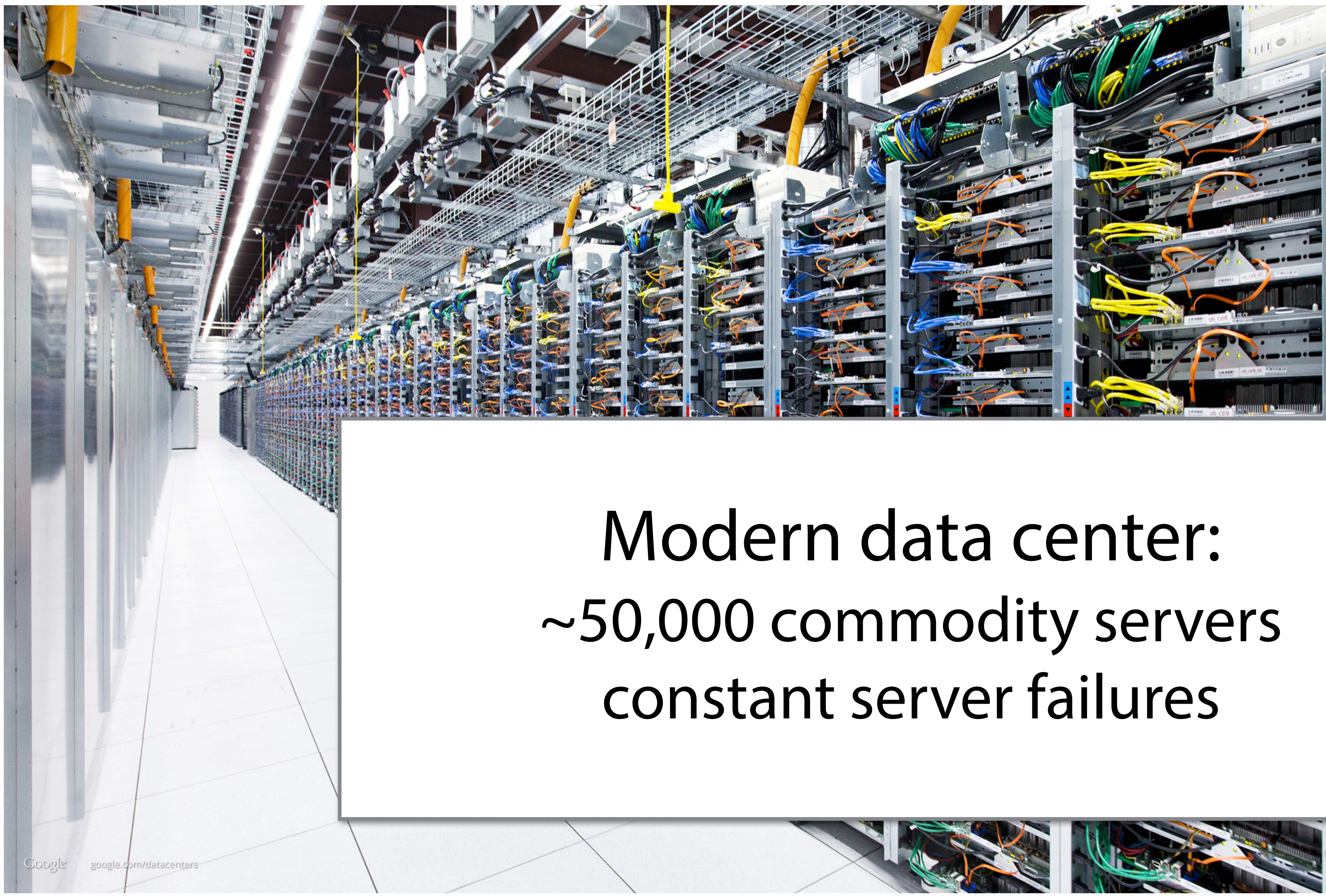
Designing Distributed Systems using Approximate Synchrony in Data Center Networks

Dan R. K. Ports
Jialin Li   Vincent Liu
Naveen Kr. Sharma   Arvind Krishnamurthy
University of Washington CSE

Today's most popular applications are ***distributed systems*** in the ***data center***

Today's most popular applications are ***distributed systems*** in the ***data center***



Modern data center:

~50,000 commodity servers

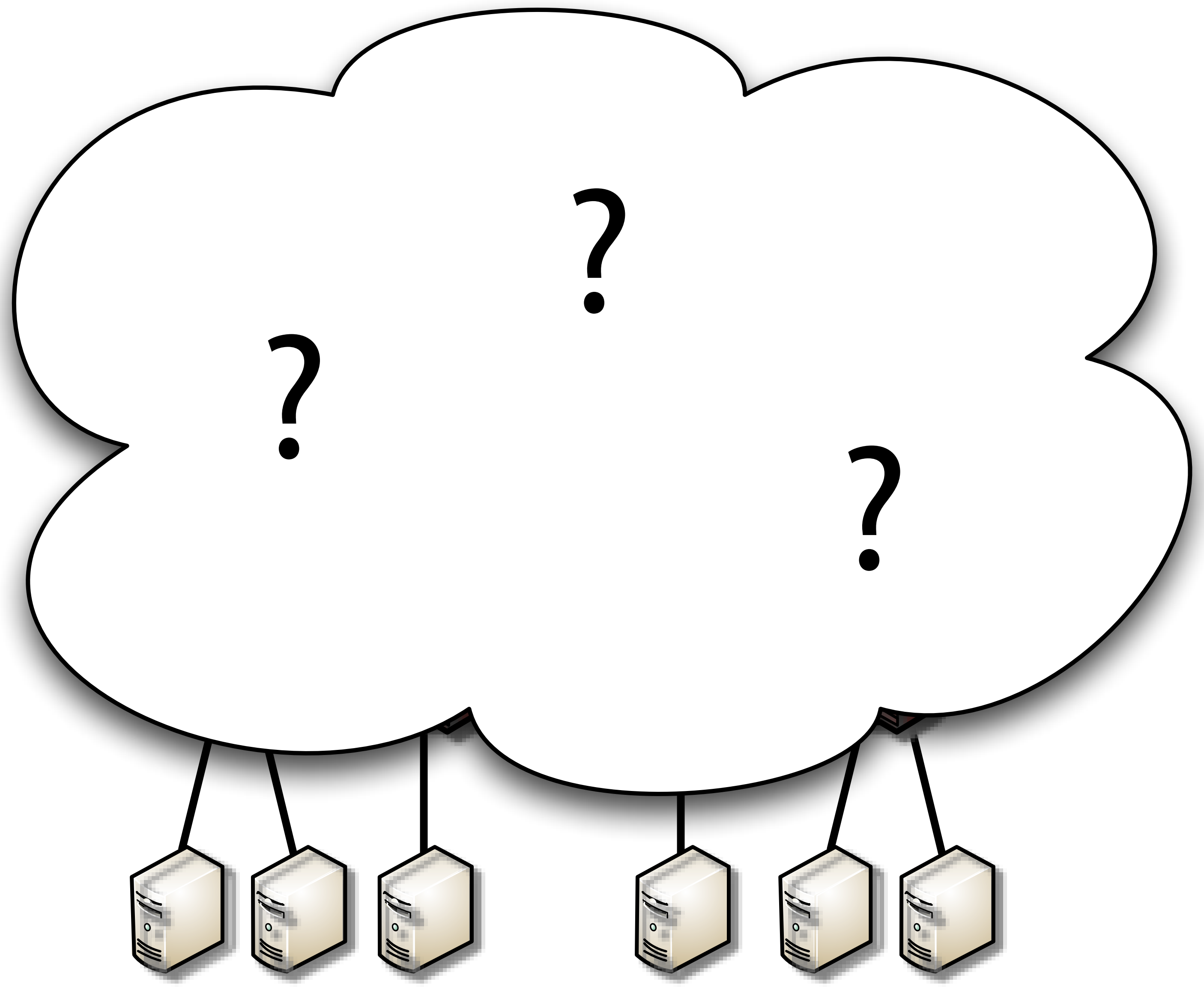
constant server failures

How do we program the data center?

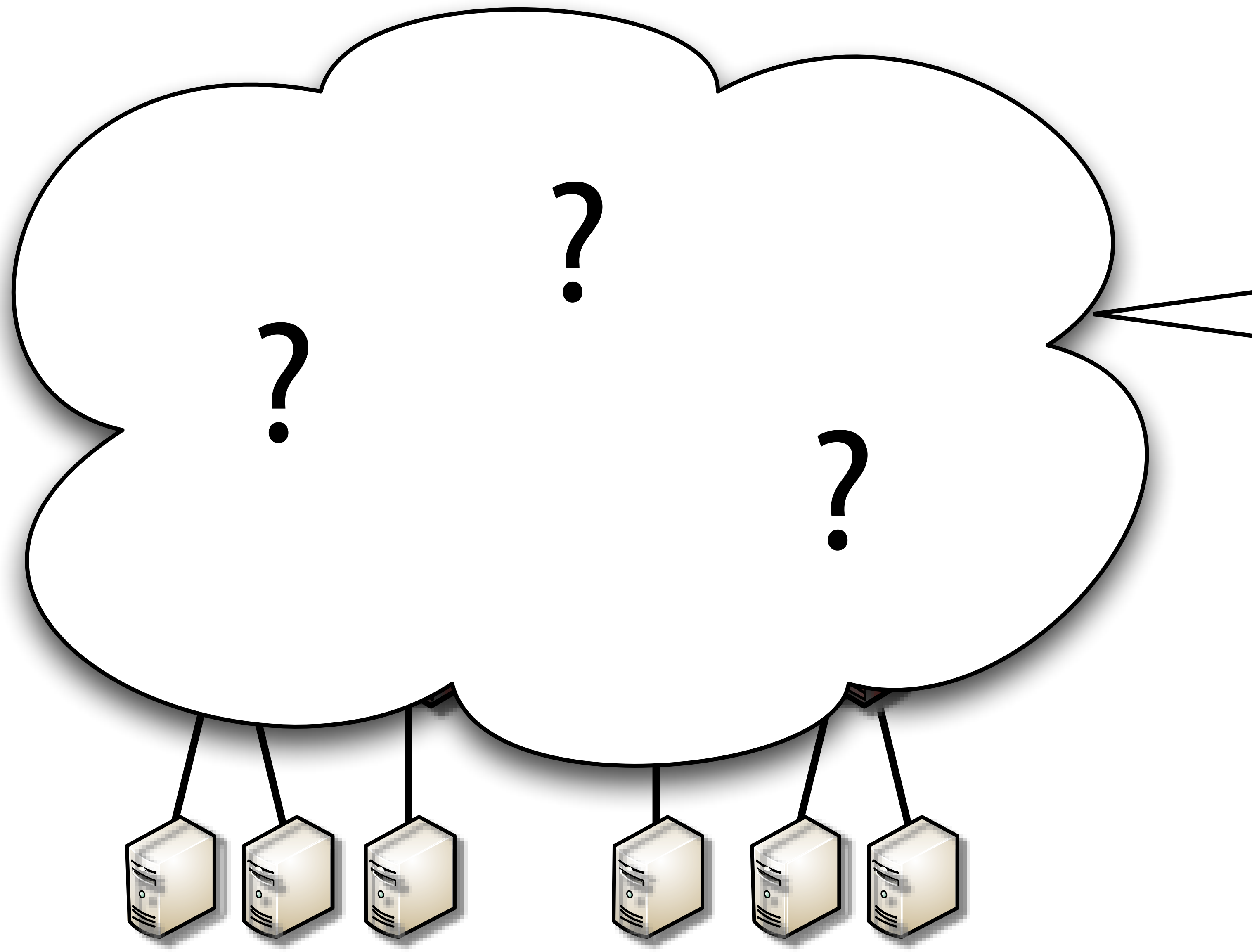
Use distributed algorithms to tolerate failures, inconsistencies

