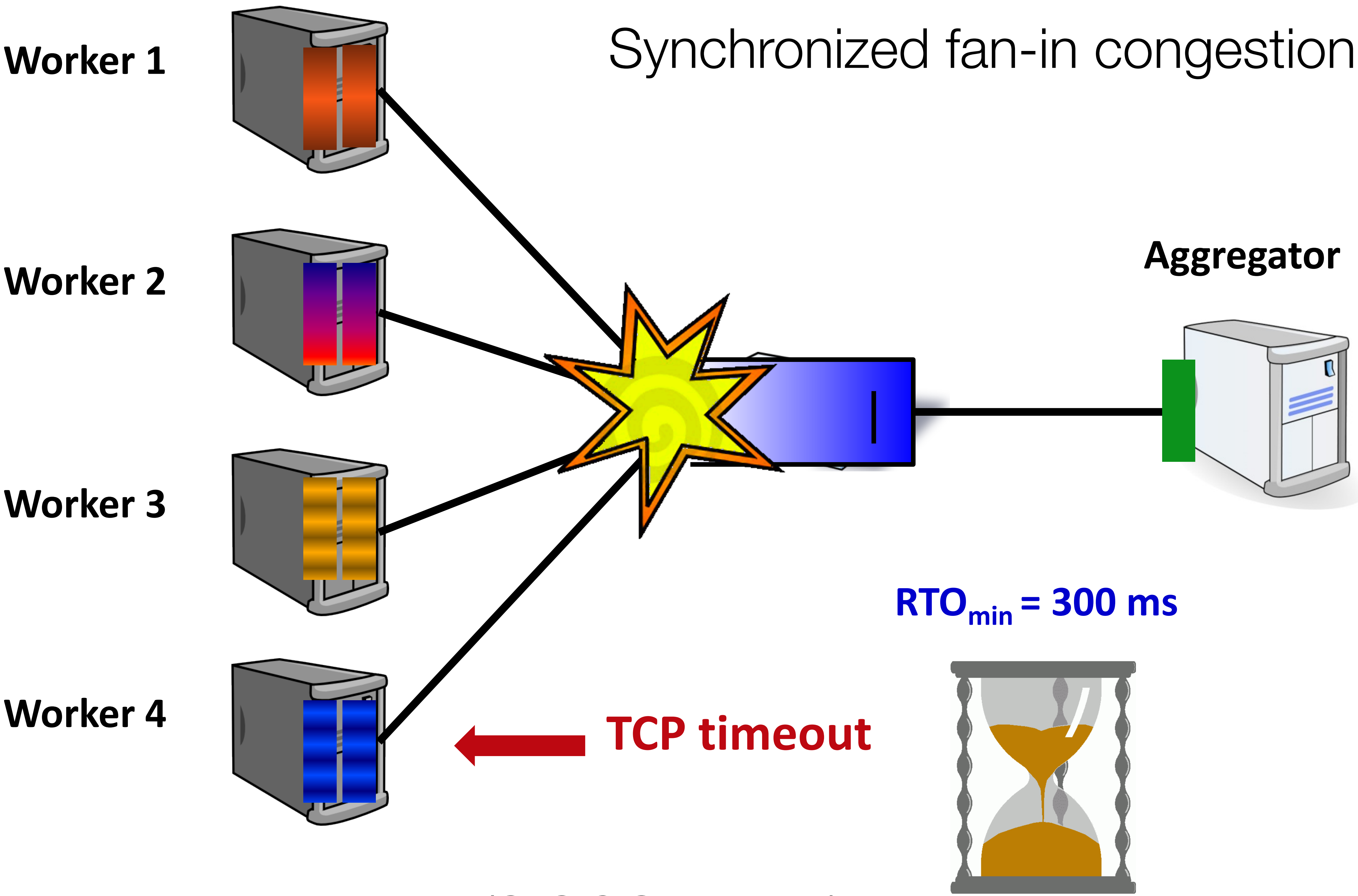
Short messages
(e.g., query, coordination) → **Low Latency**



Large flows
(e.g., data update, backup) → **High Throughput**

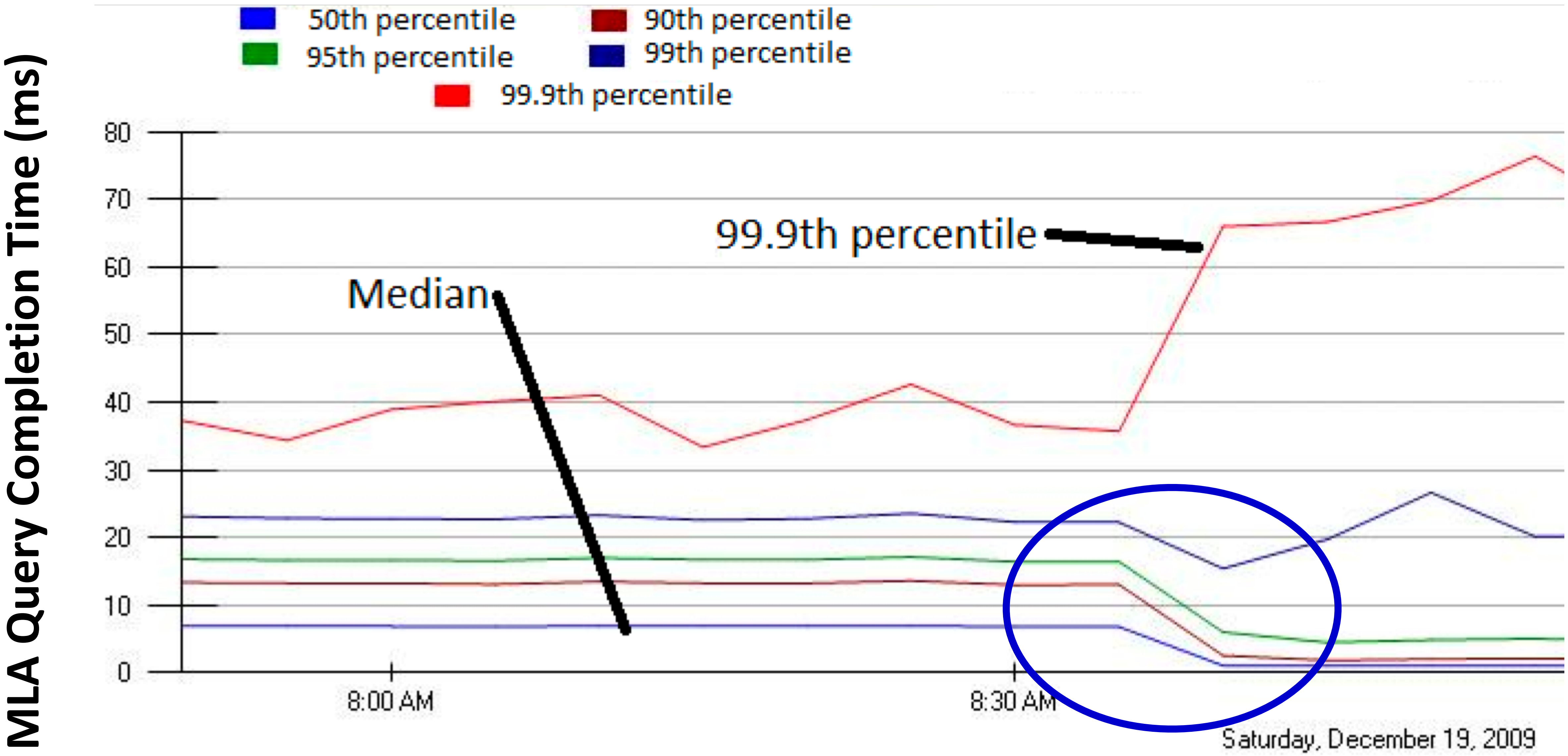


Incast



- ◇ Vasudevan et al. (SIGCOMM'09)

Incast in Bing



Jittering trades of median for high percentiles

DC Transport Requirements

1. Low Latency
   – Short messages, queries

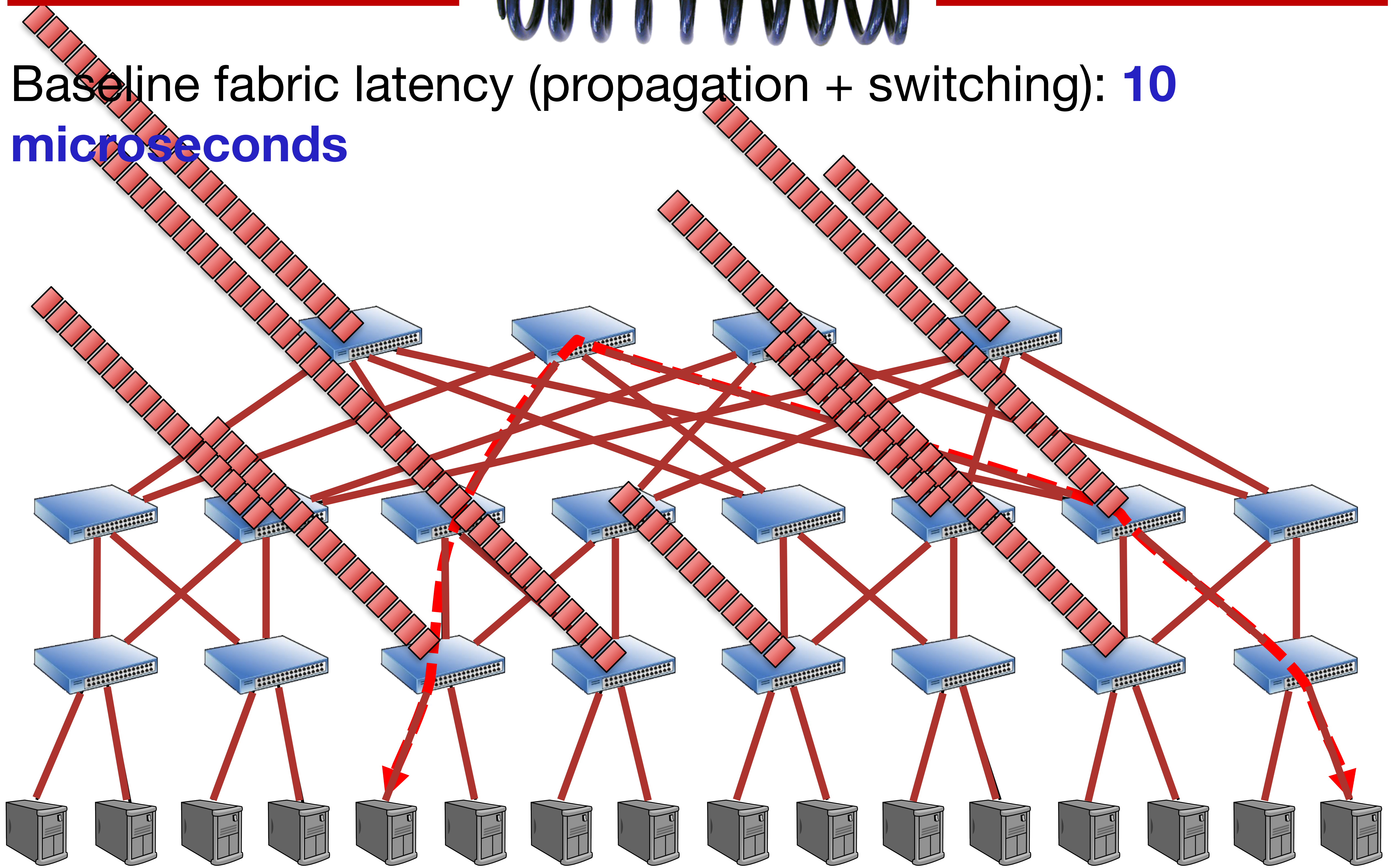
2. High Throughput
   – Continuous data updates, backups

