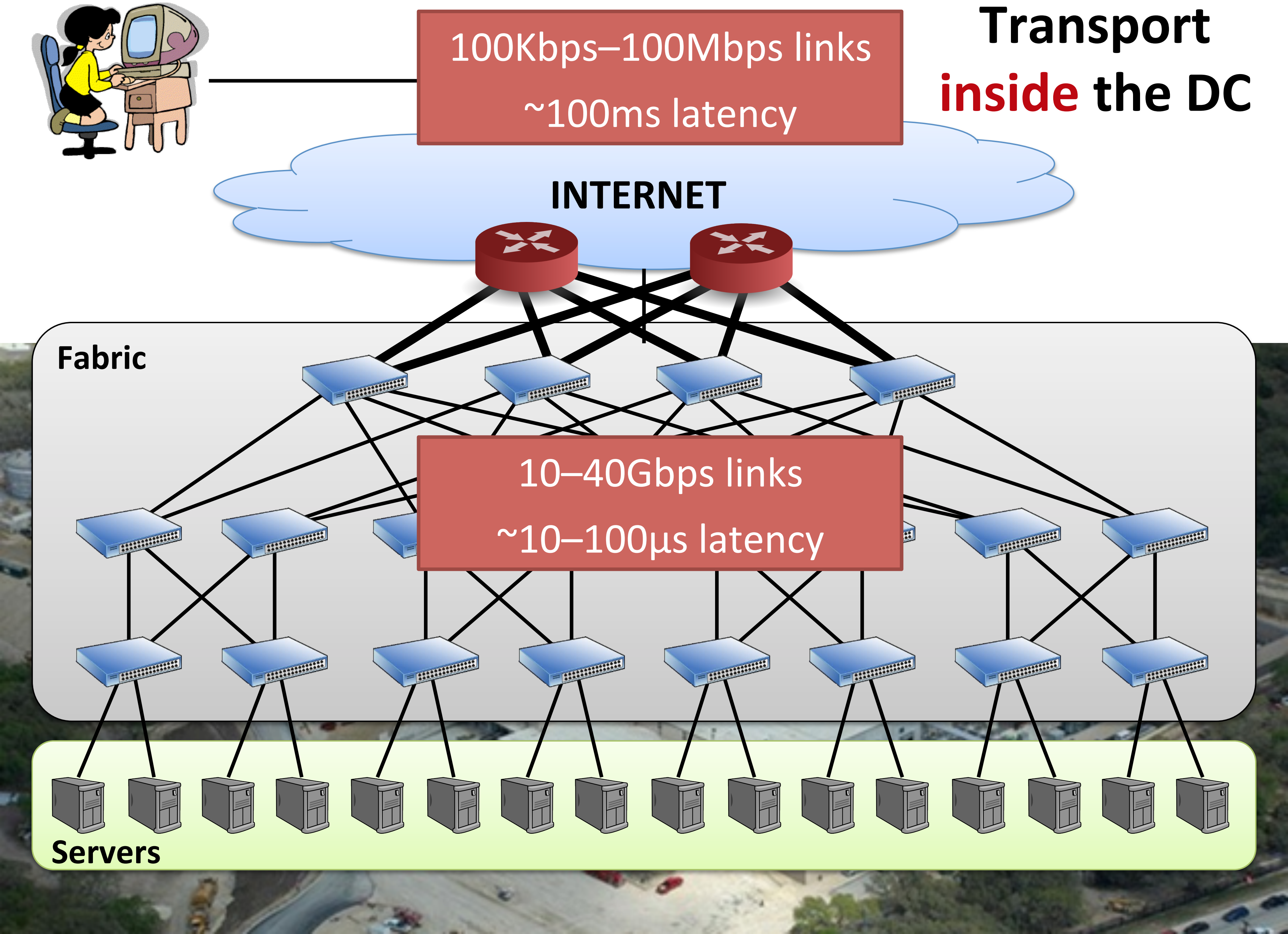
6.888:
Lecture 3
Data Center Congestion Control

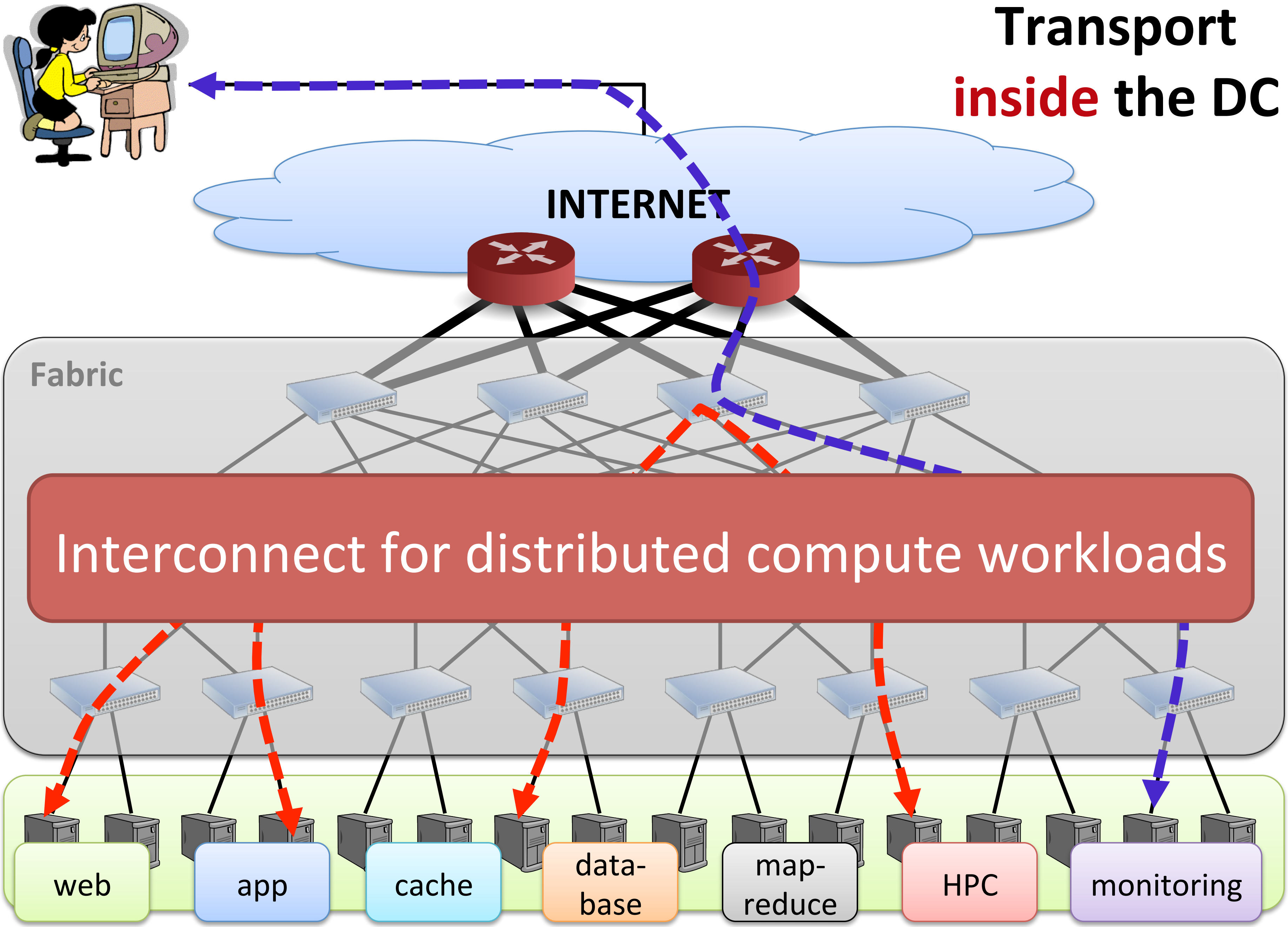
Mohammad Alizadeh

Spring 2016

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

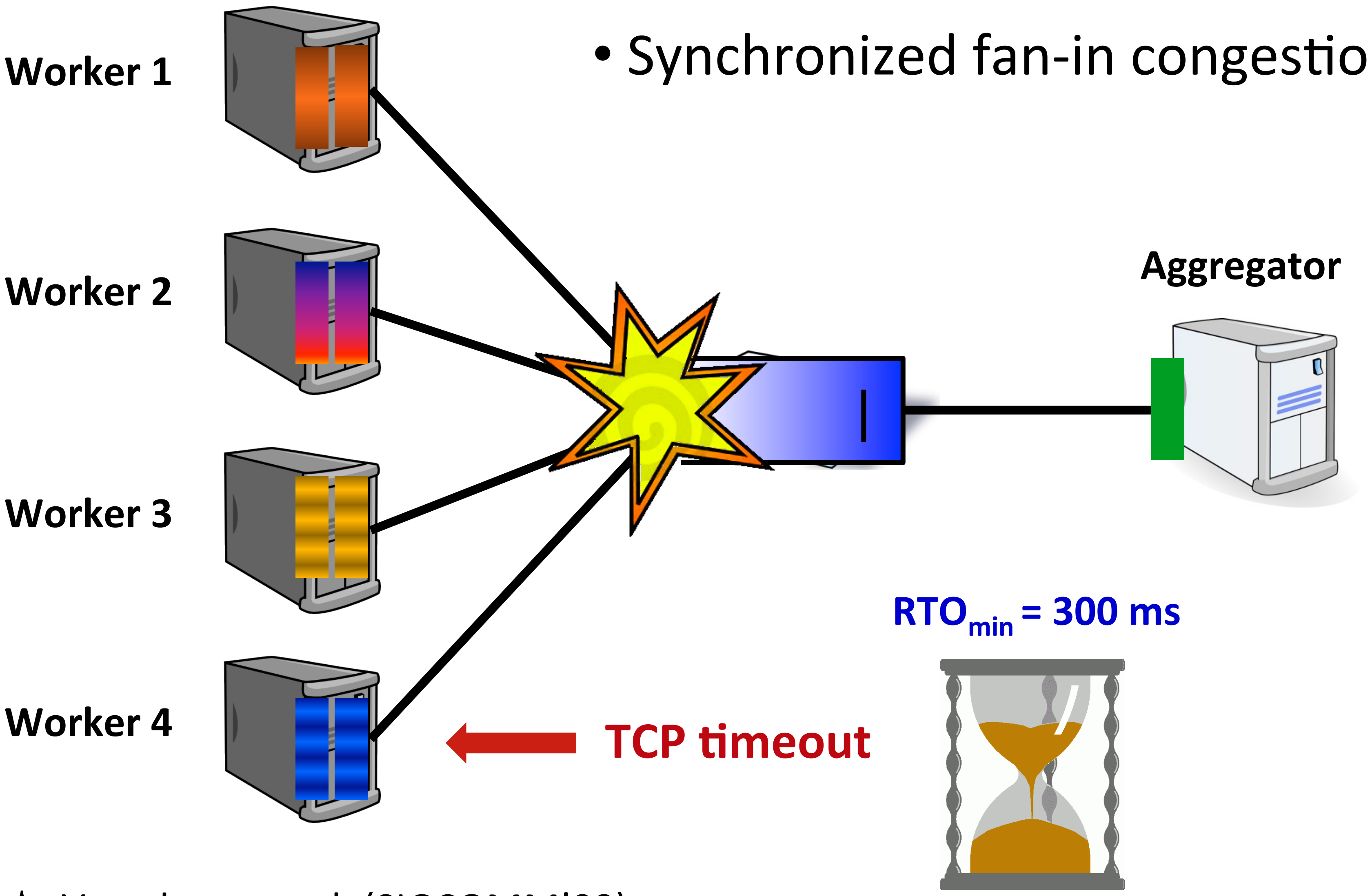
(e.g., query, coordination) → **Low Latency**

