

Advanced Computer Networks

Transport in Data Center Networks (I)

<https://pages.cs.wisc.edu/~mgliu/CS740/F25/index.html>

Ming Liu
mgliu@cs.wisc.edu

Outline

- Last lecture
 - SmartNICs
- Today
 - Transport in Data Center Networks (I)
- Announcements
 - Lab2 due 11/05/2025 11:59 PM
 - Midterm report due 11/04/2025 11:59 PM

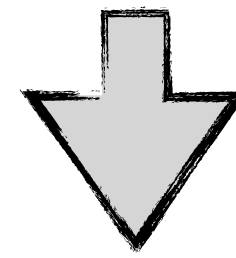
A Few Words on Project

A Few Words on Project

You'll have ~5 weeks to complete the course project.

A Few Words on Project

You'll have ~5 weeks to complete the course project.

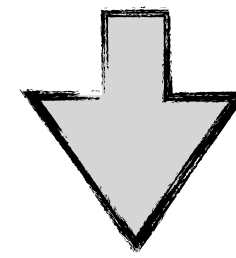


Please reshape it into a small-sized problem:

- **It's okay to answer just one question**

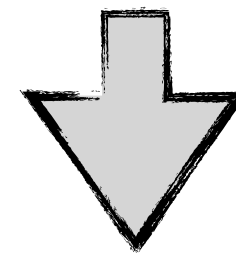
A Few Words on Project

You'll have ~5 weeks to complete the course project.



Please reshape it into a small-sized problem:

- It's okay to answer just one question



Course project = Testbed + Implementation + Evaluation

- Testbed: what are the hardware platforms, software frameworks, and workloads?
- Implementation: where are your codes running?
- Evaluation: what questions do you plan to answer and how?

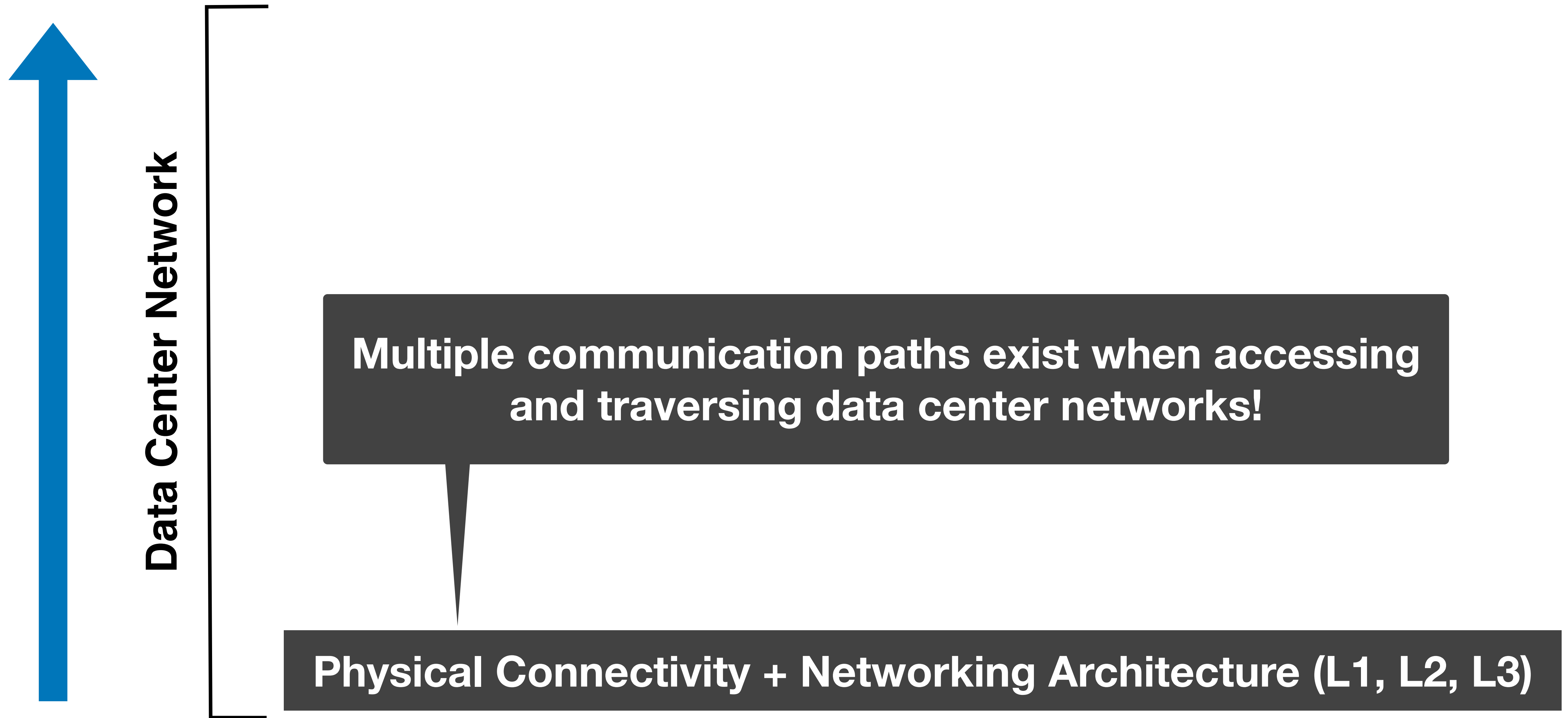
A Few Words on Project

You'll have ~5 weeks to complete the course project.

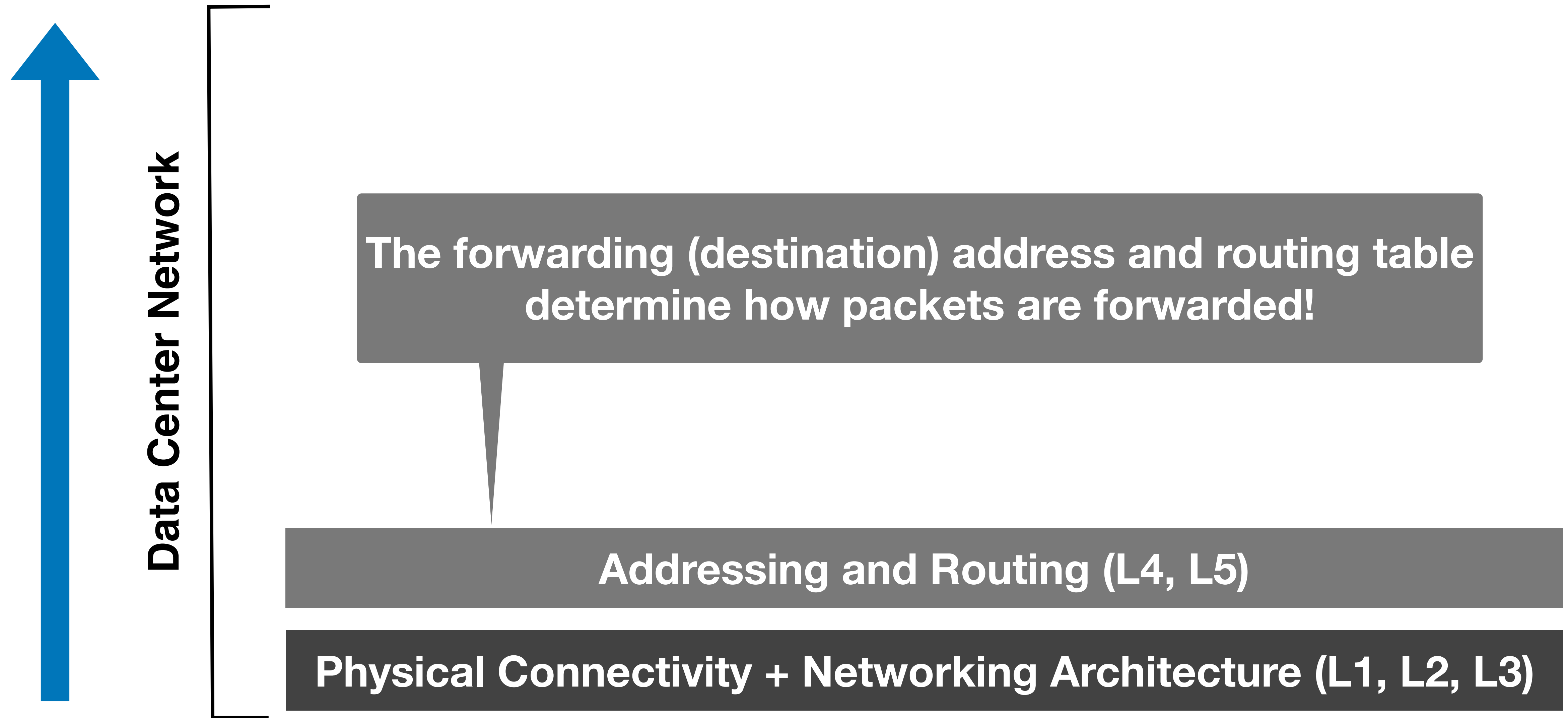
Midterm report:

- Refine your problem statement concretely;
 - Finalize your testbed;
 - Specify what you will implement;
 - Specify your outcome;
-
- Implementation: where are your codes running?
 - Evaluation: what questions do you plan to answer and how?

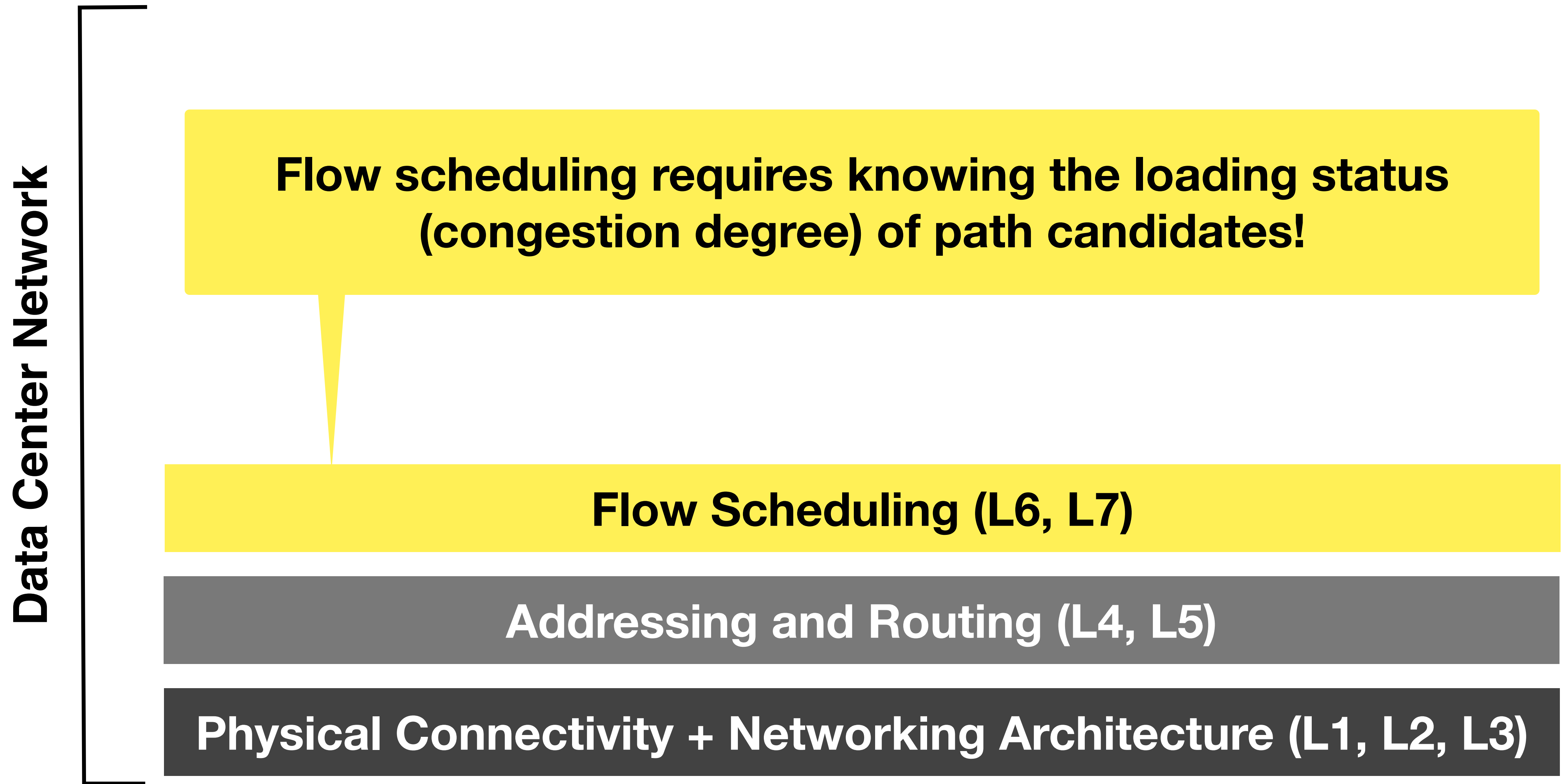
Where we are?



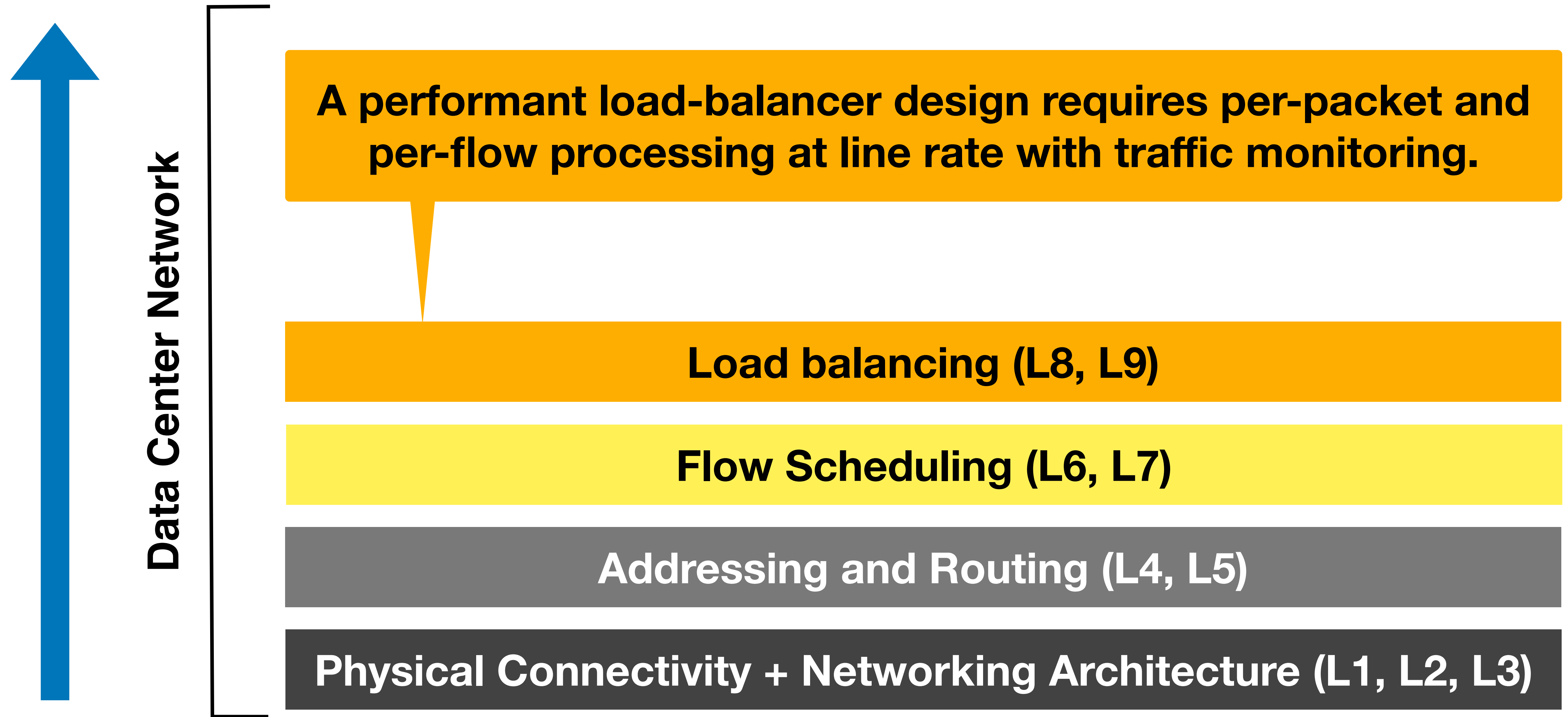
Where we are?