3. High Burst Tolerance
   – Incast

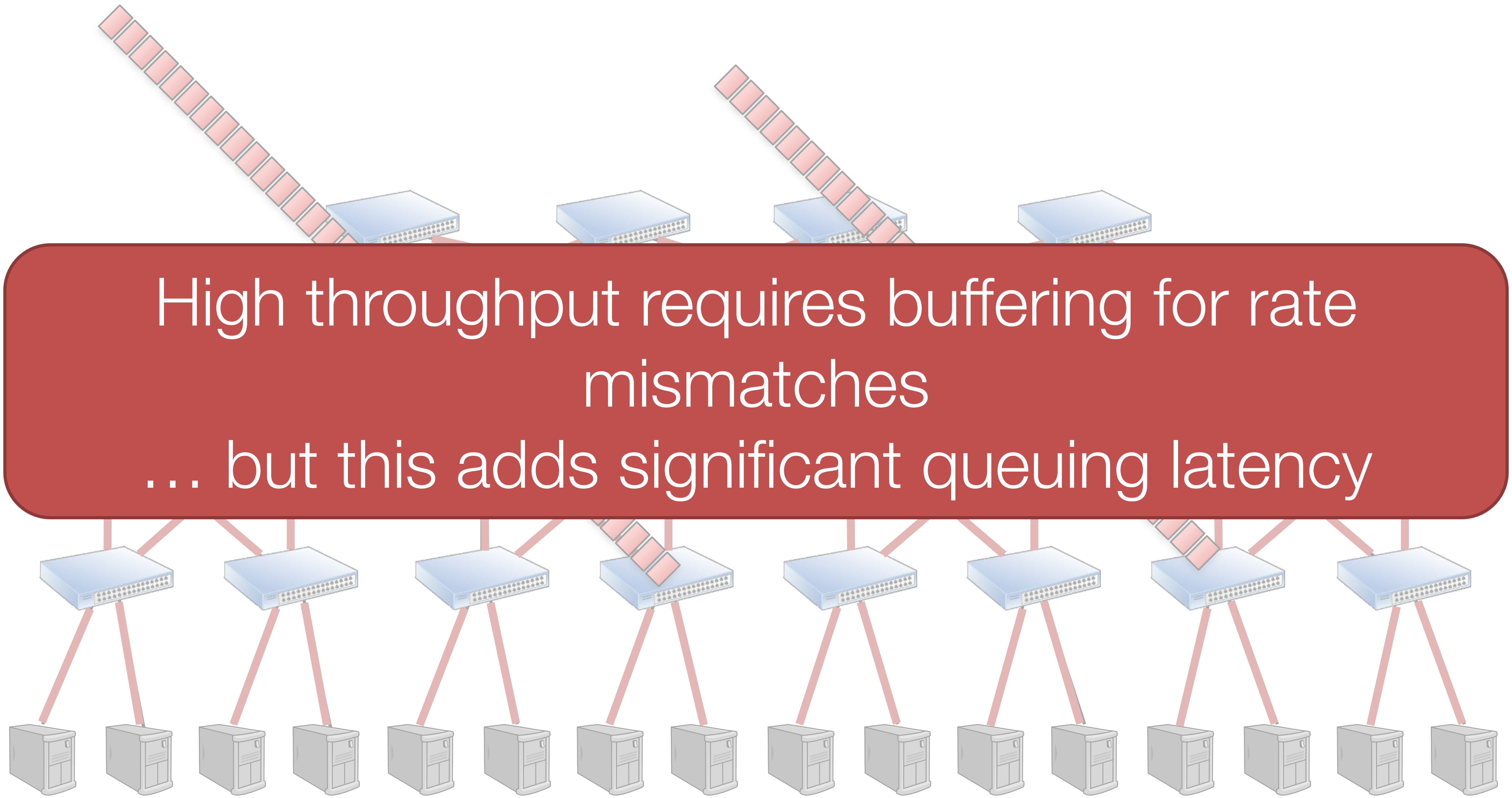
The challenge is to achieve these together

High Throughput | Low Latency

Baseline fabric latency (propagation + switching): **10 microseconds**



High Throughput  Low Latency



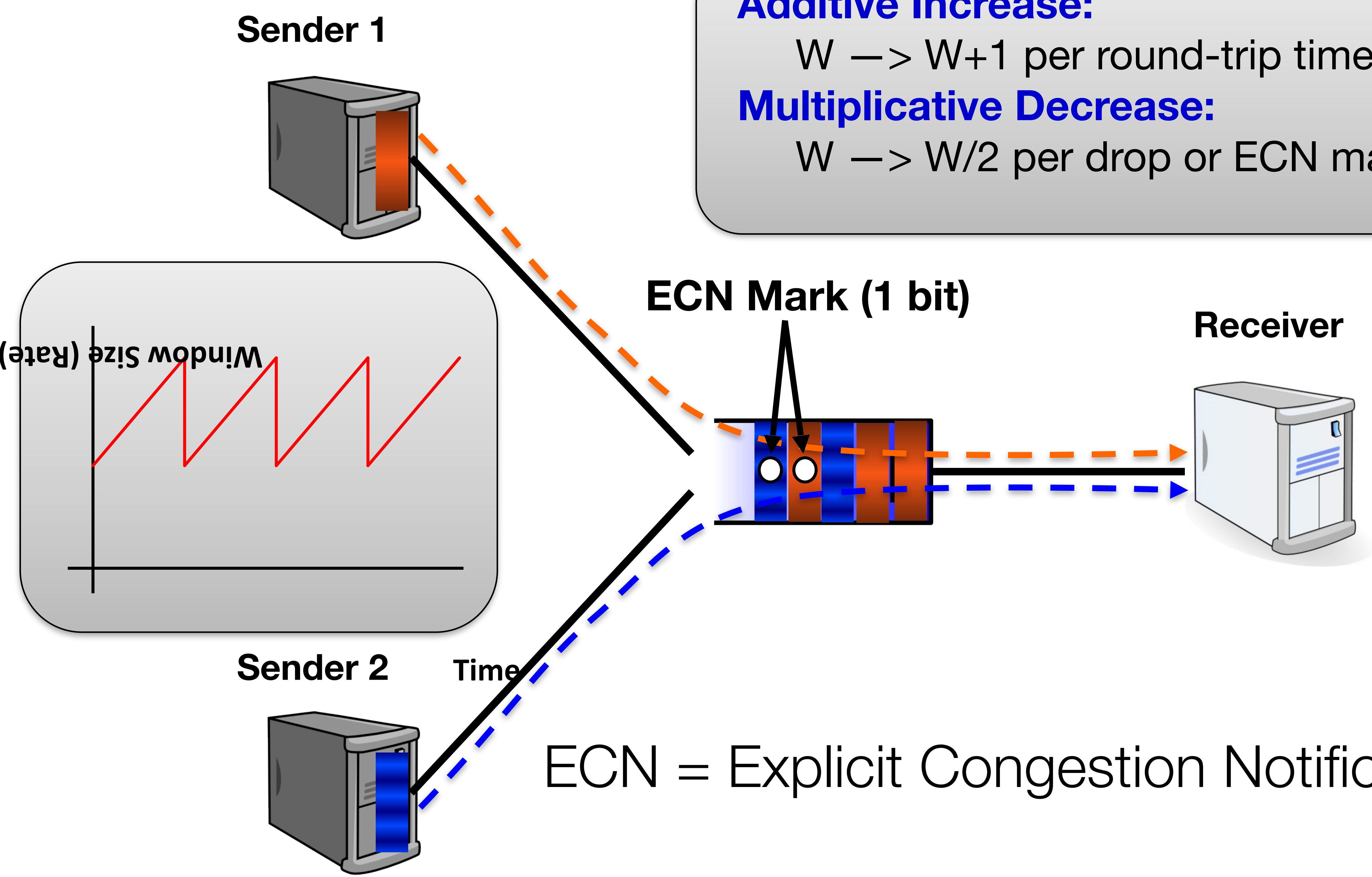
High throughput requires buffering for rate mismatches

… but this adds significant queuing latency

Data Center TCP

Review: The TCP Algorithm

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



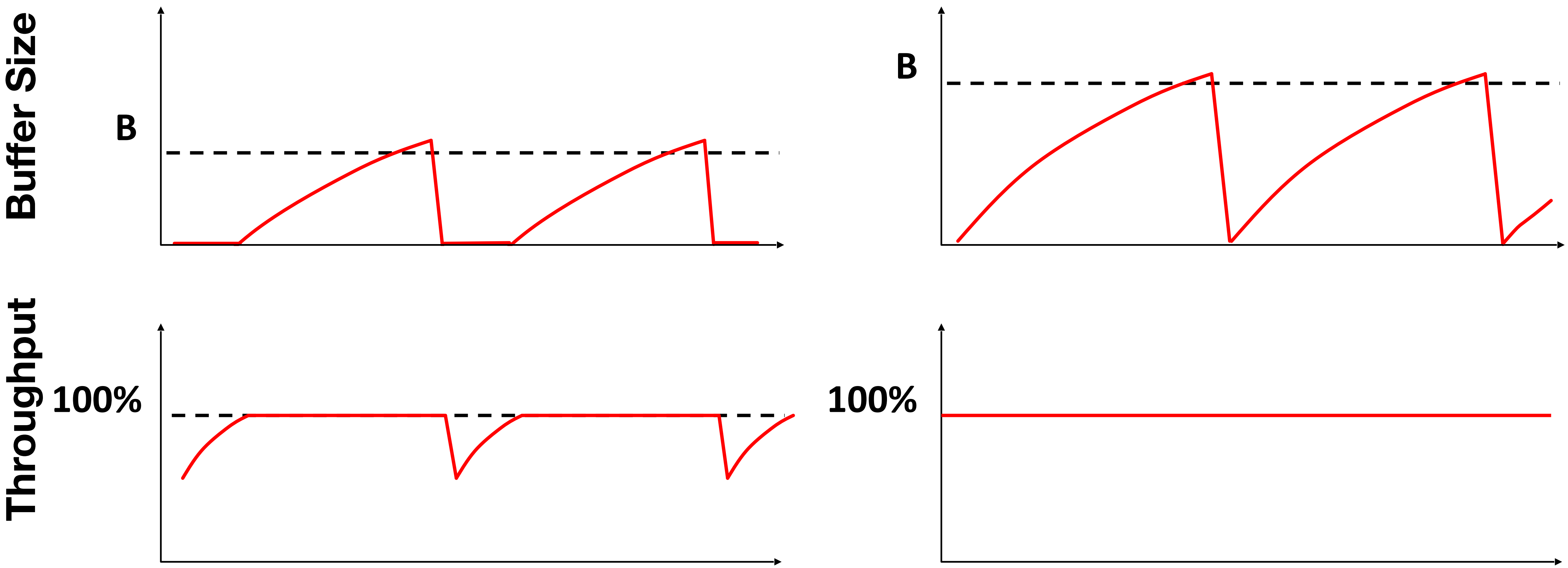
ECN = Explicit Congestion Notification

TCP Buffer Requirement

Bandwidth-delay product rule of thumb:

– A single flow needs **C×RTT** buffers for **100% Throughput.**

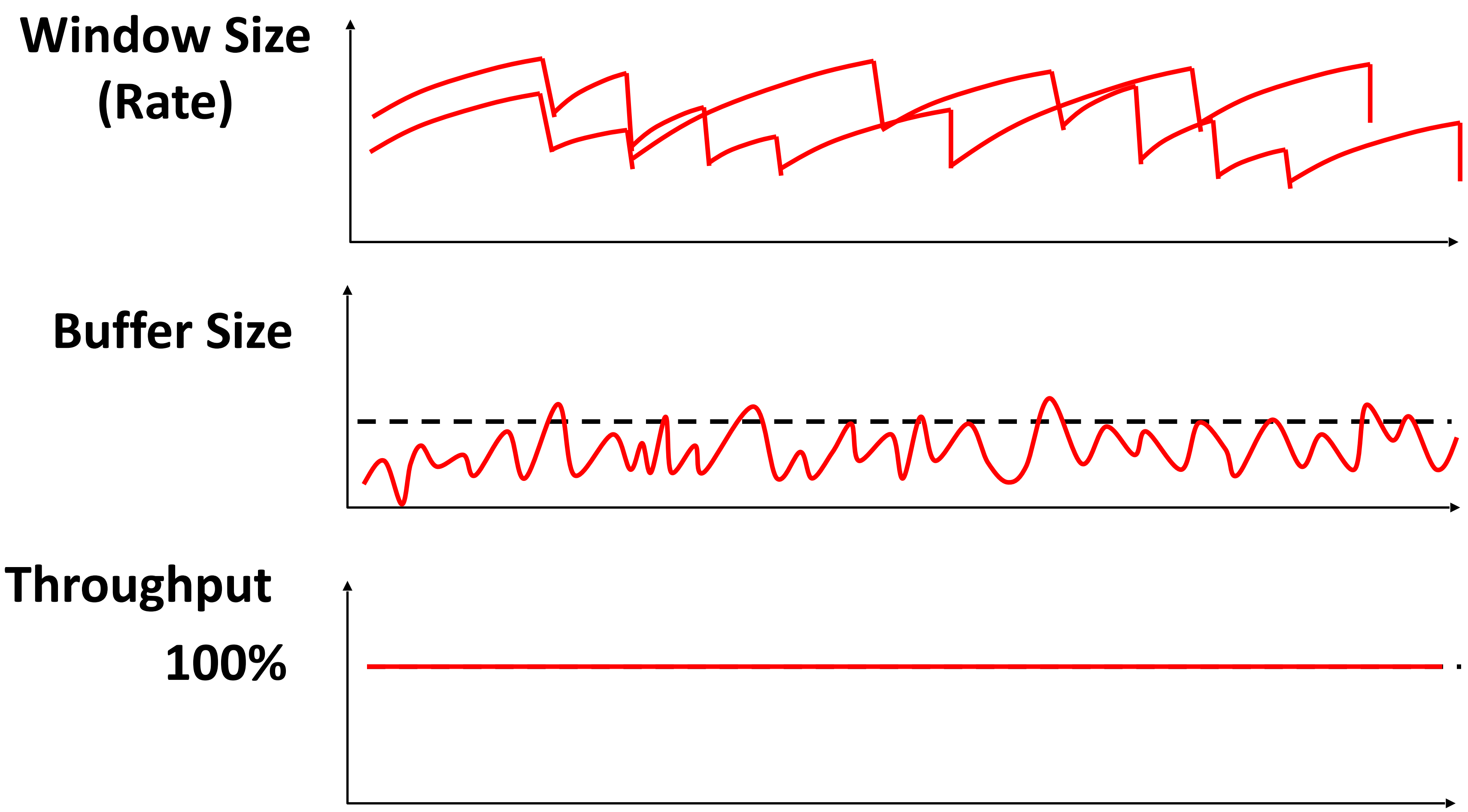
B < C×RTT **B ≥ C×RTT**



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.

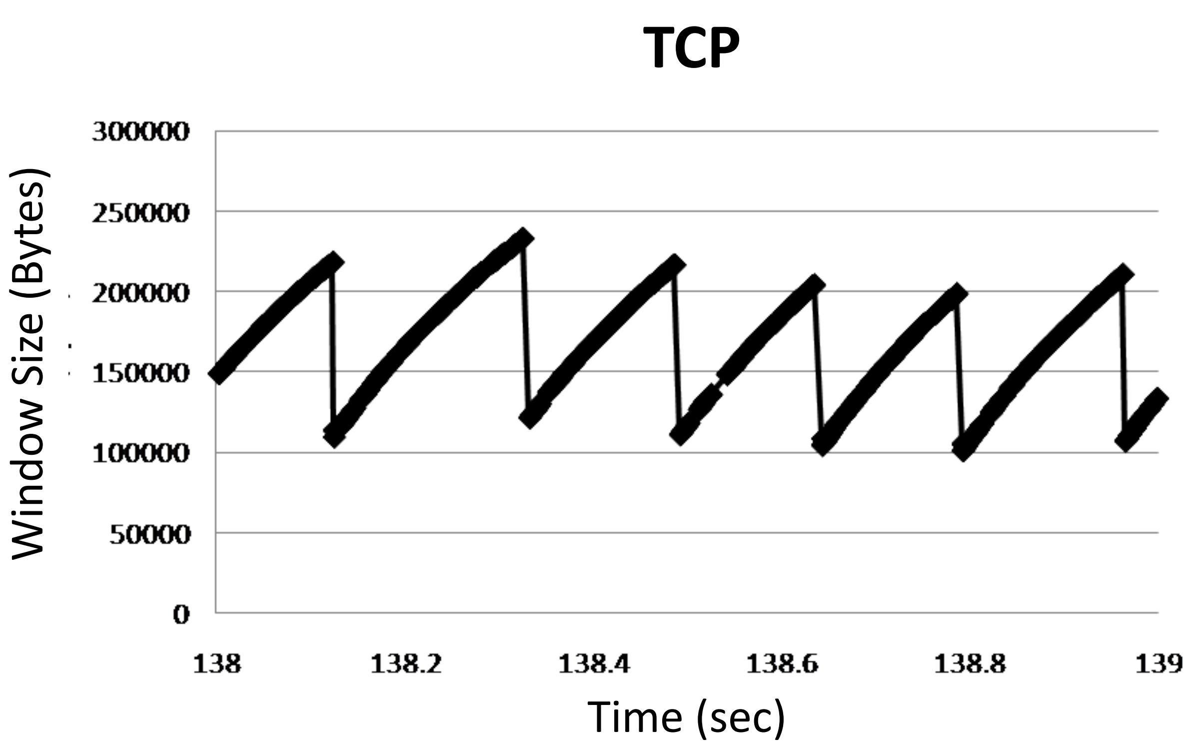
- Measurements show typically **only 1-2 large flows** at each server

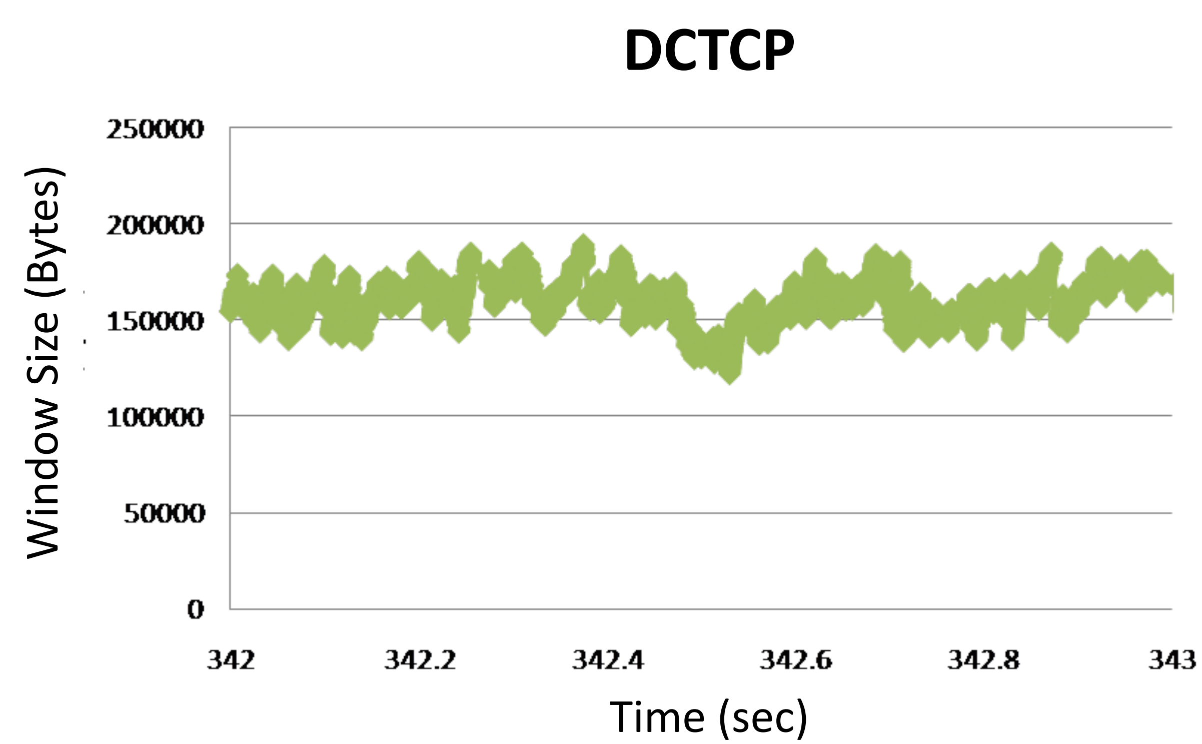
Key Observation:
Low variance in sending rate -> Small buffers suffice

DCTCP: Main Idea

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

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%

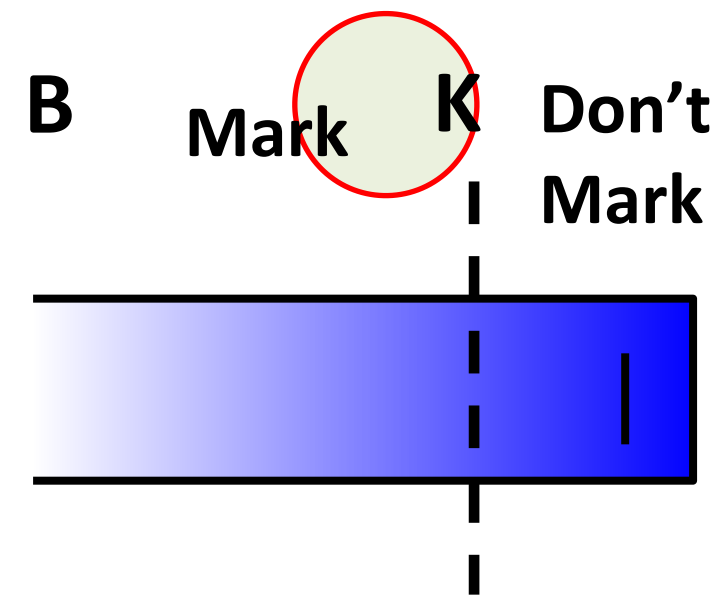




DCTCP: Algorithm

Switch side:

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



Sender side:

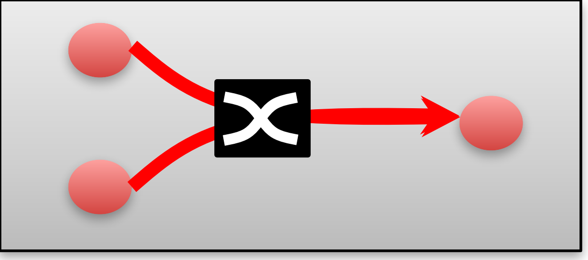
– Maintain running average of ***fraction*** of packets marked **(α)**.