Large flows

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

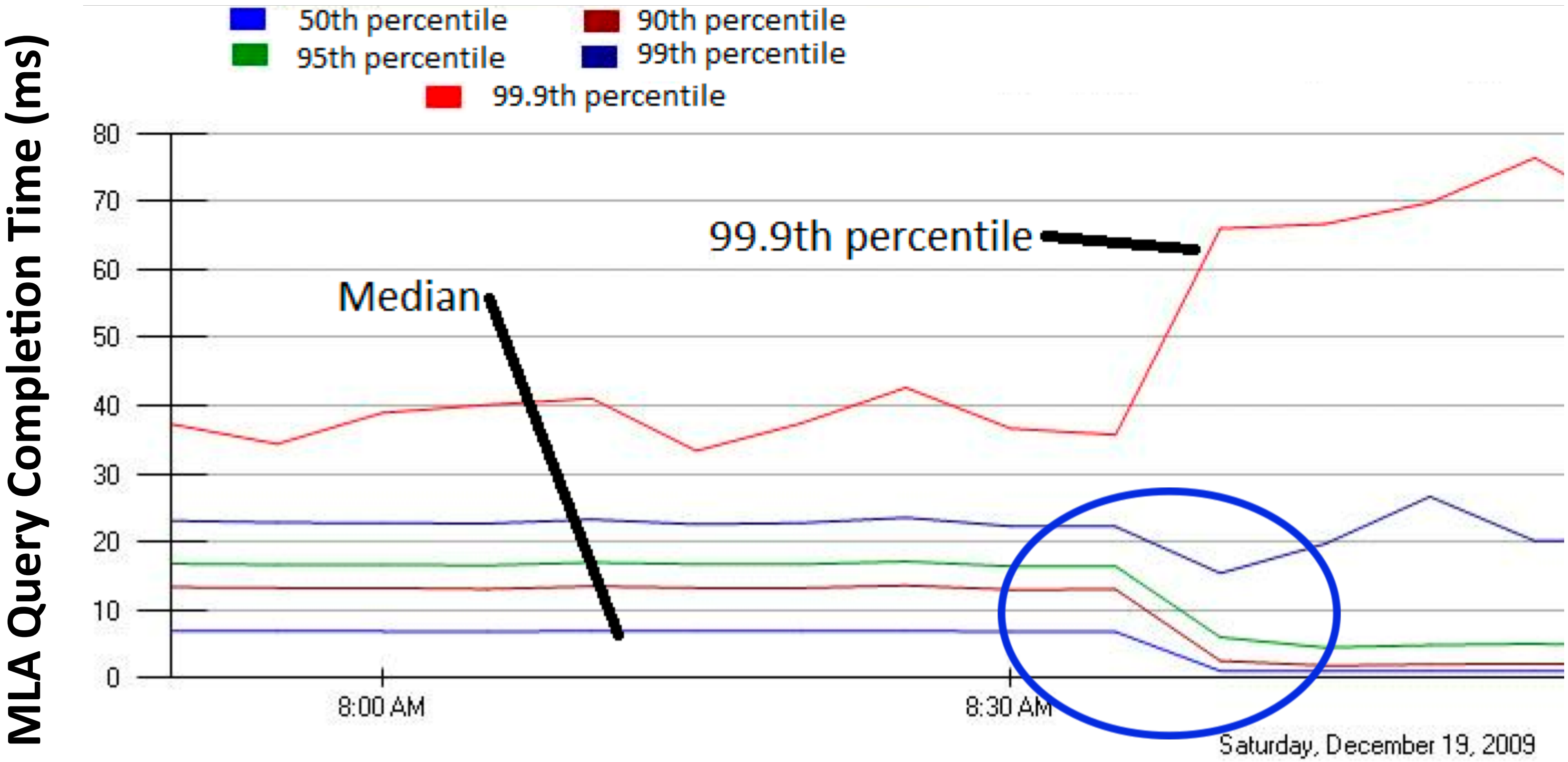
Incast

- Synchronized fan-in congestion



✧ 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

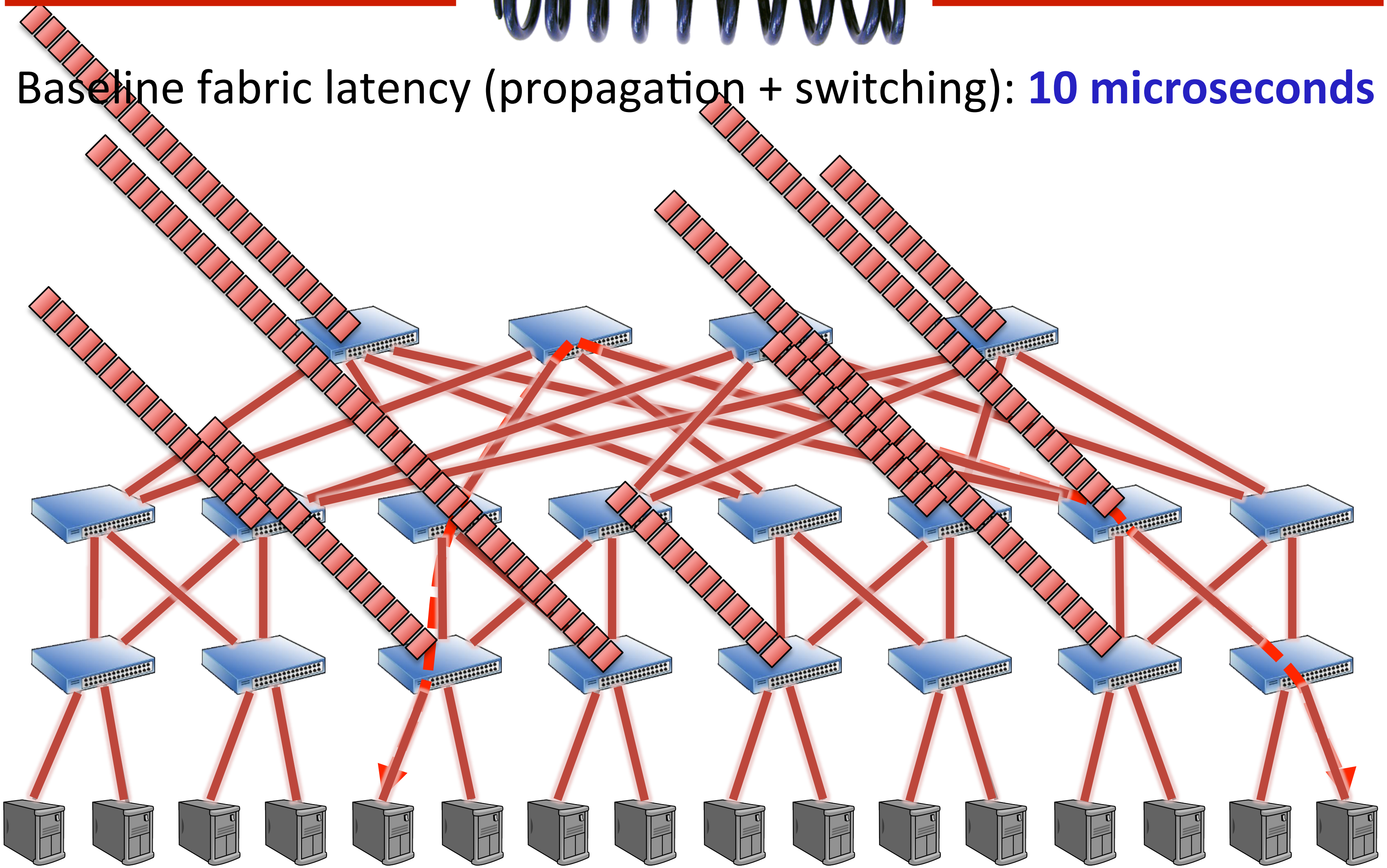
3. High Burst Tolerance

– Incast

The challenge is to achieve these *together*

High Throughput  Low Latency

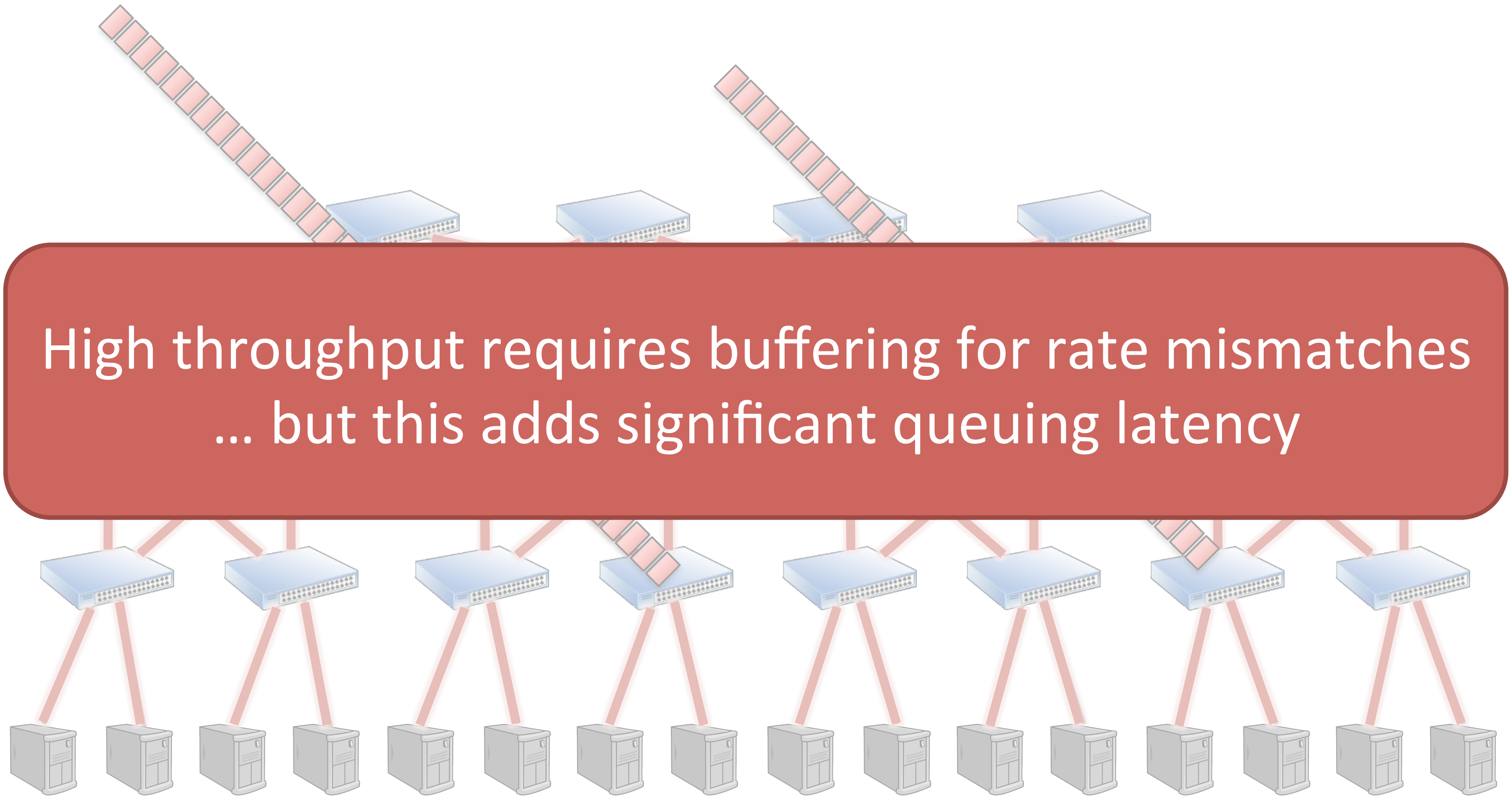
Baseline fabric latency (propagation + switching): **10 microseconds**



High Throughput

Low Latency

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



High throughput requires buffering for rate mismatches
… but this adds significant queuing latency

Data Center TCP

TCP in the Data Center

TCP [Jacobsen et al.'88] is widely used in the data center

- More than **99%** of the traffic

Operators work around TCP problems

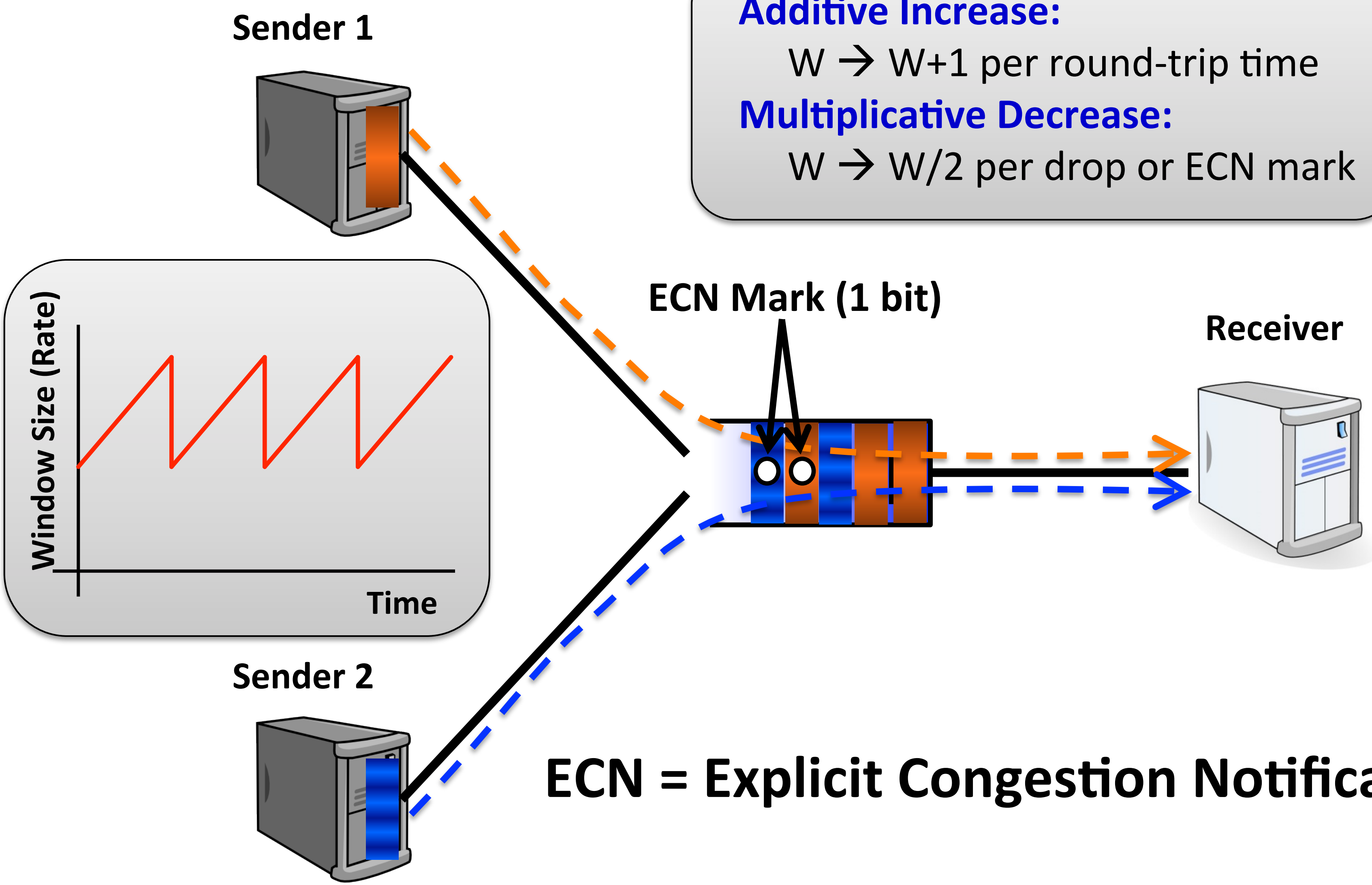
- Ad-hoc, inefficient, often expensive solutions
- TCP is deeply ingrained in applications

Practical deployment is hard
→ keep it simple!