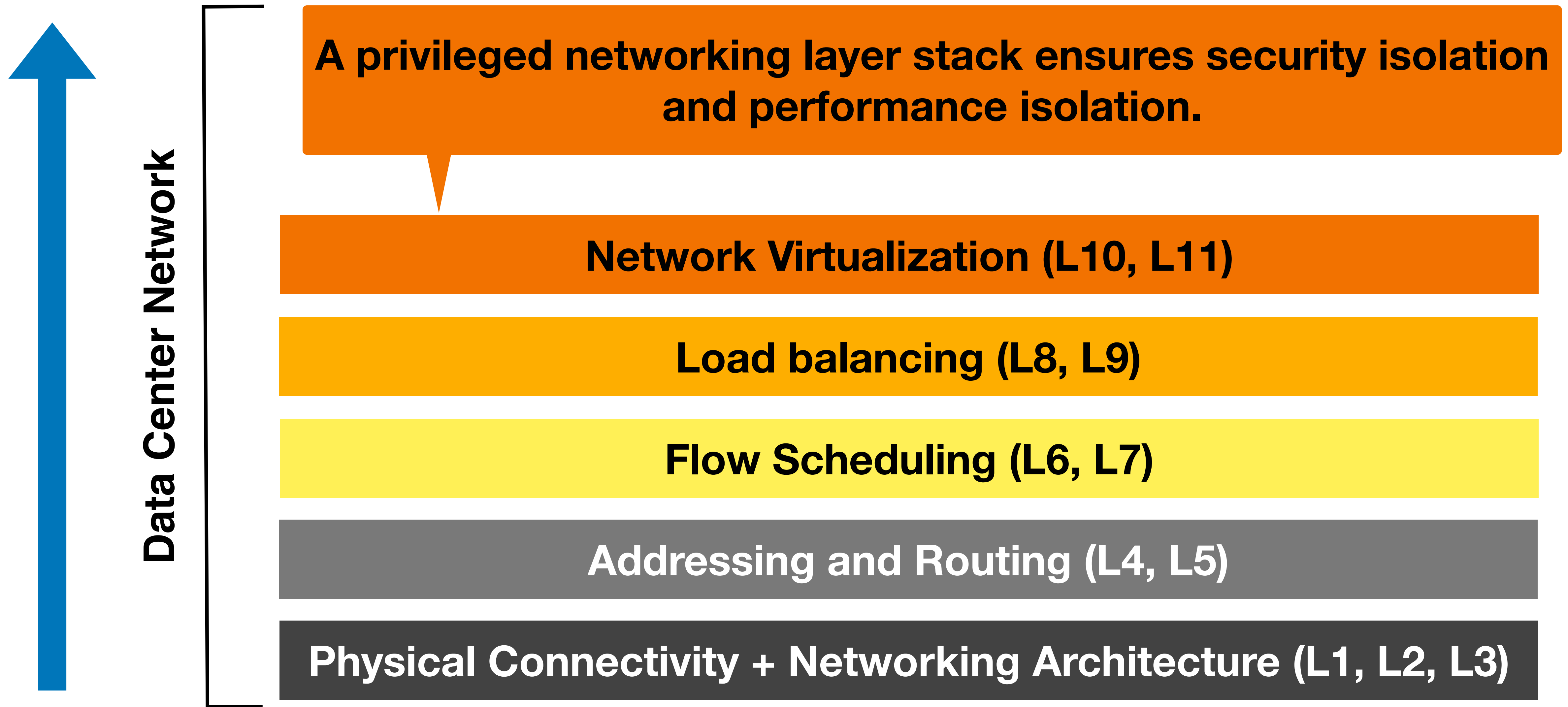
Where we are?



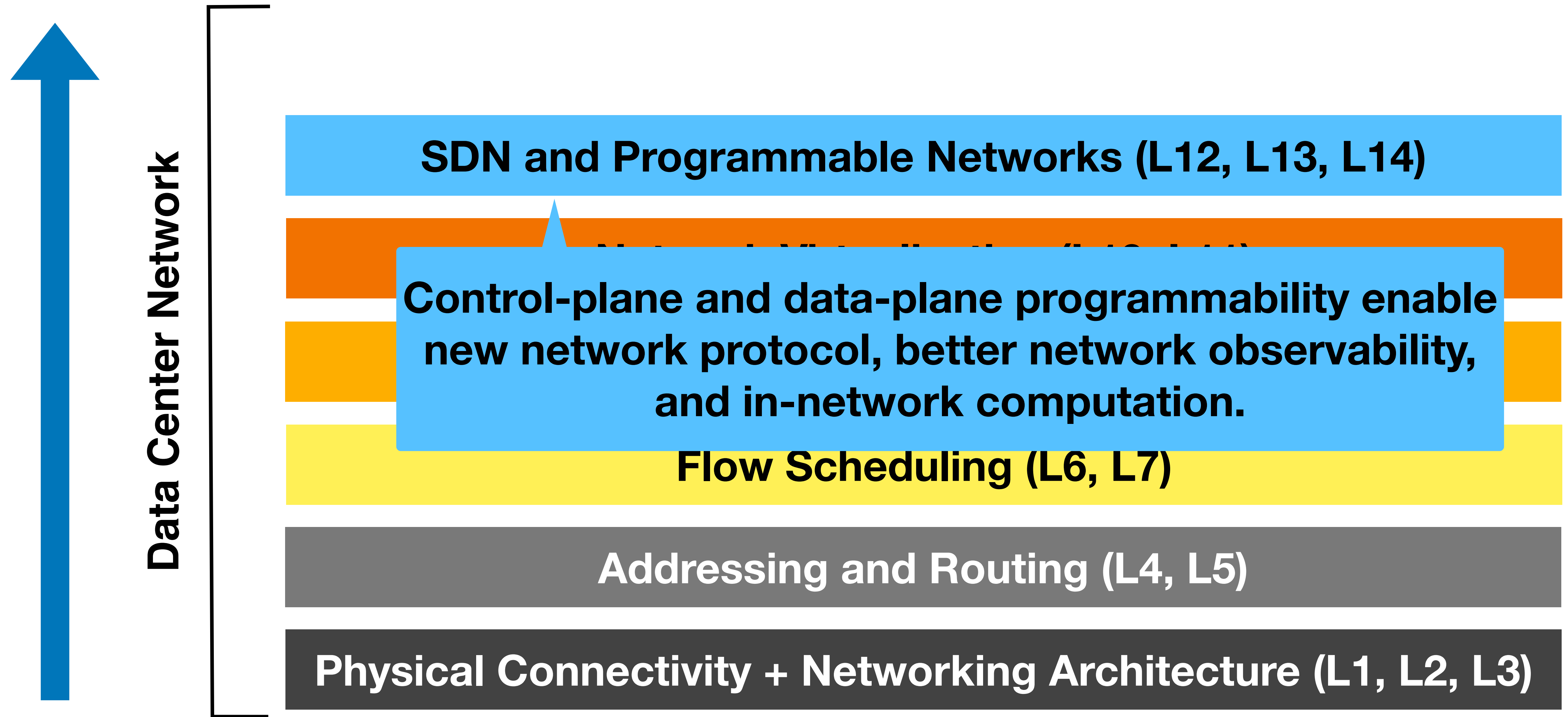
Where we are?



Where we are?



Where we are?



TCP is the dominant transport protocol in data centers.

**TCP is the dominant transport protocol in
data centers, but**

The Problem

- TCP cannot satisfy data center application demands
 - Throughput drops under burst
 - High tail latency
 - Inferior fairness guarantees
 - ...

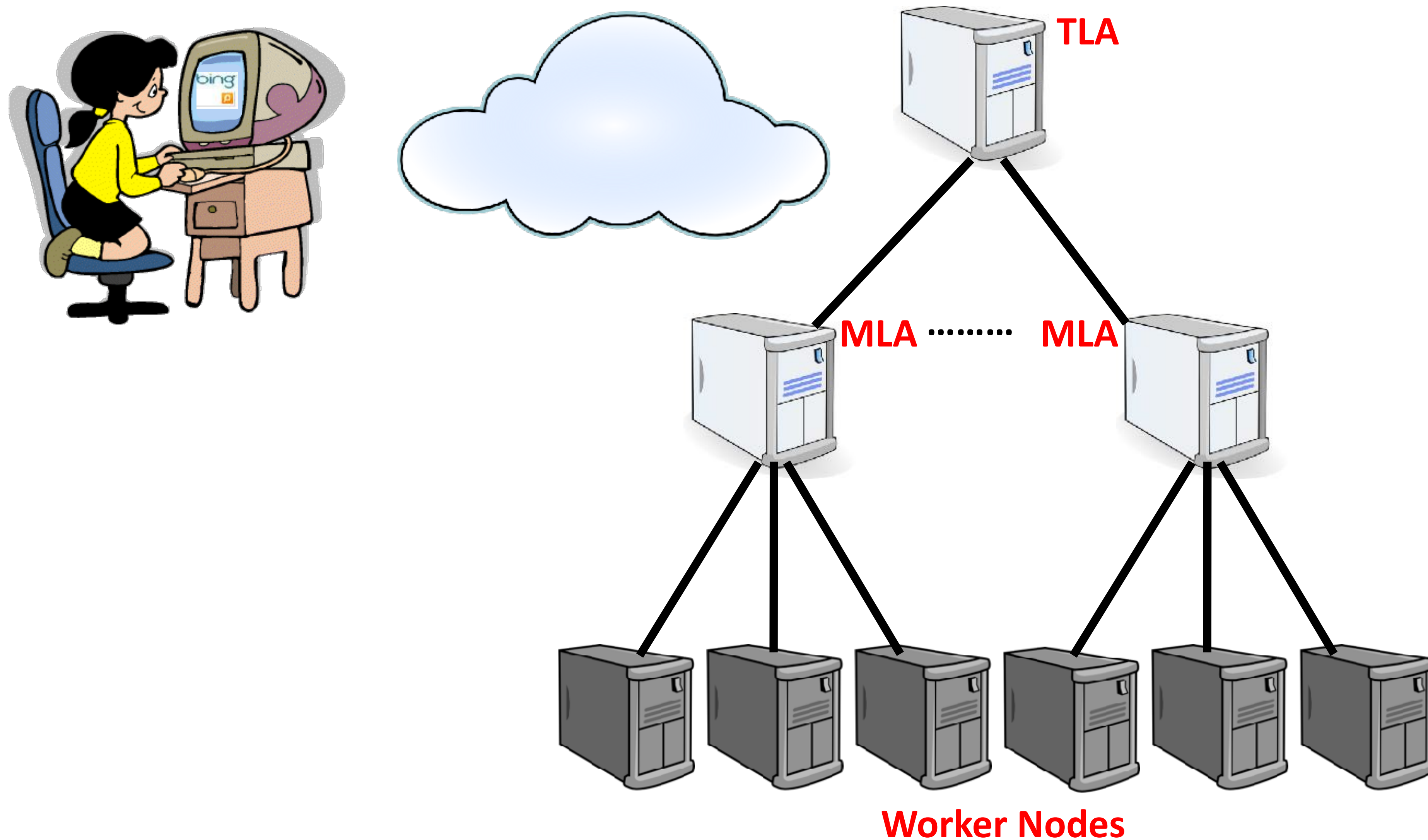
Limitations of Prior Solutions

- #1: Ad-hoc, inefficient, and expensive solutions
 - E.g., Work for app A, not app B
- #2: No solid understanding of consequences and tradeoffs
 - Not capture data center application characteristics
 - Not capture data center network characteristics

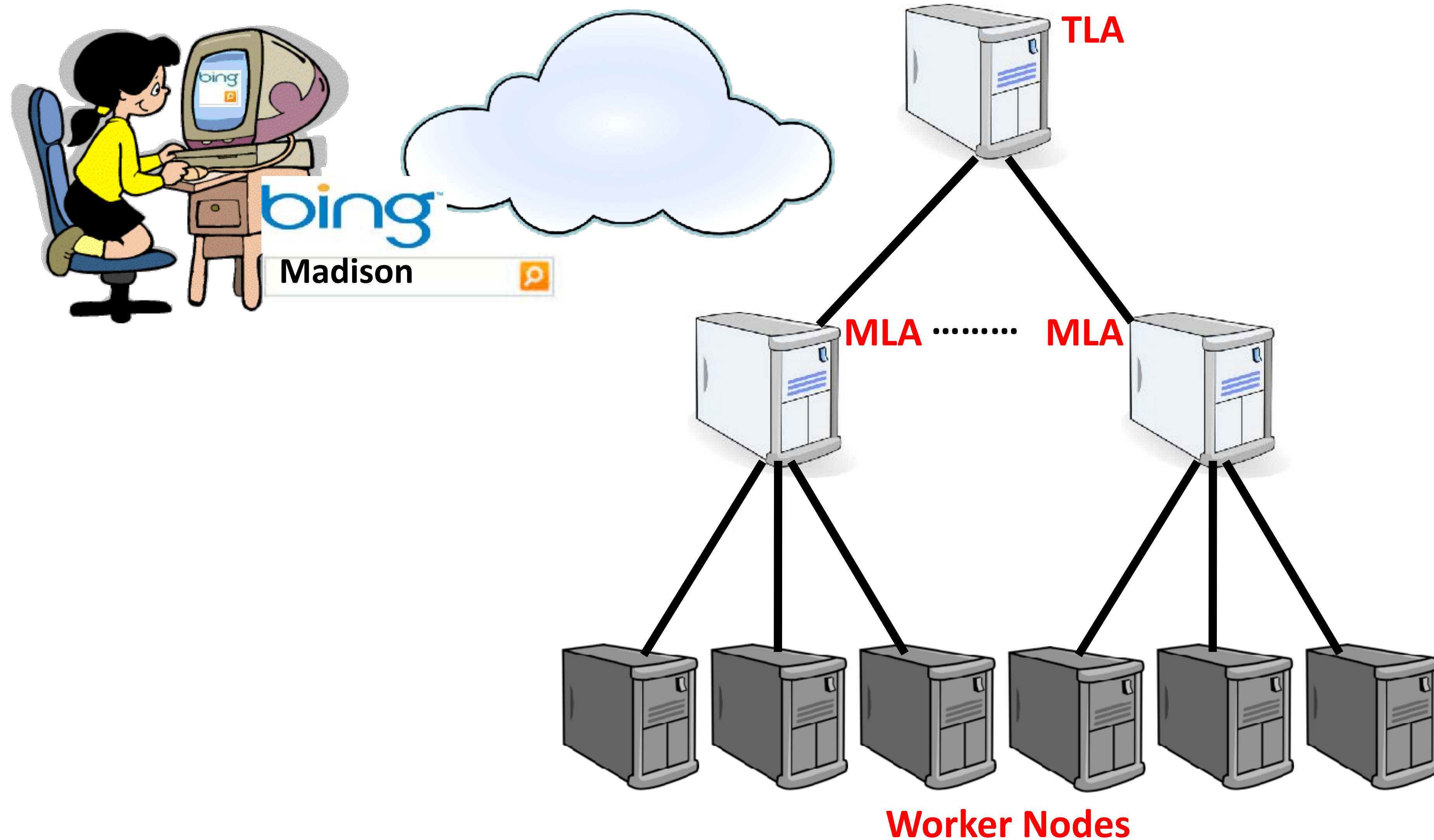
Microsoft Bing Search

- Large scale
 - Measurements from 6000 server production cluster
 - Collect more than 150TB of compressed data over a month
- Extensive instrumentation
 - Application-level
 - Socket-level
 - Selected packet-level

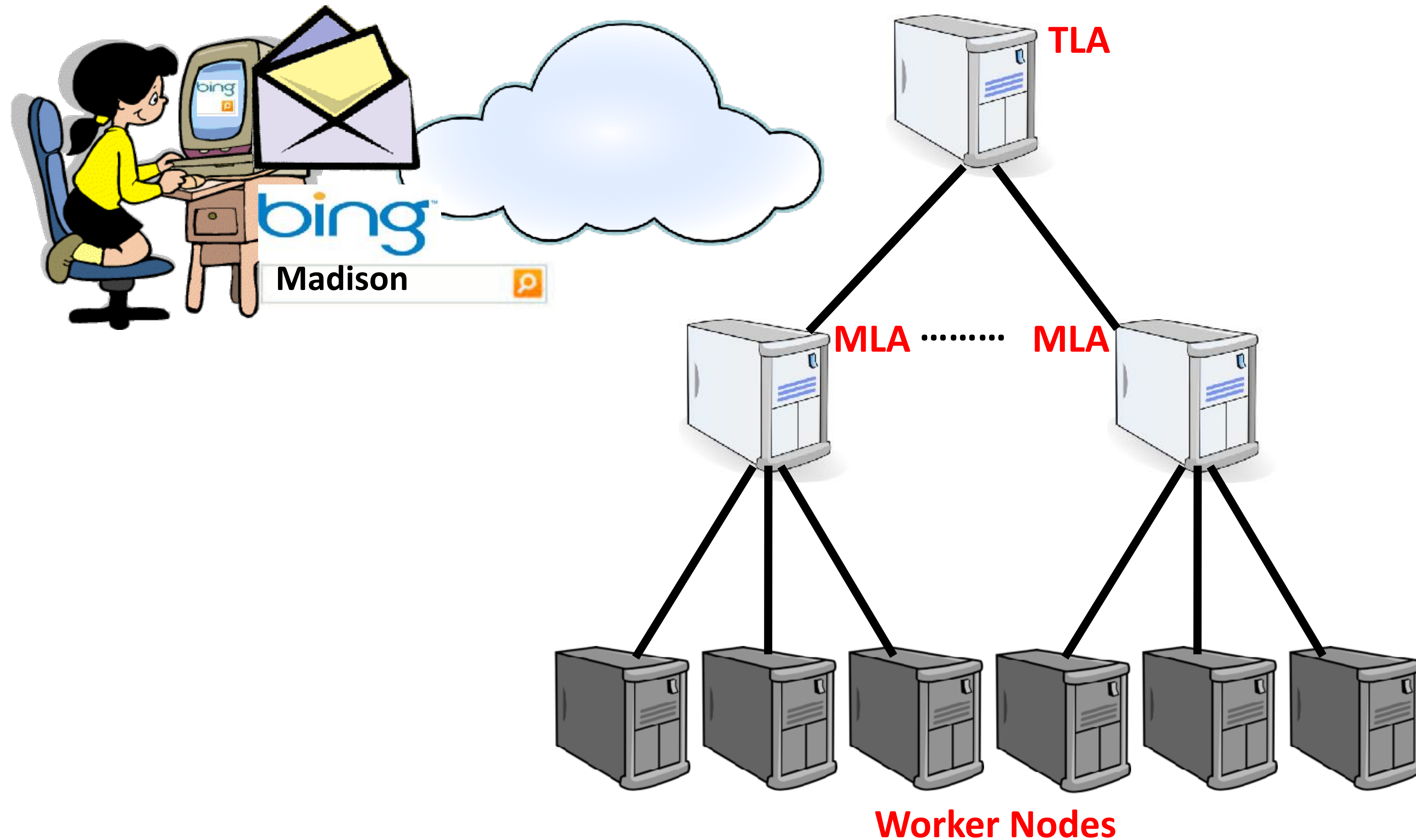
Partition/Aggregation Application Structure