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

$$W \leftarrow \left(1 - \frac{\alpha}{2}\right) W$$

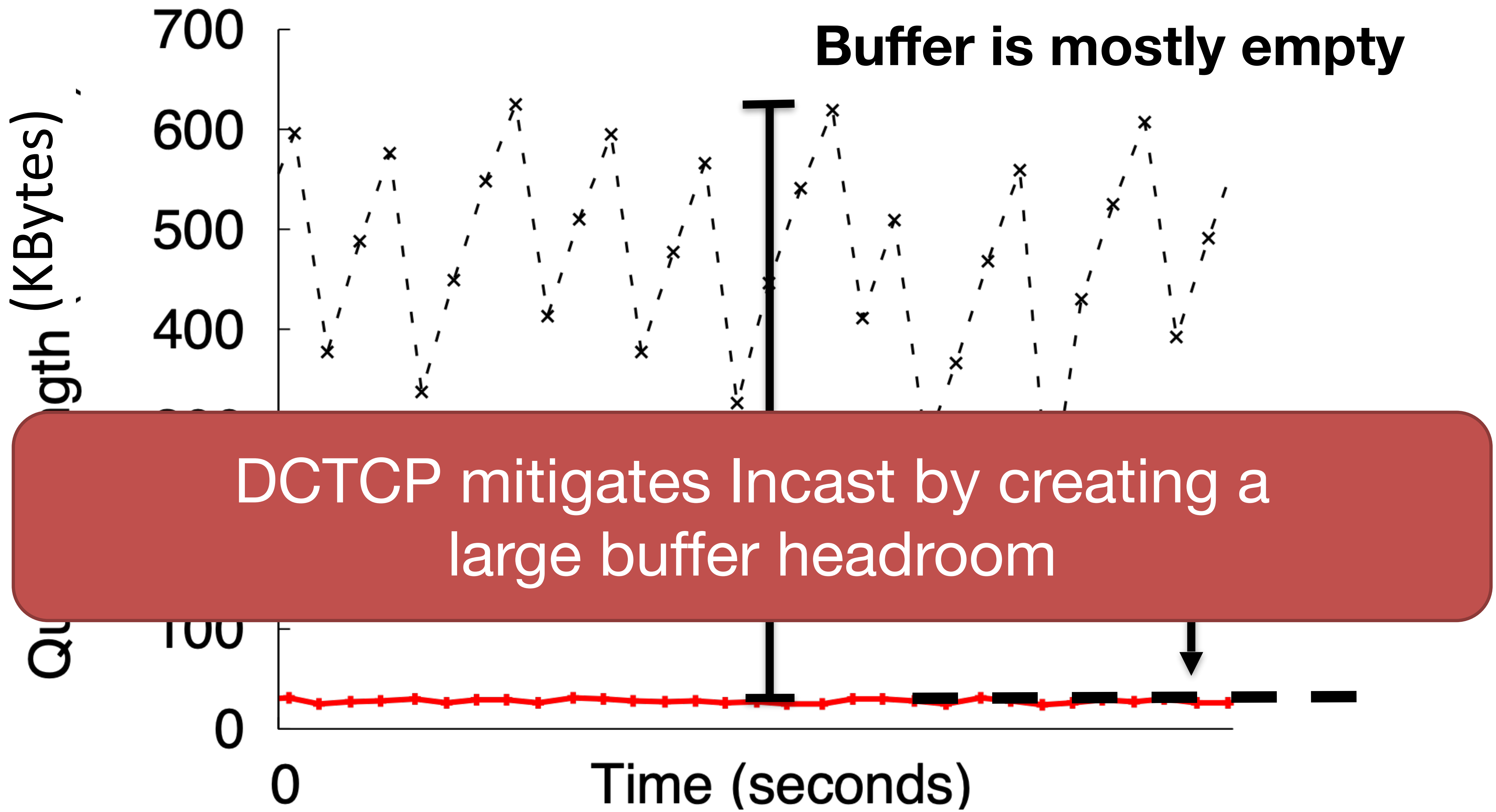
➢ **Adaptive window decreases:**

– Note: decrease factor between 1 and 2.



DCTCP vs TCP

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



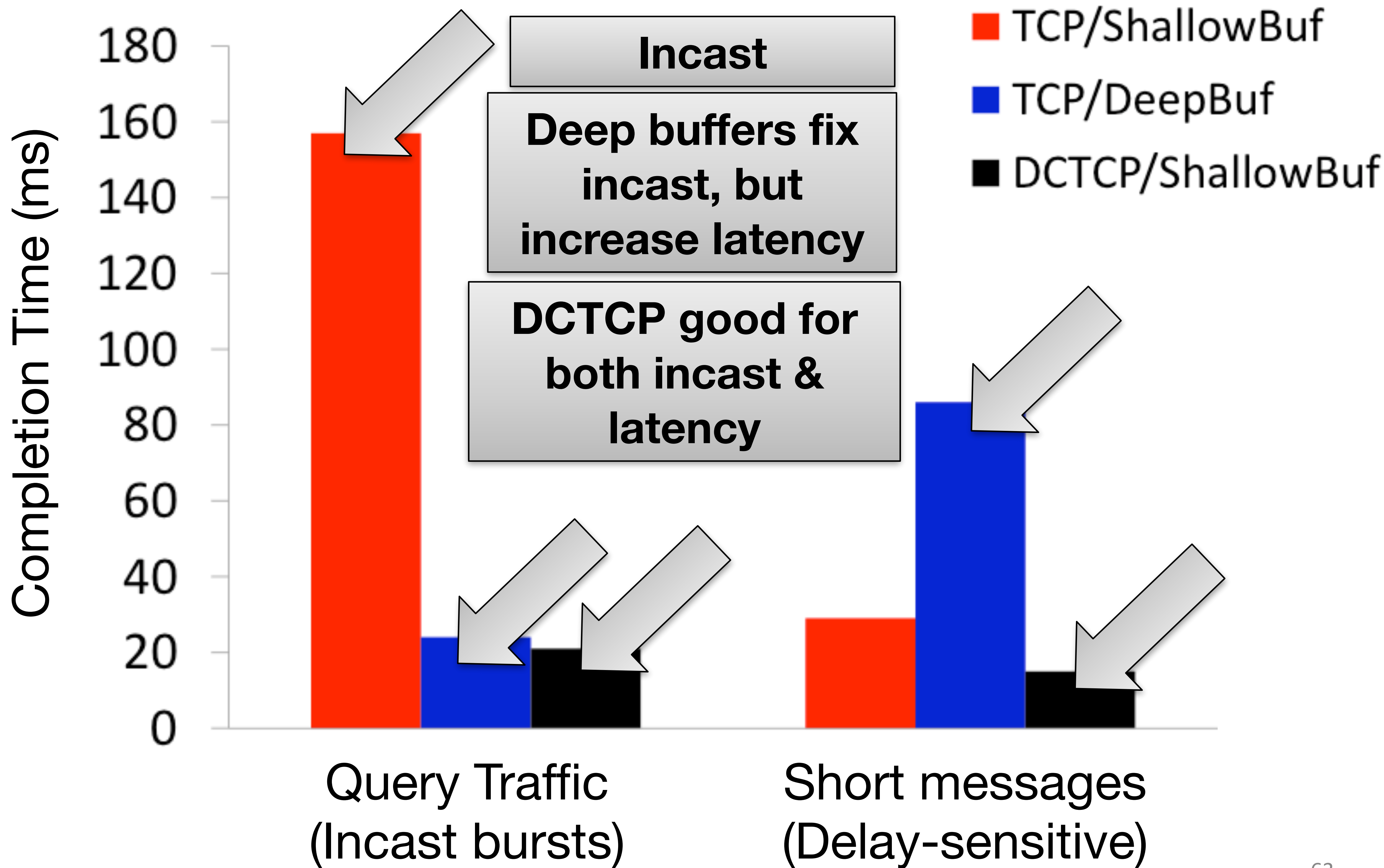
Why it Works

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

Bing Benchmark (scaled 10x)

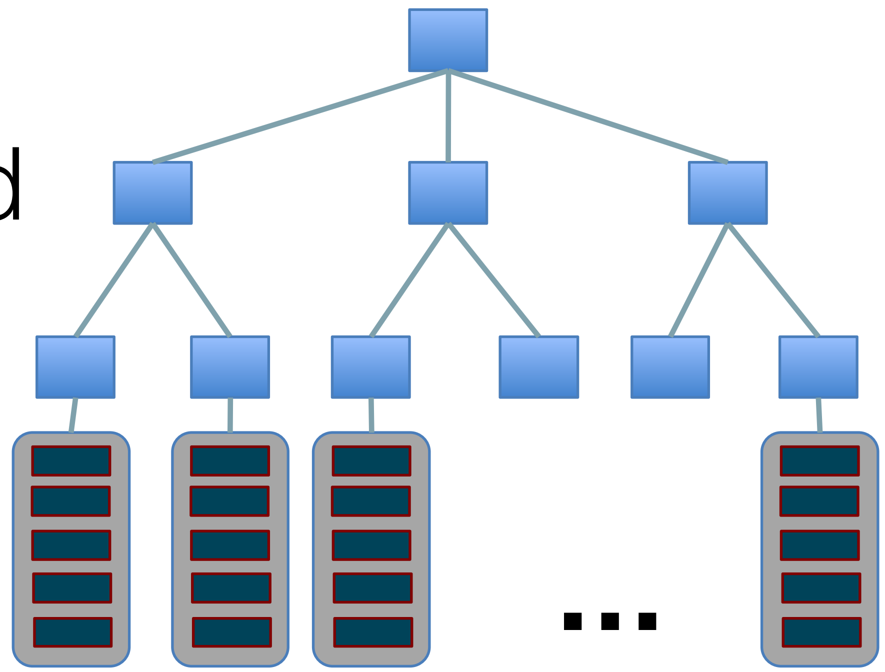


Let TCP take advantages of modern data center topologies

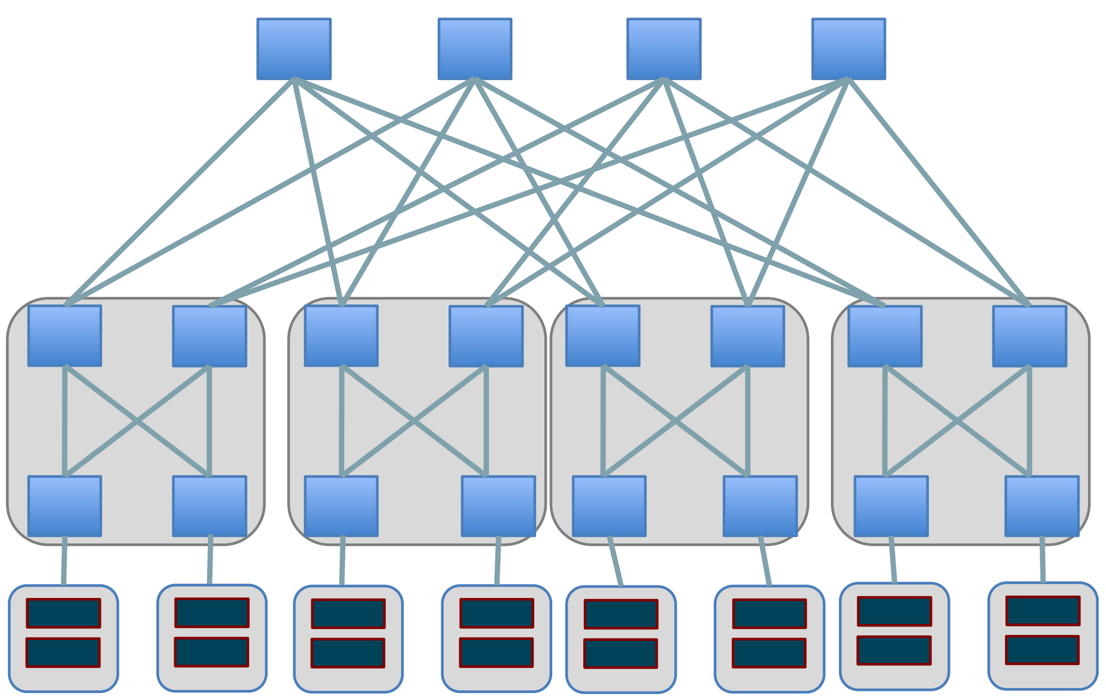
Improving Datacenter Performance and Robustness with Multipath TCP, SIGCOMM 2011