Example: Paxos state machine replication

Distributed systems and networks are typically designed independently



Distributed systems and networks are typically designed independently

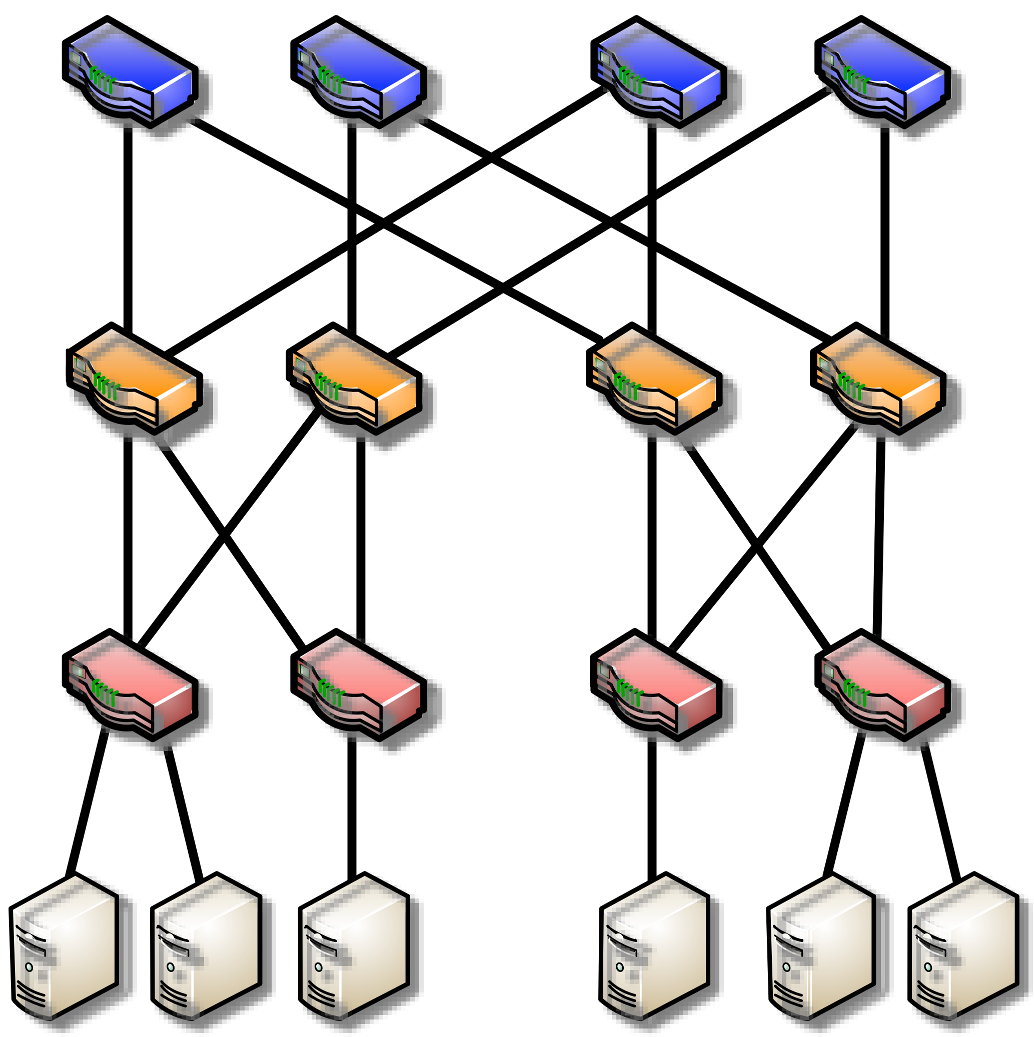


asynchronous network (Internet)

