

Data Center TCP (DCTCP)

Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye
Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, Murari Sridharan

Microsoft Research

Stanford University

Data Center Packet Transport



- Large purpose-built DCs
 - Huge investment: R&D, business
- Transport **inside** the DC
 - TCP rules (99.9% of traffic)
- How's TCP doing?

TCP in the Data Center

- We'll see TCP does not meet demands of apps.
 - Suffers from bursty packet drops, Incast [SIGCOMM '09], ...
 - Builds up large queues:
 - Adds significant latency.
 - Wastes precious buffers, esp. bad with shallow-buffered switches.
- Operators work around TCP problems.
 - Ad-hoc, inefficient, often expensive solutions
 - No solid understanding of consequences, tradeoffs

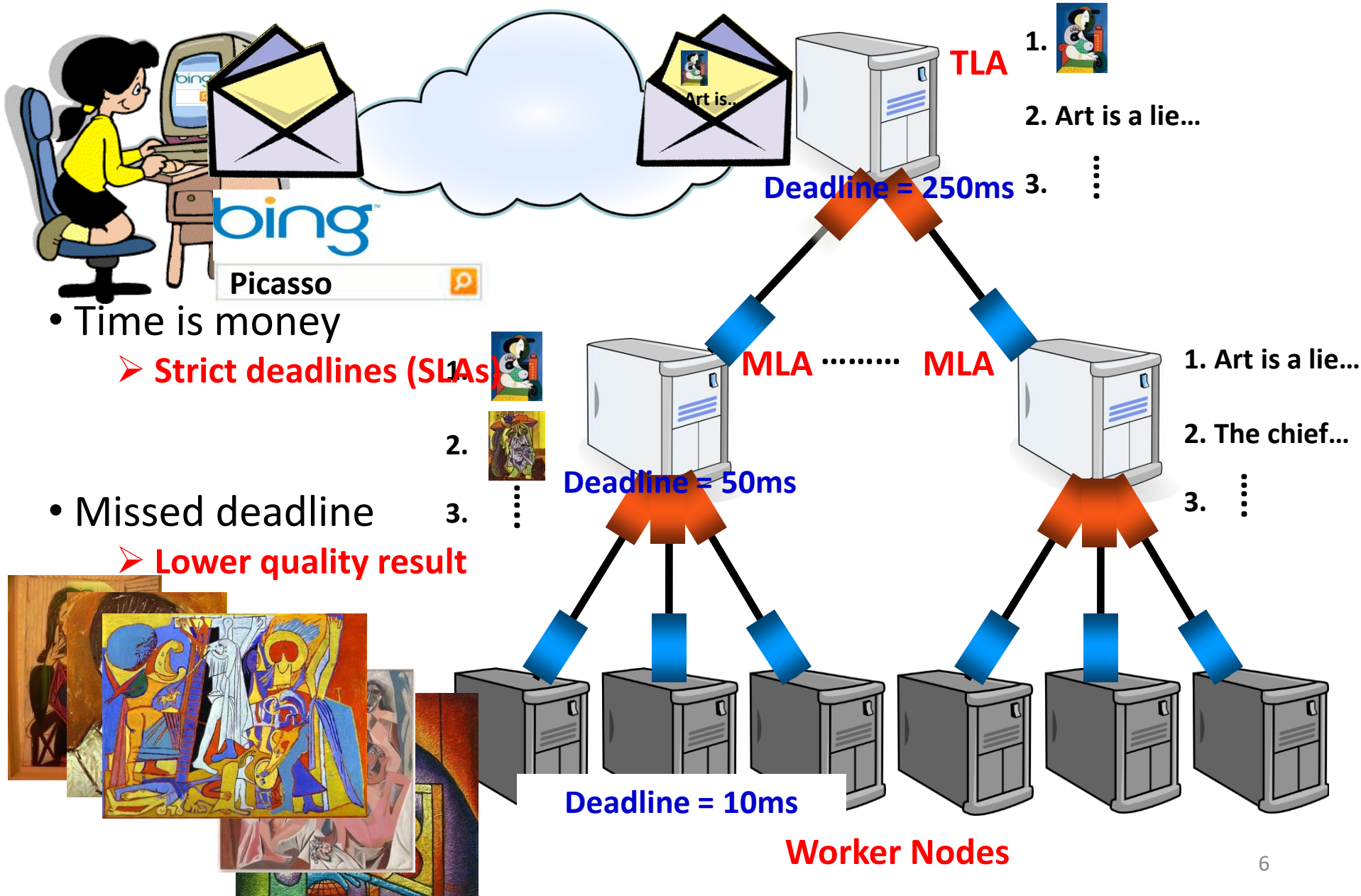
Roadmap

- What's really going on?
 - Interviews with developers and operators
 - Analysis of applications
 - Switches: shallow-buffered vs deep-buffered
 - Measurements
- A systematic study of transport in Microsoft's DCs
 - Identify **impairments**
 - Identify **requirements**
- Our solution: **Data Center TCP**

Case Study: Microsoft Bing

- Measurements from 6000 server production cluster
- Instrumentation passively collects logs
 - Application-level
 - Socket-level
 - Selected packet-level
- More than **150TB** of compressed data over a month

Partition/Aggregate Application Structure

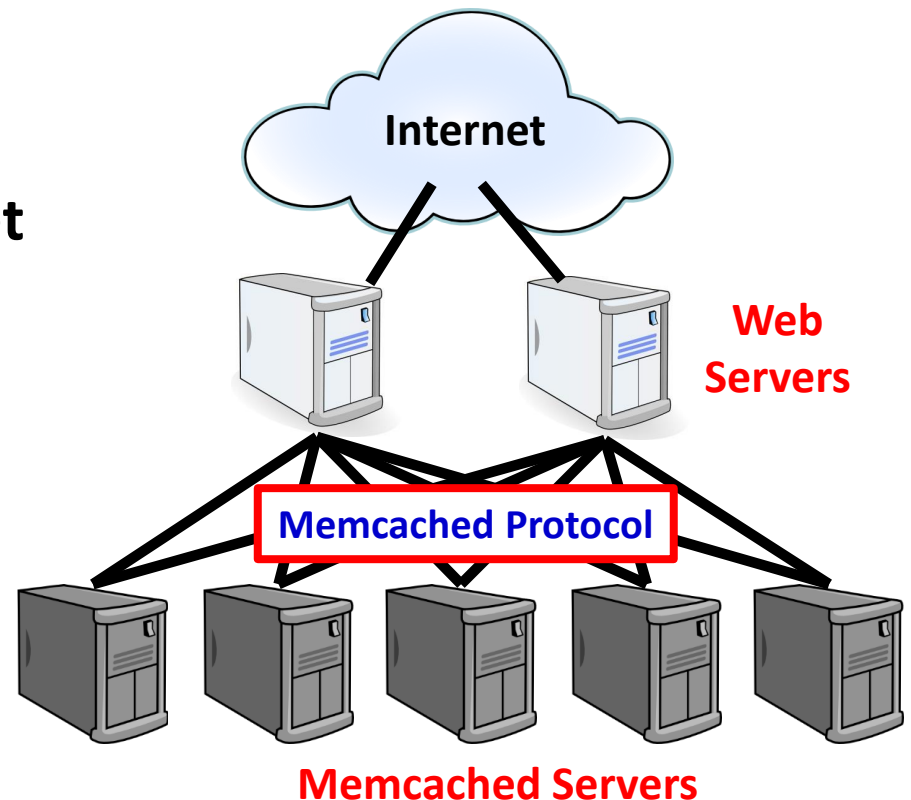


Generality of Partition/Aggregate

- The foundation for many large-scale web applications.
 - Web search, Social network composition, Ad selection, etc.
- Example: **Facebook**