Review: The TCP Algorithm

Additive Increase:
W → W+1 per round-trip time

Multiplicative Decrease:
W → W/2 per drop or ECN mark

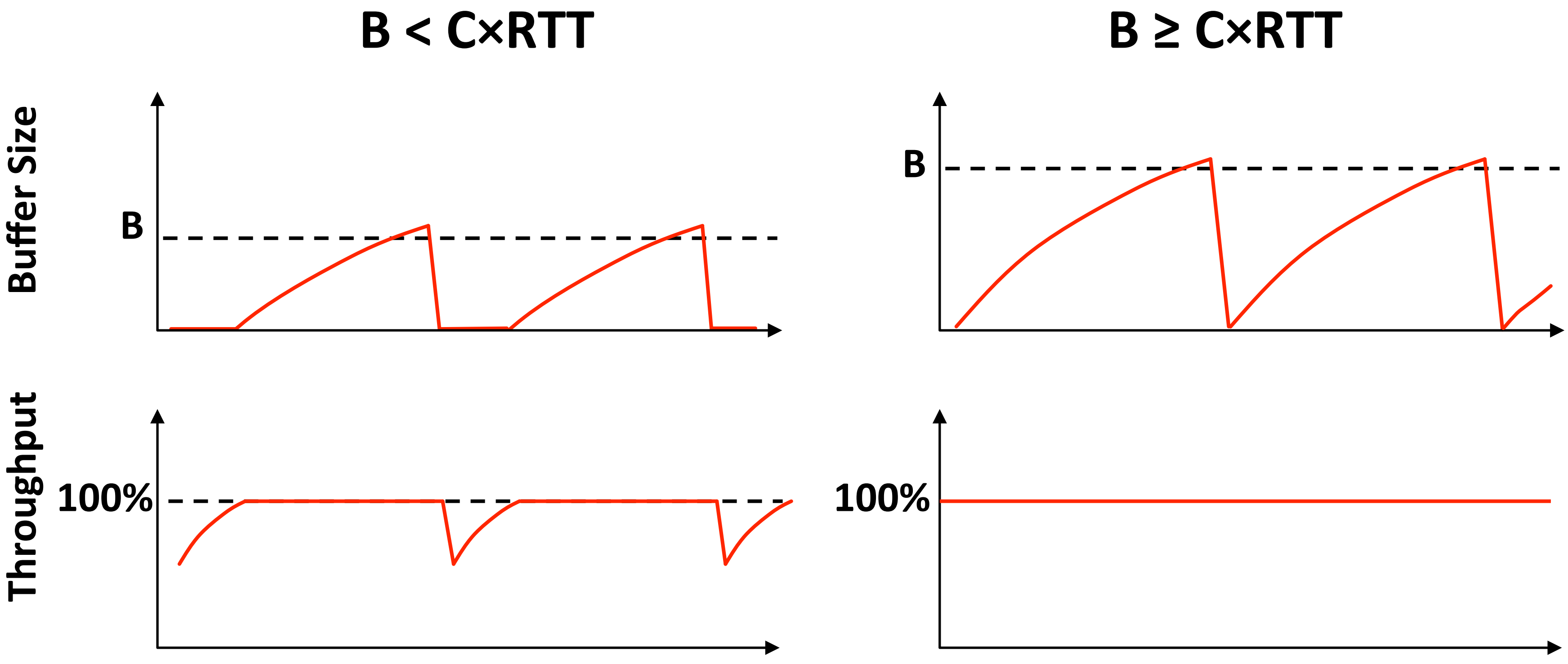


ECN = Explicit Congestion Notification

TCP Buffer Requirement

Bandwidth-delay product rule of thumb:

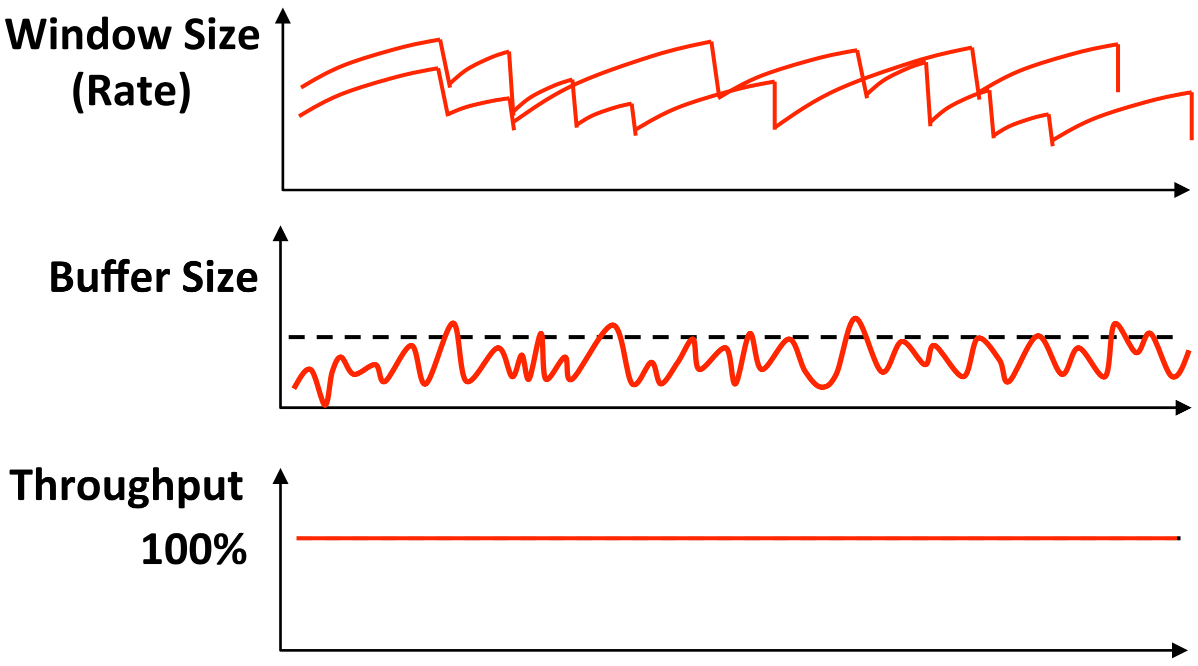
- A single flow needs **C×RTT** buffers for **100% Throughput.**



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.

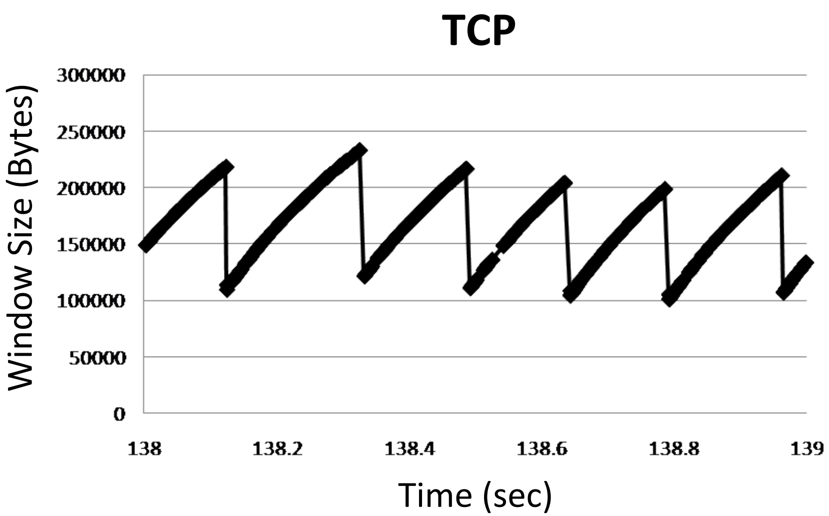
- Measurements show typically **only 1-2 large flows** at each server

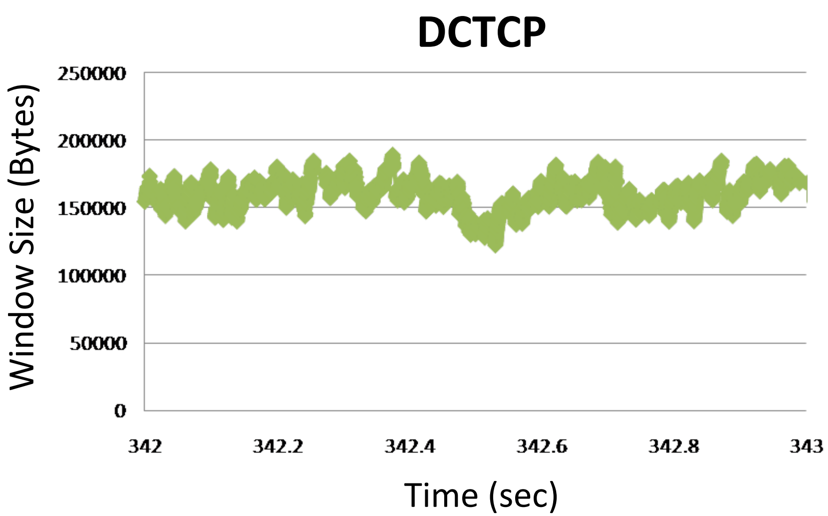
Key Observation:
Low variance in sending rate → Small buffers suffice

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%

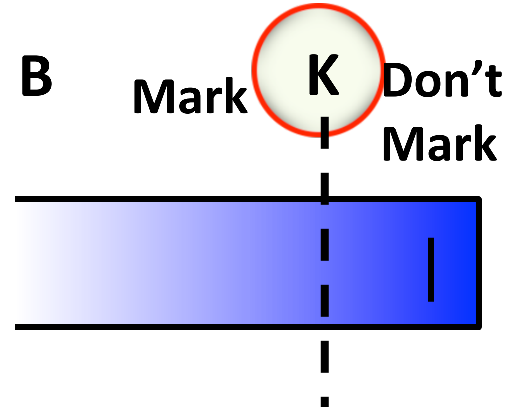




DCTCP: Algorithm

Switch side:

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



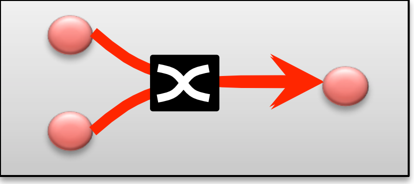
Sender side:

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

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

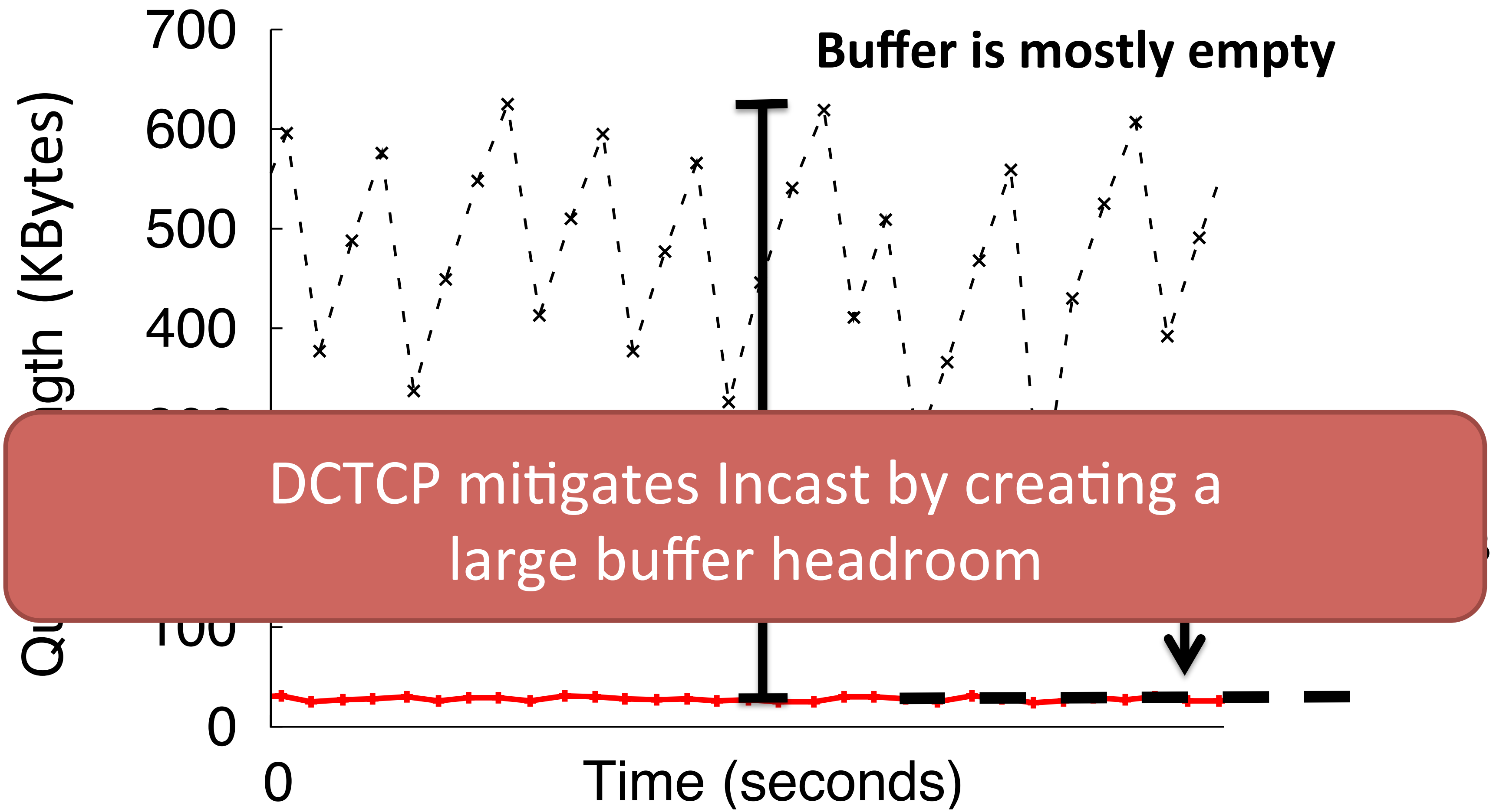
➢ **Adaptive window decreases:** $W \leftarrow (1 - \frac{\alpha}{2})W$

– Note: decrease factor between 1 and 2.