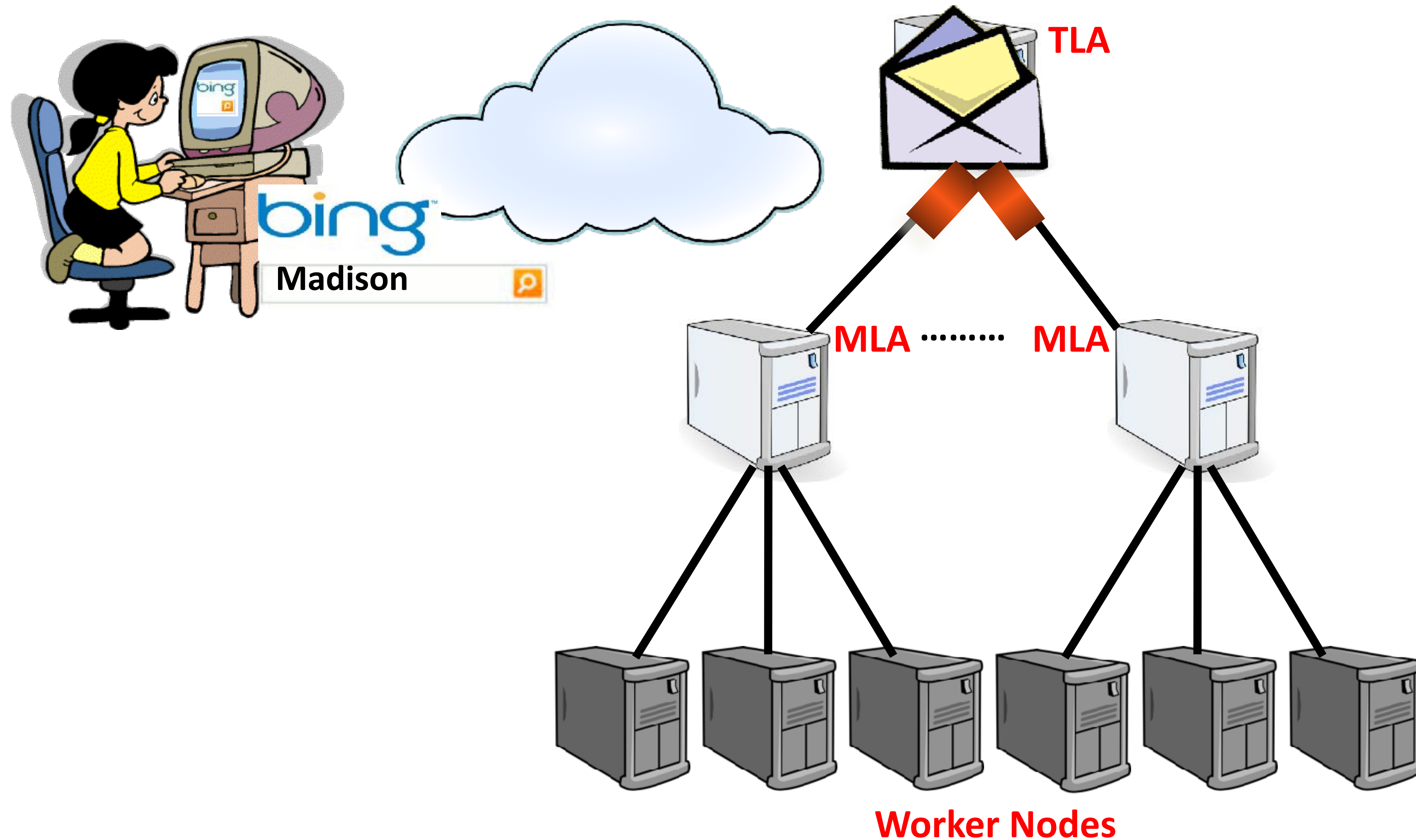
Partition/Aggregation Application Structure



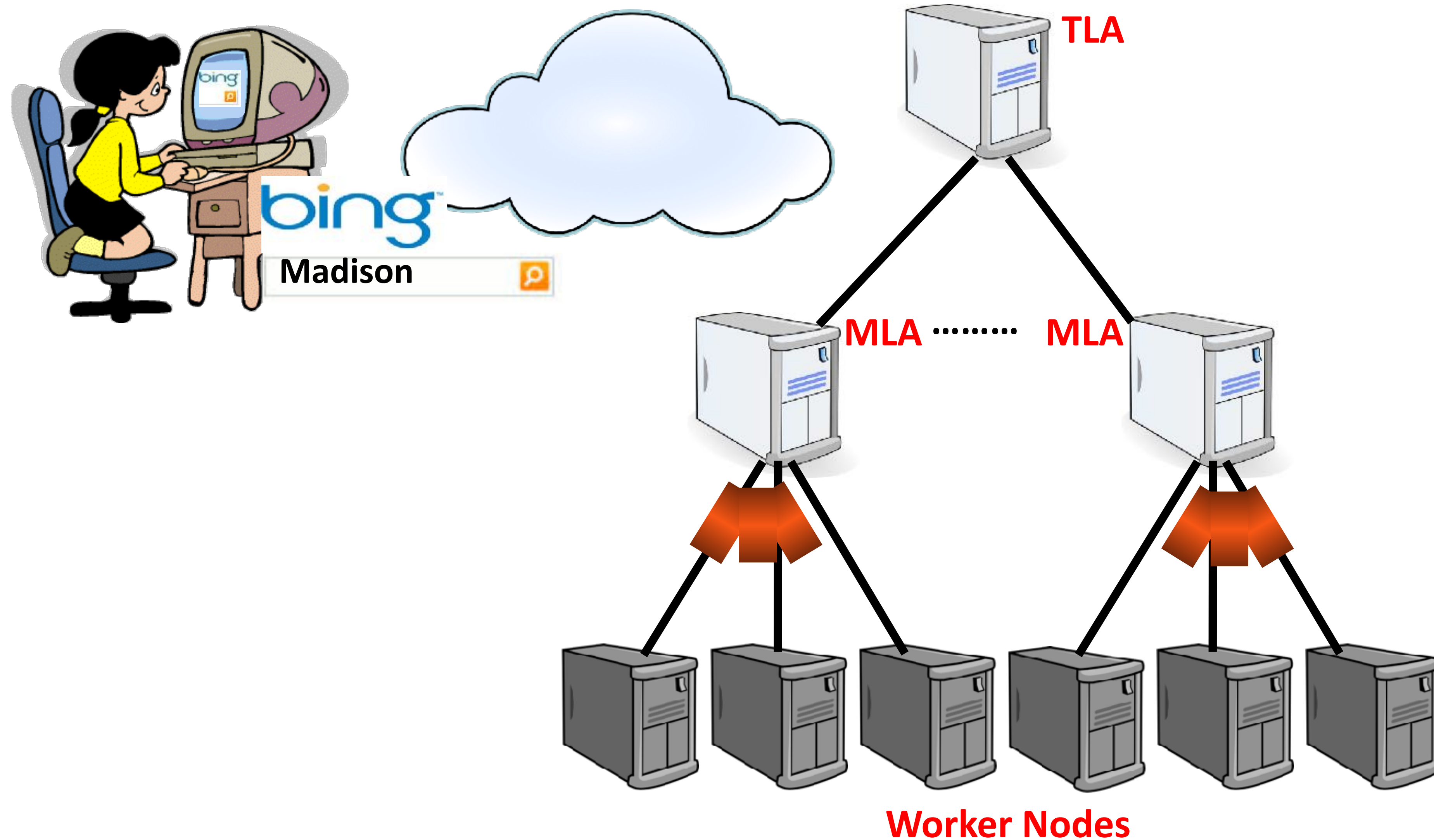
Partition/Aggregation Application Structure



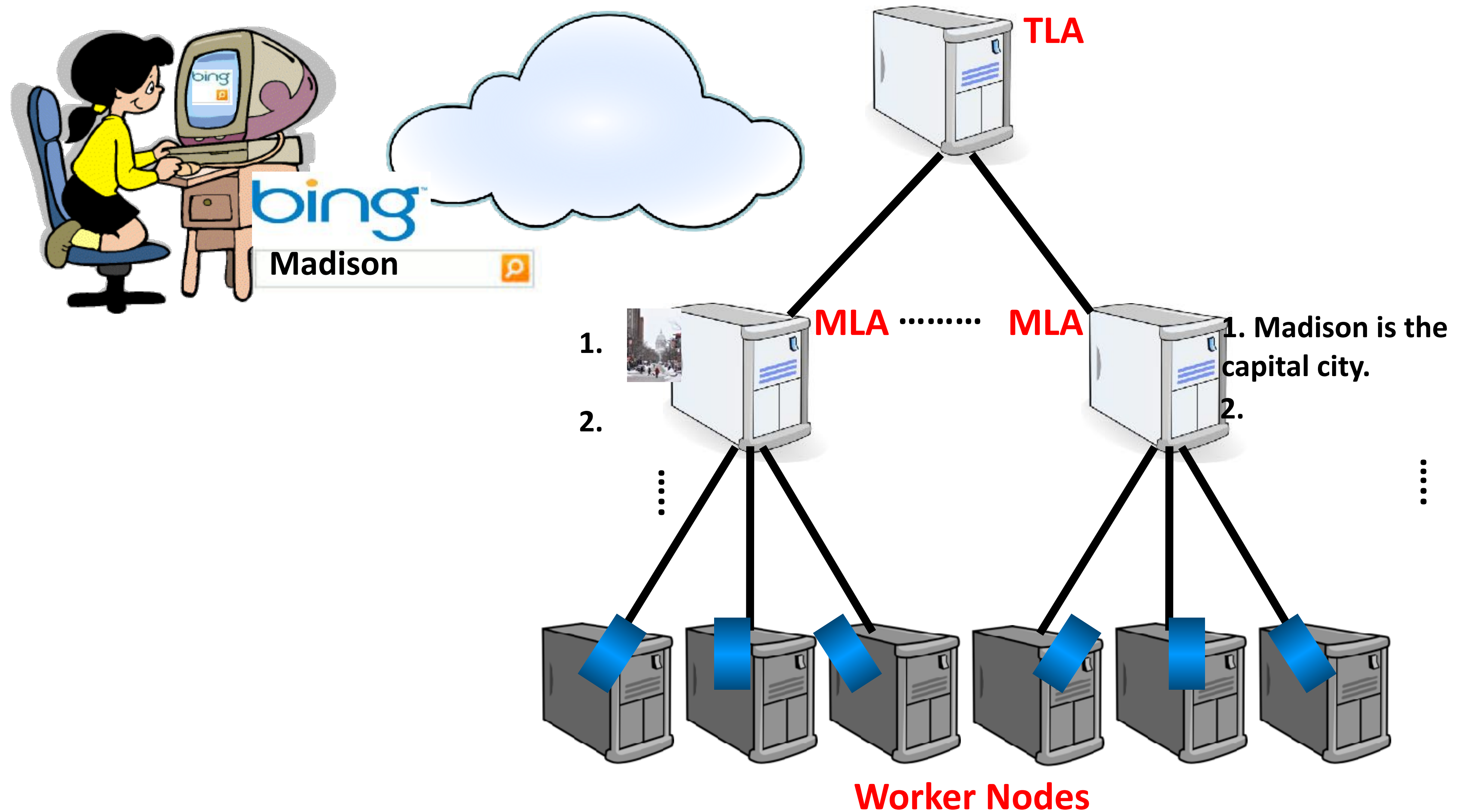
Partition/Aggregation Application Structure



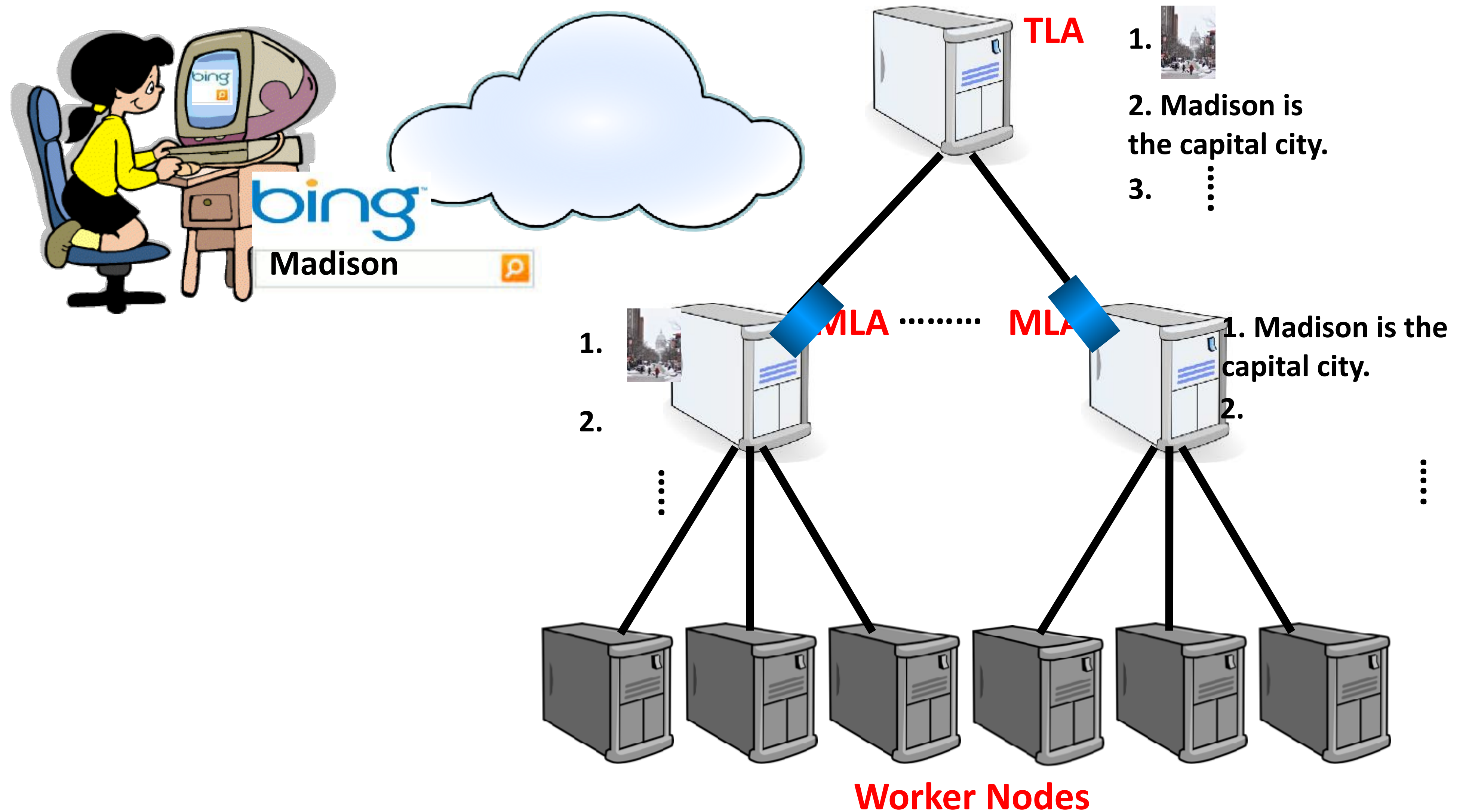
Partition/Aggregation Application Structure