Partition/Aggregate ~ Multiget

- Aggregators: **Web Servers**
- Workers: **Memcached Servers**



Workloads

- Partition/Aggregate
(Query)



Delay-sensitive



- Short messages [50KB-1MB]
(Coordination, Control state)



Delay-sensitive



- Large flows [1MB-50MB]
(Data update)



Throughput-sensitive



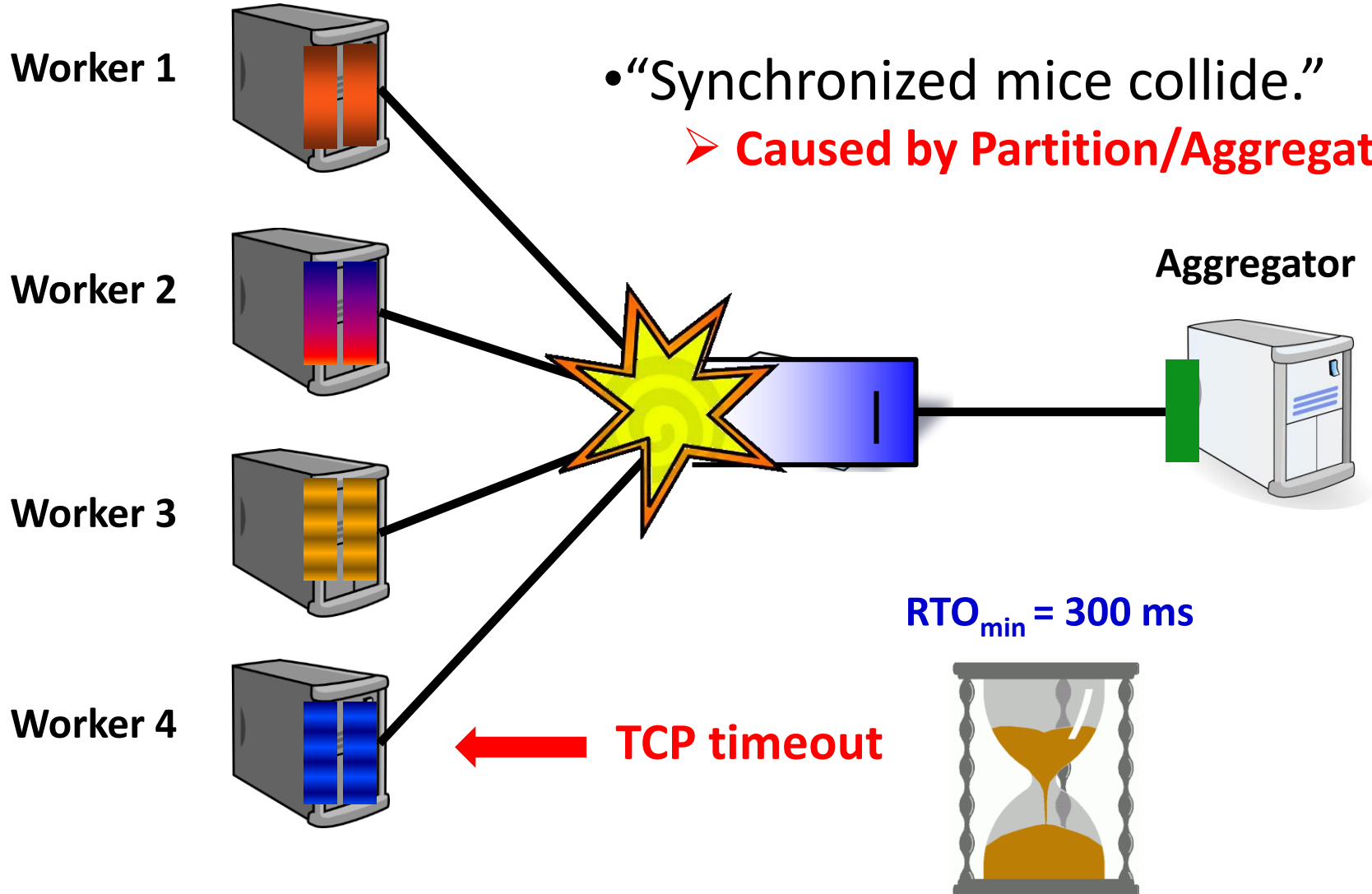
Impairments

- Incast
- Queue Buildup
- Buffer Pressure

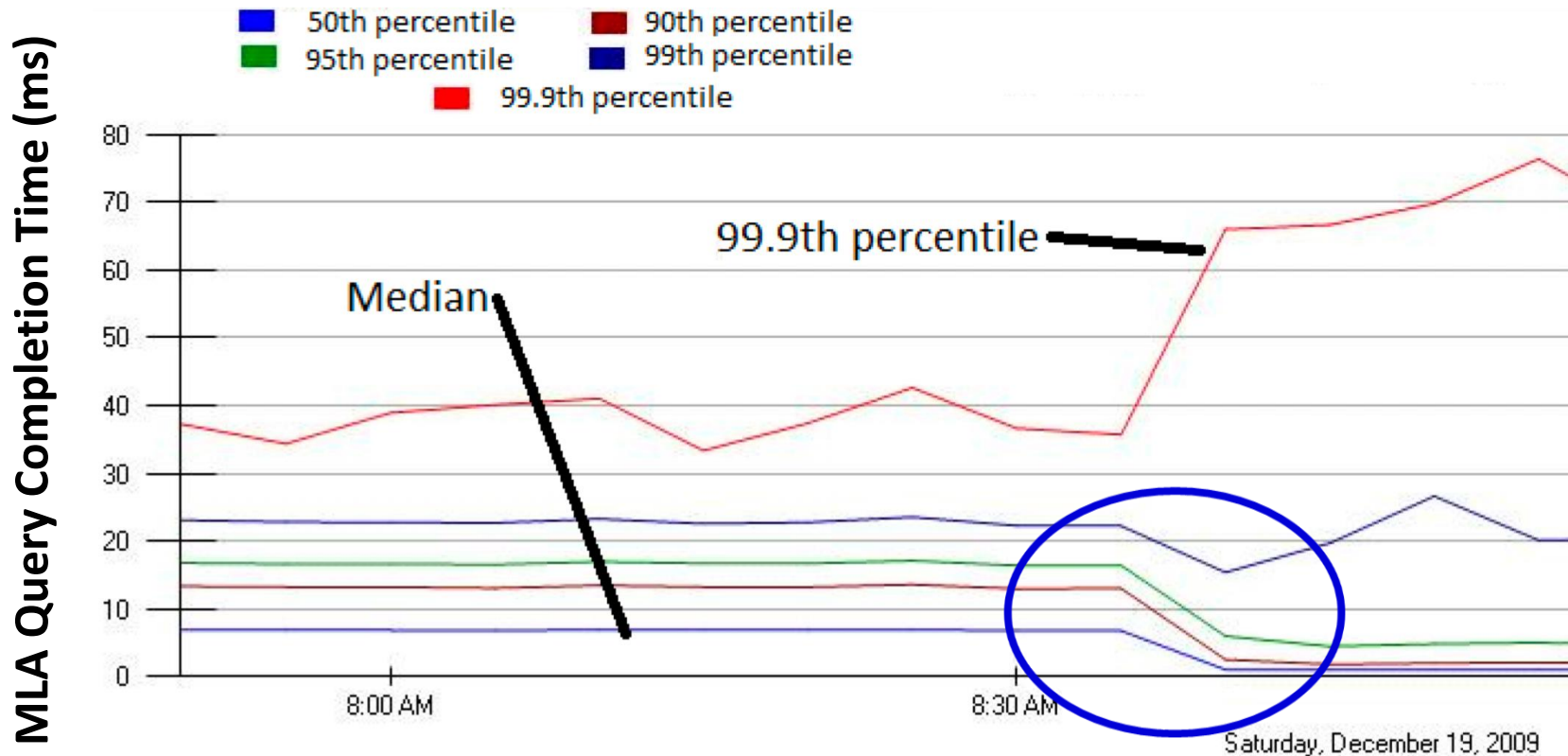
Incast

- “Synchronized mice collide.”

➤ **Caused by Partition/Aggregate.**



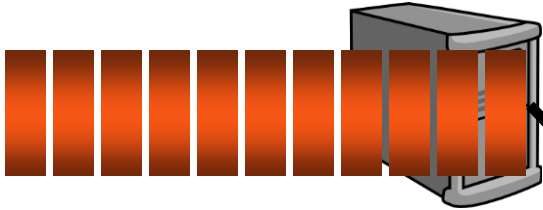
Incast Really Happens



Jittering 99.9th percentile is being tracked. ntiles.

Queue Buildup

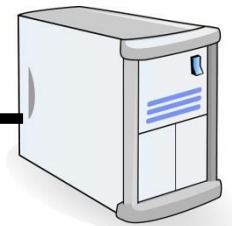
Sender 1



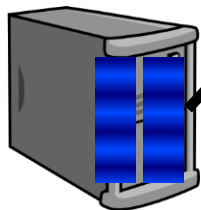
- Big flows buildup queues.

➤ Increased latency for short flows.

Receiver



Sender 2



- Measurements in Bing cluster

➤ For 90% packets: $RTT < 1ms$

➤ For 10% packets: $1ms < RTT < 15ms$

Data Center Transport Requirements

1. High Burst Tolerance

- Incast due to Partition/Aggregate is common.

2. Low Latency

- Short flows, queries

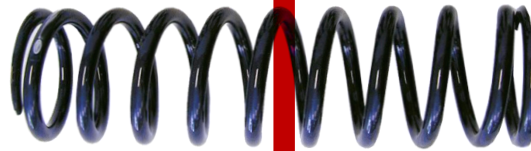
3. High Throughput

- Continuous data updates, large file transfers

The challenge is to achieve these three together.

Tension Between Requirements

High Throughput
High Burst Tolerance



Low Latency

Deep Buffers:

- Queuing Delays
Increase Latency

Shallow Buffers:

- Bad for Bursts &
Throughput

Reduced RTO_{min} (SIGCOMM '09)

- Doesn't Help Latency

AQM – RED:

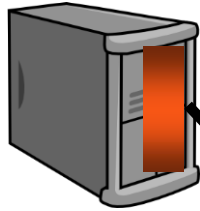
- Avg Queue Not Fast
Enough for Incast

Objective:
Low Queue Occupancy & High Throughput

The DCTCP Algorithm

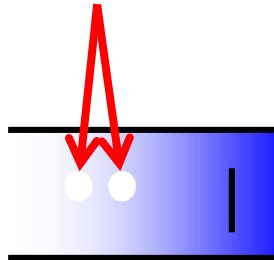
Review: The TCP/ECN Control Loop

Sender 1

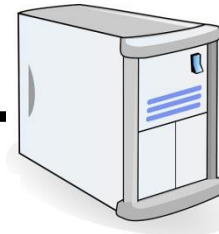


ECN = Explicit Congestion Notification

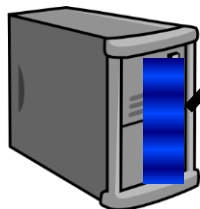
ECN Mark (1 bit)