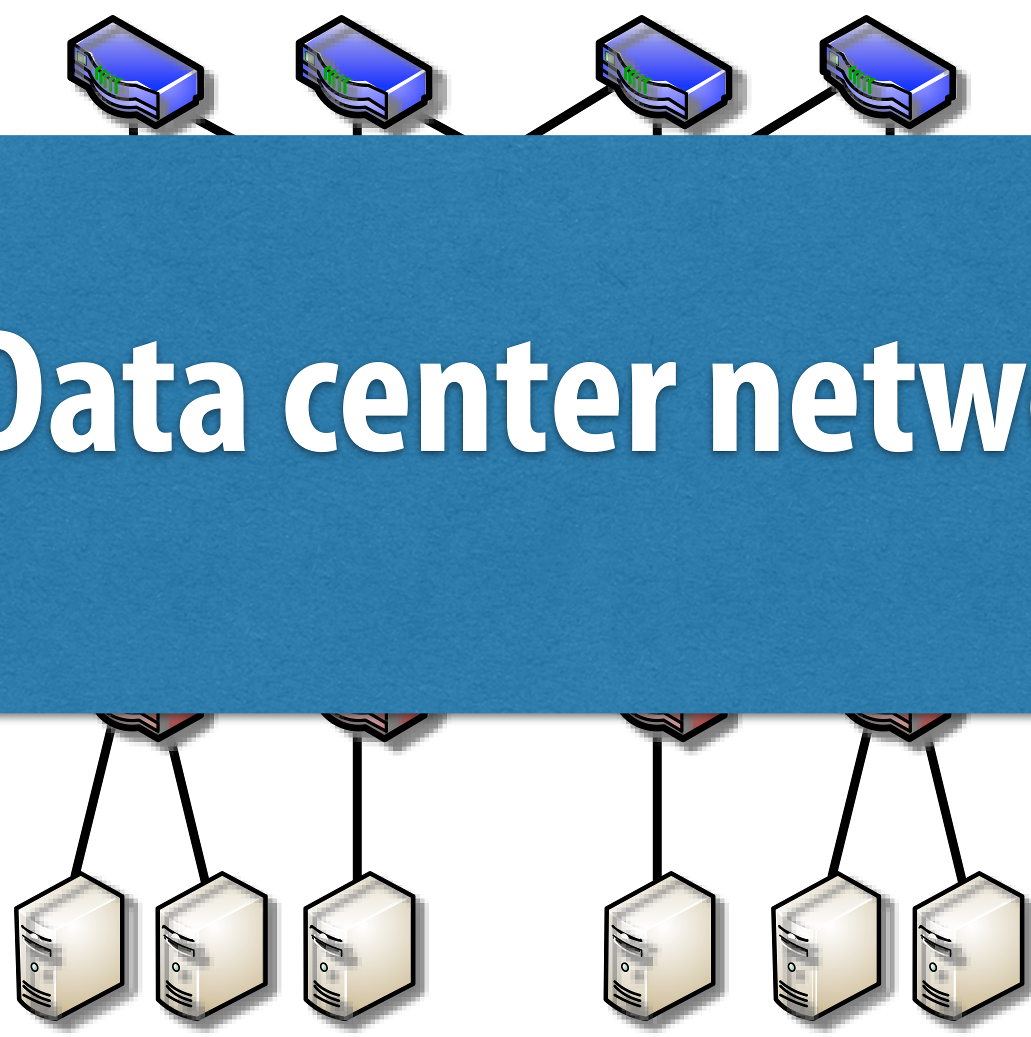
packets may be arbitrarily

- dropped
- delayed
- reordered

Distributed systems and networks are typically designed independently



Distributed systems and networks are typically designed independently



Data center networks are different!

Data Center Networks Are Different

Data center networks are more ***predictable***

- known topology, routes, predictable latencies

Data center networks are more ***reliable***

Data center networks are ***extensible***

- single administrative domain makes changes possible
- software-defined networking exposes sophisticated line-rate processing capability

Data Center Networks Are Different

Data center networks are more ***predictable***

- known topology, routes, predictable latencies

We should co-design distributed systems and data center networks!

Data center networks are ***extensible***

- single administrative domain makes changes possible
- software-defined networking exposes sophisticated line-rate processing capability

Co-Designing Networks and Distributed Systems

Design the ***data center network*** to support ***distributed applications***

Design ***distributed applications*** around the properties of the ***data center network***

This Talk

A concrete instantiation:

improving replication performance using
Speculative Paxos and ***Mostly-Ordered Multicast***

This Talk

A concrete instantiation:

improving replication performance using
Speculative Paxos and ***Mostly-Ordered Multicast***

new replication protocol

new network primitive

This Talk

A concrete instantiation:

improving replication performance using
Speculative Paxos and ***Mostly-Ordered Multicast***

new replication protocol

new network primitive

3x throughput and 40% lower latency than conventional approach

Outline

1. Co-designing Distributed Systems and Data Center Networks

2. **Background: State Machine Replication & Paxos**

3. Mostly-Ordered Multicast and Speculative Paxos

4. Evaluation

State Machine Replication

Used to tolerate failures in datacenter applications

- keep critical management services online (e.g., Google's Chubby, Zookeeper)
- persistent storage in distributed databases (e.g., Spanner, H-Store)

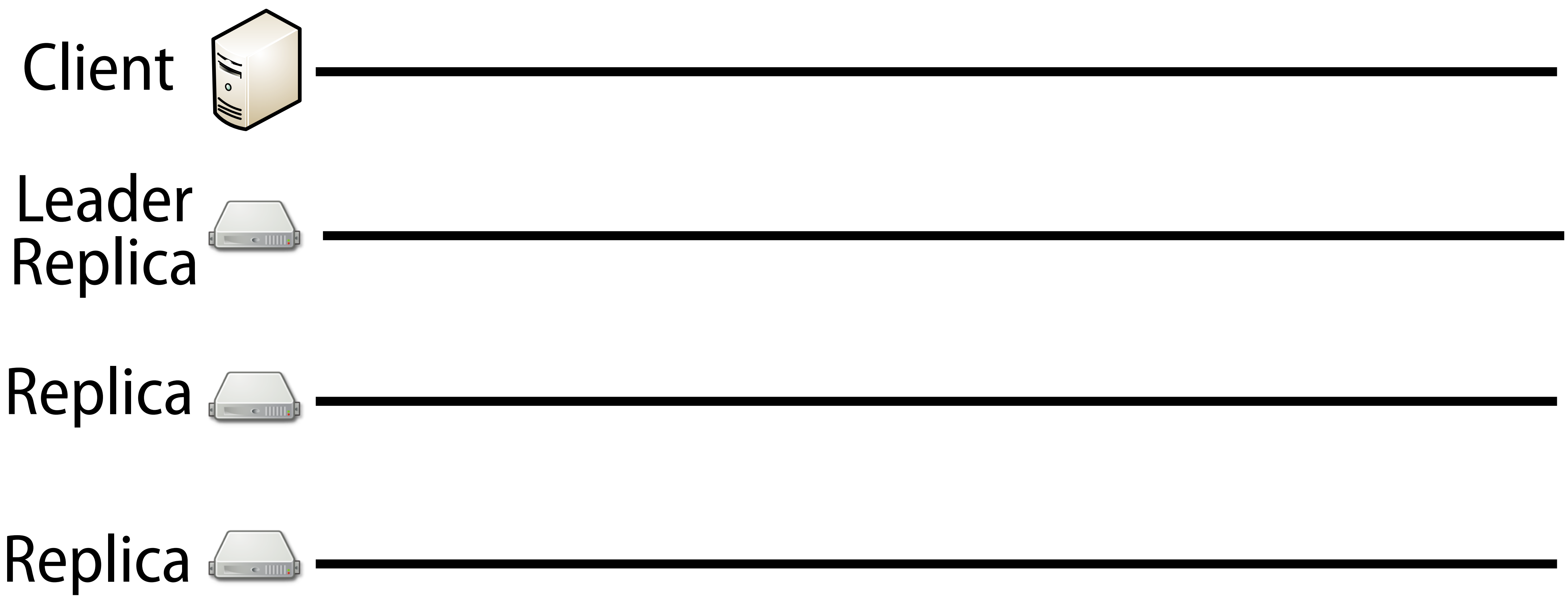
Strongly consistent (linearizable) replication, i.e.,