To satisfy demand, modern data centers provide many parallel paths

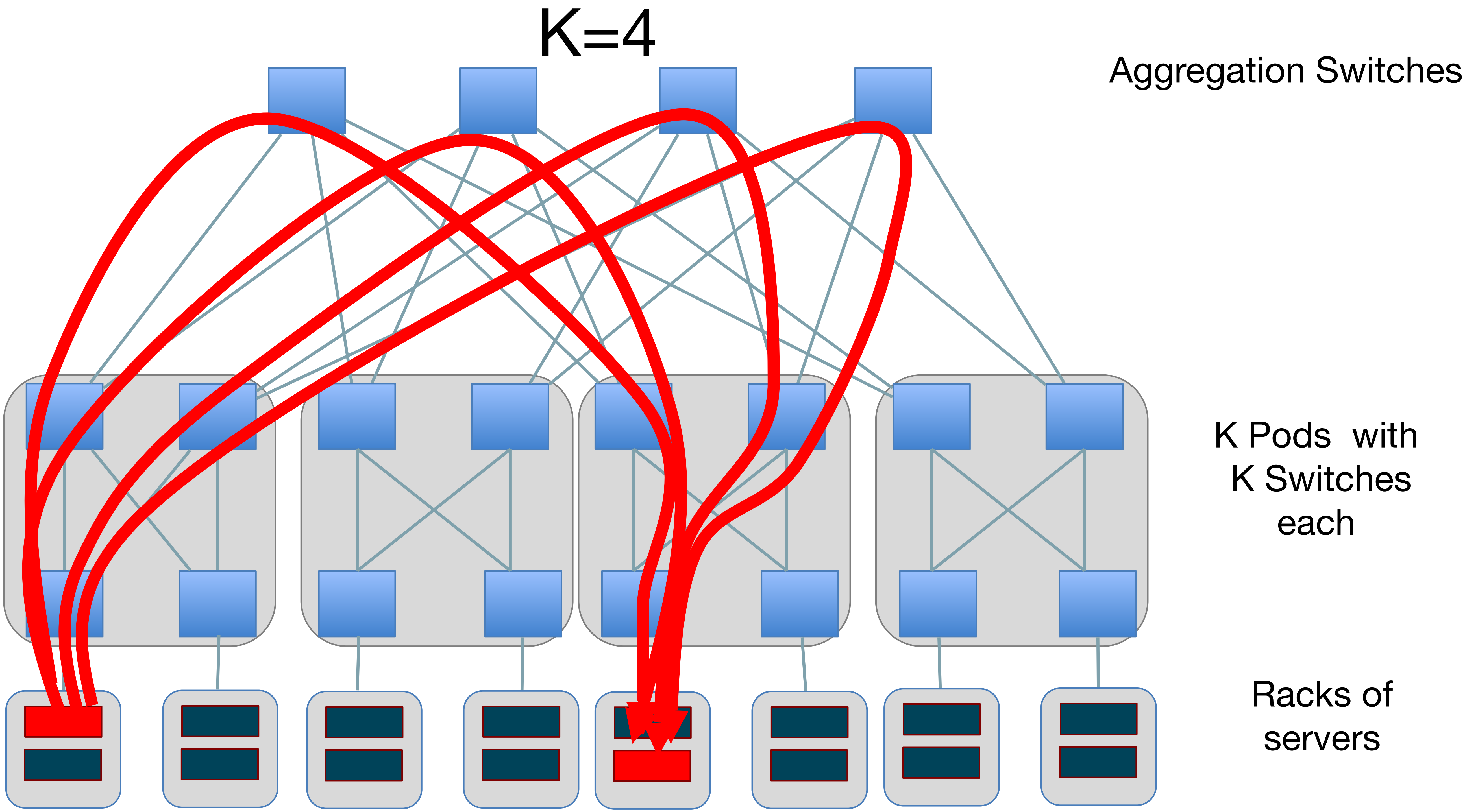
- Traditional topologies are tree-based
  - Poor performance
  - Not fault tolerant



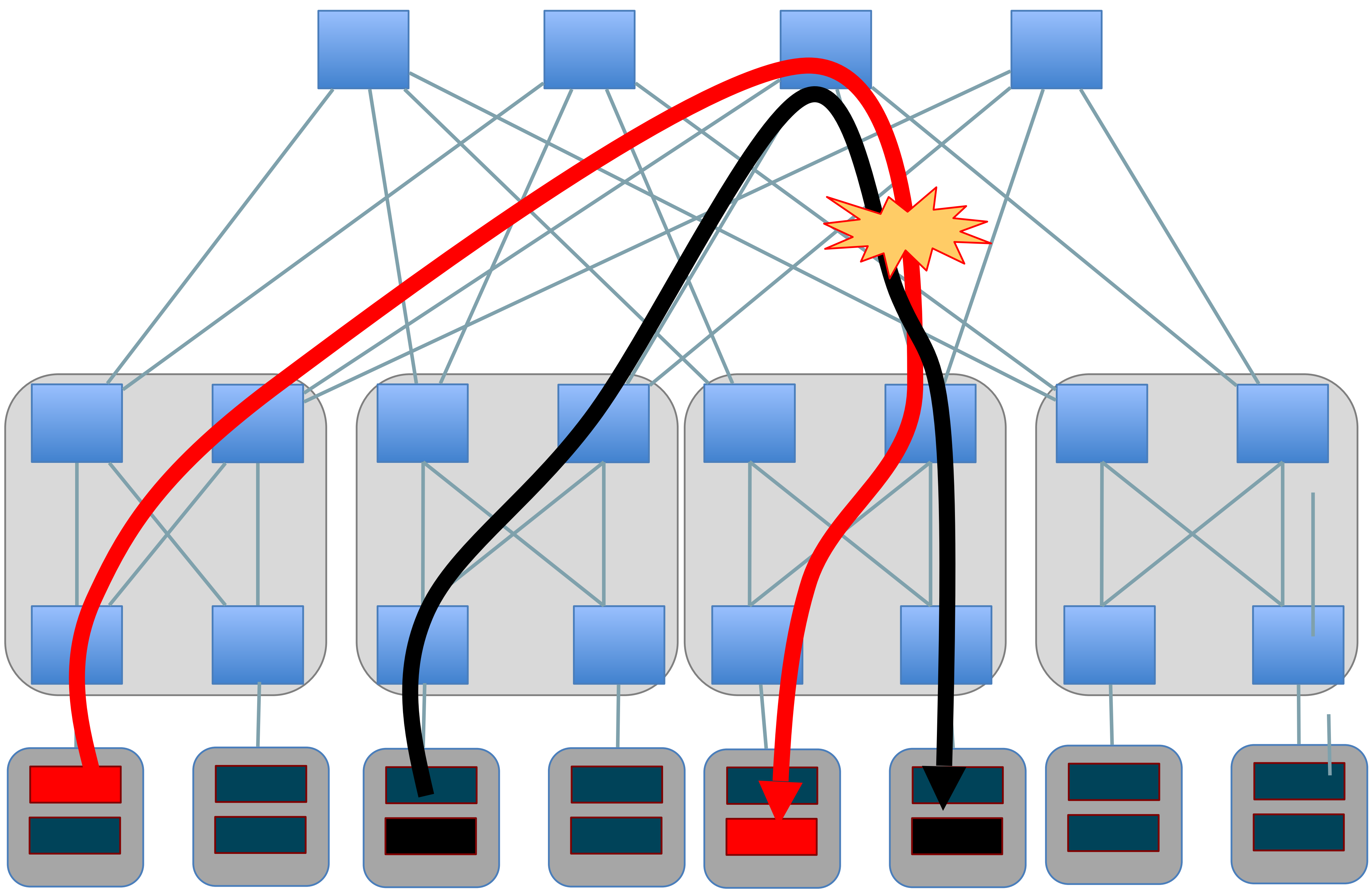
- Shift towards multipath topologies: FatTree, BCube, VL2, ….