Receiver

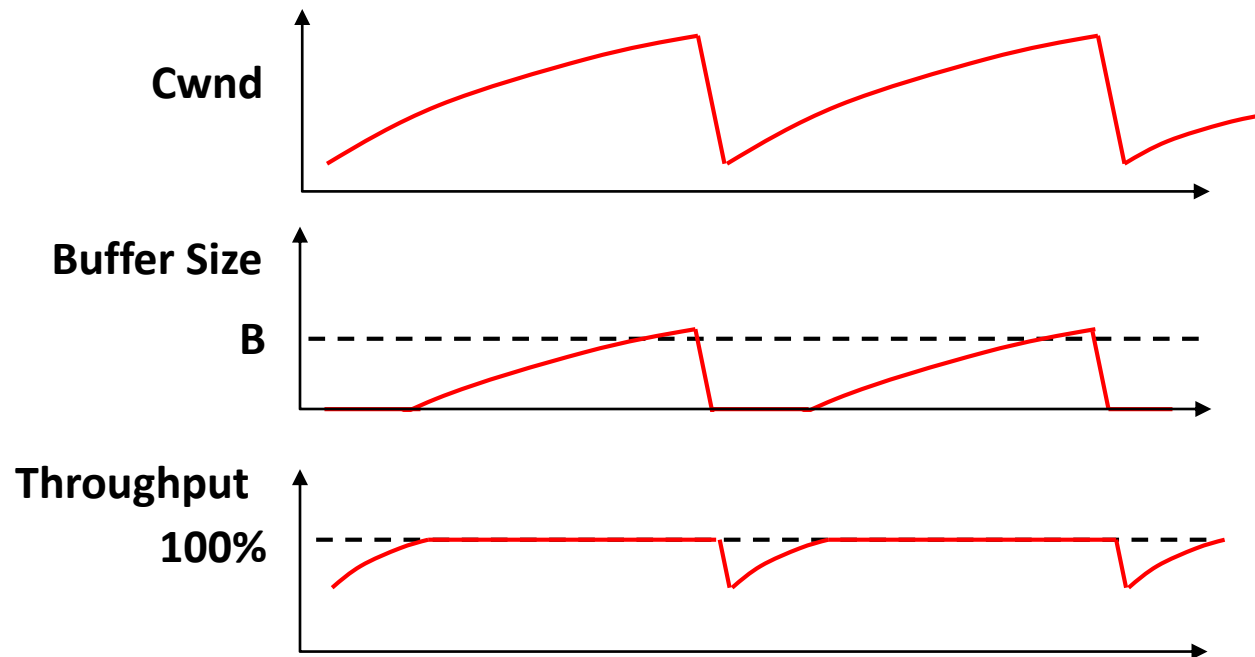


Sender 2



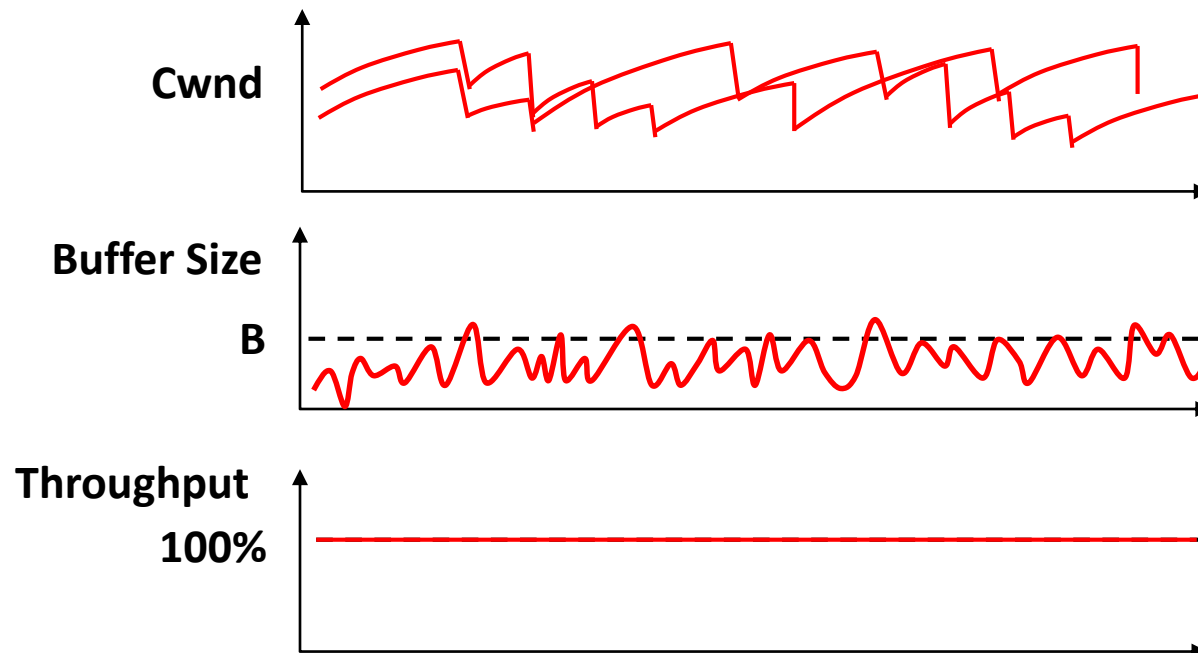
Small Queues & TCP Throughput: The Buffer Sizing Story

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



Small Queues & TCP Throughput: The Buffer Sizing Story

- Bandwidth-delay product rule of thumb:
 - A single flow needs $C \times RTT$ buffers for **100% Throughput**.
- Appenzeller rule of thumb (SIGCOMM '04):
 - Large # of flows: $C \times RTT / \sqrt{N}$ is enough.



Small Queues & TCP Throughput: The Buffer Sizing Story

- Bandwidth-delay product rule of thumb:
 - A single flow needs $C \times RTT$ buffers for **100% Throughput**.
- Appenzeller rule of thumb (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 1-2 big flows** at each server, **at most 4**.

Small Queues & TCP Throughput: The Buffer Sizing Story

- Bandwidth-delay product rule of thumb:
 - A single flow needs $C \times RTT$ buffers for **100% Throughput**.