all replicas execute same operations in same order

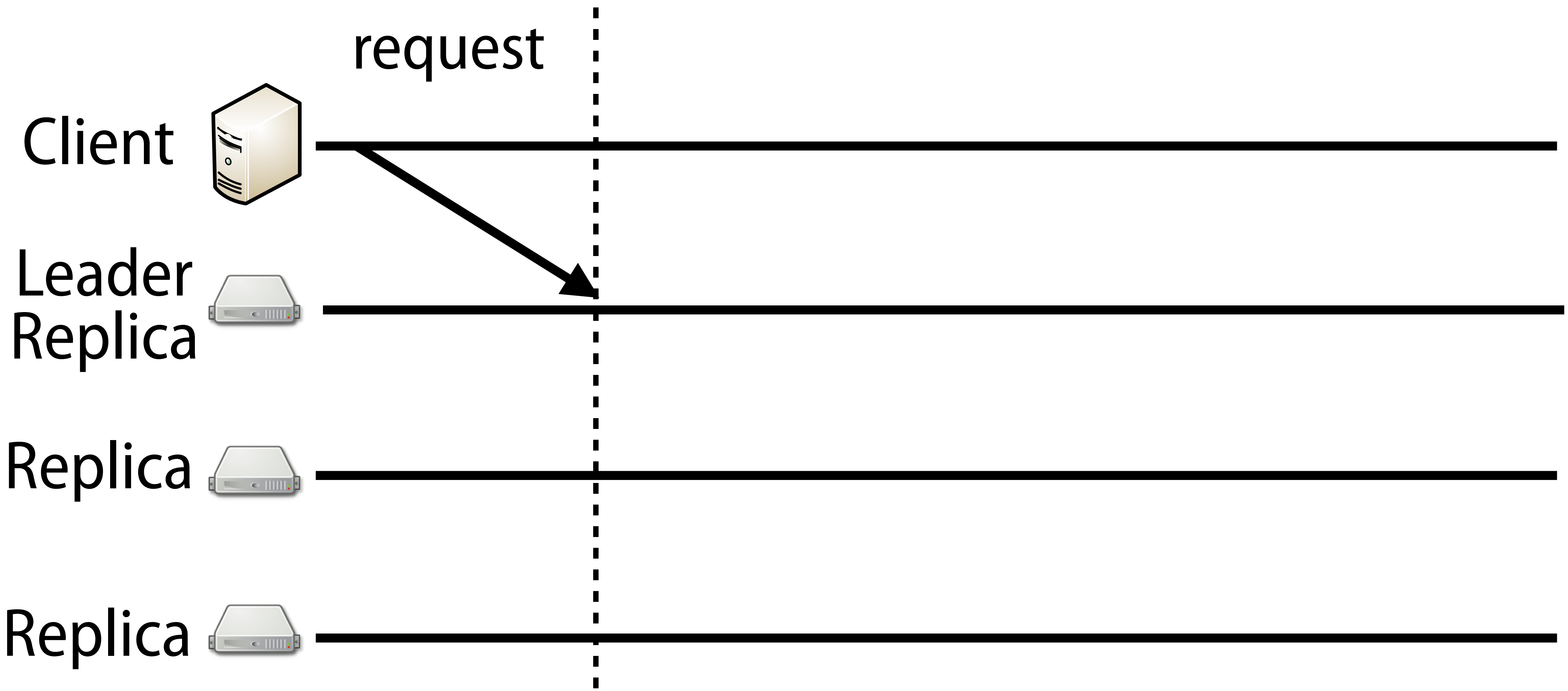
. . . *even when up to half replicas fail*