DCTCP vs TCP

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

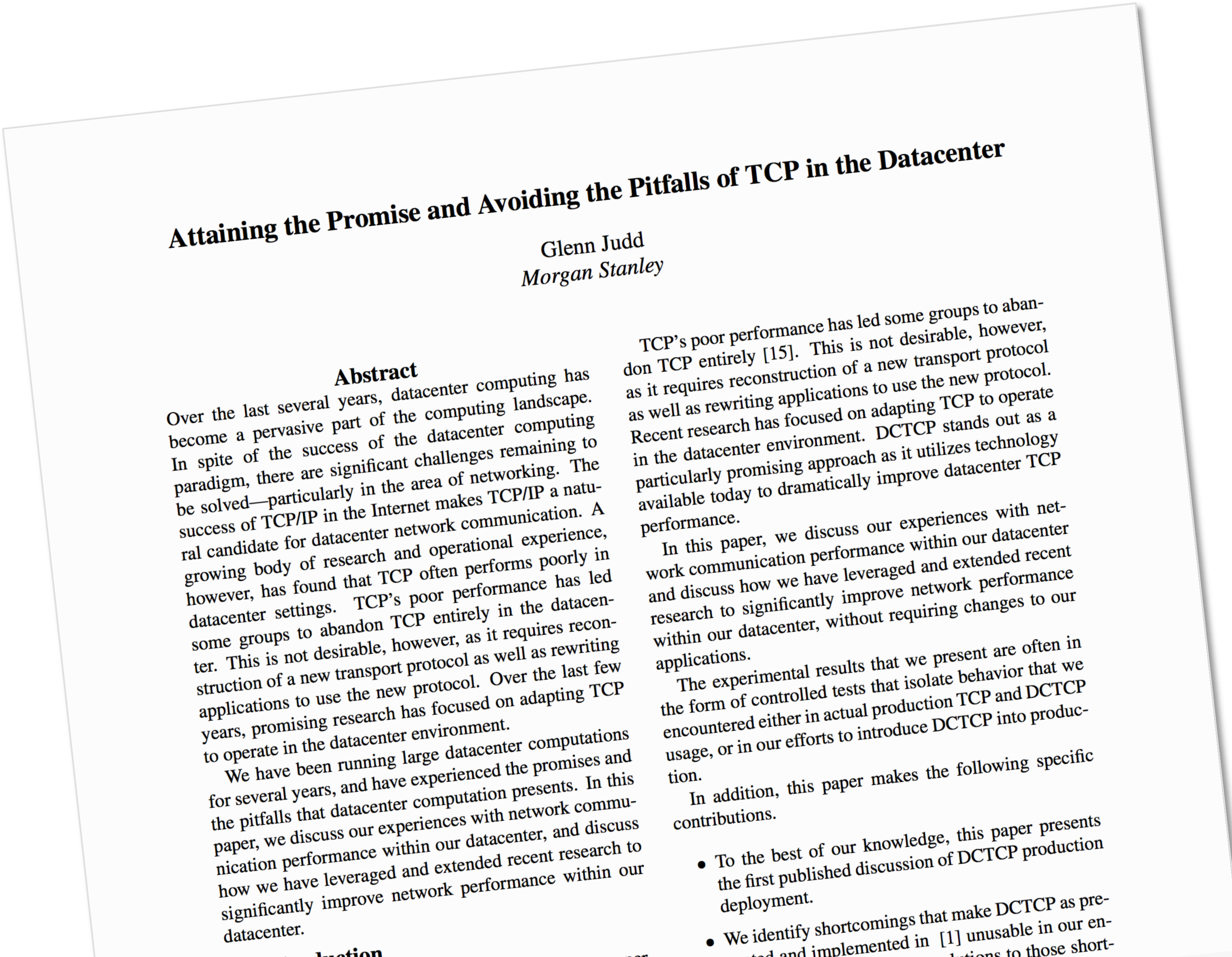


DCTCP mitigates Incast by creating a large buffer headroom

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

DCTCP Deployments



Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter

Glenn Judd
Morgan Stanley

Abstract

Over the last several years, datacenter computing has become a pervasive part of the computing landscape. In spite of the success of the datacenter computing paradigm, there are significant challenges remaining to be solved—particularly in the area of networking. The success of TCP/IP in the Internet makes TCP/IP a natural candidate for datacenter network communication. A growing body of research and operational experience, however, has found that TCP often performs poorly in datacenter settings. TCP's poor performance has led some groups to abandon TCP entirely in the datacenter. This is not desirable, however, as it requires reconstruction of a new transport protocol as well as rewriting applications to use the new protocol. Over the last few years, promising research has focused on adapting TCP to operate in the datacenter environment.

We have been running large datacenter computations for several years, and have experienced the promises and the pitfalls that datacenter computation presents. In this paper, we discuss our experiences with network communication performance within our datacenter, and discuss how we have leveraged and extended recent research to significantly improve network performance within our datacenter.

TCP's poor performance has led some groups to abandon TCP entirely [15]. This is not desirable, however, as it requires reconstruction of a new transport protocol as well as rewriting applications to use the new protocol. Recent research has focused on adapting TCP to operate in the datacenter environment. DCTCP stands out as a particularly promising approach as it utilizes technology available today to dramatically improve datacenter TCP performance.

In this paper, we discuss our experiences with network communication performance within our datacenter and discuss how we have leveraged and extended recent research to significantly improve network performance within our datacenter, without requiring changes to our applications.

The experimental results that we present are often in the form of controlled tests that isolate behavior that we encountered either in actual production TCP and DCTCP usage, or in our efforts to introduce DCTCP into production.

In addition, this paper makes the following specific contributions.

- To the best of our knowledge, this paper presents the first published discussion of DCTCP production deployment.
- We identify shortcomings that make DCTCP as pre-

Discussion

What You Said?

Austin: *“The paper's performance comparison to RED seems arbitrary, perhaps RED had traction at the time? Or just convenient as the switches were capable of implementing it?”*

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**

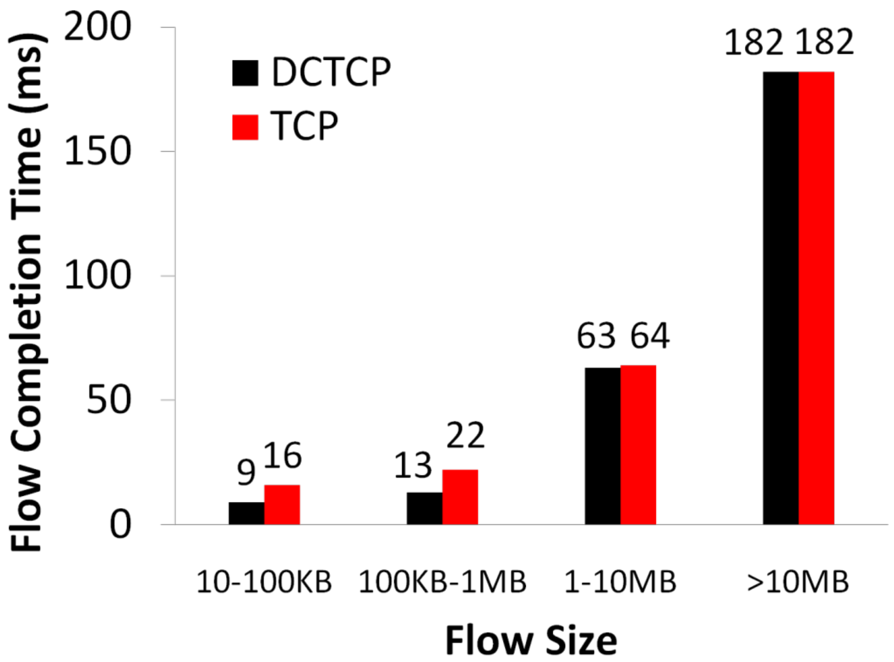
Numerous micro-benchmarks

- **Throughput and Queue Length**
- **Multi-hop**
- **Queue Buildup**
- **Buffer Pressure**
- **Fairness and Convergence**
- **Incast**
- **Static vs Dynamic Buffer Mgmt**

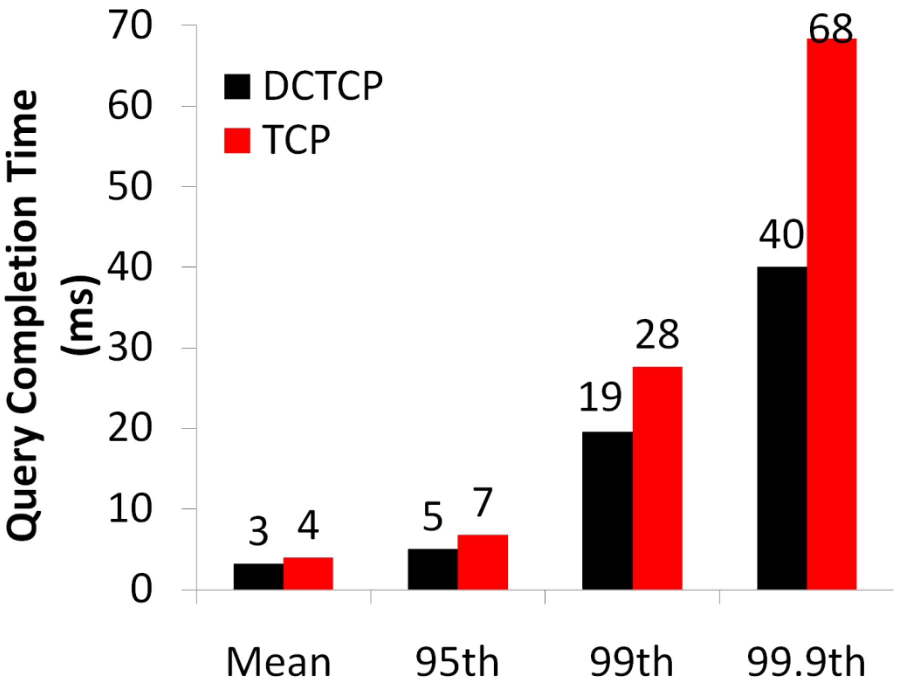
Bing cluster benchmark

Bing Benchmark (baseline)

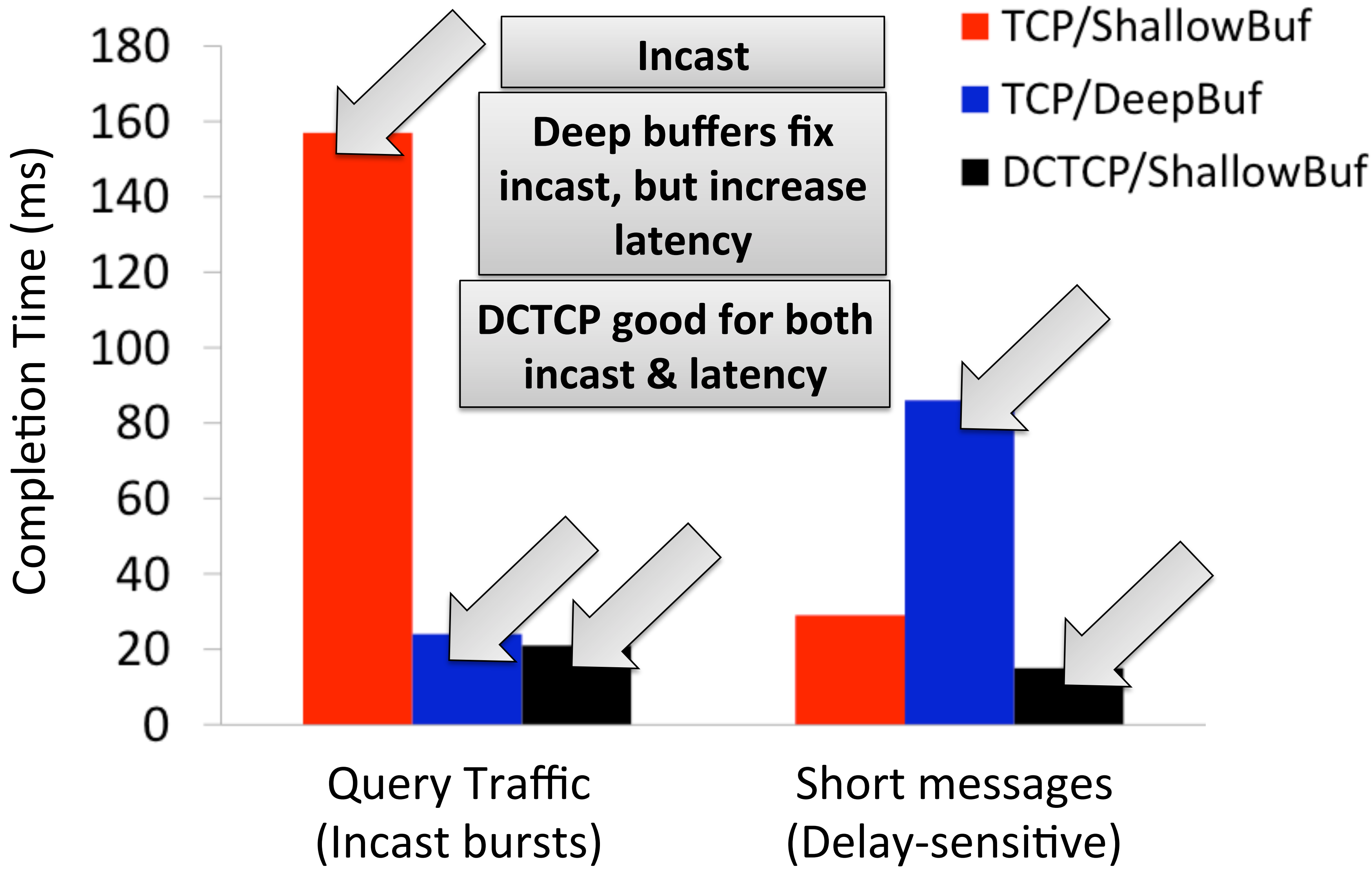
Background Flows



Query Flows



Bing Benchmark (scaled 10x)

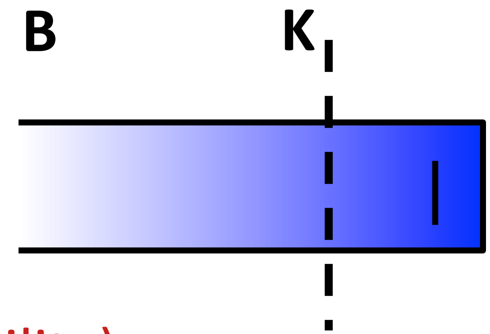


What You Said

Amy: *"I find it unsatisfying that the details of many congestion control protocols (such at these) are so complicated! ... can we create a parameter-less congestion control protocol that is similar in behavior to DCTCP or TIMELY?"*