- Appenzeller rule of thumb (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 1-2 big flows** at each server, **at most 4**.

Real Rule of Thumb:
Low Variance in Sending Rate → Small Buffers Suffice

Two Key Ideas

1. React in proportion to the **extent** of congestion, not its **presence**.
 - ✓ Reduces **variance** in sending rates, lowering queuing requirements.

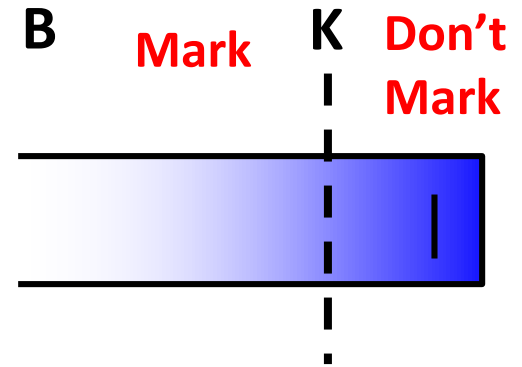
ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%

2. Mark based on **instantaneous** queue length.
 - ✓ Fast feedback to better deal with bursts.

Data Center TCP Algorithm

Switch side:

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



Sender side:

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

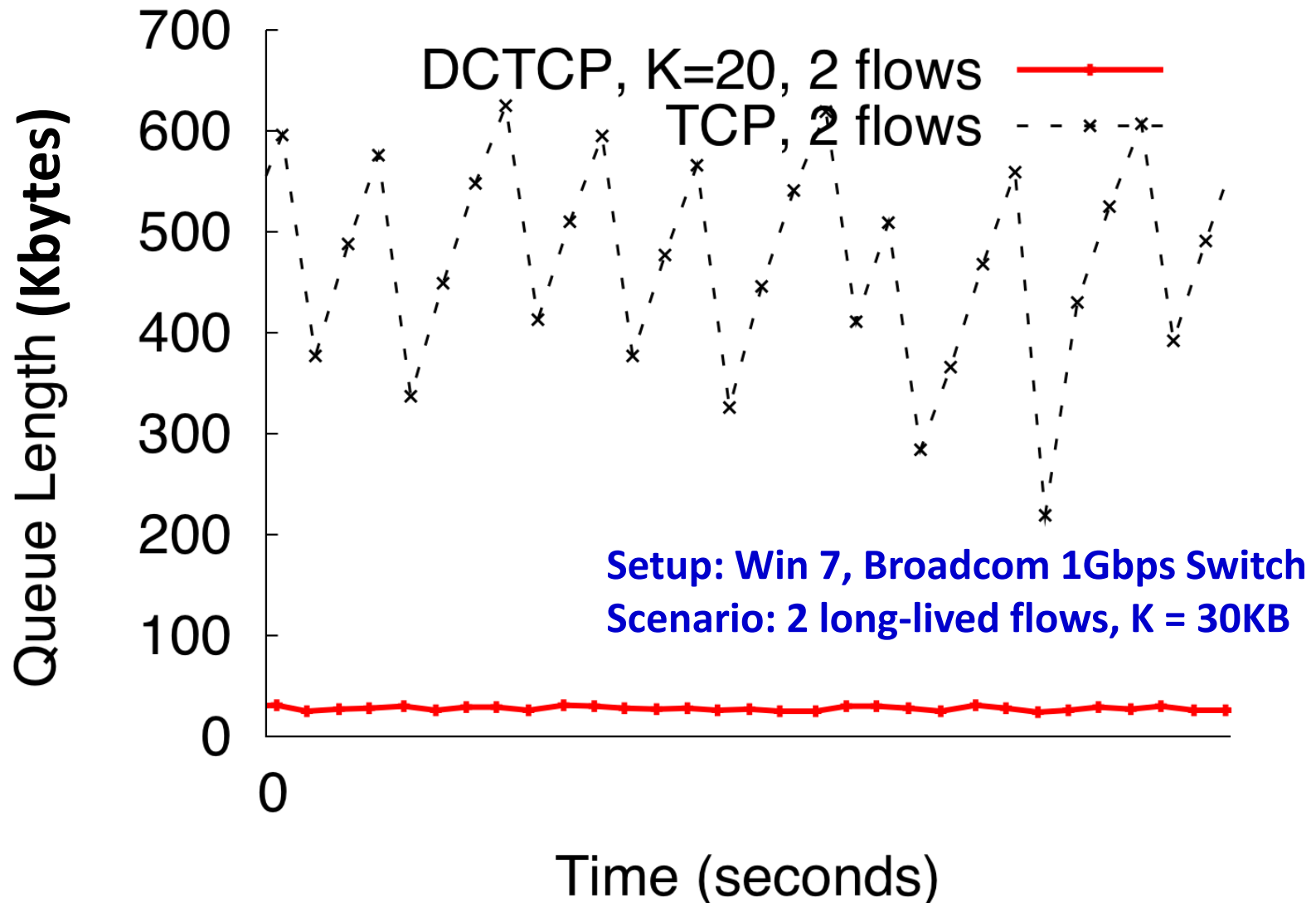
In each RTT:

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

➤ **Adaptive window decreases:** $cwnd \leftarrow (1 - \frac{\alpha}{2})cwnd$

- Note: decrease factor between 1 and 2.