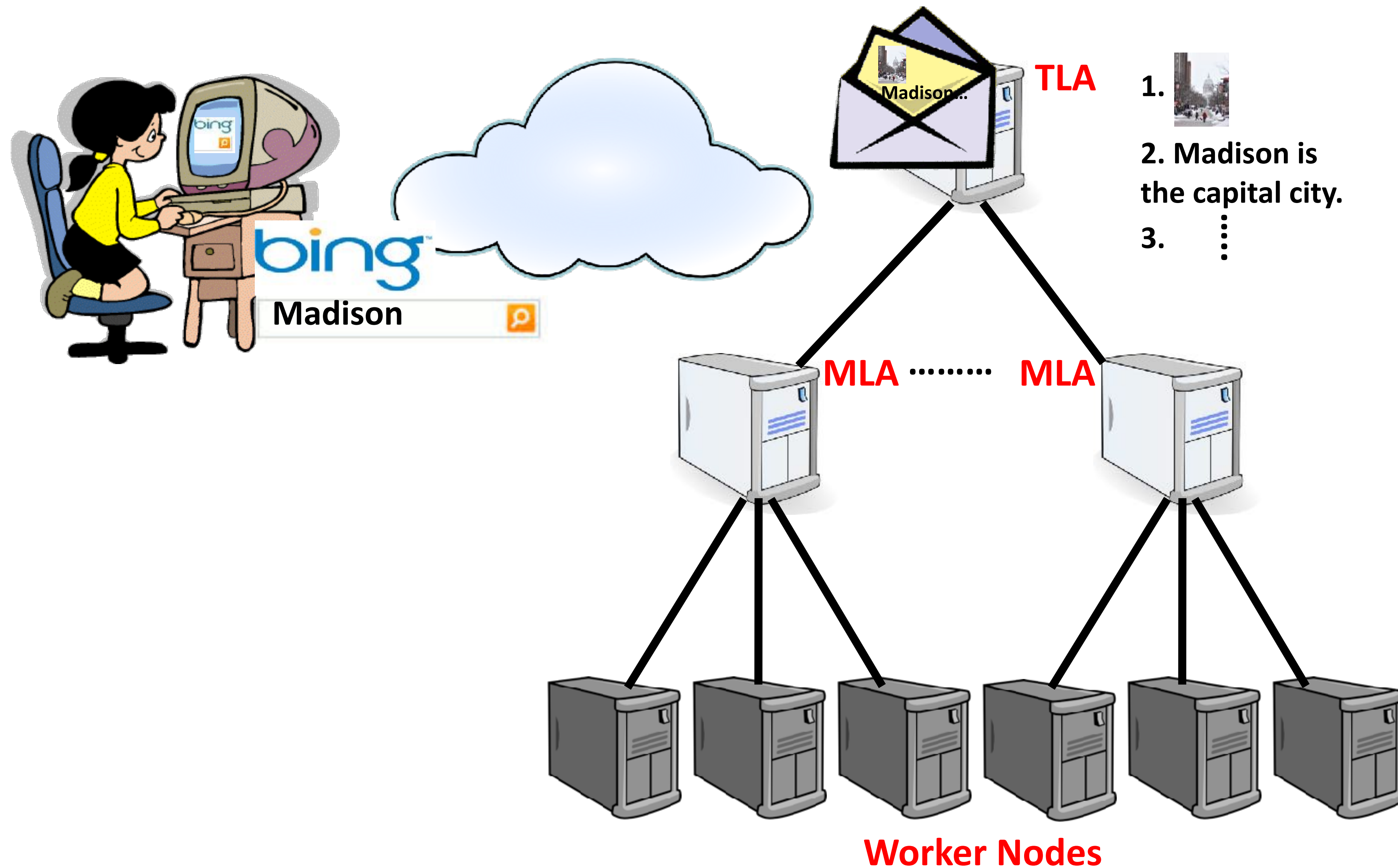
Partition/Aggregation Application Structure



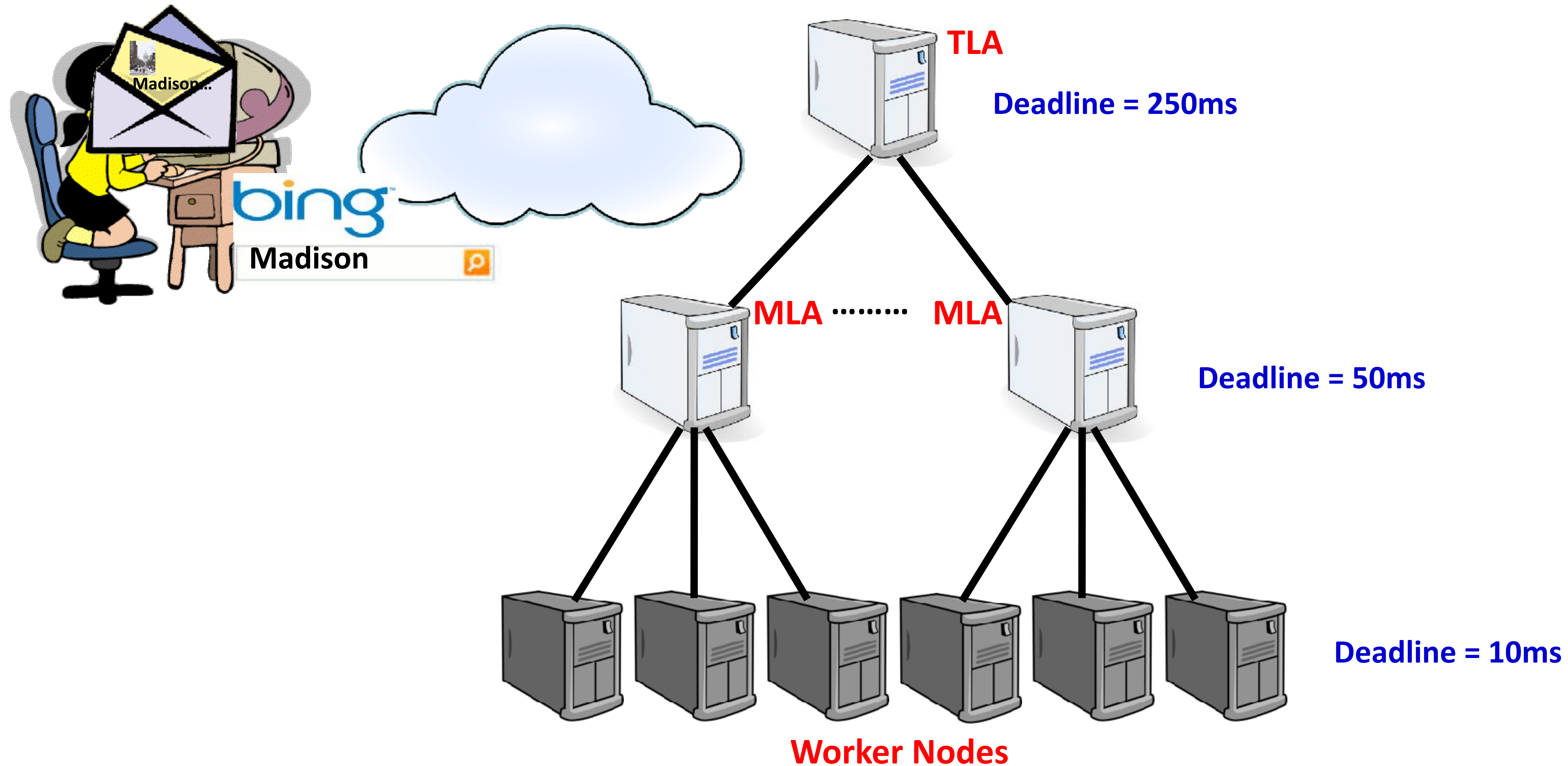
Partition/Aggregation Application Structure



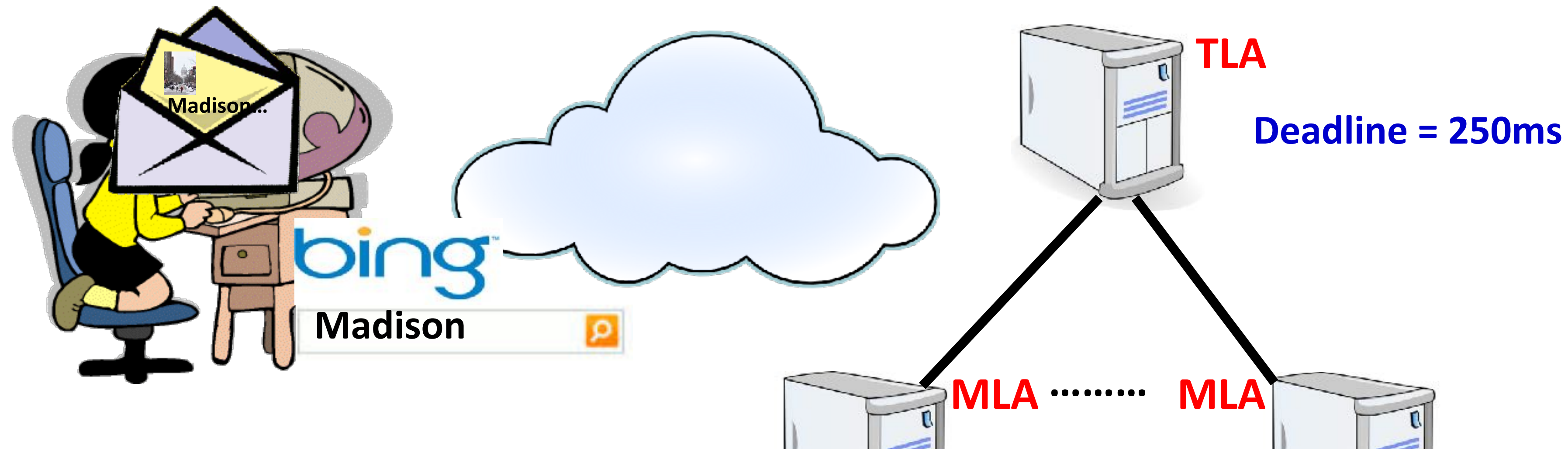
Partition/Aggregation Application Structure



Partition/Aggregation Application Structure



Partition/Aggregation Application Structure



- Time is money => Strict deadline (SLAs)
- Missed deadline => Lower quality results



Worker Nodes

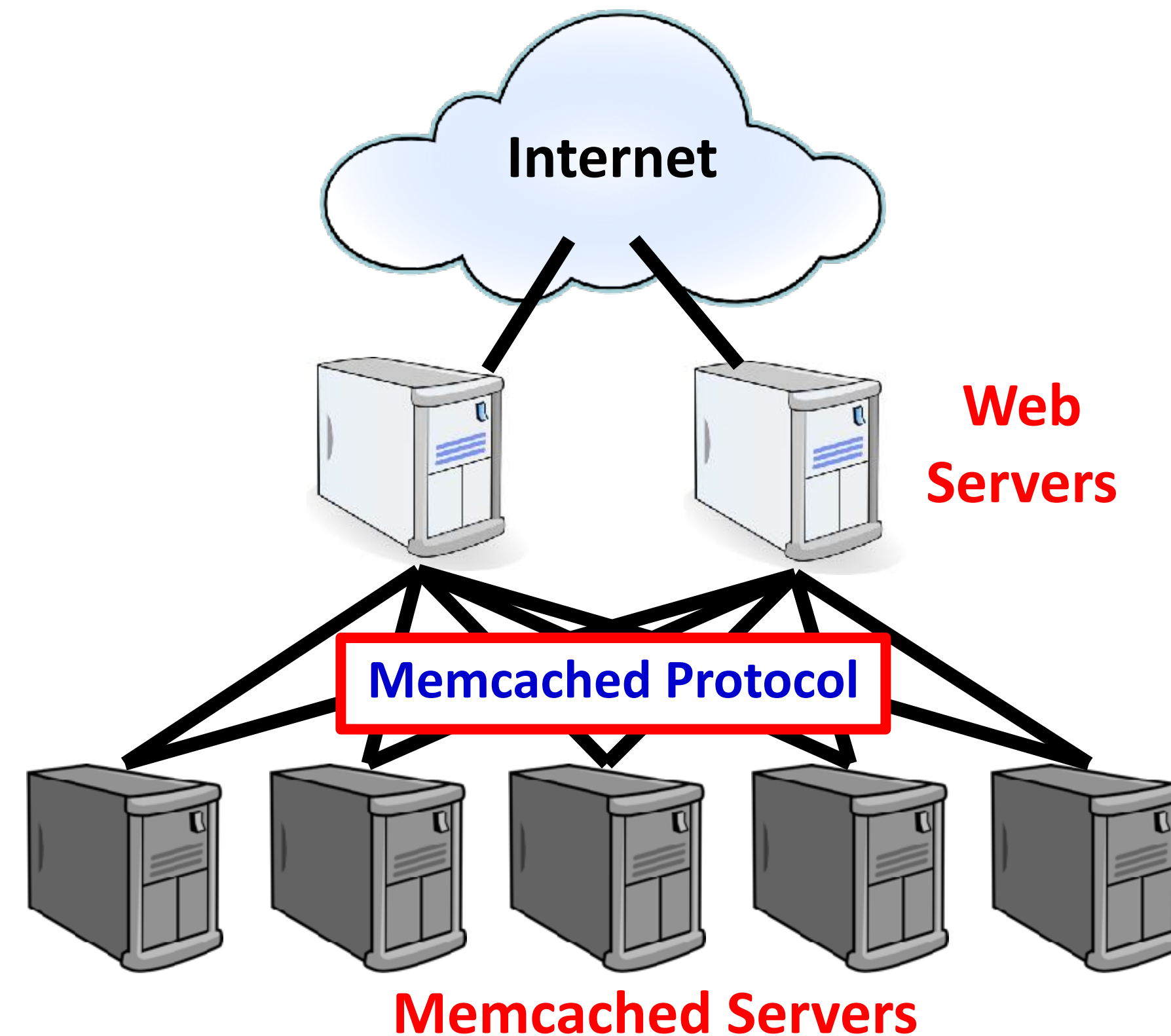
Deadline = 10ms

Partition/Aggregation application structure is everywhere!

Generality of Partition/Aggregation

- The foundation for many large-scale web applications.
 - Social network, ad selection/recommendation, AI inference,...

- Example: Facebook
 - Aggregators: Web servers
 - Workers: Memcached servers



Workload Characteristics

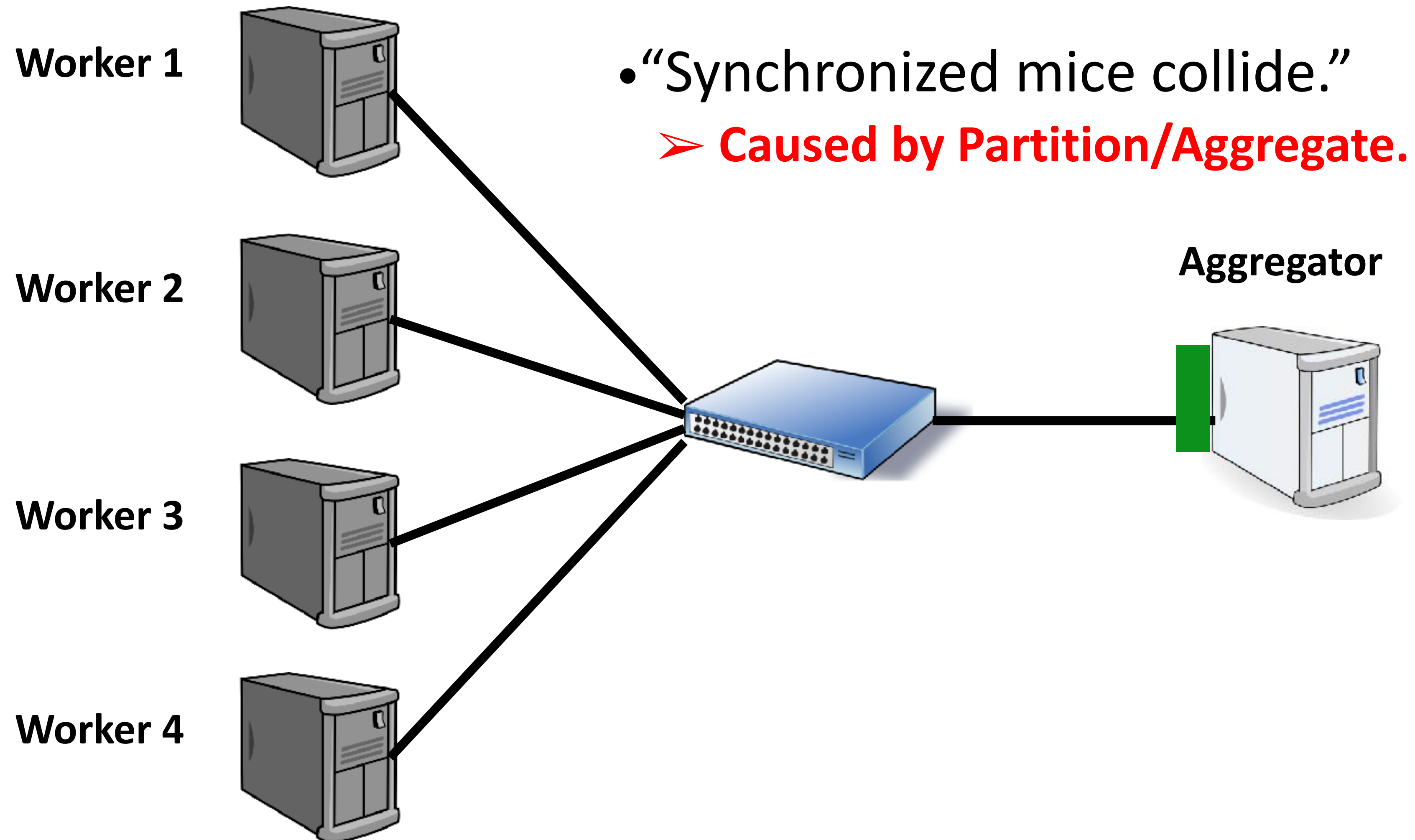
- #1: Split/Merge
 - Query-like execution model
- #2: Short message [50K-1MB]
 - Coordination, control state,...
- #3: Large flows [1MB-50MB]
 - Data retrieval and update