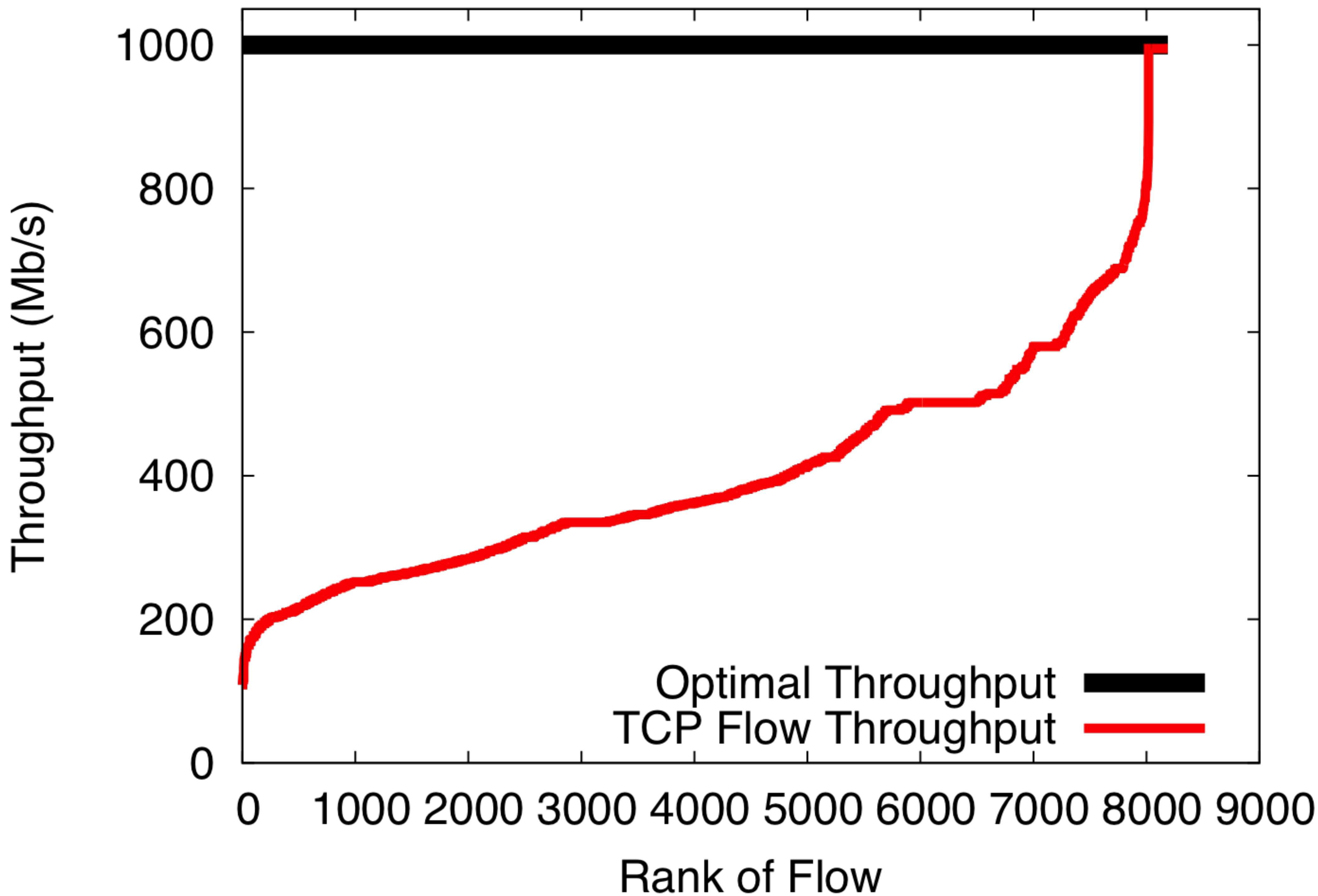
Fat Tree Topology



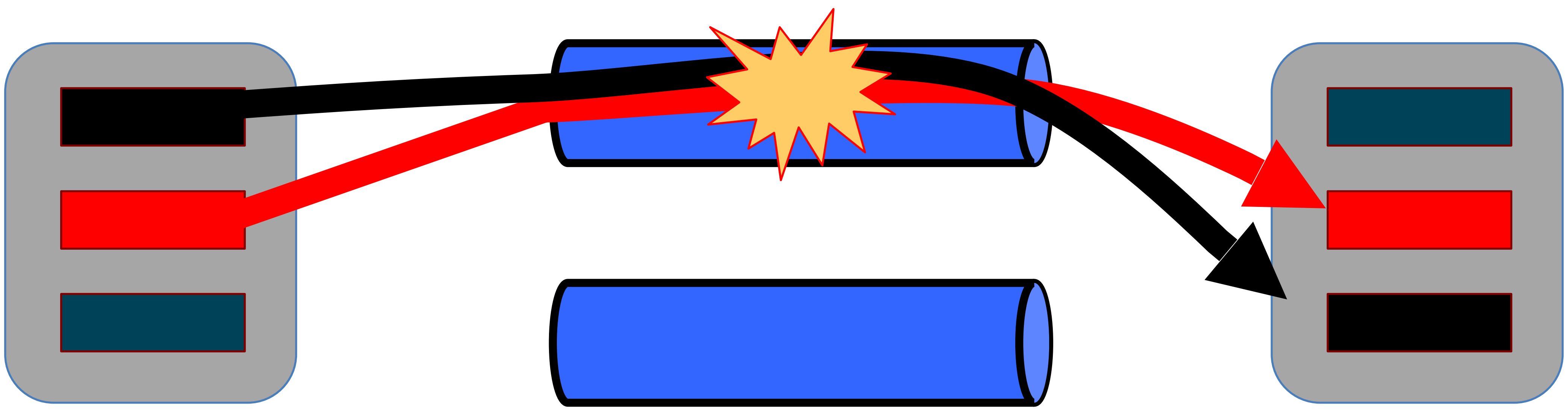
Collisions

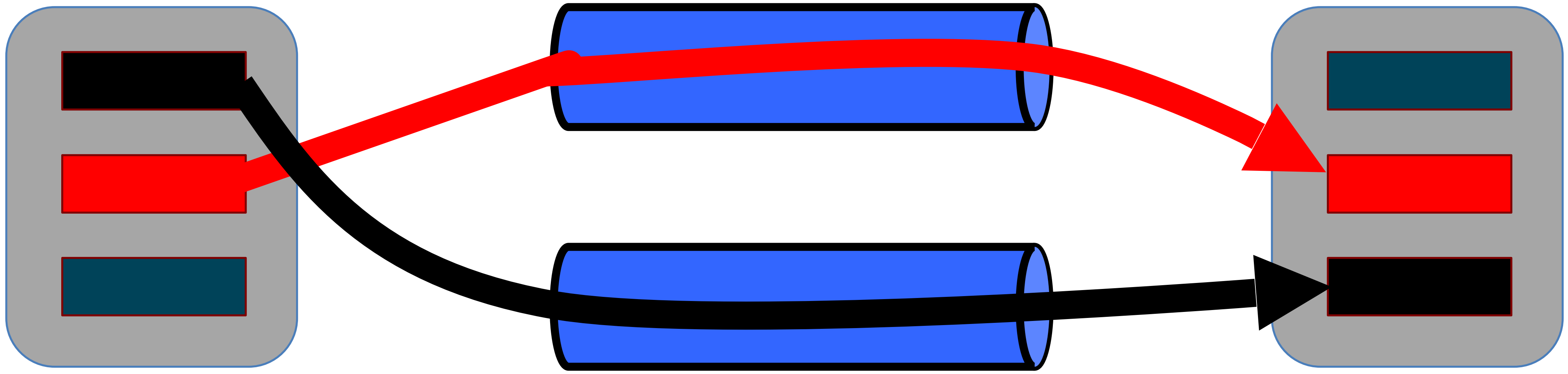


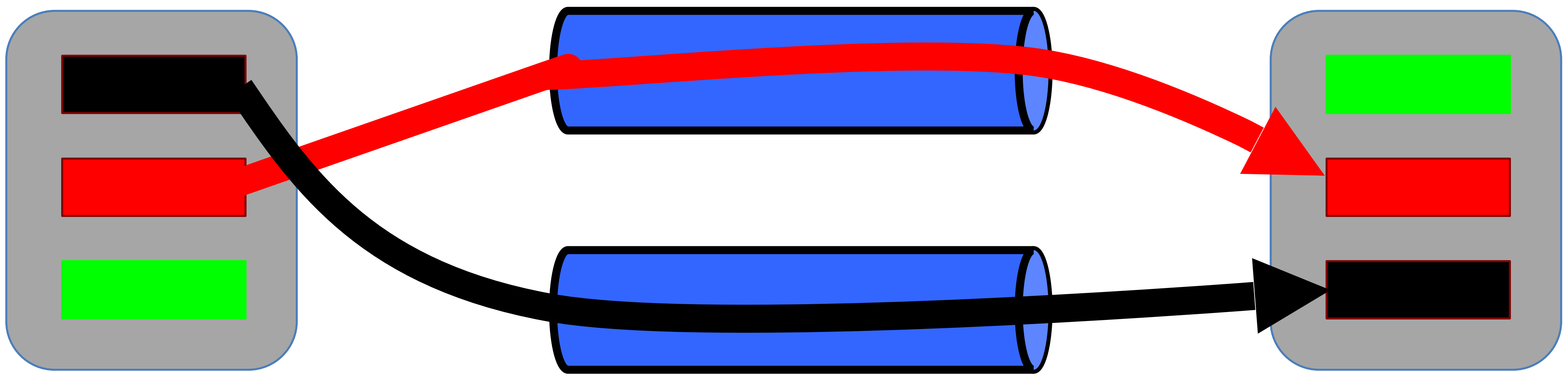
Single-path TCP collisions reduce throughput