DCTCP in Action



Why it Works

1. High Burst Tolerance

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

2. Low Latency

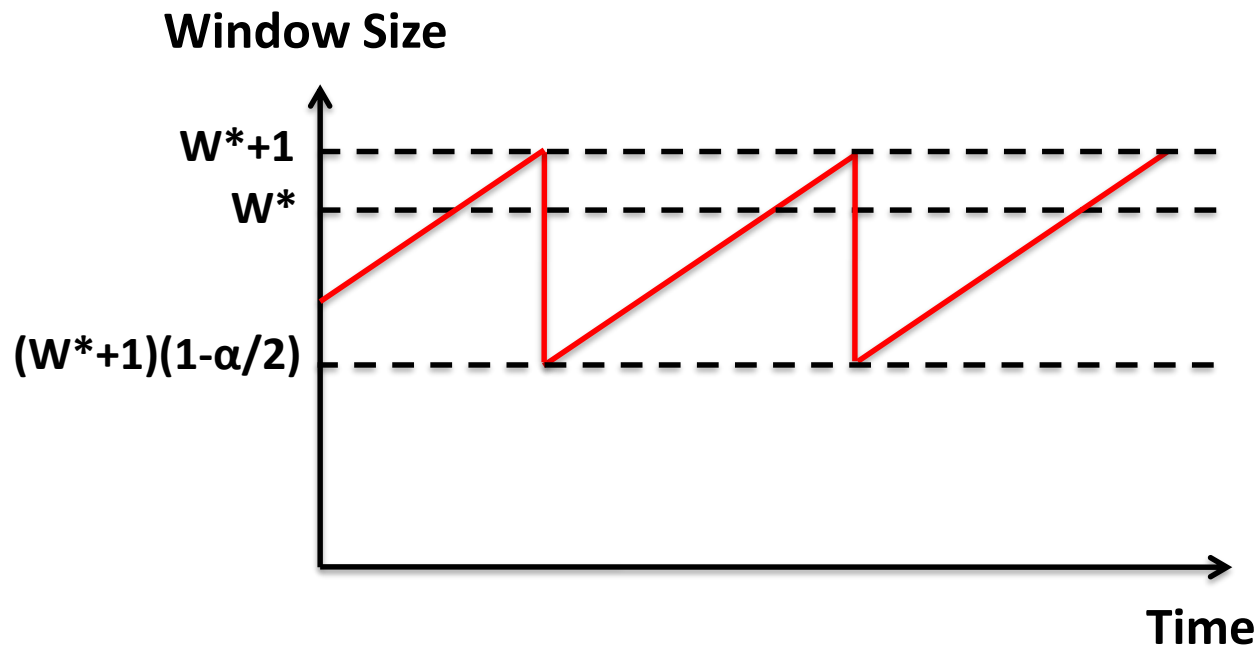
- ✓ **Small buffer occupancies** → low queuing delay.

3. High Throughput

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

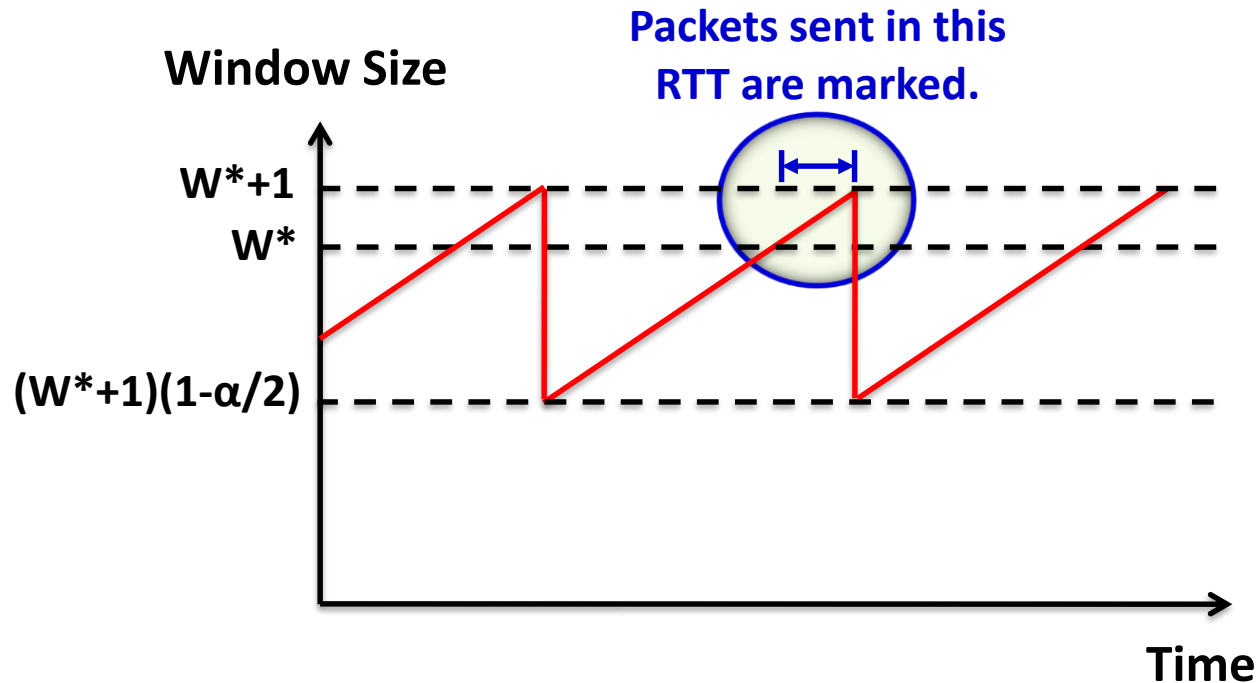
Analysis

- How low can DCTCP maintain queues without loss of throughput?
- How do we set the DCTCP parameters?
 - **Need to quantify queue size oscillations (Stability).**



Analysis

- How low can DCTCP maintain queues without loss of throughput?
- How do we set the DCTCP parameters?
 - **Need to quantify queue size oscillations (Stability).**