Workload Characteristics

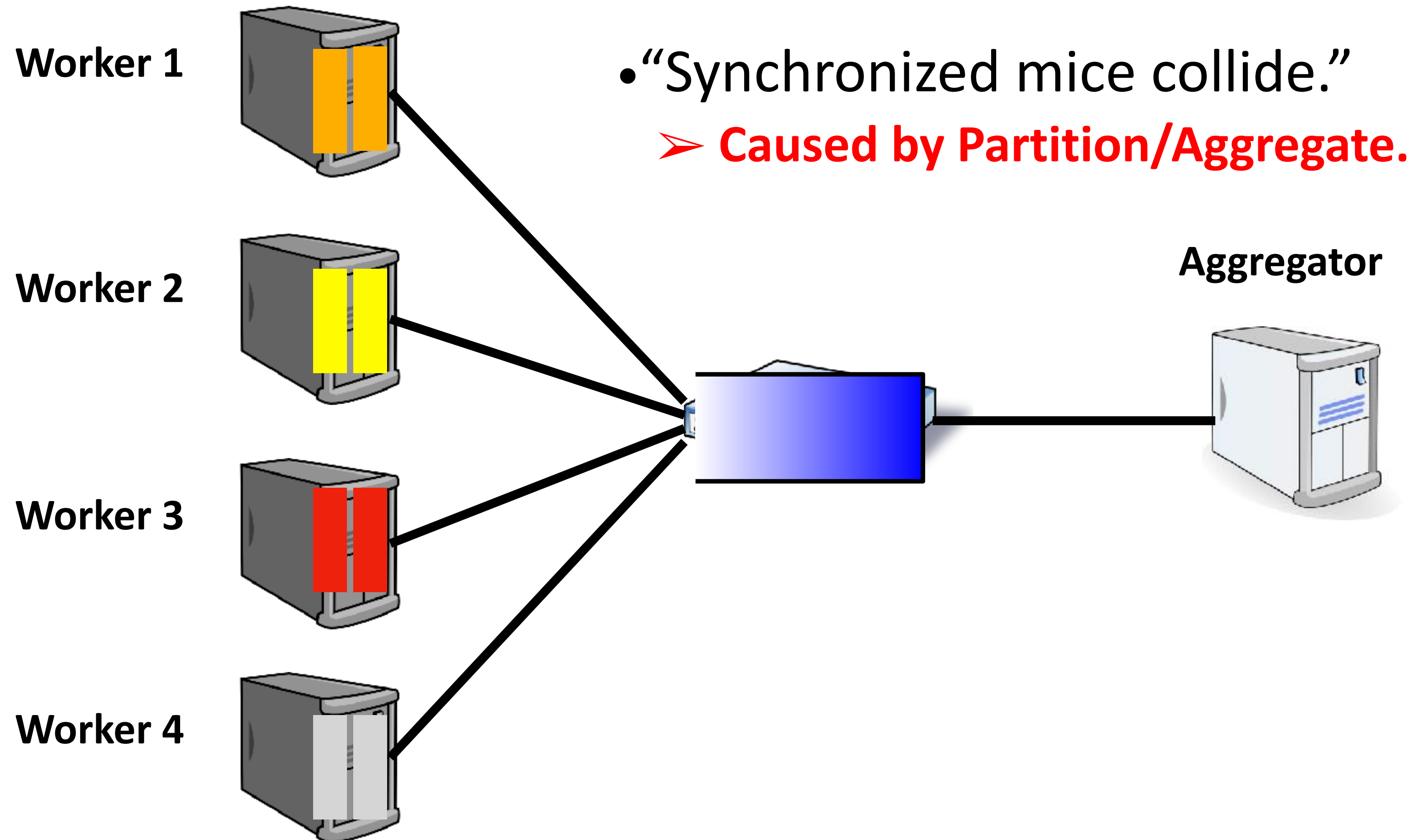
- #1: Split/Merge
 - Query-like execution model→ Delay-sensitive
- #2: Short message [50K-1MB]
 - Coordination, control state,...→ Delay-sensitive
- #3: Large flows [1MB-50MB]
 - Data retrieval and update→ Throughput-sensitive

Given these, what are the performance abnormalities?