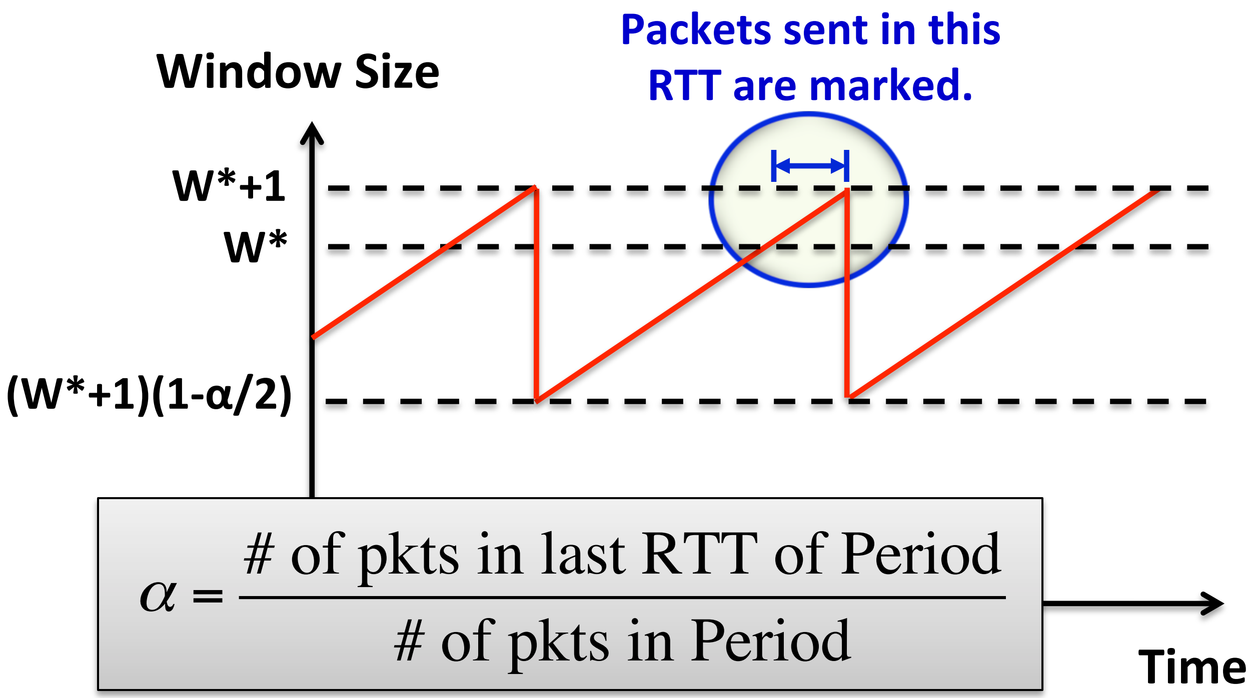
Hongzi: "Is there a general guideline to tune the parameters, like alpha, beta, delta, N, T_low, T_high, in the system?"

A bit of Analysis



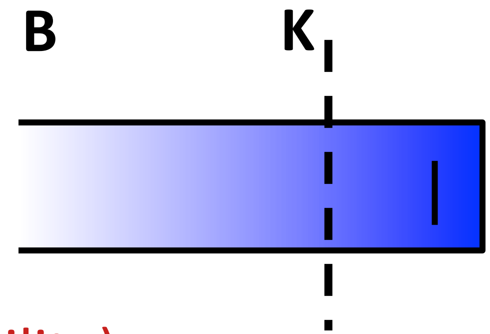
How much buffering does DCTCP need for 100% throughput?

- Need to quantify queue size oscillations (Stability).



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

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**

What assumptions does the model make?

What You Said

Anurag: *"In both the papers, one of the difference I saw from TCP was that these protocols don't have the "slow start" phase, where the rate grows exponentially starting from 1 packet/RTT."*

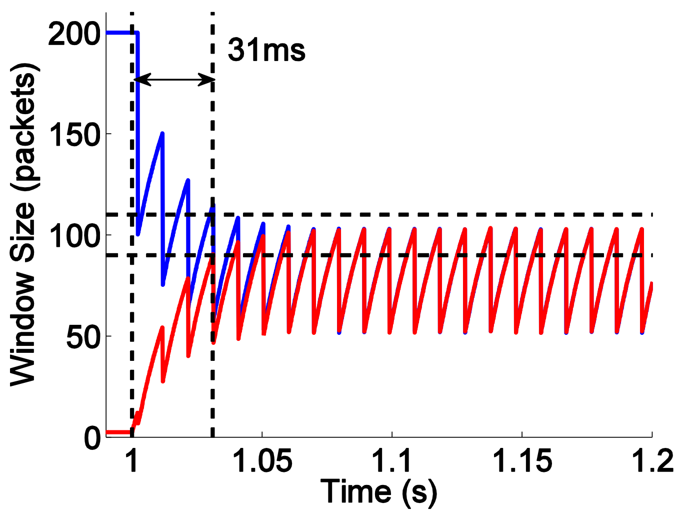
Convergence Time

DCTCP takes at most ~40% more **RTTs** than TCP

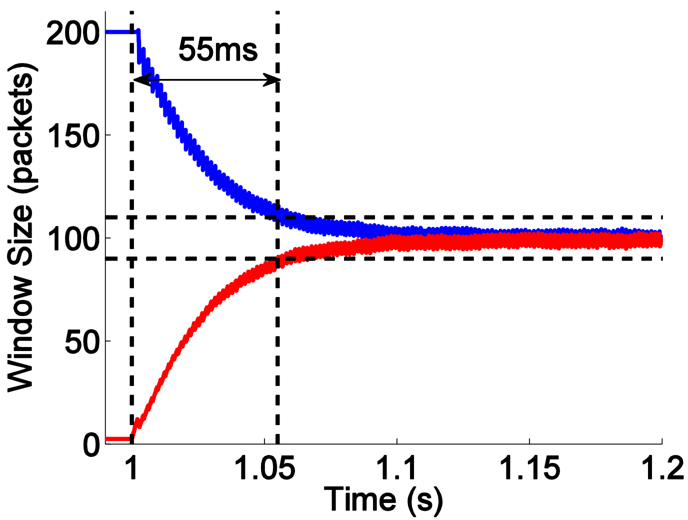
– "Analysis of DCTCP: Stability, Convergence, and Fairness," SIGMETRICS 2011

Intuition: DCTCP makes smaller adjustments than TCP, but makes them much more frequently

TCP



DCTCP



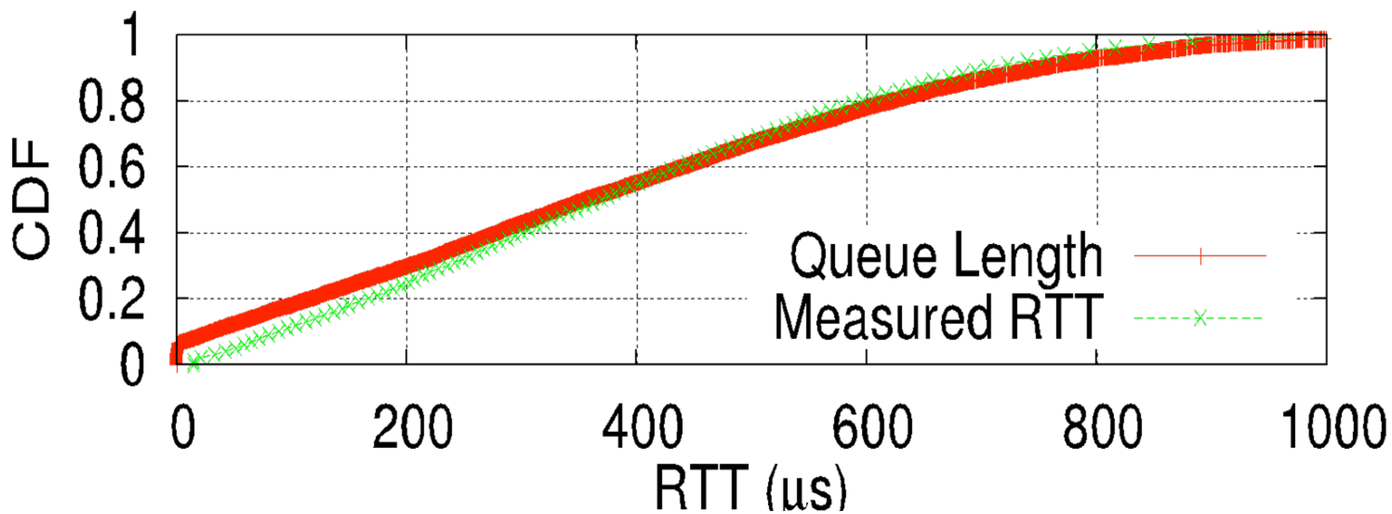
TIMELY

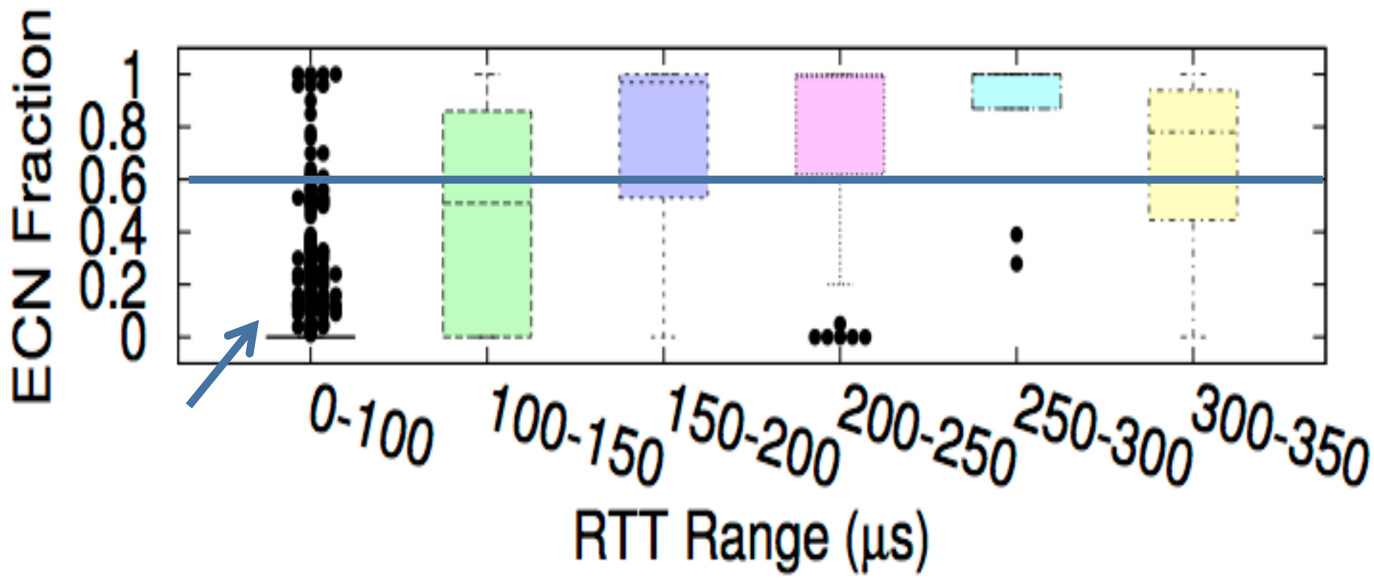
✧ Slides by Radhika Mittal (Berkeley)

Qualities of RTT

- Fine-grained and informative
- Quick response time
- No switch support needed
- End-to-end metric
- Works seamlessly with QoS

RTT correlates with queuing delay





What You Said

Ravi: *"The first thing that struck me while reading these papers was how different their approaches were. DCTCP even states that delay-based protocols are "susceptible to noise in the very low latency environment of data centers" and that "the accurate measurement of such small increases in queuing delay is a daunting task". Then, I noticed that there is a 5 year gap between these two papers… "*

Arman: "They had to resort to extraordinary measures to ensure that the timestamps accurately reflect the time at which a packet was put on wire…"