Analysis

- How low can DCTCP maintain queues without loss of throughput?
- How do we set the DCTCP parameters?
 - **Need to quantify queue size oscillations (Stability).**

$$K > \frac{1}{7} C \times RTT$$

85% Less Buffer than TCP

Evaluation

- Implemented in Windows stack.
- Real hardware, **1Gbps and 10Gbps** experiments
 - **90 server testbed**
 - **Broadcom Triumph** 48 1G ports – 4MB shared memory
 - **Cisco Cat4948** 48 1G ports – 16MB shared memory
 - **Broadcom Scorpion** 24 10G ports – 4MB shared memory
- Numerous micro-benchmarks
 - **Throughput and Queue Length**
 - **Multi-hop**
 - **Queue Buildup**
 - **Buffer Pressure**
 - **Fairness and Convergence**
 - **Incast**
 - **Static vs Dynamic Buffer Mgmt**
- **Cluster traffic benchmark**

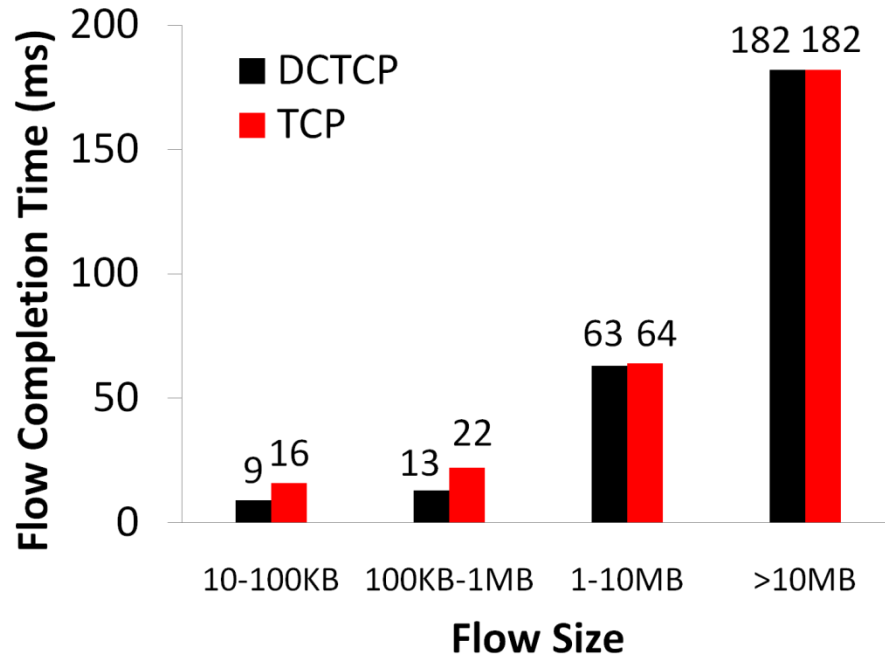
Cluster Traffic Benchmark

- Emulate traffic within 1 Rack of Bing cluster
 - 45 1G servers, 10G server for external traffic
- Generate query, and background traffic
 - Flow sizes and arrival times follow distributions seen in Bing
- Metric:
 - Flow completion time for queries and background flows.

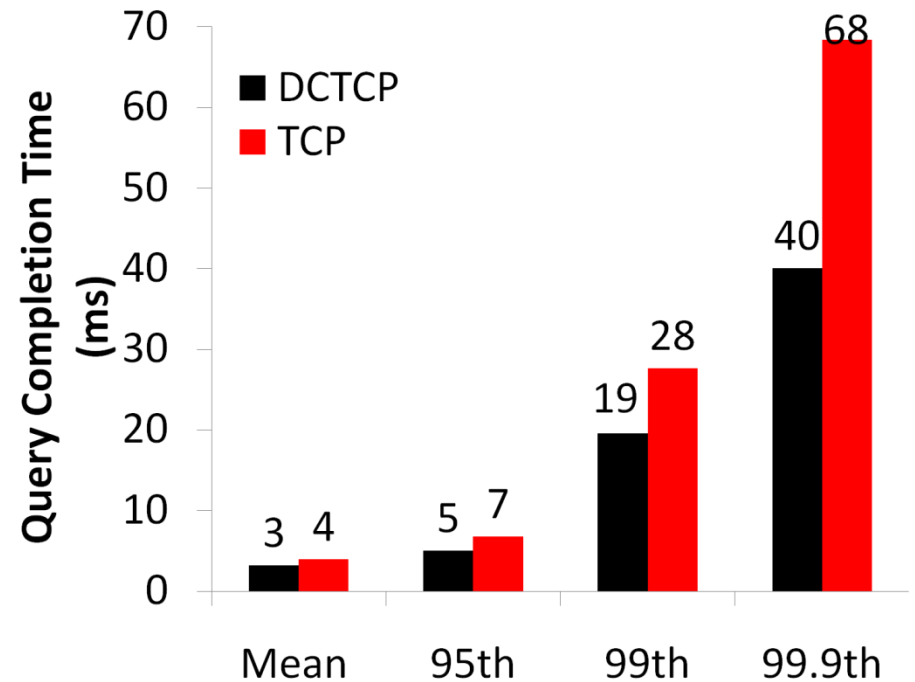
We use $RTO_{\min} = 10\text{ms}$ for both TCP & DCTCP.