. . . *even when messages are lost*

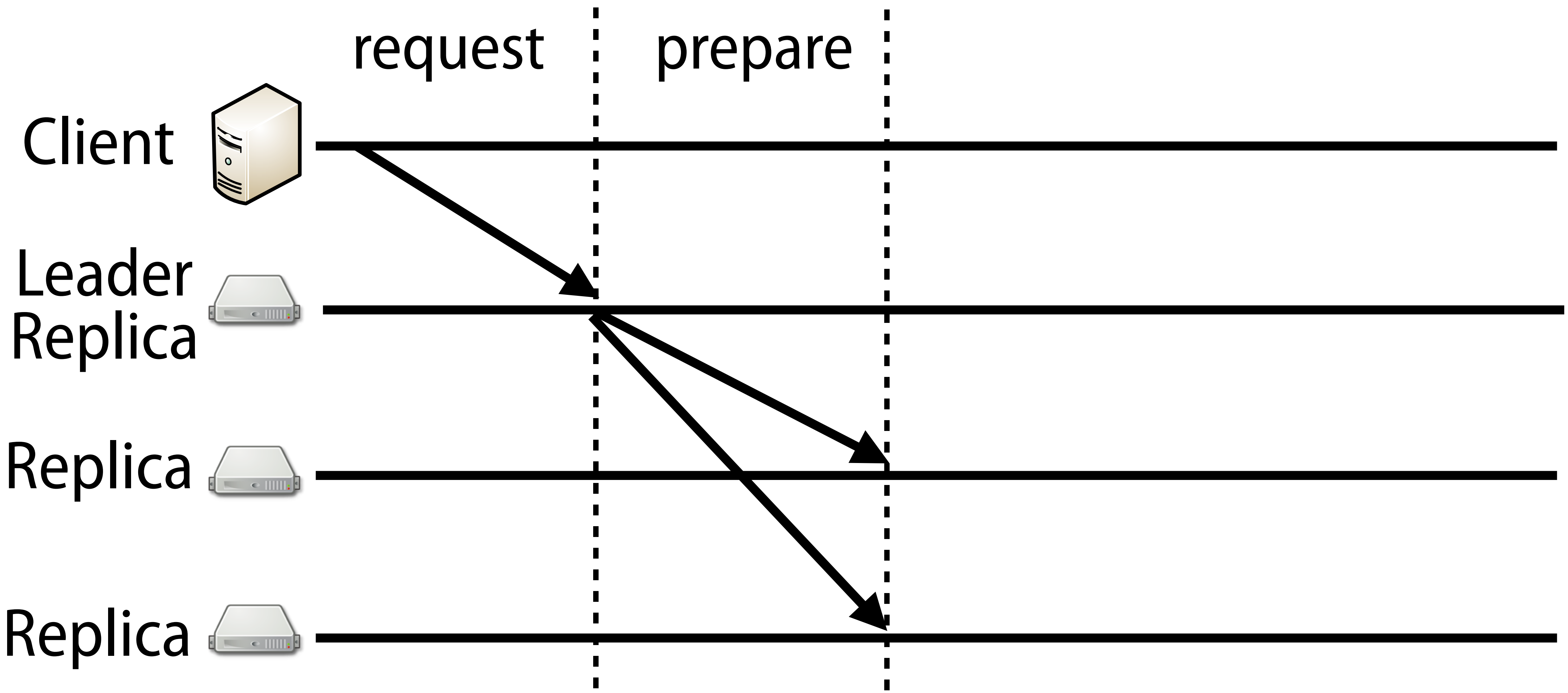
Example: Paxos