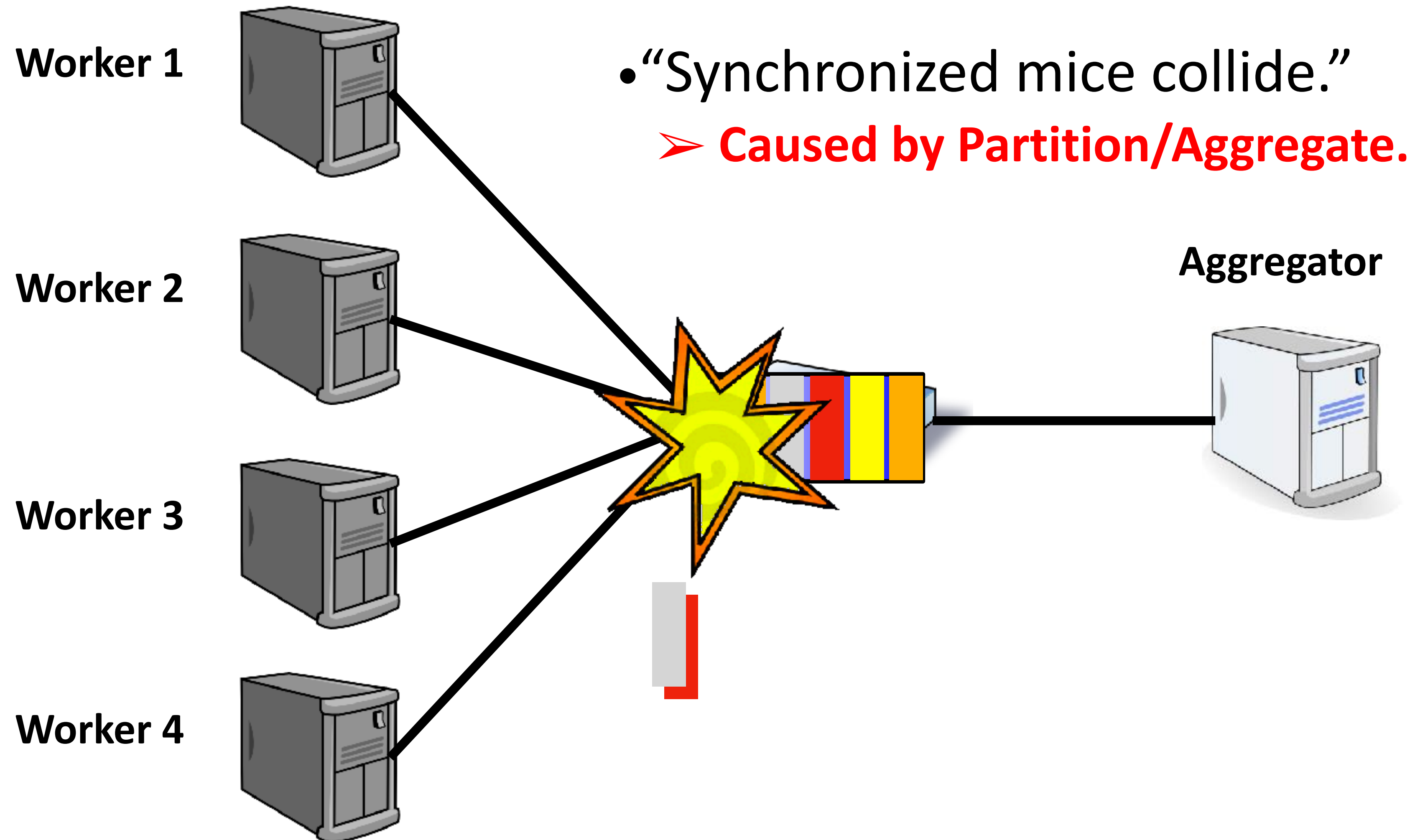
Issue #1: Incast



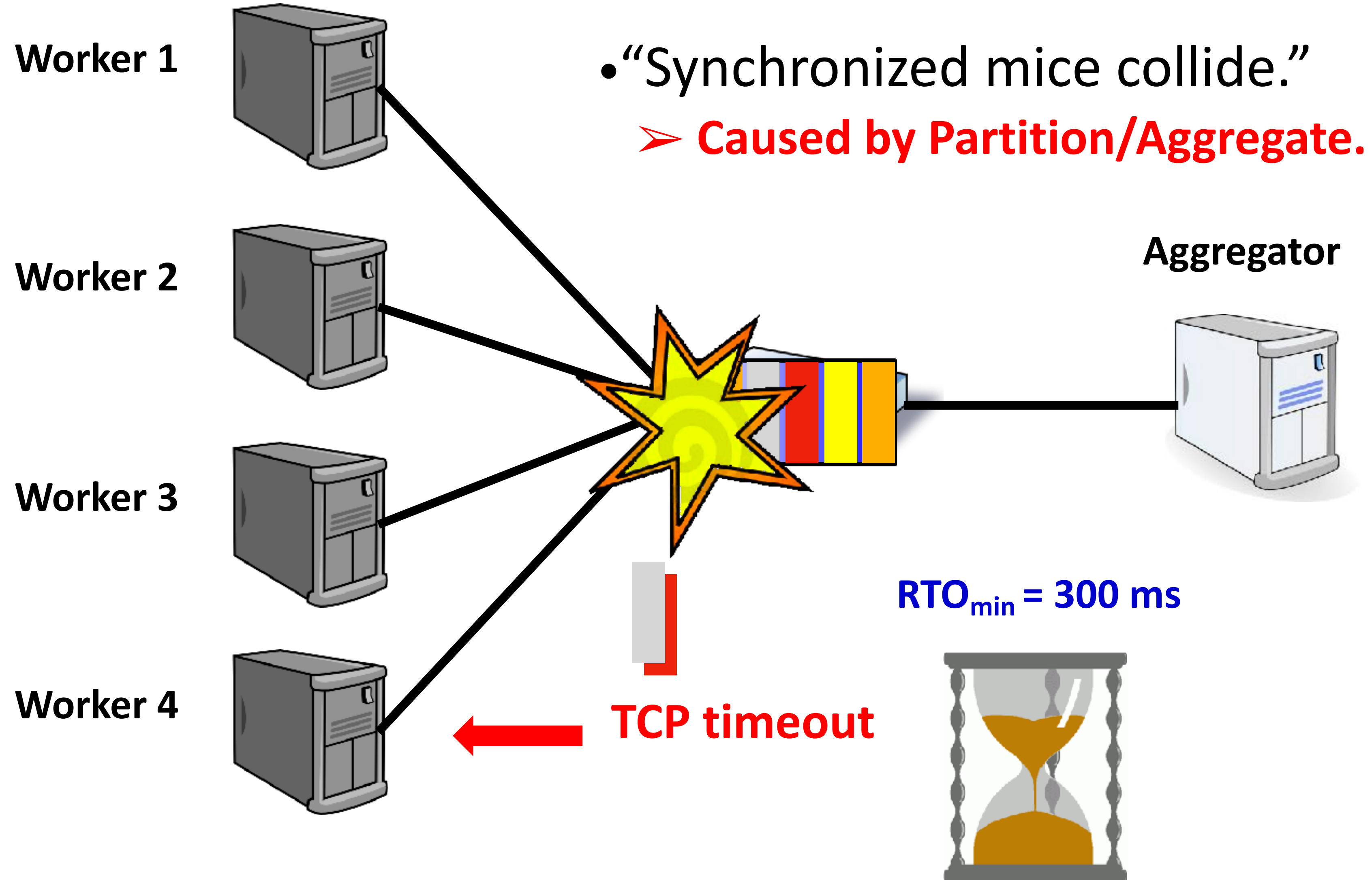
Issue #1: Incast



Issue #1: Incast

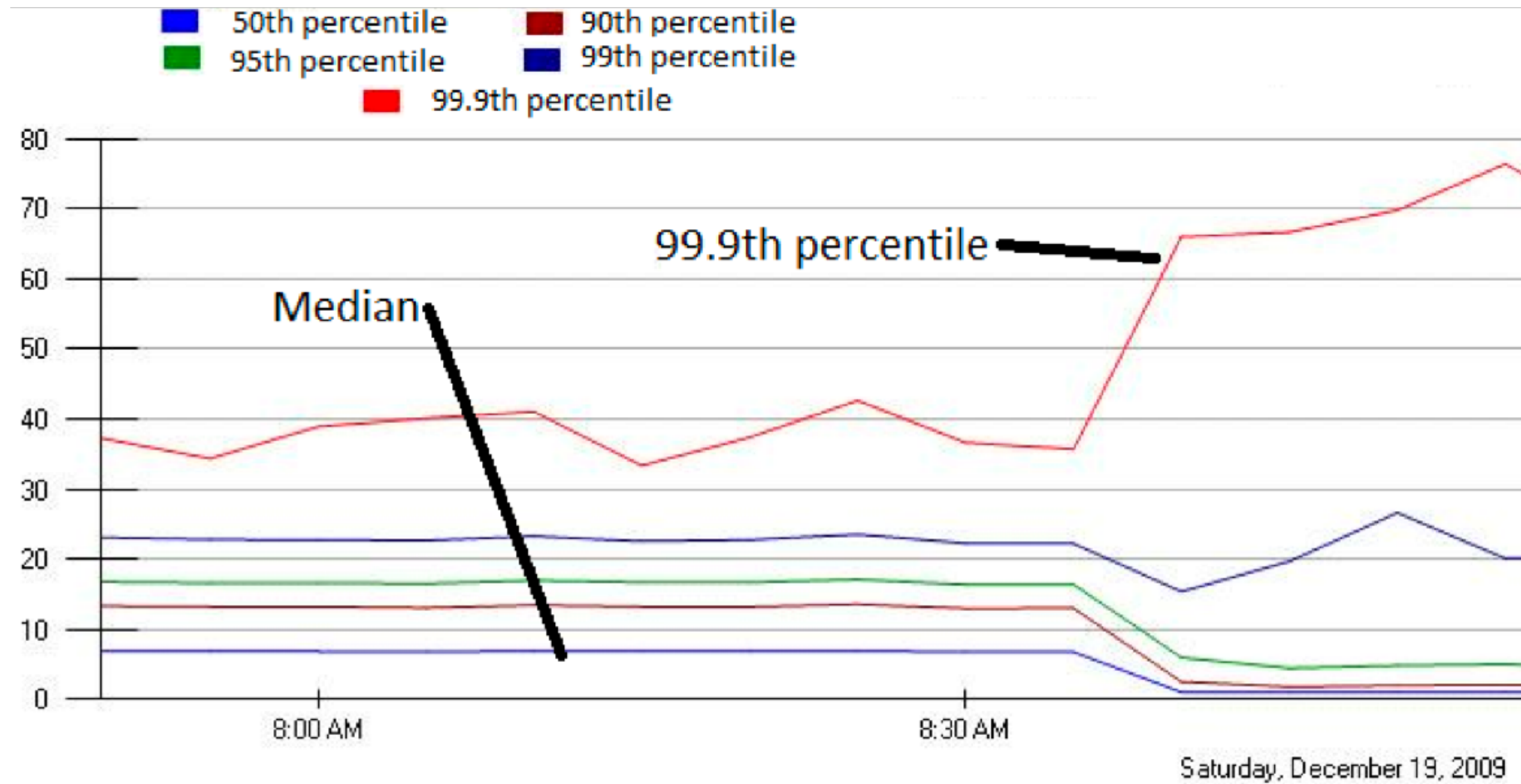


Issue #1: Incast

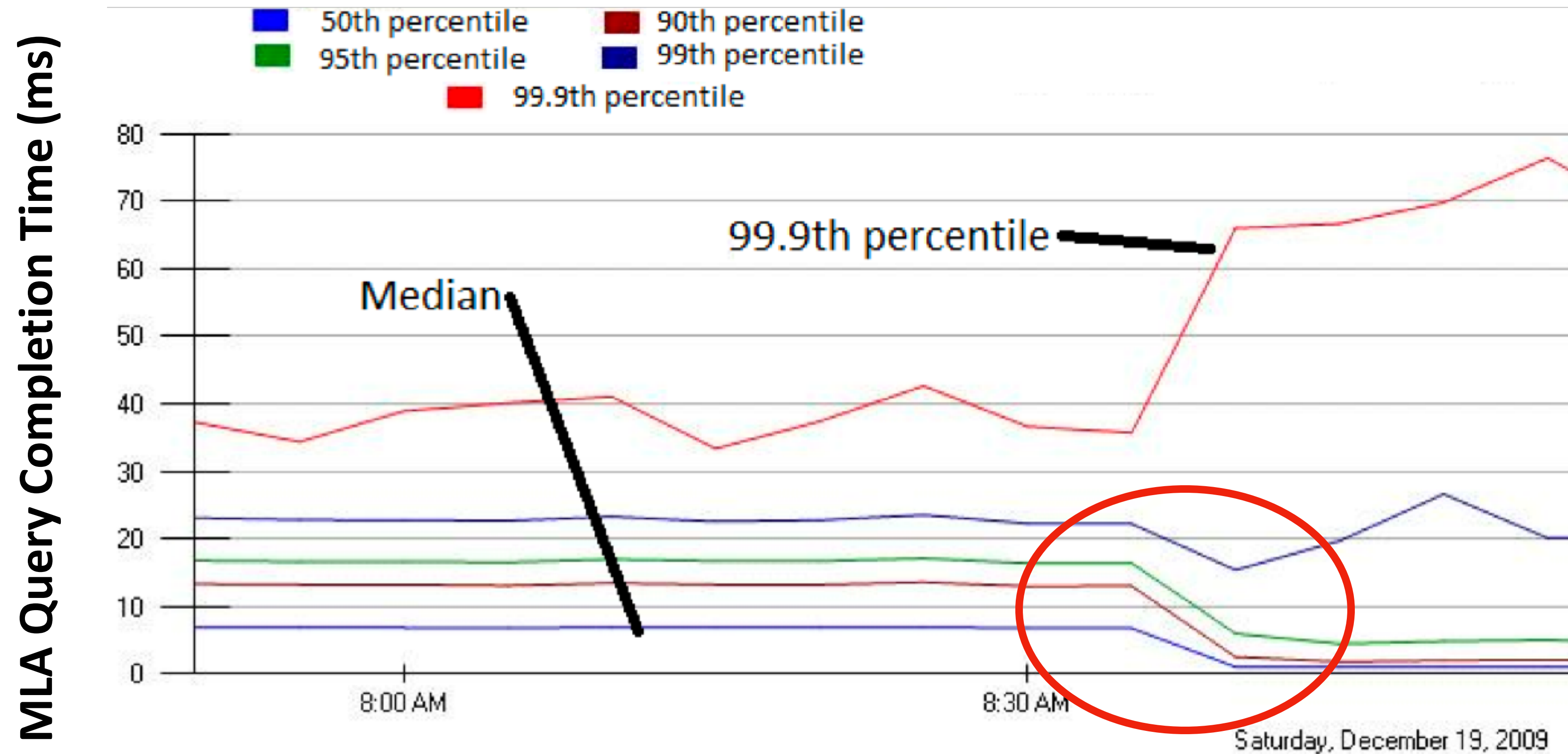


Performance under Incast

MLA Query Completion Time (ms)

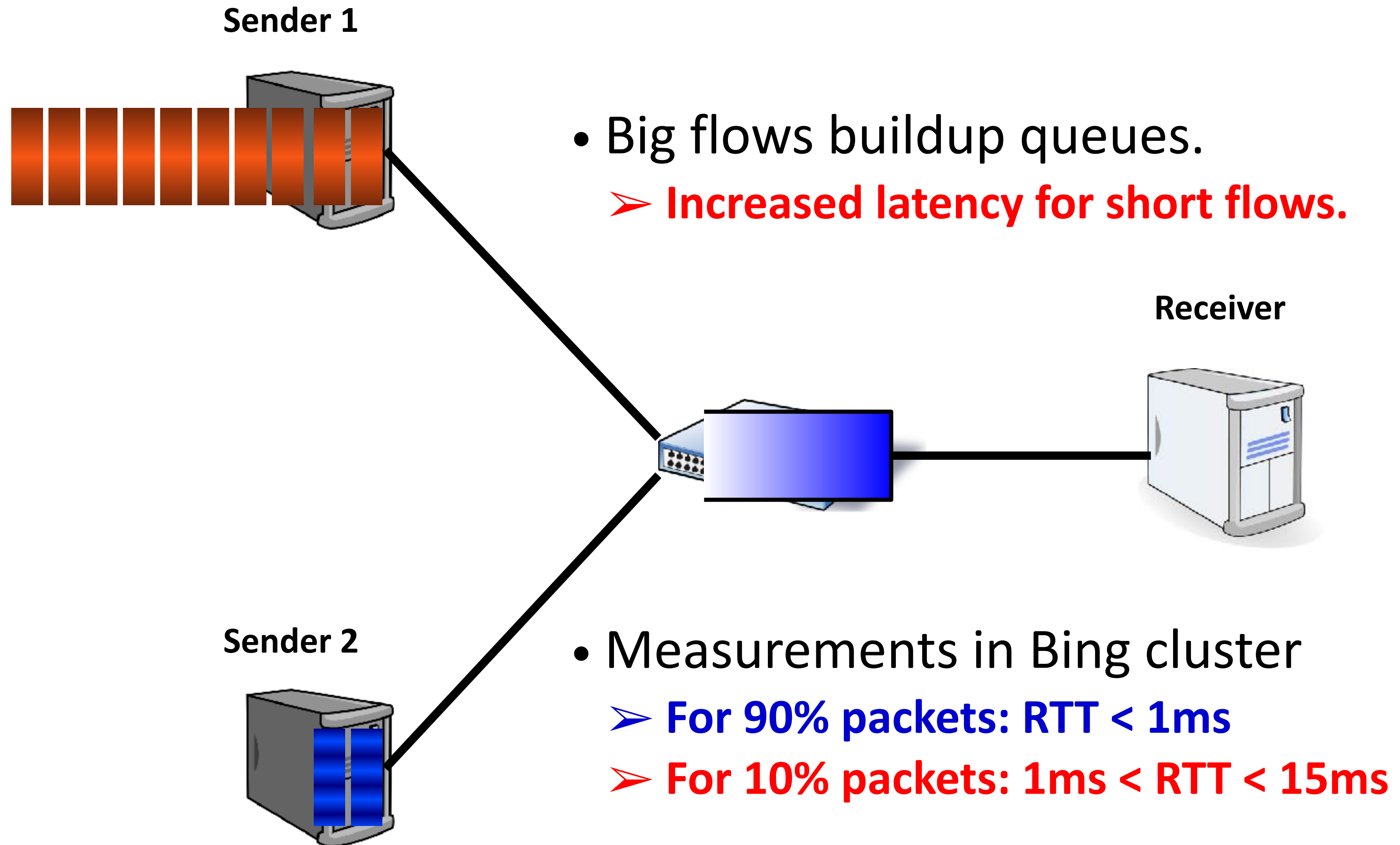


Performance under Incast



- Requests are jittered over 10ms windows
- Jittering switched off around 8:30am

Issue #2: Queue Buildup

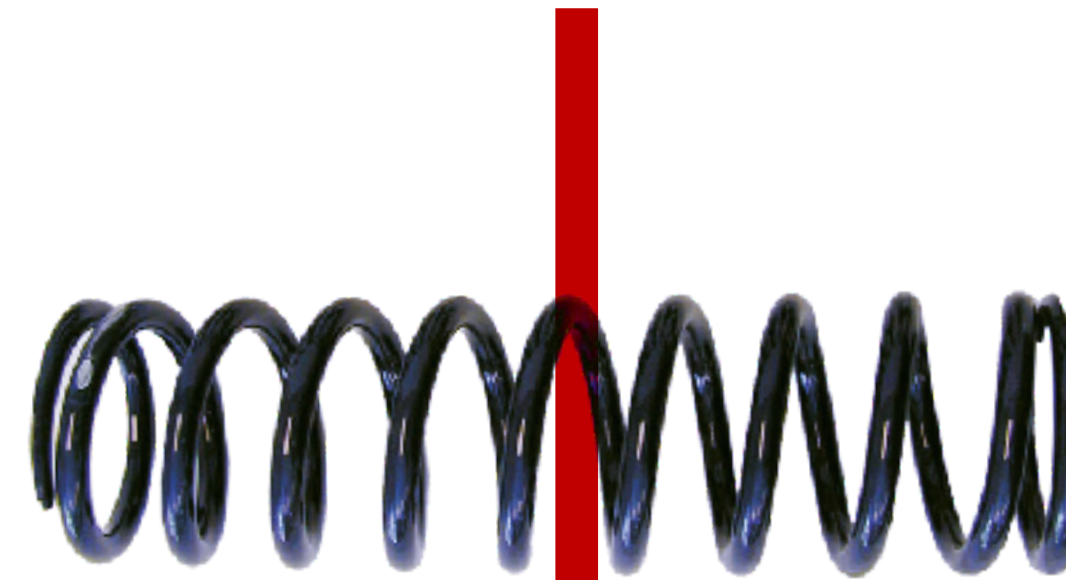


Design Goals of the “Enhanced” TCP

- #1: High burst performance
 - Tolerate the incast due to partition/aggregation
- #2: Low average/tail latency
 - Short flows
- #3: High throughput
 - Continuous data updates and large file transfers

Tension Between Requirements

**High Throughput
High Burst Tolerance**



Low Latency

Deep Buffers:

- Queuing Delays & Increase Latency

**Reduced RTO_{min}
(SIGCOMM '09)**

- Doesn't Help Latency

Shallow Buffers:

- Bad for Bursts & Throughput

AQM – RED:

- Avg Queue Not Fast Enough for Incast

Objective:

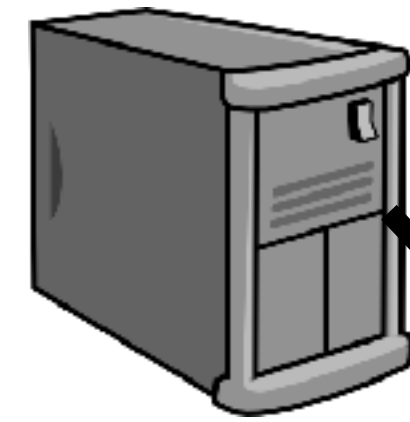
Low Queue Occupancy & High Throughput

How does DCTCP solve the problem?

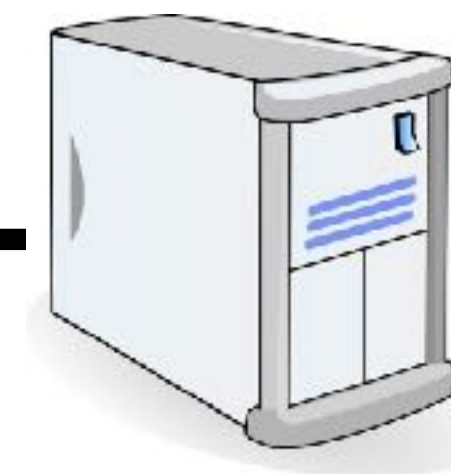
TCP/ECN Control Loop

Sender 1

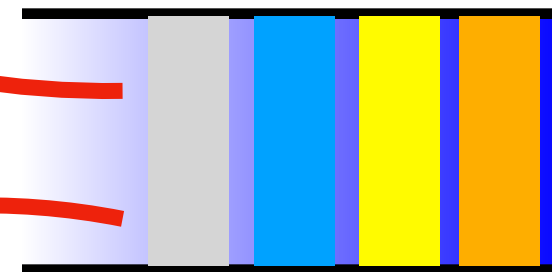
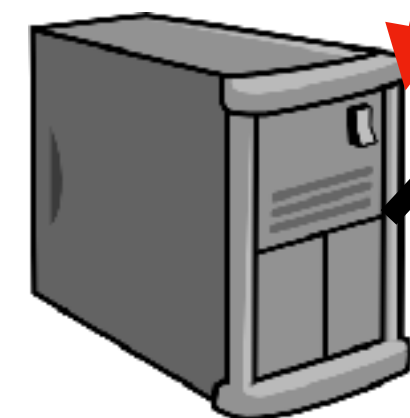
ECN = Explicit Congestion Notification



Receiver

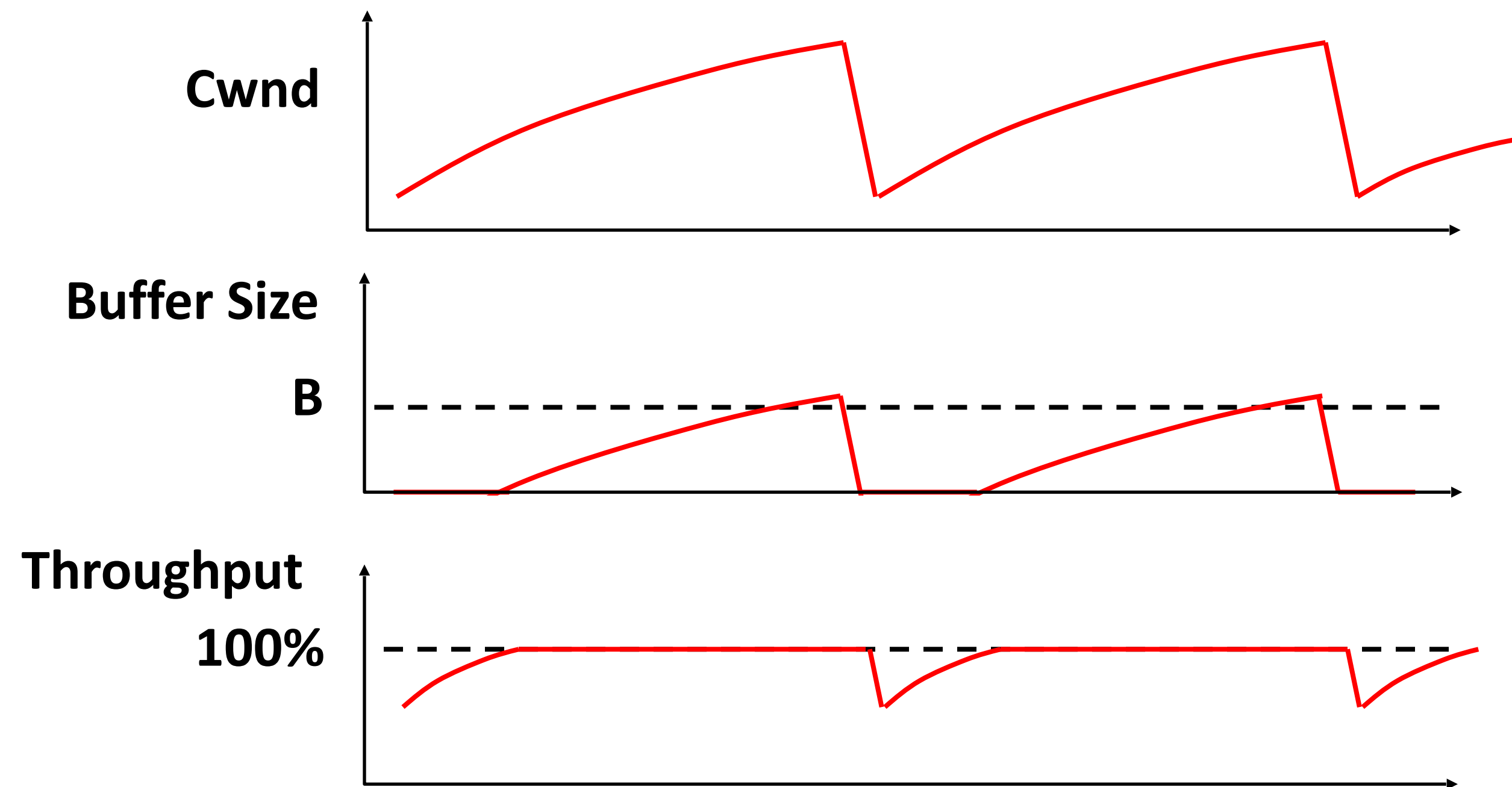


Sender 2



Buffer Sizing: Small Queue

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



Buffer Sizing: Small Queue

- 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: $C \times \frac{RTT}{\sqrt{N}}$ is enough

Buffer Sizing: Small Queue

- 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: $C \times \frac{RTT}{\sqrt{N}}$ is enough
- Can't rely on stat-mux benefit in the data center
 - Measurements show typically 1-2 big flows at each server, at most 4

Buffer Sizing: Small Queue

- 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: $C \times \frac{RTT}{\sqrt{N}}$ is enough
- Can't rely on stat-mux benefit in the data center
 - 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

Key Ideas of DCTCP

- #1: React proportional the extent of congestion, not its presence
 - Reduce variances in sending rates, lowering queue requirement

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%

Key Ideas of DCTCP

- #1: React proportional the extent of congestion, not its presence
 - Reduce variances in sending rates, lowering queue requirement