Accurate RTT Measurement

Hardware Assisted RTT Measurement

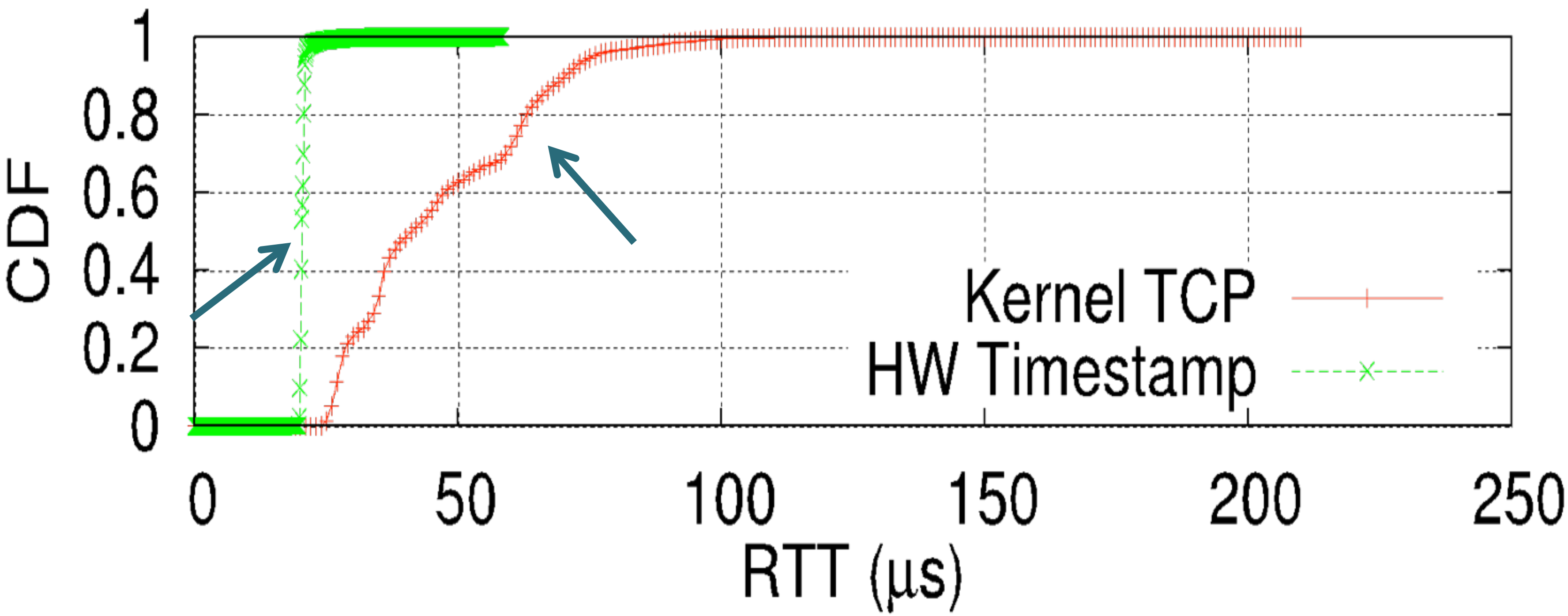
Hardware Timestamps

- mitigate noise in measurements

Hardware Acknowledgements

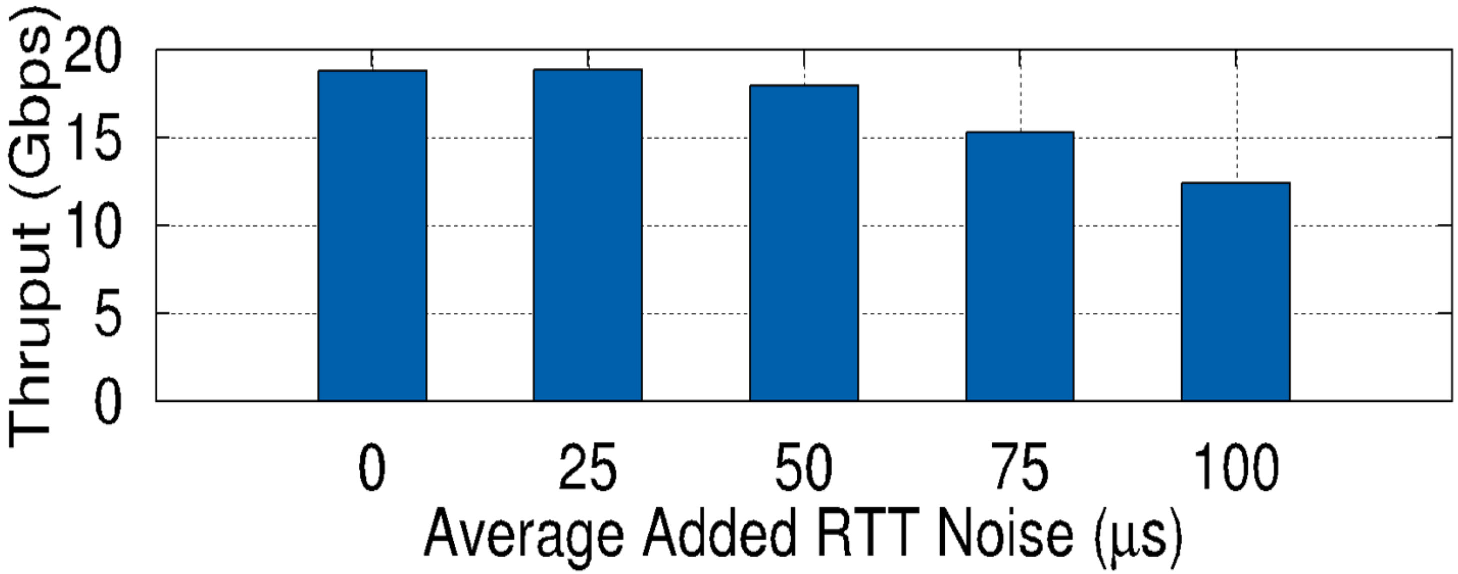
- avoid processing overhead

Hardware vs Software Timestamps



Kernel Timestamps introduce significant noise in RTT measurements compared to HW Timestamps.

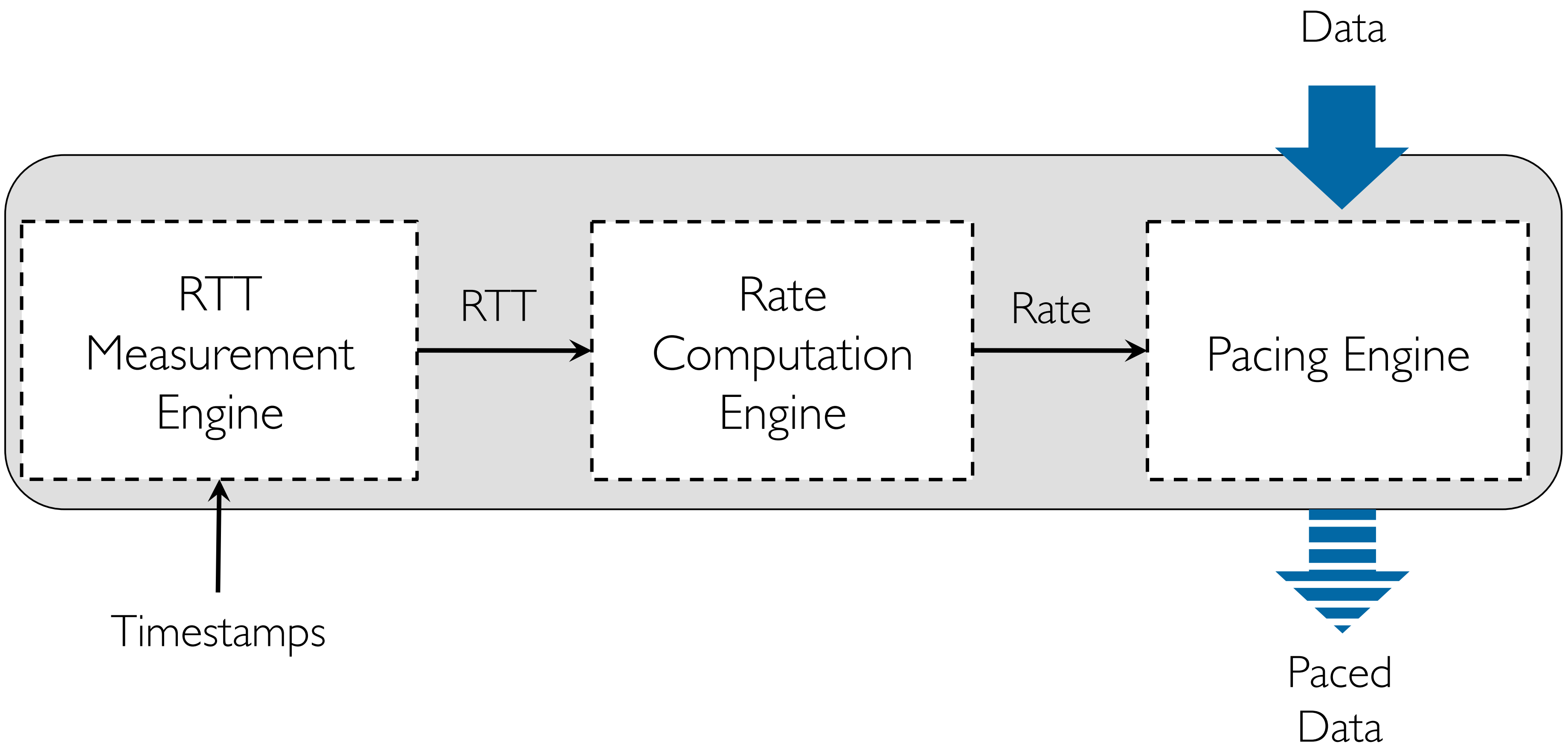
Impact of RTT Noise



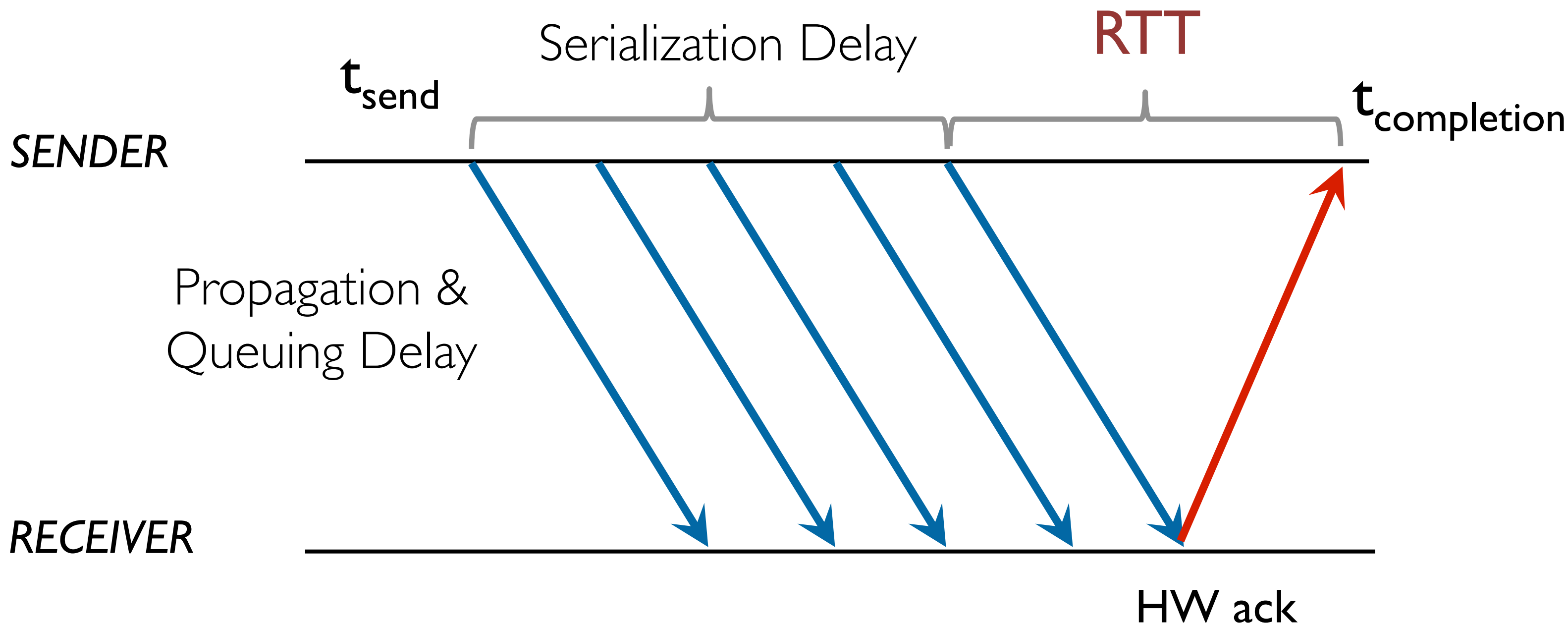
Throughput degrades with increasing noise in RTT.
Precise RTT measurement is crucial.

TIMELY Framework

Overview


Data
RTT
Measurement
Engine
RTT
Rate
Computation
Engine
Rate
Pacing Engine
Timestamps
Paced
Data

RTT Measurement Engine



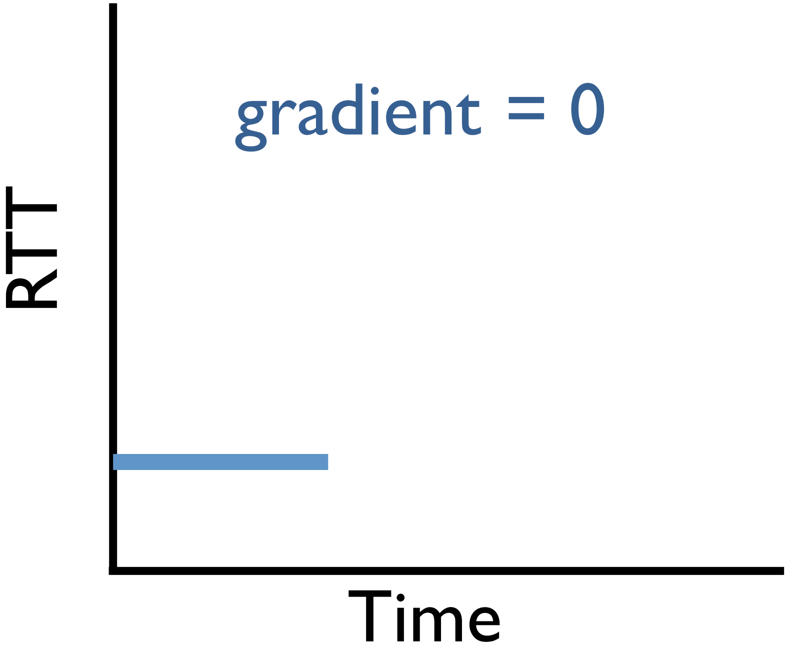
$$RTT = t_{completion} - t_{send} - \text{Serialization Delay}$$