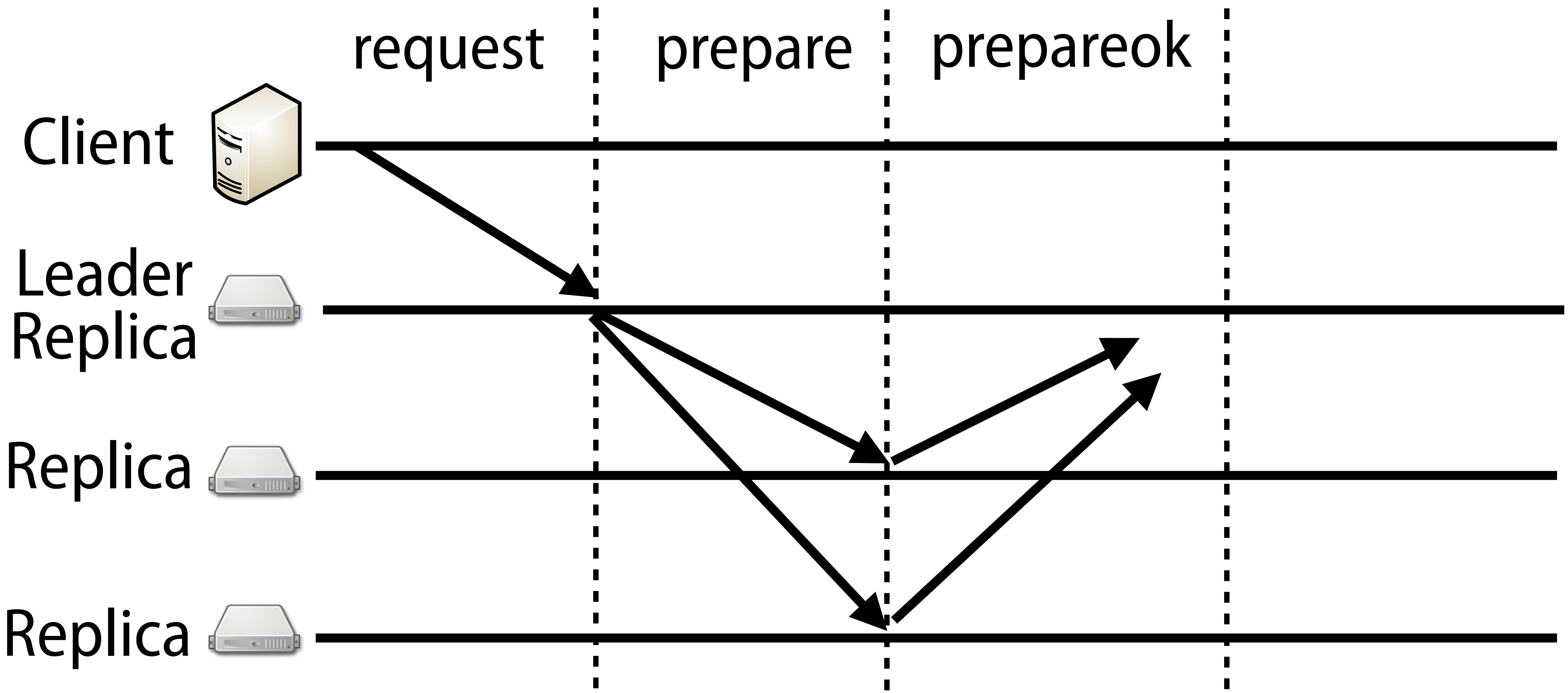
Example: Paxos



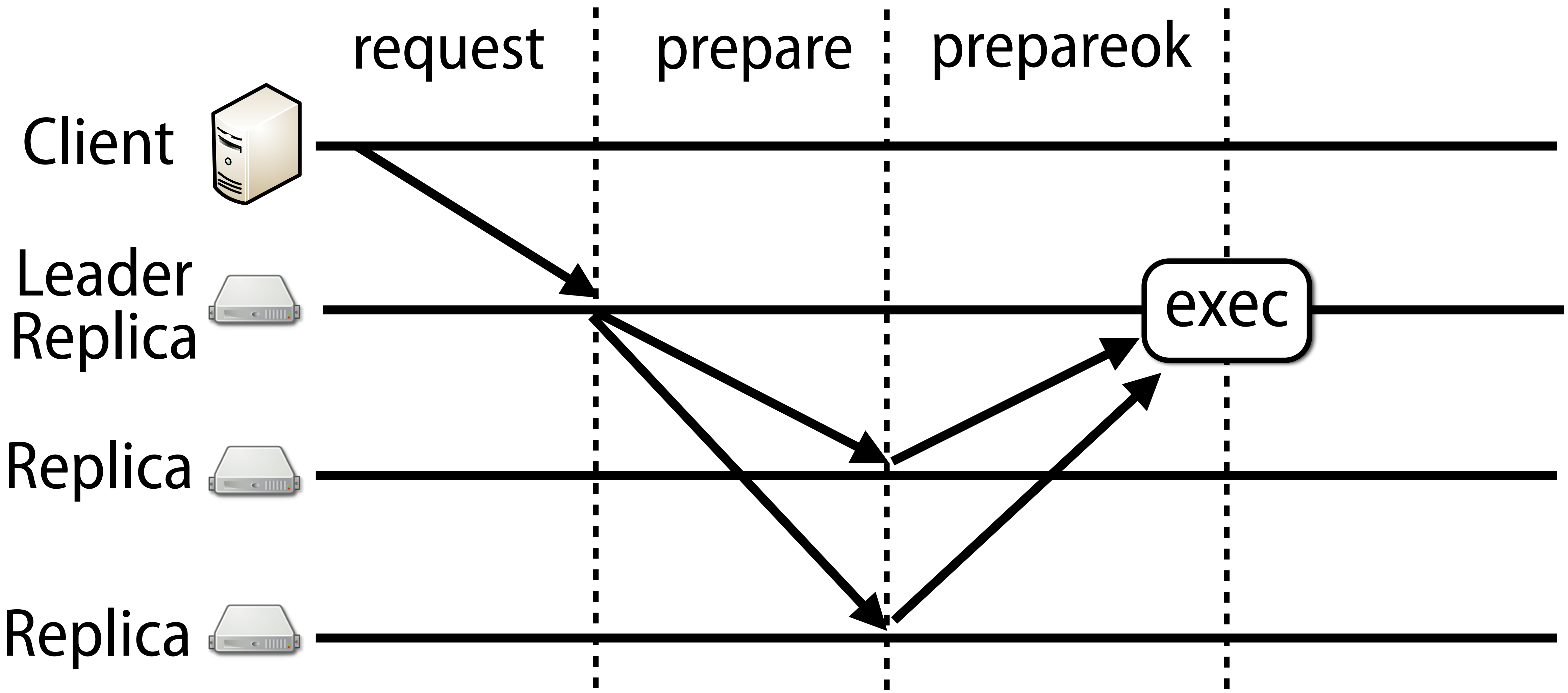
Example: Paxos