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

DCTCP Algorithm

Sender

Switch

Receiver

DCTCP Algorithm

Sender

Switch

Receiver

Sender side:

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

$$F = \frac{\# \text{ of marked ACKs}}{\text{Total \# of ACKs}}$$

$$\alpha \leftarrow (1 - g)\alpha + gF$$

$$Cwnd \leftarrow (1 - \frac{\alpha}{2})Cwnd$$

DCTCP Algorithm

Sender

Switch

Receiver

Sender side:

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

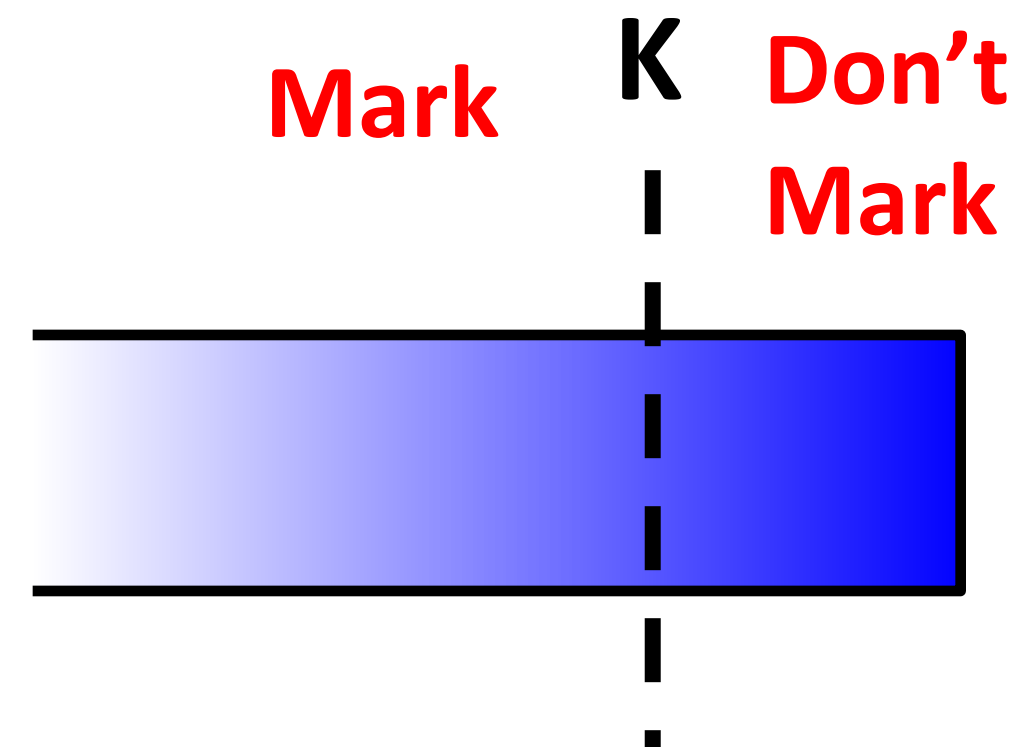
$$F = \frac{\# \text{ of marked ACKs}}{\text{Total \# of ACKs}}$$

$$\alpha \leftarrow (1 - g)\alpha + gF$$

$$Cwnd \leftarrow (1 - \frac{\alpha}{2})Cwnd$$

Switch side:

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



DCTCP Algorithm

Sender

Switch

Receiver

Sender side:

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

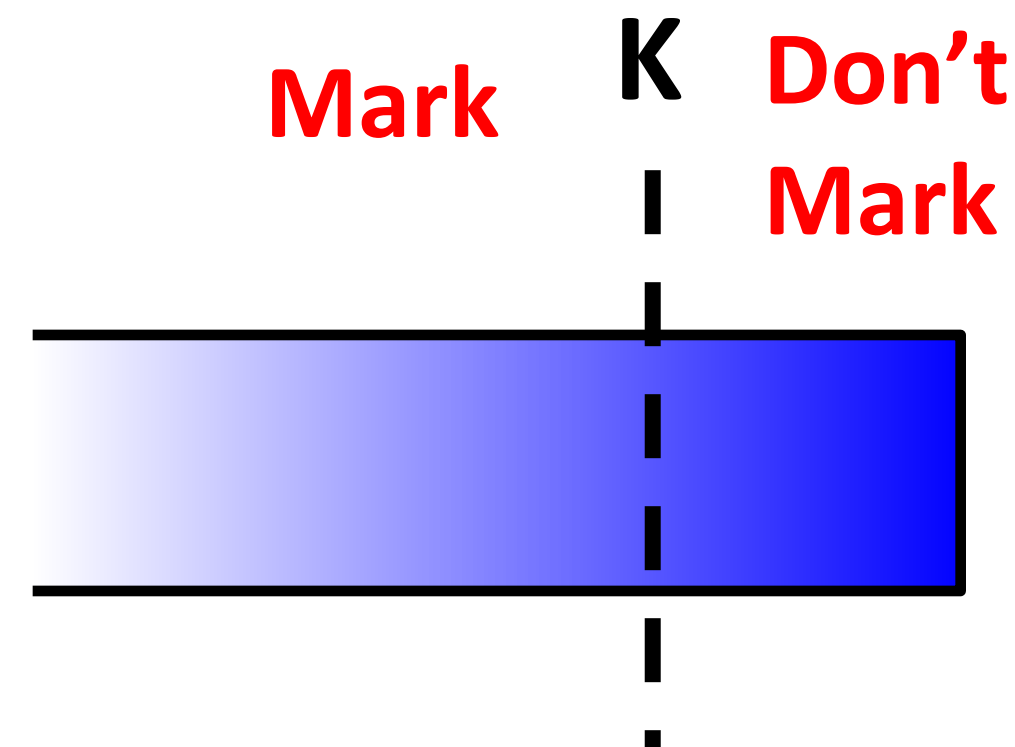
$$F = \frac{\# \text{ of marked ACKs}}{\text{Total \# of ACKs}}$$

$$\alpha \leftarrow (1 - g)\alpha + gF$$

$$Cwnd \leftarrow (1 - \frac{\alpha}{2})Cwnd$$

Switch side:

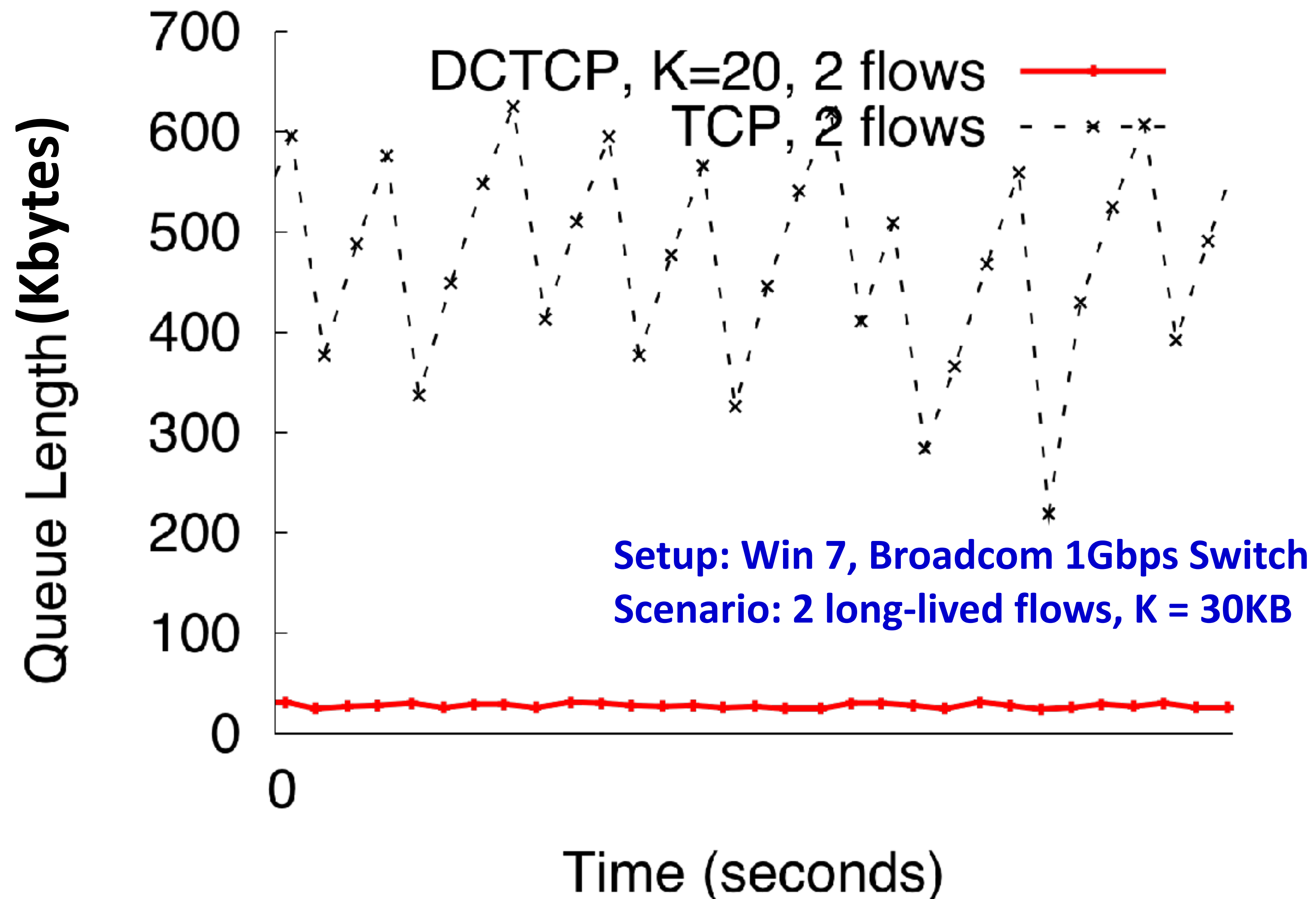
- Mark packets when **Queue Length > K**.



Receiver side:

- Piggyback

DCTCP Effect



Why does DCTCP work?

Efficiency of DCTCP

- #1: High burst performance

Efficiency of DCTCP

- #1: High burst performance

Aggressive marking —> Sources react before packets are dropped

Efficiency of DCTCP

- #1: High burst performance

Aggressive marking —> Sources react before packets are dropped

- #2: Low average/tail latency

Efficiency of DCTCP

- #1: High burst performance

Aggressive marking —> Sources react before packets are dropped

- #2: Low average/tail latency

Small buffer occupancy —> Low queueing delay

Efficiency of DCTCP

- #1: High burst performance

Aggressive marking —> Sources react before packets are dropped

- #2: Low average/tail latency

Small buffer occupancy —> Low queueing delay

- #3: High throughput

Efficiency of DCTCP

- #1: High burst performance

Aggressive marking —> Sources react before packets are dropped

- #2: Low average/tail latency

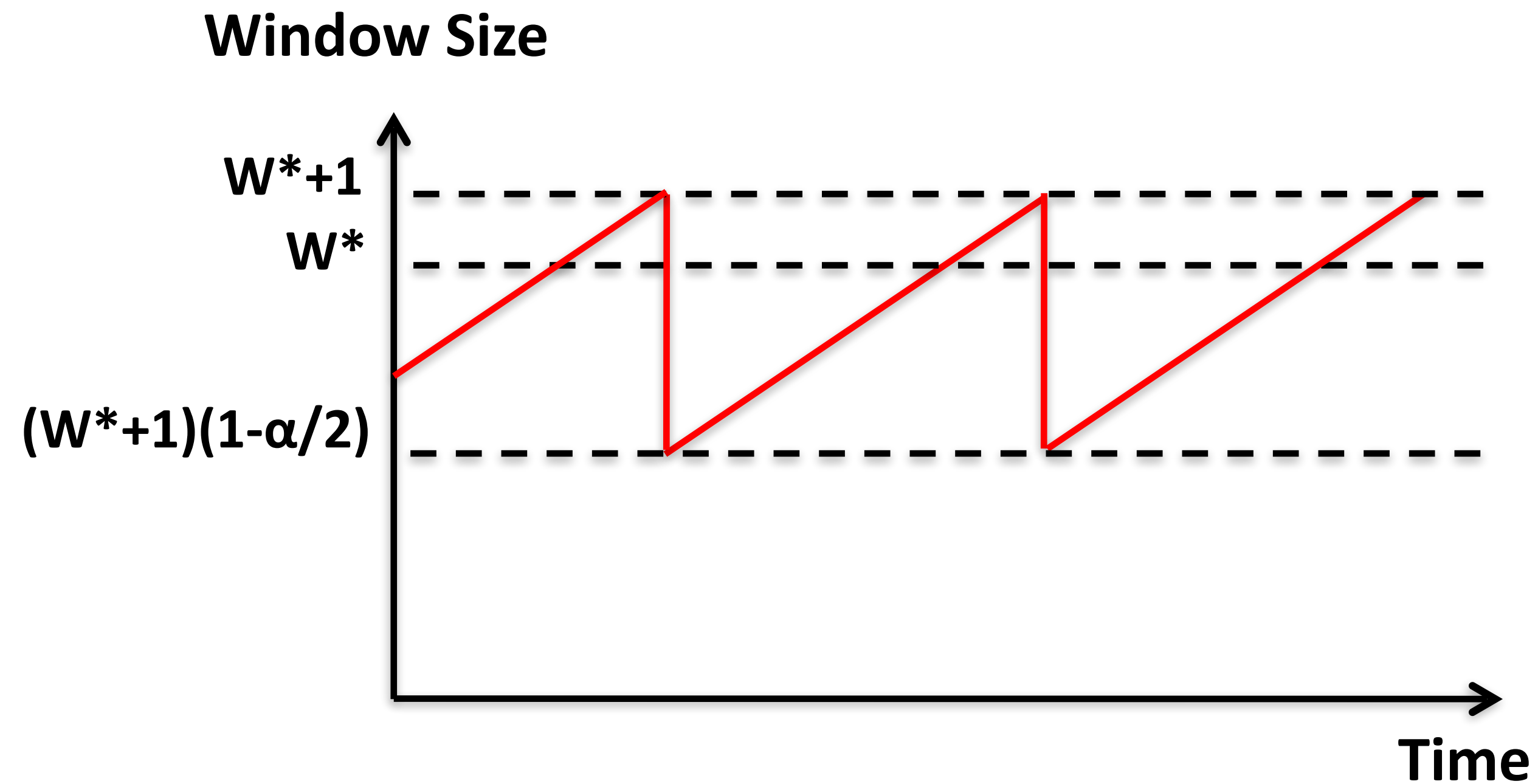
Small buffer occupancy —> Low queueing delay

- #3: High throughput

ECN averaging —> smooth rate adjustment, low variance

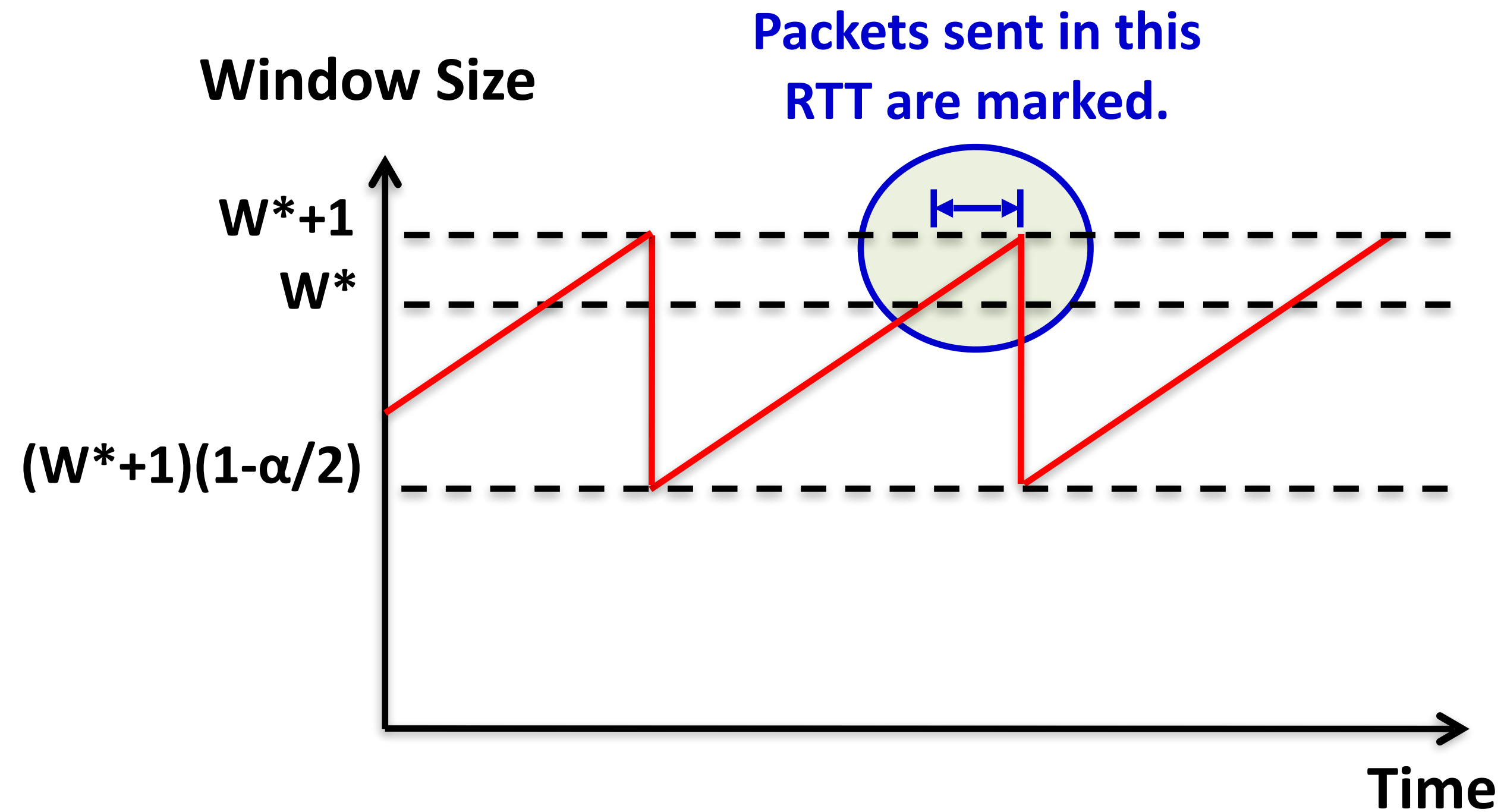
DCTCP Analysis

- How long can DCTCP maintain queues without throughput drop?