Algorithm Overview

Gradient-based Increase / Decrease

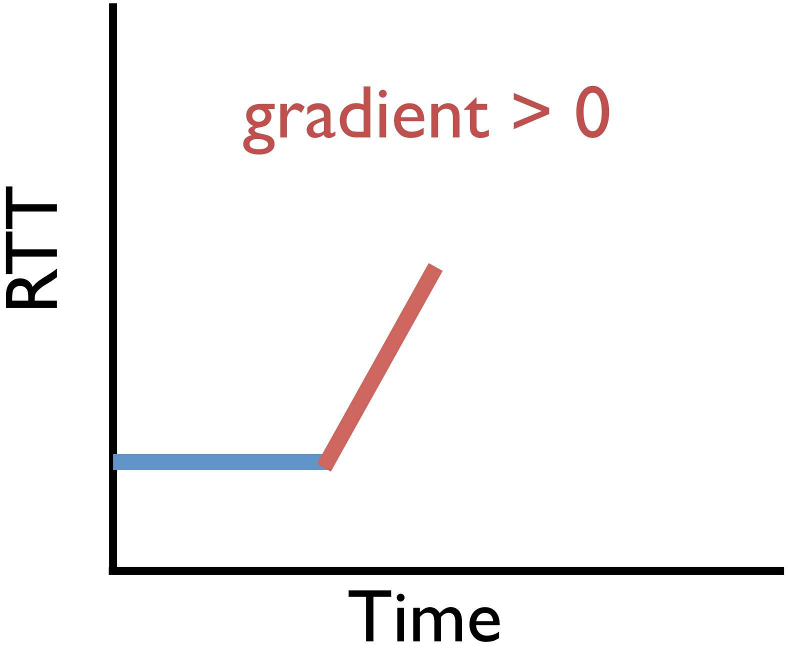
Algorithm Overview

Gradient-based Increase / Decrease



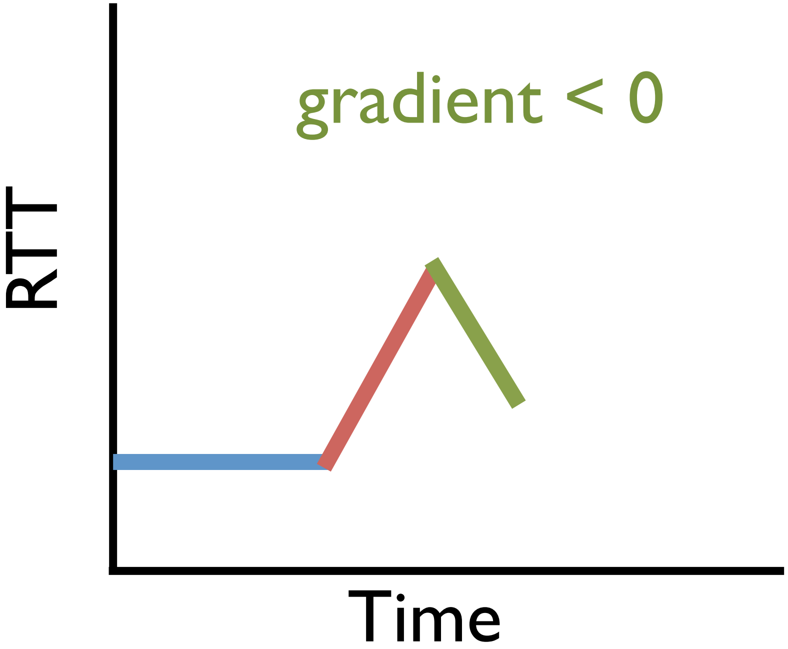
Algorithm Overview

Gradient-based
Increase / Decrease



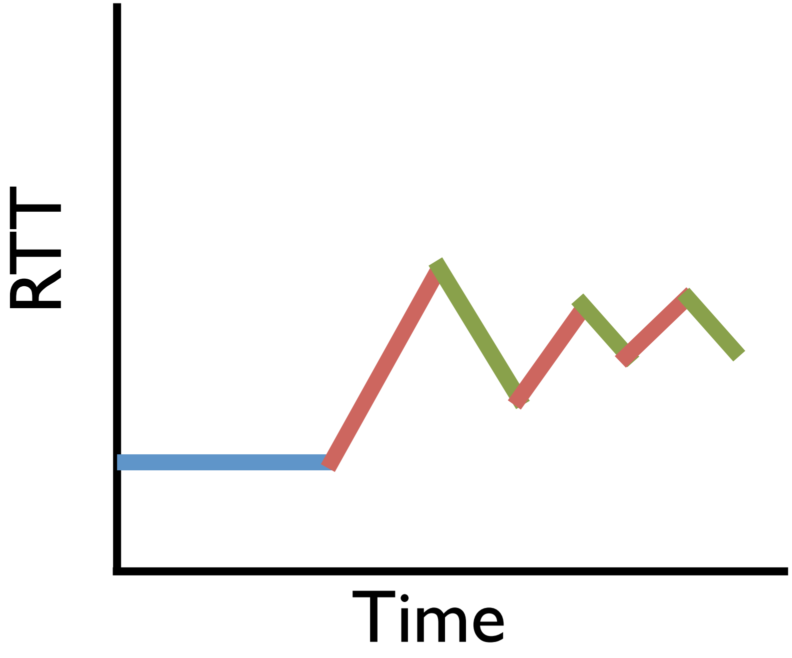
Algorithm Overview

Gradient-based Increase / Decrease



Algorithm Overview

Gradient-based Increase / Decrease

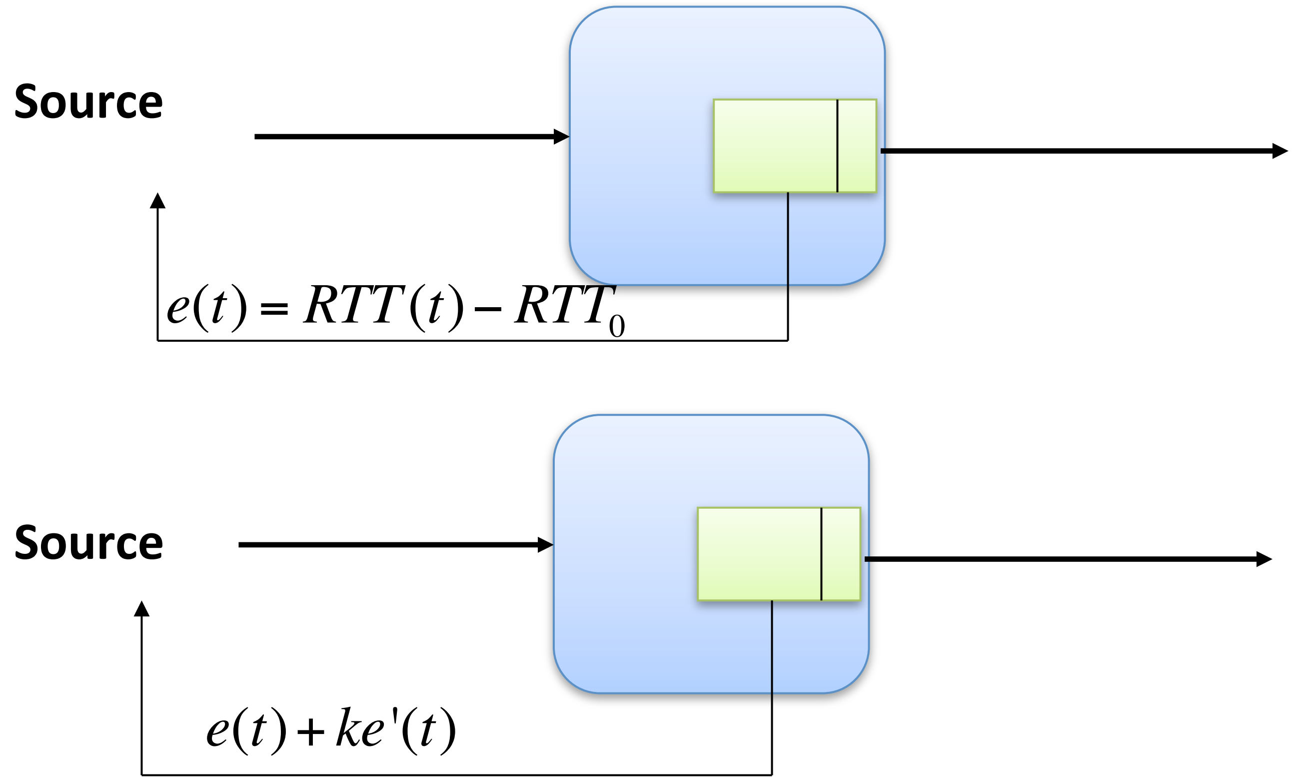


Algorithm Overview

Gradient-based Increase / Decrease

To navigate the throughput-latency tradeoff and ensure stability.

Why Does Gradient Help Stability?



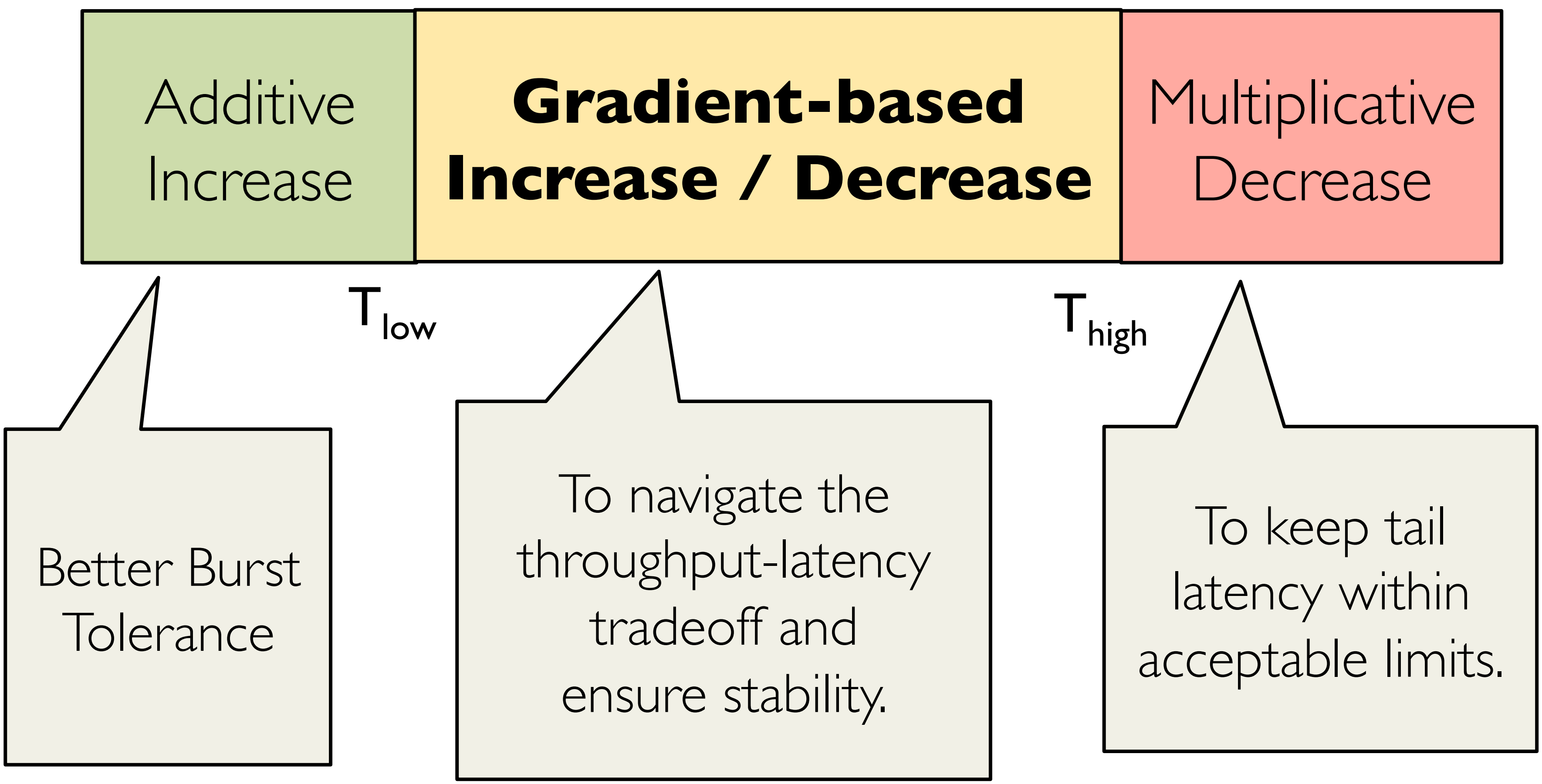
Feedback higher order derivatives

Observe not only error, but *change* in error – “anticipate” future state

What You Said

Arman: *"I also think that deducing the queue length from the gradient model could lead to miscalculations. For example, consider an Incast scenario, where many senders transmit simultaneously through the same path. Noting that every packet will see a long, yet steady, RTT, they will compute a near-zero gradient and hence the congestion will continue."*

Algorithm Overview



Discussion

Implementation Set-up

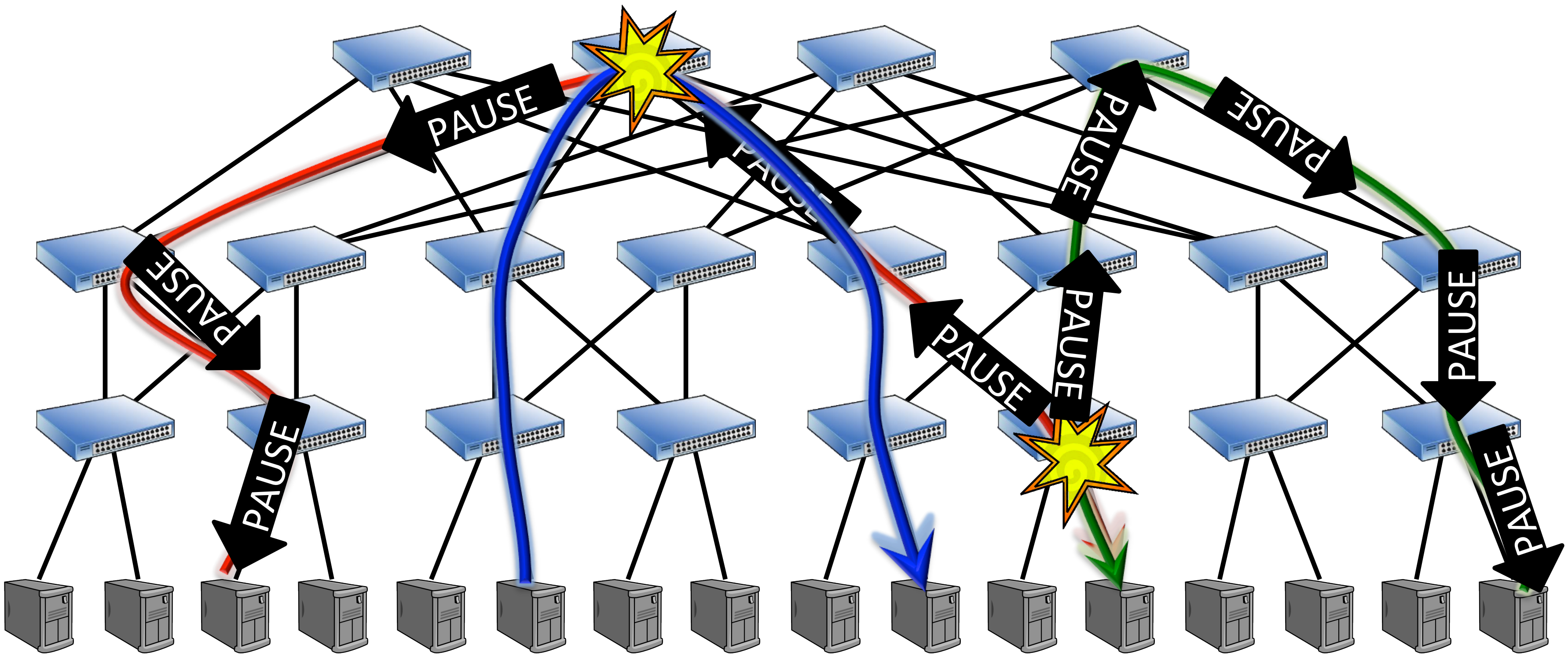
TIMELY is implemented in the context of RDMA.