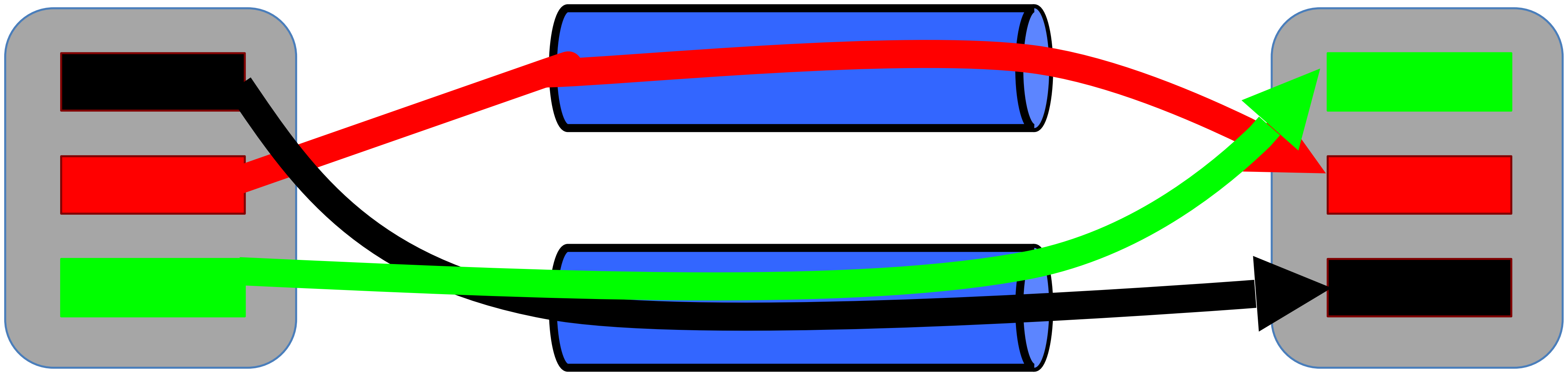
Collision



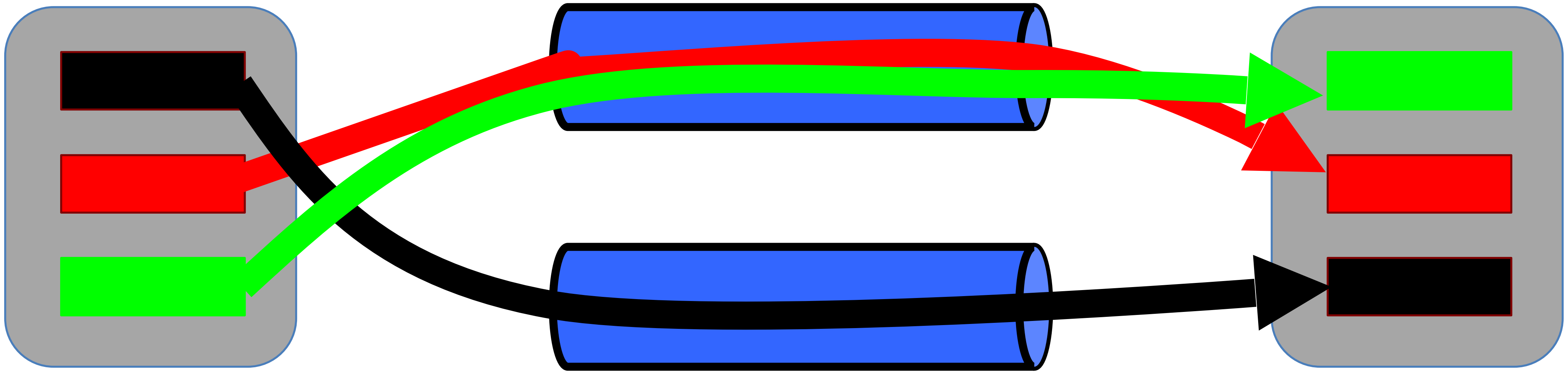




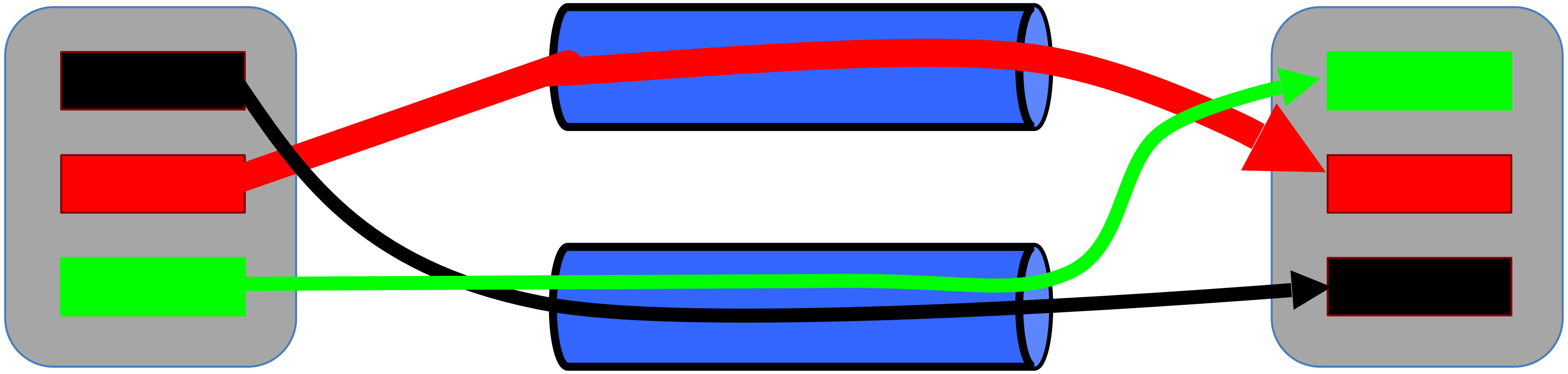
Not fair



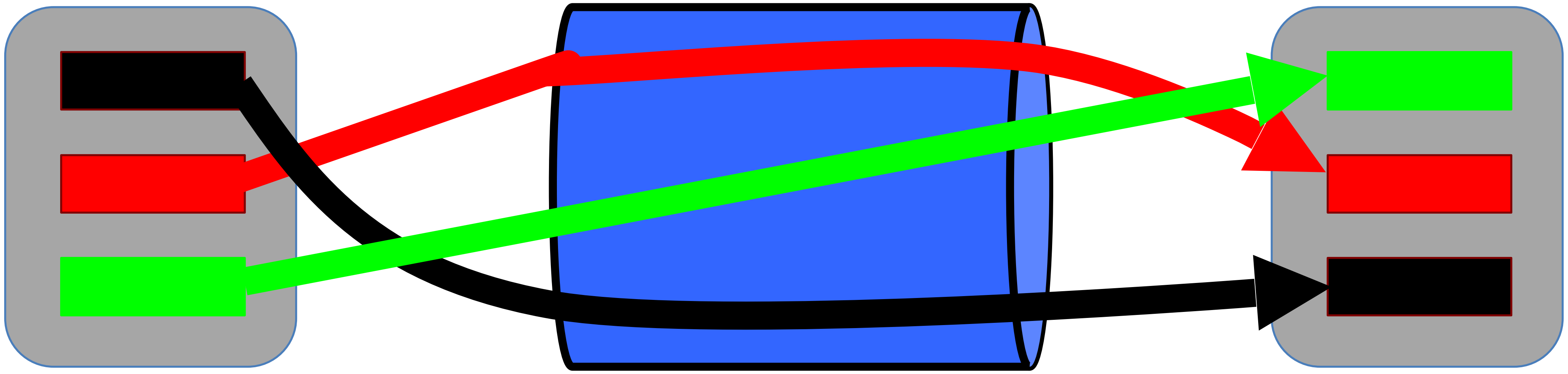
Not fair



No matter how you do it,
mapping each flow to a path is the wrong goal



Instead, we should pool capacity from different links

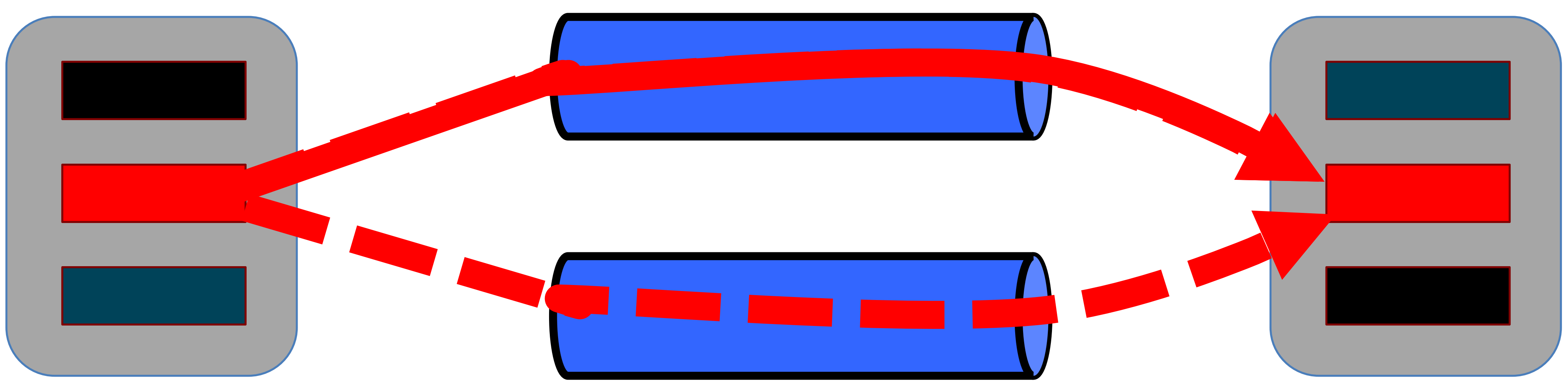


Multipath Transport can pool datacenter networks

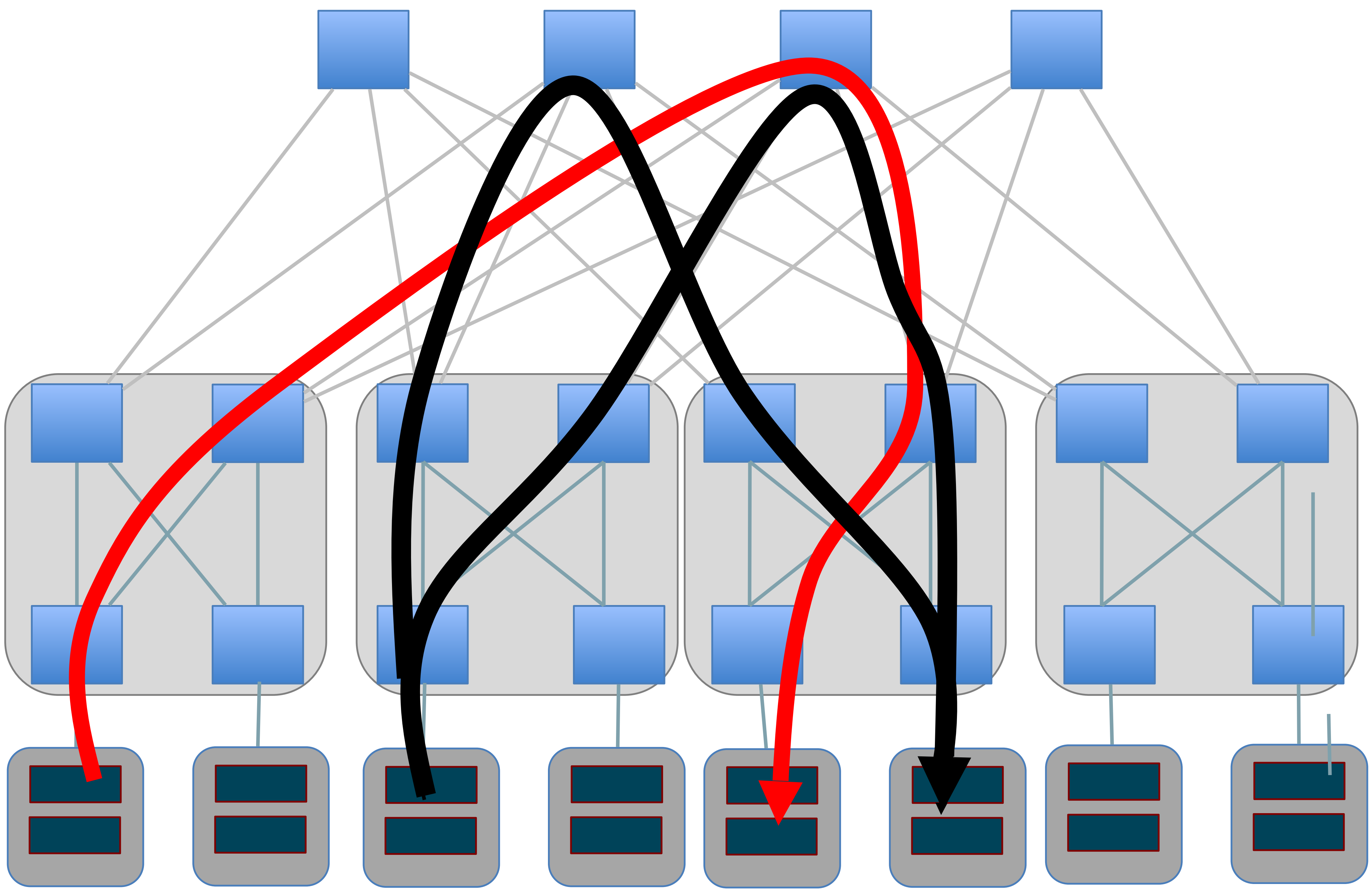
- Instead of using one path for each flow, use many random paths
- Don't worry about collisions.
- Just don't send (much) traffic on colliding paths

Multipath TCP Primer [IETF MPTCP WG]

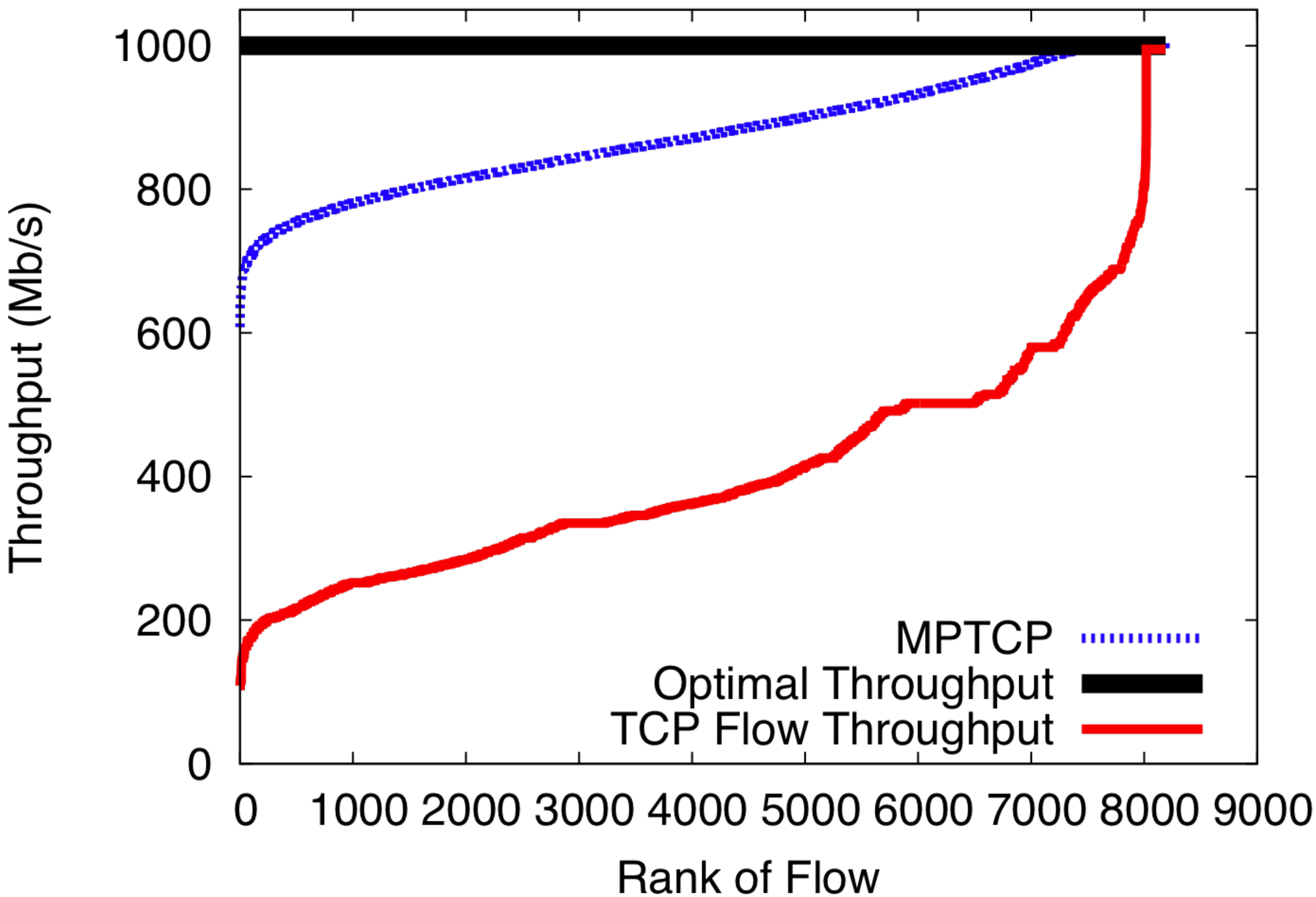
- MPTCP is a drop-in replacement for TCP
- MPTCP spreads application data over multiple subflows