Baseline

Background Flows

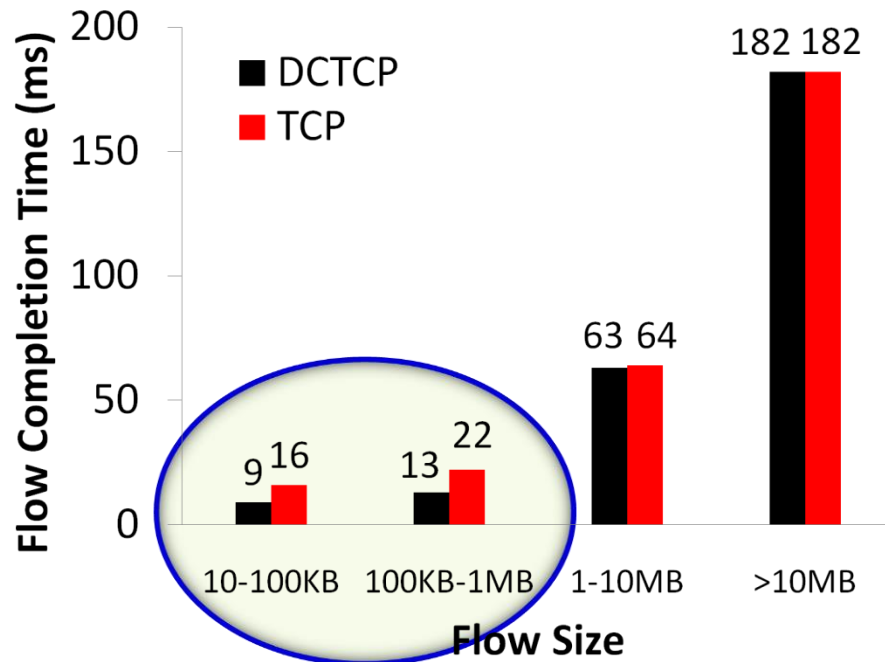


Query Flows

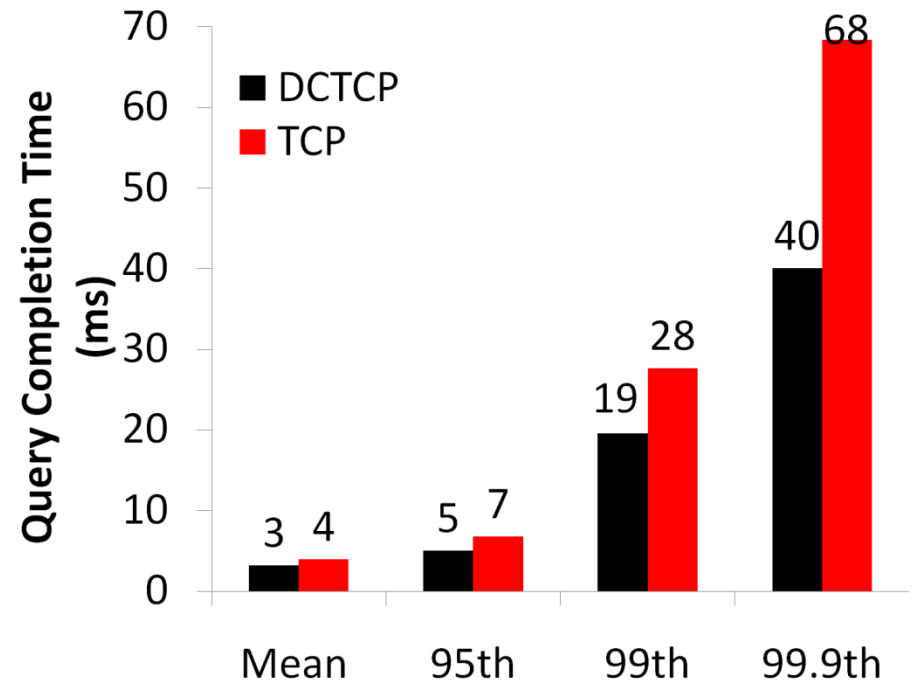


Baseline

Background Flows



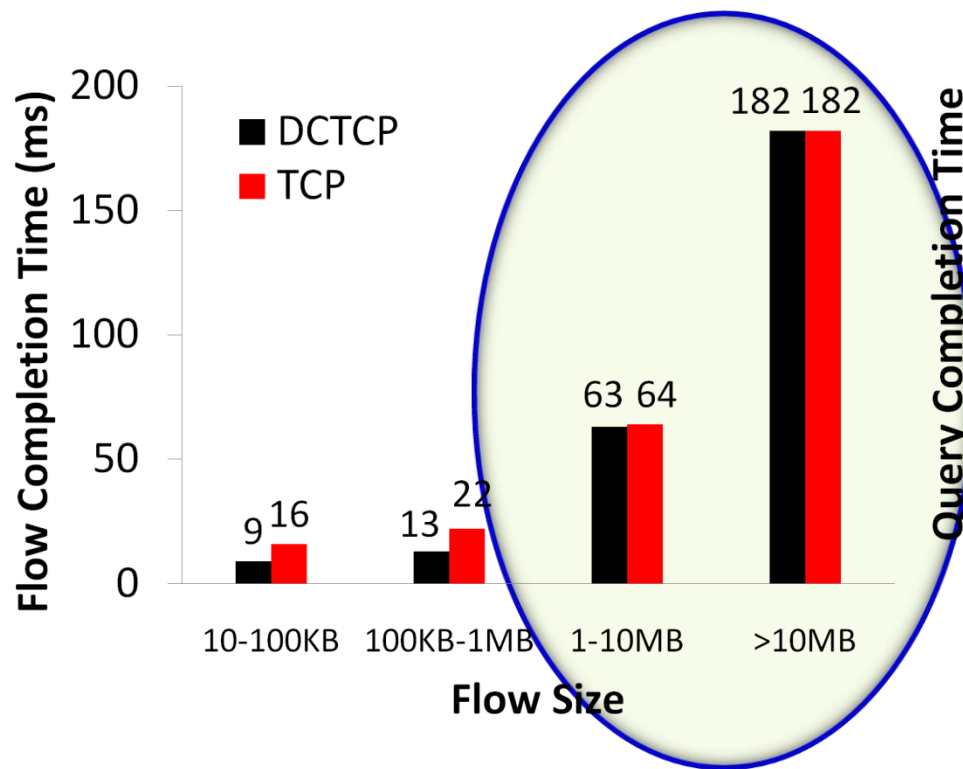
Query Flows



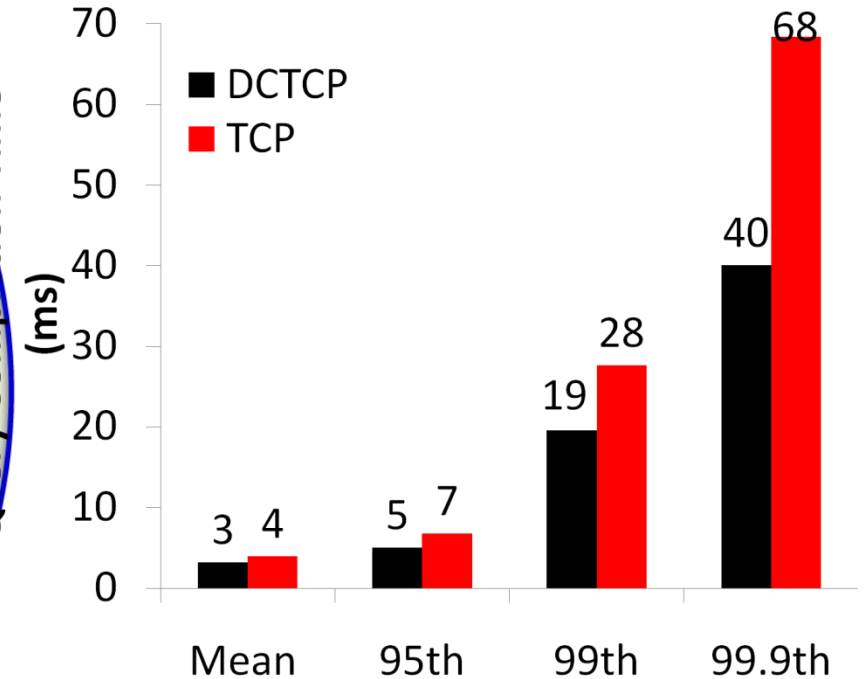
✓ Low latency for short flows.