- RDMA write and read primitives used to invoke NIC services.

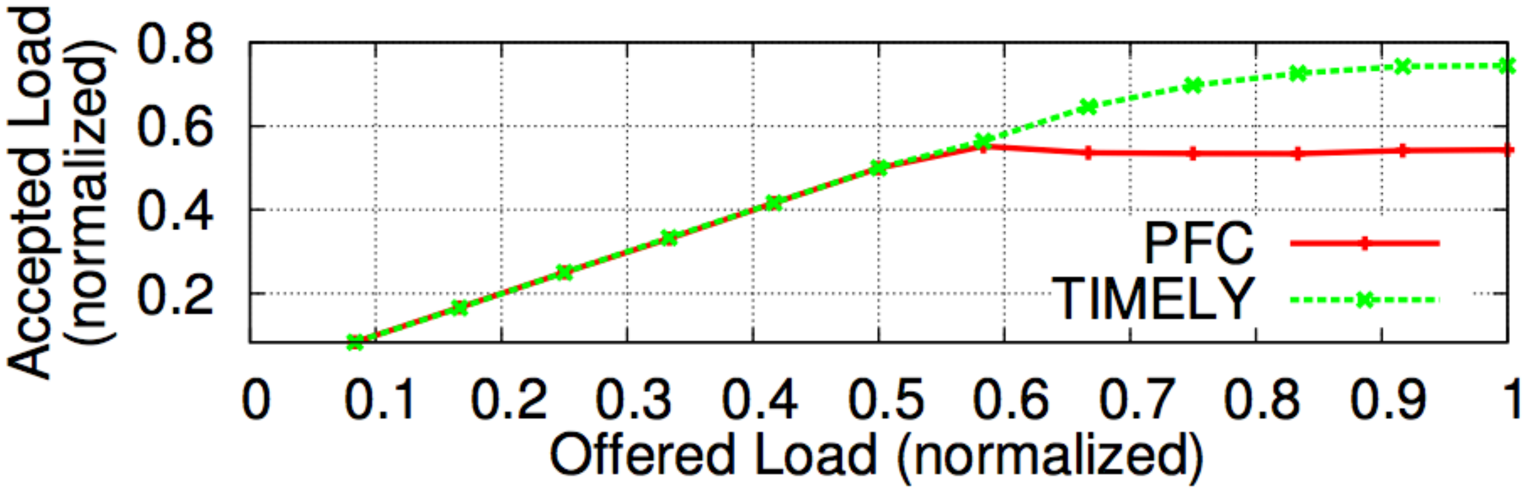
Priority Flow Control is enabled in the network fabric.

- RDMA transport in the NIC is sensitive to packet drops.
- PFC sends out pause frames to ensure lossless network.

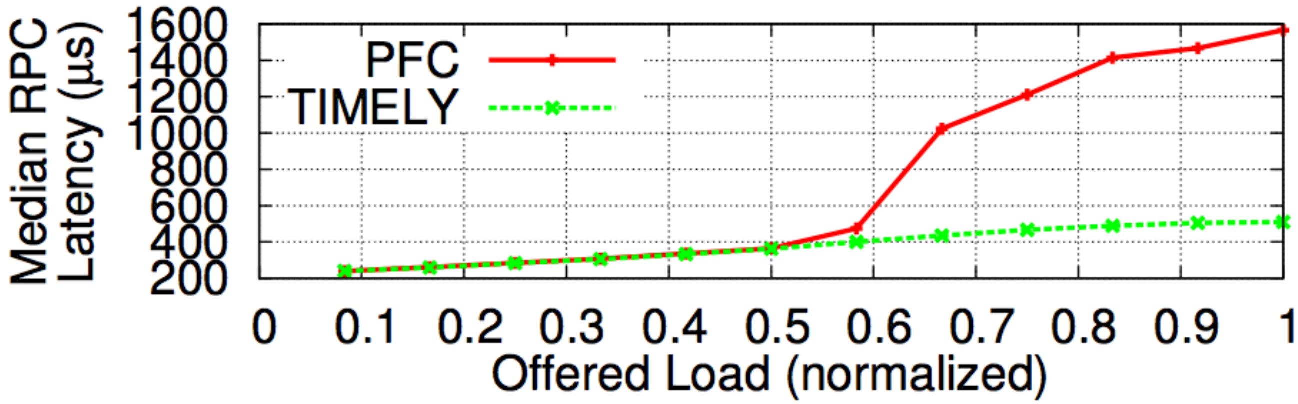
"Congestion Spreading" in Lossless Networks

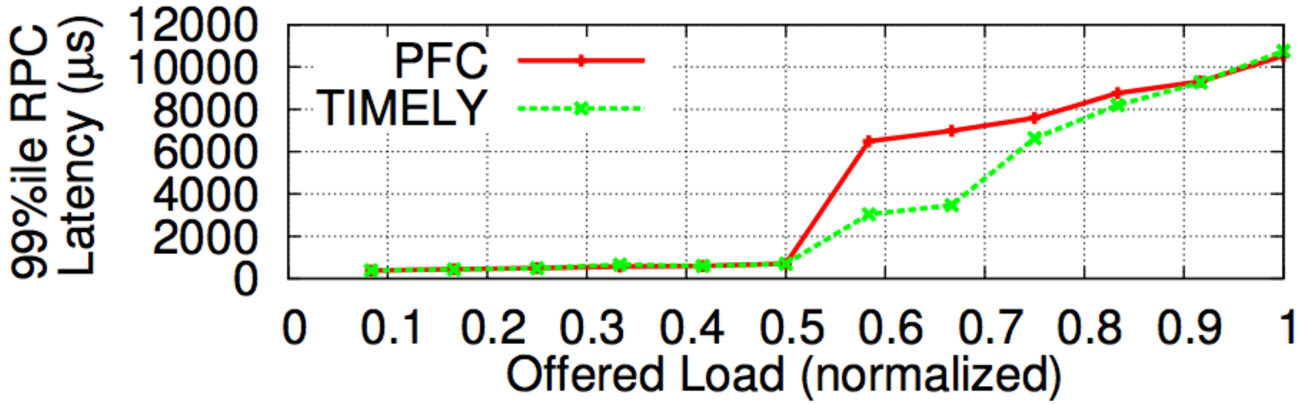


TIMELY vs PFC



TIMELY vs PFC





What You Said

Amy: *"I was surprised to see that TIMELY performed so much better than DCTCP. Did the lack of an OS-bypass for DCTCP impact performance? I wish that the authors had offered an explanation for this result."*

Next time: Load Balancing

Improving Datacenter Performance and Robustness w[ith] Multipath TCP

Costin Raiciu†*, Sebastien Barre‡, Christopher Pluntke†,
Adam Greenhalgh†, Damon Wischik†, Mark Handley†
†University College London, ‡Universite Catholique de Louvain, *Universitatea Politehnica Bucuresti
{c.raiciu, c.pluntke, a.greenhalgh, d.wischik, mjh}@cs.ucl.ac.uk, sebastien.barre@uclouvain

ABSTRACT

The latest large-scale data centers offer higher aggregate bandwidth and robustness by creating multiple paths in the core of the network. To utilize this bandwidth requires different flows take different paths, which poses a challenge. In short, a single-path transport seems ill-suited to such networks.

We propose using Multipath TCP as a replacement for TCP in such data centers, as it can effectively and seamlessly use available bandwidth, giving improved throughput and better fairness on many topologies. We investigate what causes these benefits, teasing apart the contribution of each of the mechanisms used by MPTCP.

Using MPTCP lets us rethink data center networks, with a different mindset as to the relationship between transport protocols, routing and topology. MPTCP enables topologies that single path TCP cannot utilize. As a proof-of-concept, we present a dual-homed variant of the FatTree topology. With MPTCP, this outperforms FatTree for a wide range of workloads, but costs the same.

In existing data centers, MPTCP is readily deployable leveraging widely deployed technologies such as ECMP. We have run MPTCP on Amazon EC2 and found that it outperforms TCP by a factor of three when there is path diversity. But the biggest benefits will come when data centers are designed for multipath transports.

Categories and Subject Descriptors

C.2.2[Computer-Comms Nets]: Network Protocols

General Terms: Algorithms, Design, Performance

Topologies like these have started to be deployed; Amazon's la[test] EC2 data center has such a redundant structure - between cert[ain] pairs of hosts there are many alternative paths. Typically switch[es] run a variant of ECMP routing, randomly hashing flows to equa[l] cost paths to balance load across the topology. However, with mos[t] such topologies it takes many simultaneous TCP connections per host to generate sufficient flows to come close to balancing traffic. With more typical load levels, using ECMP on these multi-stage topologies causes flows to collide on at least one link with high probability. In traffic patterns that should be able to fill the network, we have observed flows that only manage 10% of the throughput they might expect and total network utilization below 50%.

In this paper we examine the use of Multipath TCP [4] within large data centers. Our intuition is that by exploring multiple paths simultaneously and by linking the congestion response of subflows on different paths to move traffic away from congestion, MPTCP will lead to both higher network utilization and fairer allocation of capacity to flows.

From a high-level perspective, there are four main components to a data center networking architecture:

- Physical topology
- Routing over the topology
- Selection between multiple paths supplied by routing
- Congestion control of traffic on the selected paths

These are not independent; the performance of one will depend on the choices made by those preceding it in the list, and in some cases by those after it in the list. The insight that we evaluate in

Presto: Edge-based Load Balancing for Fast Datacenter Networks

Keqiang He† Eric Rozner* Kanak Agarwal*
Wes Felter* John Carter* Aditya Akella†
†University of Wisconsin–Madison *IBM Research

ABSTRACT

Datacenter networks deal with a variety of workloads, ranging from latency-sensitive small flows to bandwidth-hungry large flows. Load balancing schemes based on flow hashing, *e.g.*, ECMP, cause congestion when hash collisions occur and can perform poorly in asymmetric topologies. Recent proposals to load balance the network require centralized traffic engineering, multipath-aware transport, or expensive specialized hardware. We propose a mechanism that avoids these limitations by (i) pushing load-balancing functionality into the soft network edge (*e.g.*, virtual switches) such that no changes are required in the transport layer, customer VMs, or networking hardware, and (ii) load balancing on fine-grained, near-uniform units of data (flowcells) that fit within end-host segment offload optimizations used to support fast networking speeds. We design and implement such a soft-edge load balancing scheme, called Presto, and evaluate it on a 10 Gbps physical testbed. We demonstrate the computational impact of packet reordering on receivers and propose a mechanism to handle reordering in the TCP receive offload functionality. Presto's performance closely tracks that of a single, non-blocking switch over many workloads and is adaptive to failures and topology asymmetry.

CCS Concepts

•**Networks** → **Network architectures; End nodes; Programmable networks; Data center networks;** *Data path algorithms; Network experimentation;*

Keywords

Load Balancing; Software-Defined Networking;

1. INTRODUCTION

Datacenter networks must support an increasingly div[erse] set of workloads. Small latency-sensitive flows to sup[port] real-time applications such as search, RPCs, or gaming s[hare] the network with large throughput-sensitive flows for v[ideo], big data analytics, or VM migration. Load balancing th[e] network is crucial to ensure operational efficiency and su[itable] application performance. Unfortunately, popular loa[d bal]ancing schemes based on flow hashing, *e.g.*, ECMP, [cause] congestion when hash collisions occur [3, 17, 19, 22, ... 63] and perform poorly in asymmetric topologies [4, ...]

A variety of load balancing schemes aim to add[ress the] problems of ECMP. Centralized schemes, such as He[dera] and Planck [54], collect network state and reroute [elephant] flows when collisions occur. These approaches ar[e funda]mentally reactive to congestion and are very coars[e-grained] due to the large time constraints of their control [loops]. [Trans]port-level solutions such as MPTCP [61] can react faster b[ut] require extra network infrastructure [54]... widespread adoption and are difficult to enforce in multi-tenant datacenters where customers often deploy c[ustom] VMs. In-network reactive distributed load balanc[ing ...] *e.g.*, CONGA [4] and Juniper VCF [28], can be e[ffective but] require specialized networking hardware.

The shortcomings of the above approaches ca[use us to re]examine the design space for load balancing [in datacenter] networks. ECMP, despite its limitations, is a [highly practi]cal solution due to its proactive nature and state[less behavior]. Conceptually, ECMP's flaws are not internal to [it] but are caused by asymmetry in network topol[ogy (or capac]ities) and variation in flow sizes. *In a symm[etric network] topology where all flows are "mice", ECM[P should pro]vide near optimal load balancing*; indeed, pr[ior work ...]