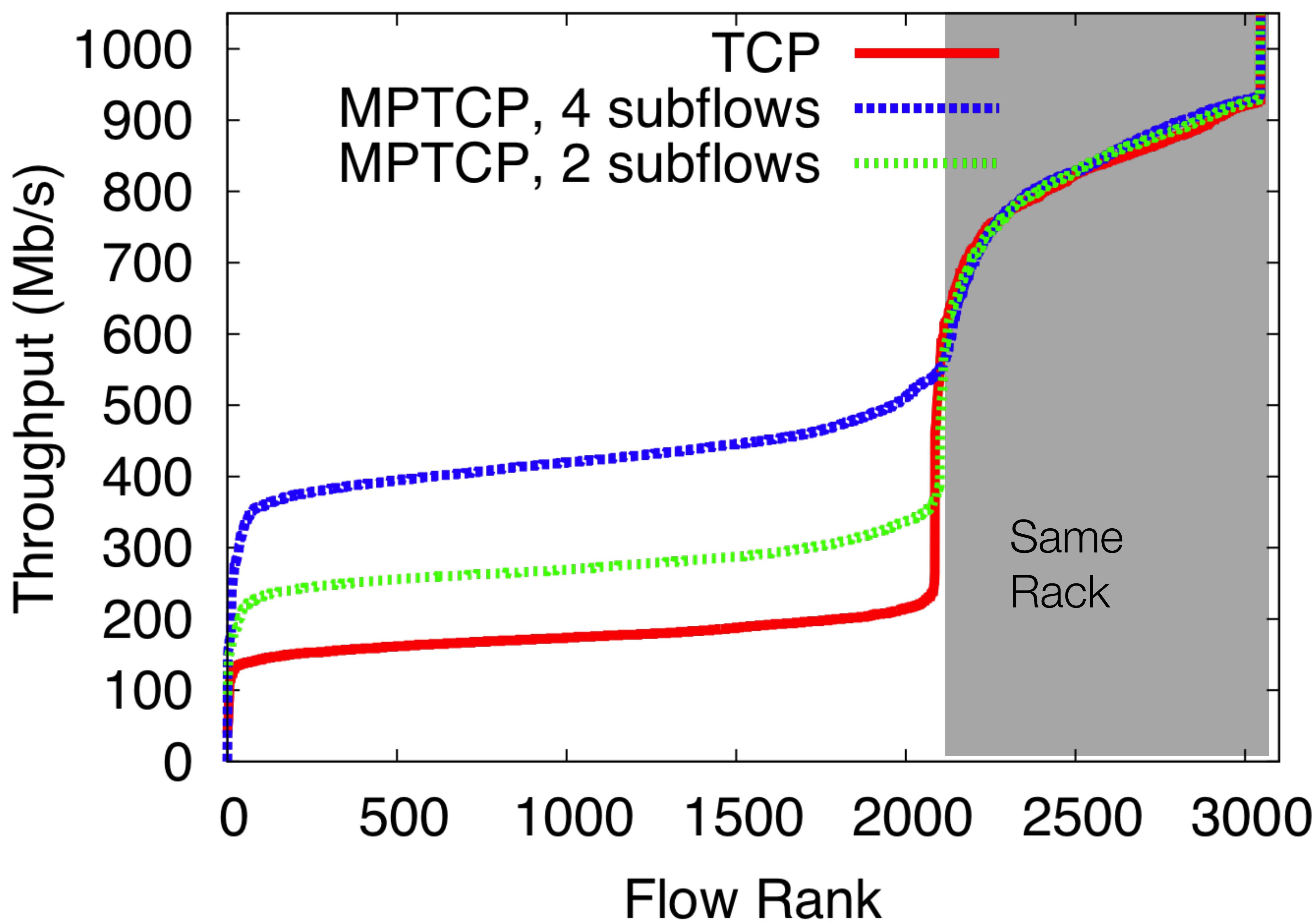
Multipath TCP: Congestion Control [NSDI, 2011]



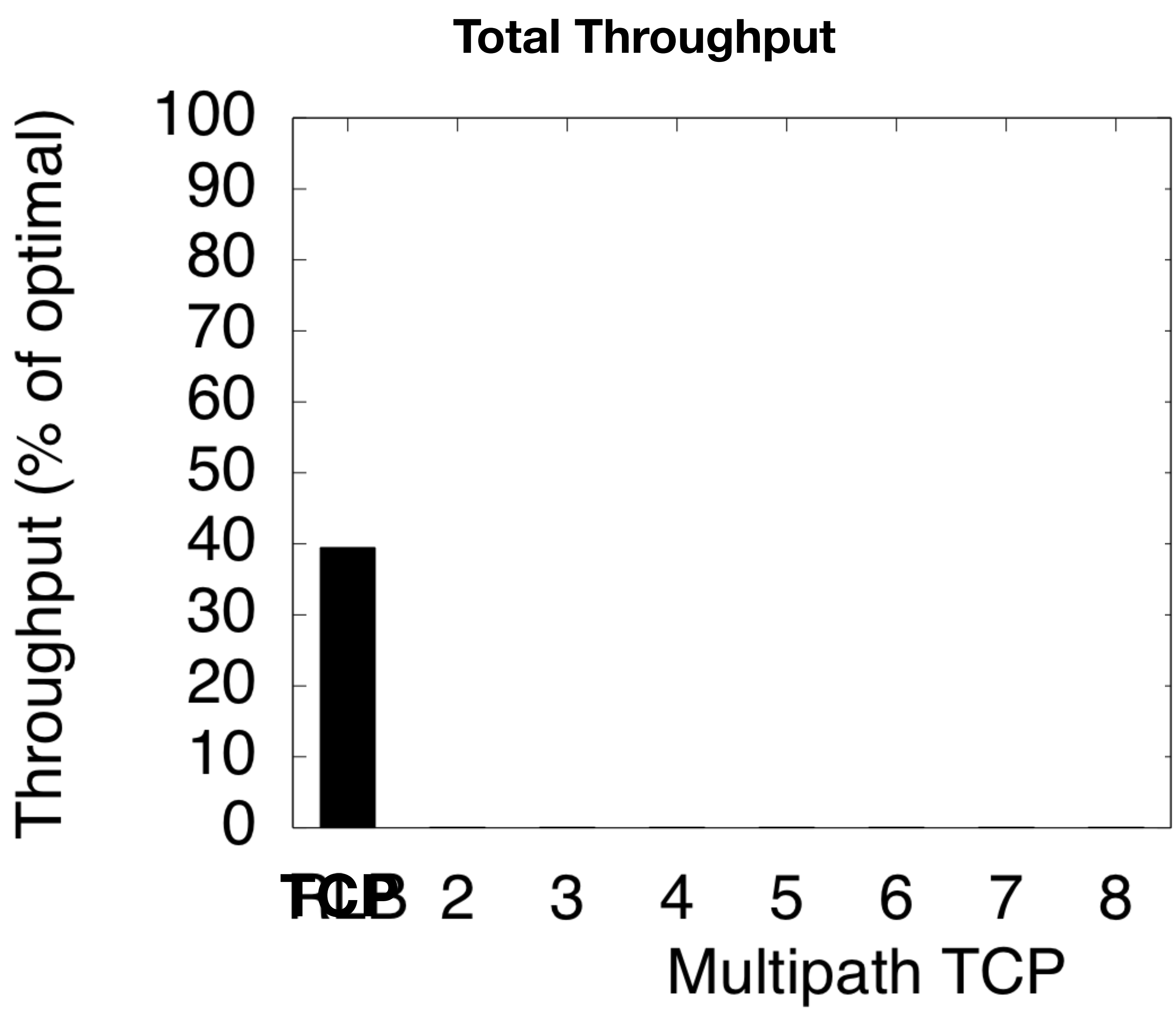
MPTCP better utilizes the Fat-Tree network



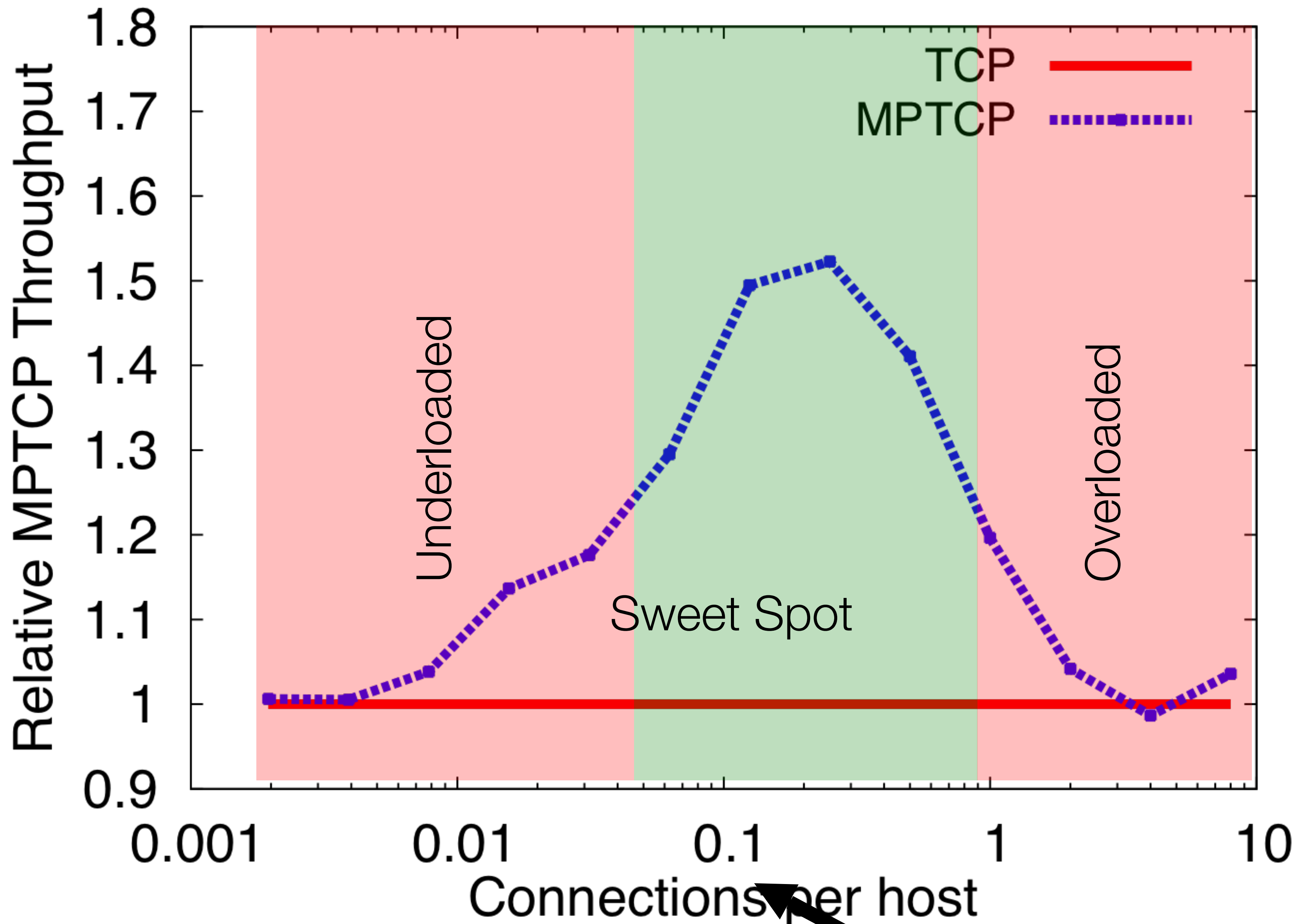
MPTCP improves performance on EC2



At most 8 subflows are needed



Performance improvements depend on traffic matrix



Increase Load