Baseline

Background Flows



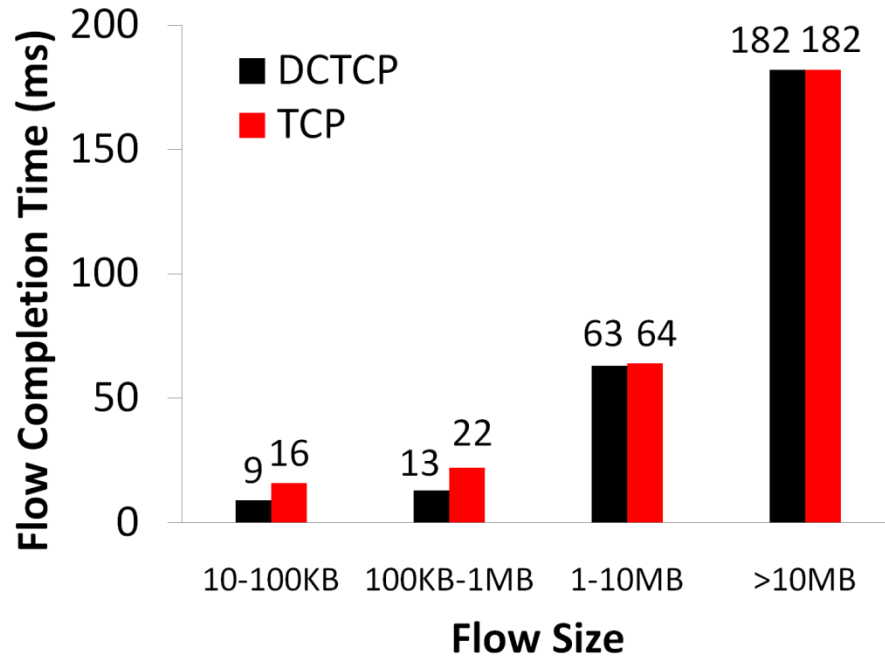
Query Flows



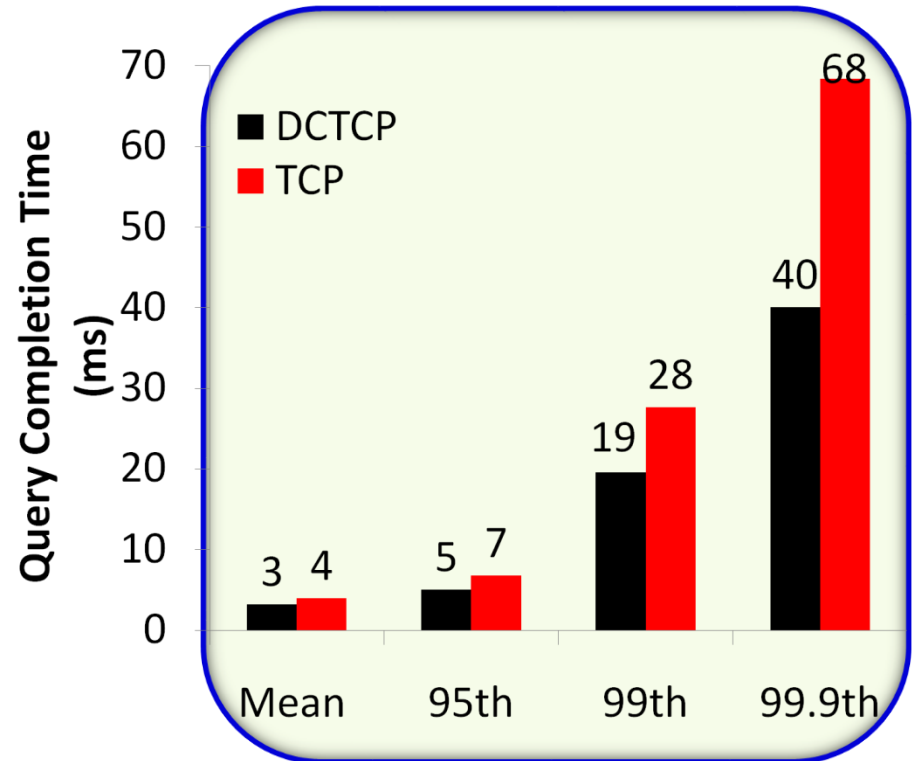
- ✓ Low latency for short flows.
- ✓ High throughput for long flows.

Baseline

Background Flows



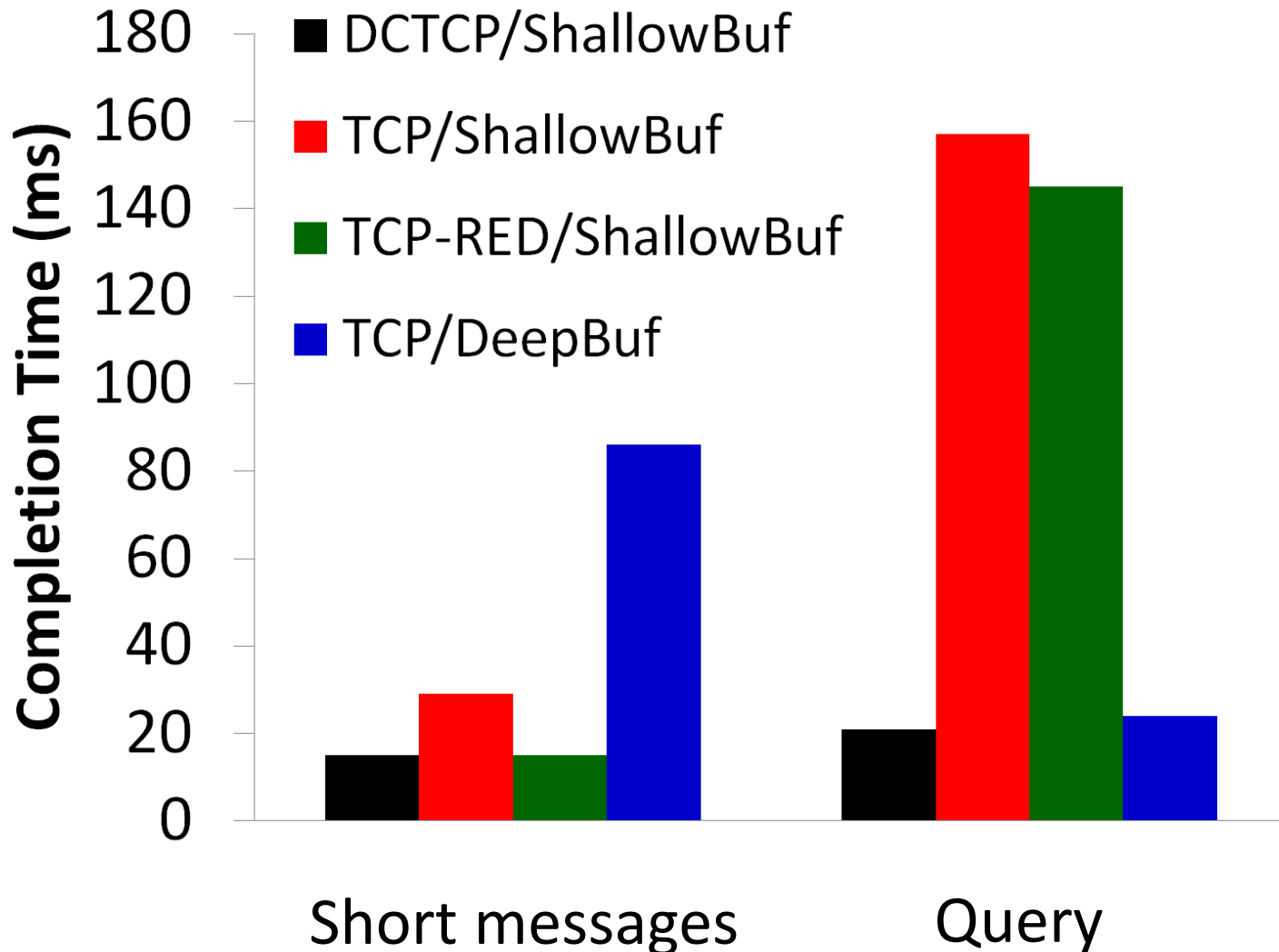
Query Flows



- ✓ Low latency for short flows.
- ✓ High throughput for long flows.
- ✓ High burst tolerance for query flows.

Scaled Background & Query

10x Background, 10x Query



Conclusions

- DCTCP satisfies all our requirements for Data Center packet transport.
 - ✓ **Handles bursts well**
 - ✓ **Keeps queuing delays low**
 - ✓ **Achieves high throughput**
- Features:
 - ✓ **Very simple change to TCP and a single switch parameter.**
 - ✓ **Based on mechanisms already available in Silicon.**