DCTCP Analysis

- How long can DCTCP maintain queues without throughput drop?



DCTCP Analysis

- How long can DCTCP maintain queues without throughput drop?
- How do we set the DCTCP parameters?

DCTCP Analysis

- How long can DCTCP maintain queues without throughput drop?
- How do we set the DCTCP parameters?

Need to quantify queue size oscillations (stability)

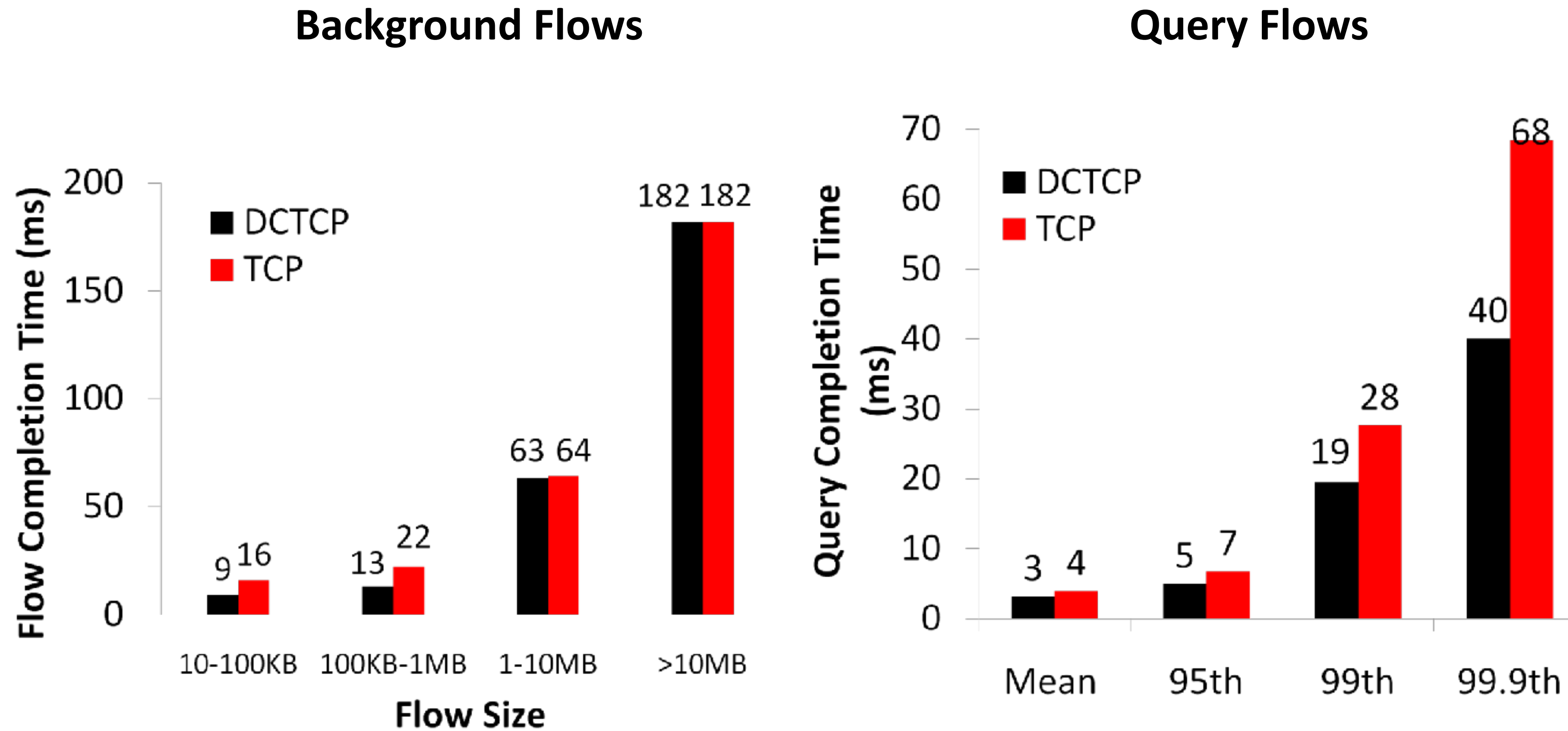
DCTCP Analysis

- How long can DCTCP maintain queues without throughput drop?
- How do we set the DCTCP parameters?

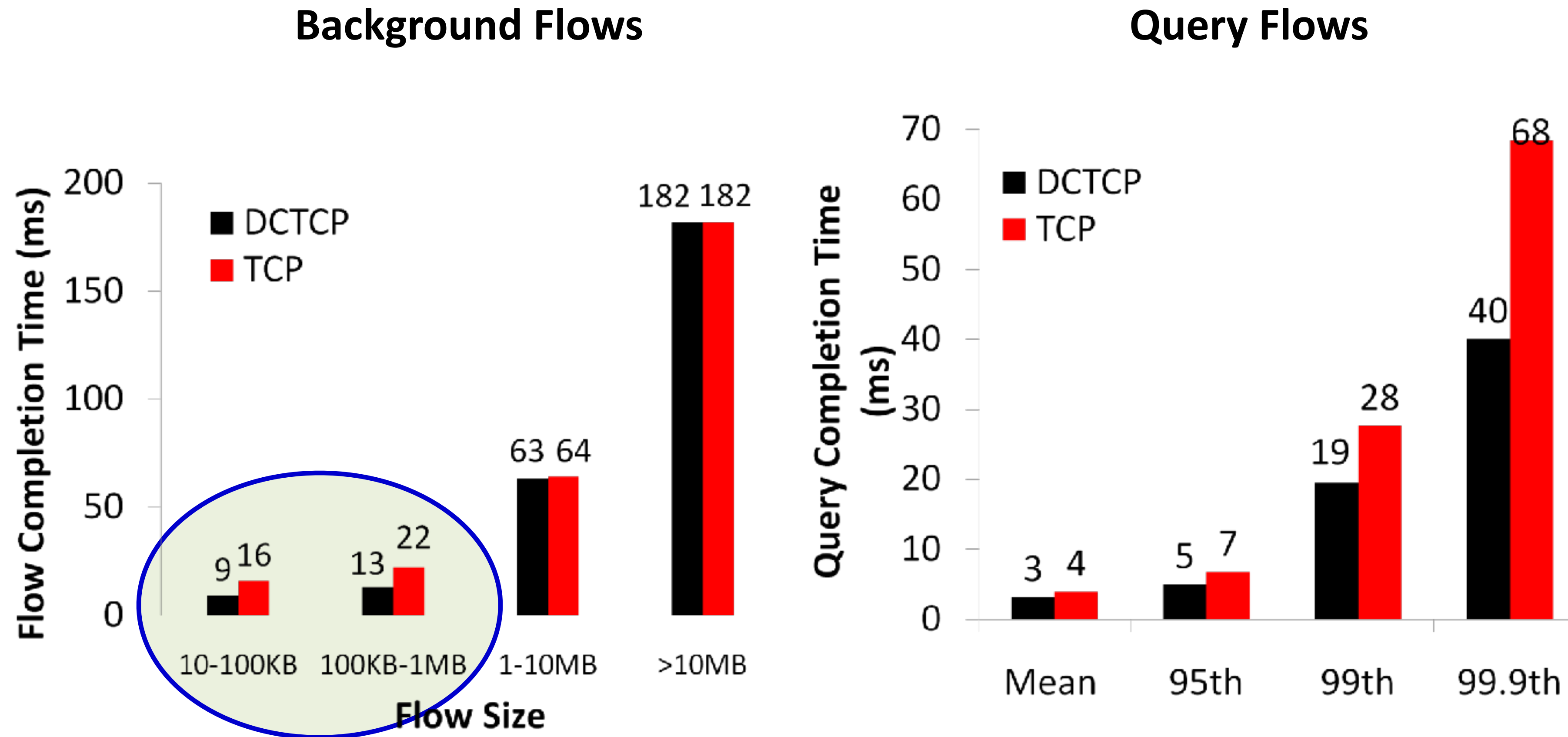
$$K > \frac{1}{7} C \times RTT$$

85% Less Buffer than TCP

DCTCP Performance

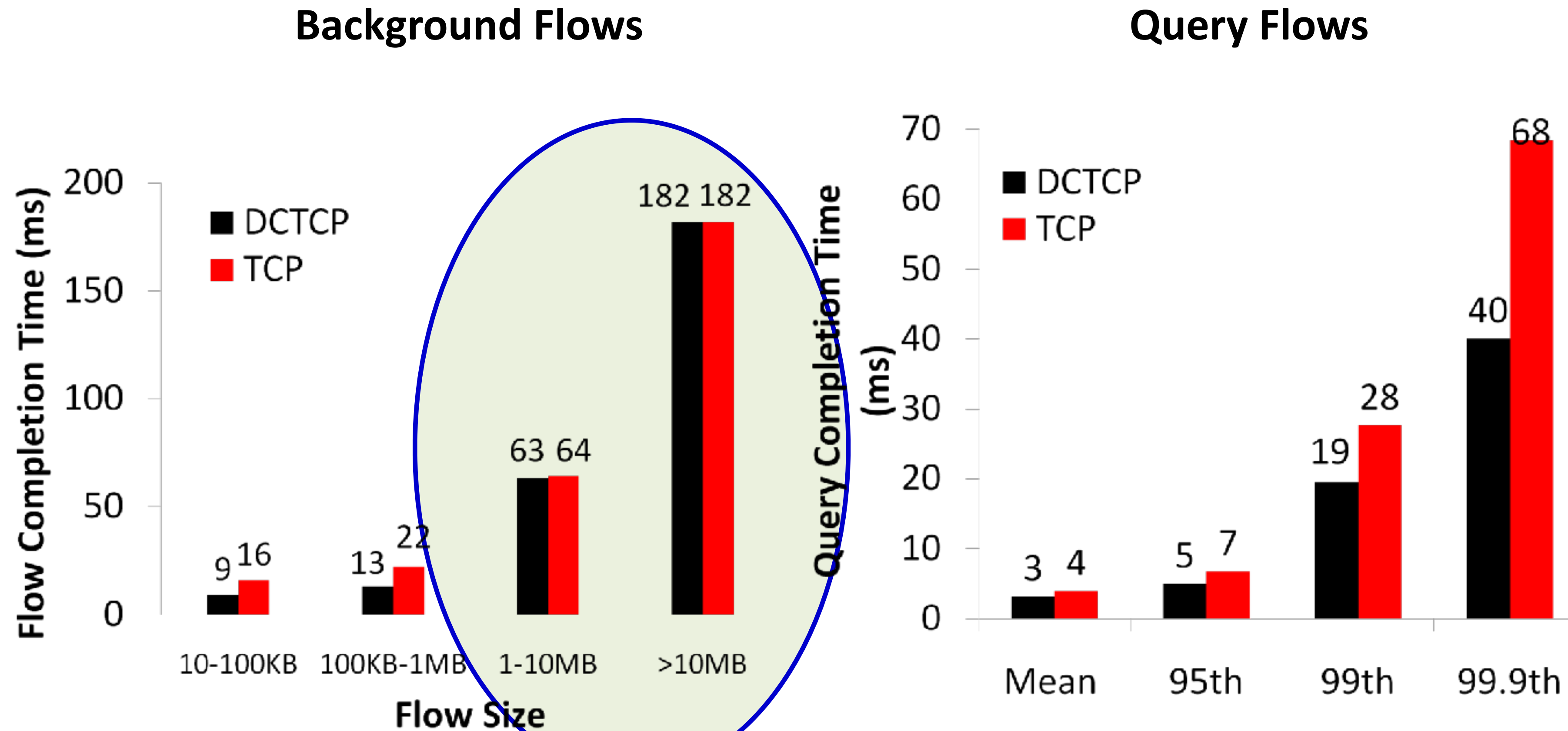


DCTCP Performance



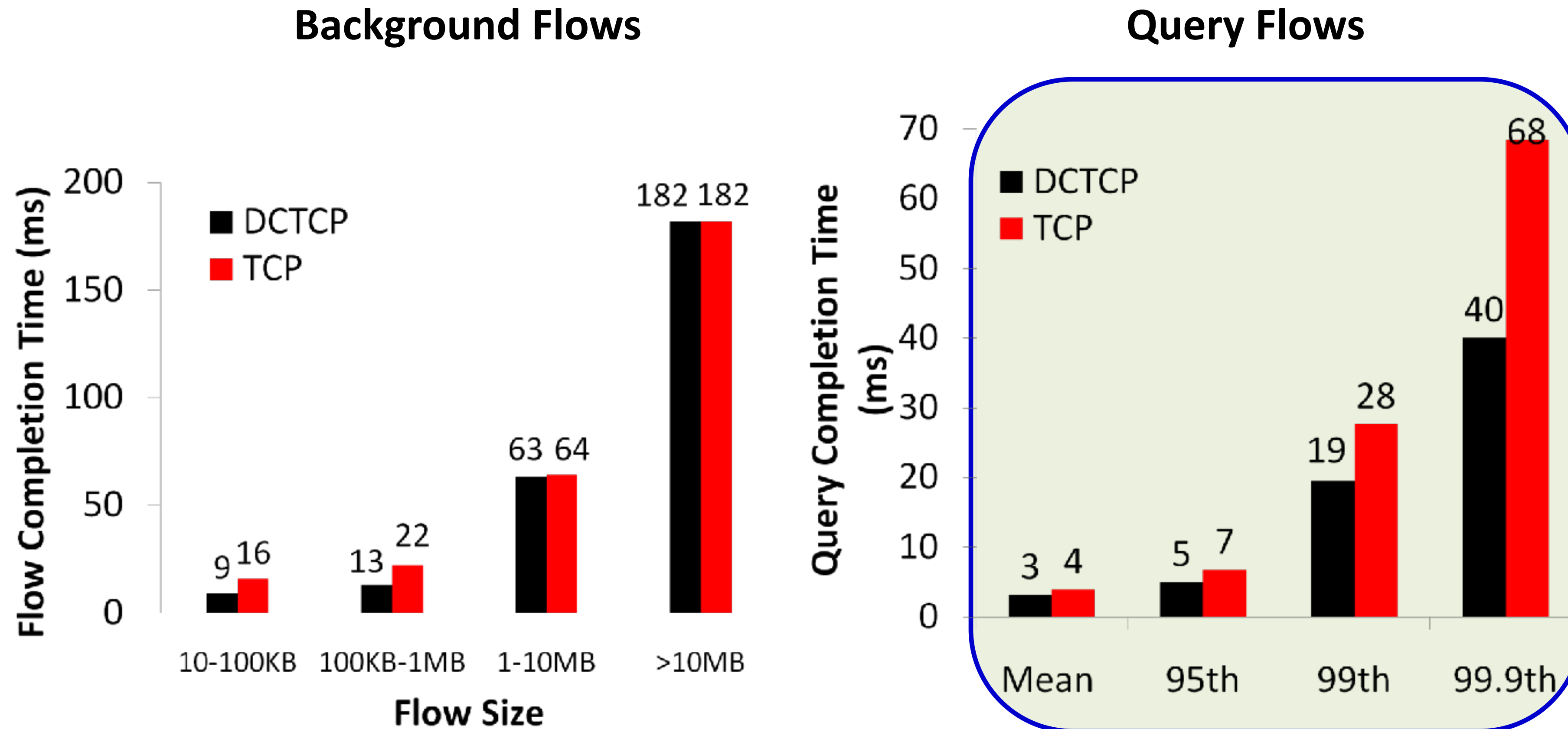
✓ Low latency for short flows

DCTCP Performance



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

DCTCP Performance



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

Summary

- Today
 - DCTCP
- Next
 - NDP (Sigcomm'17)