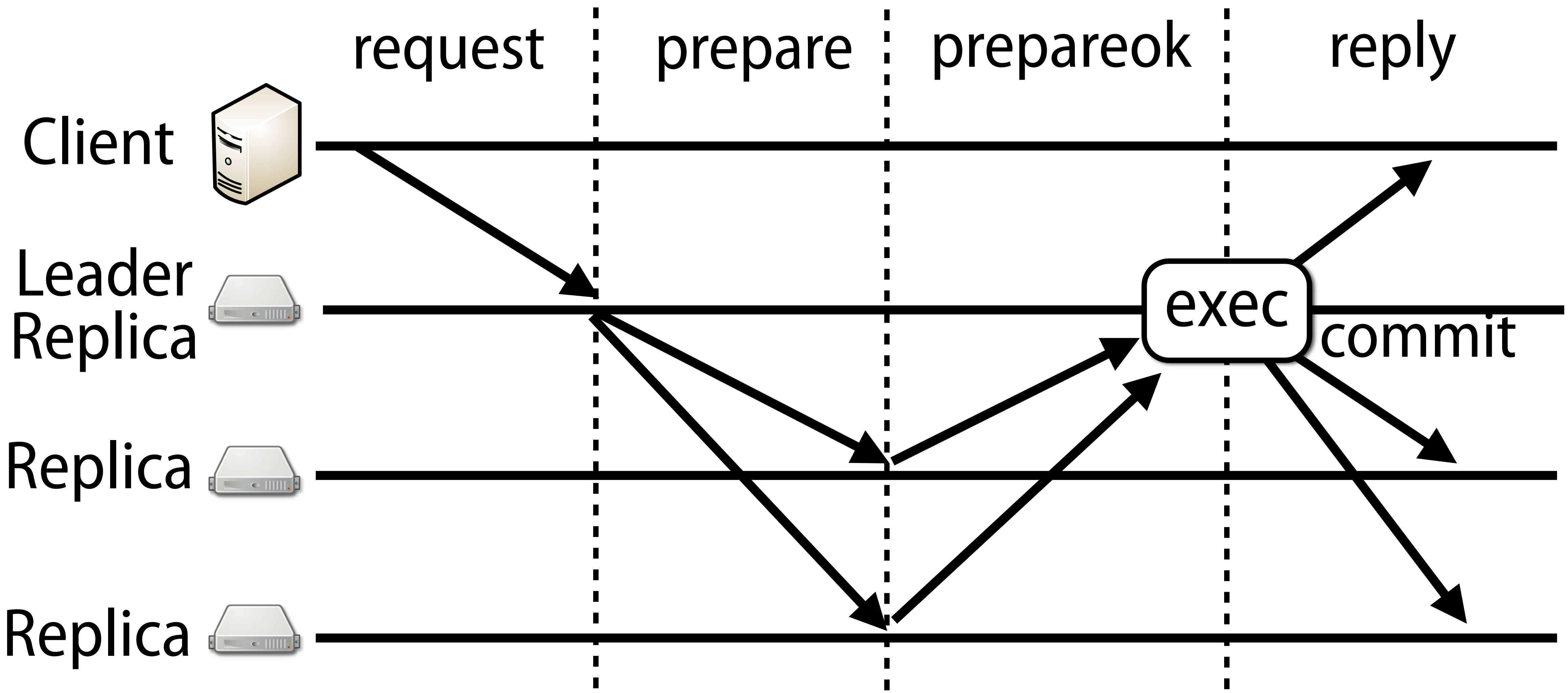
Example: Paxos



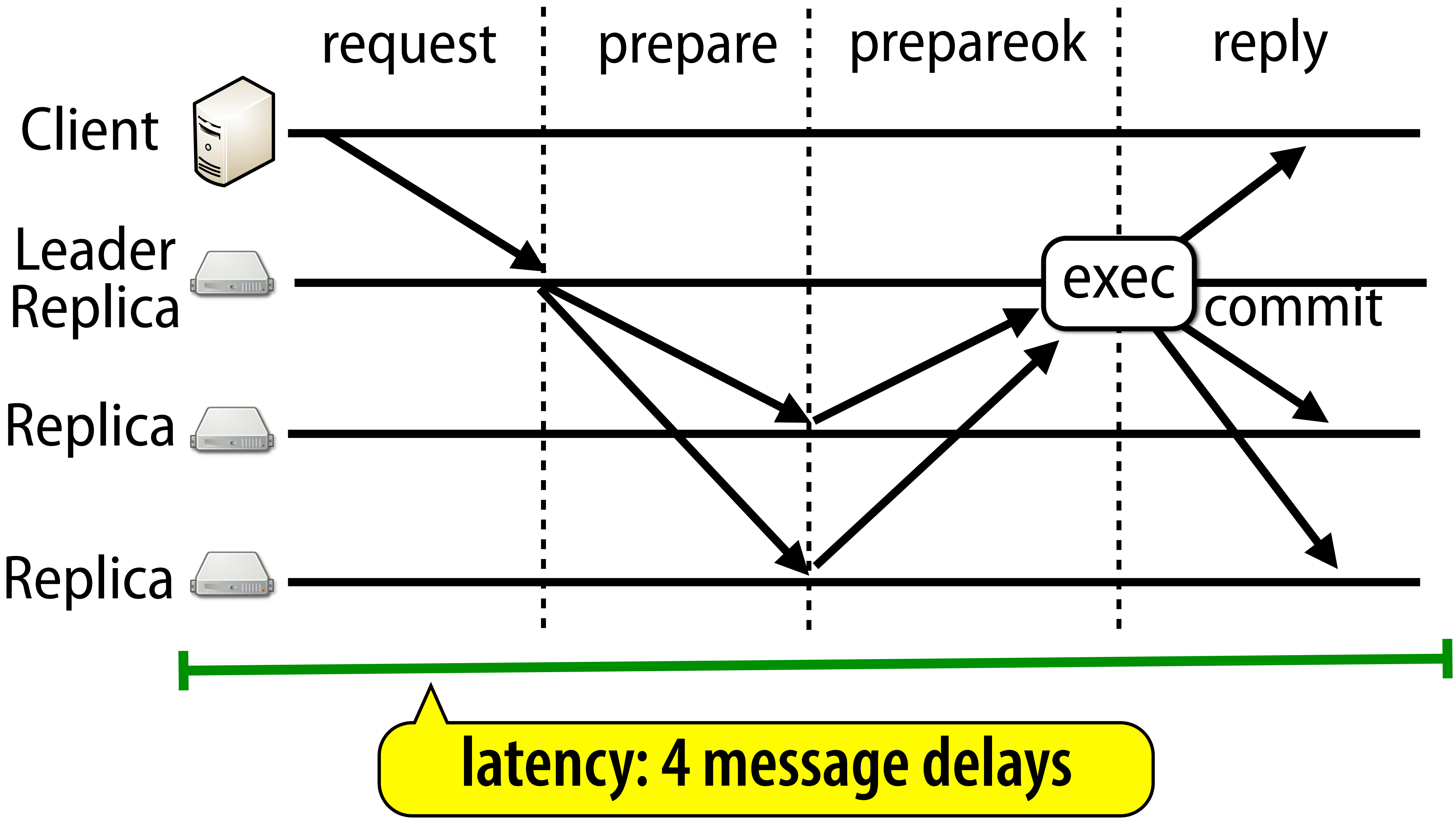
Example: Paxos



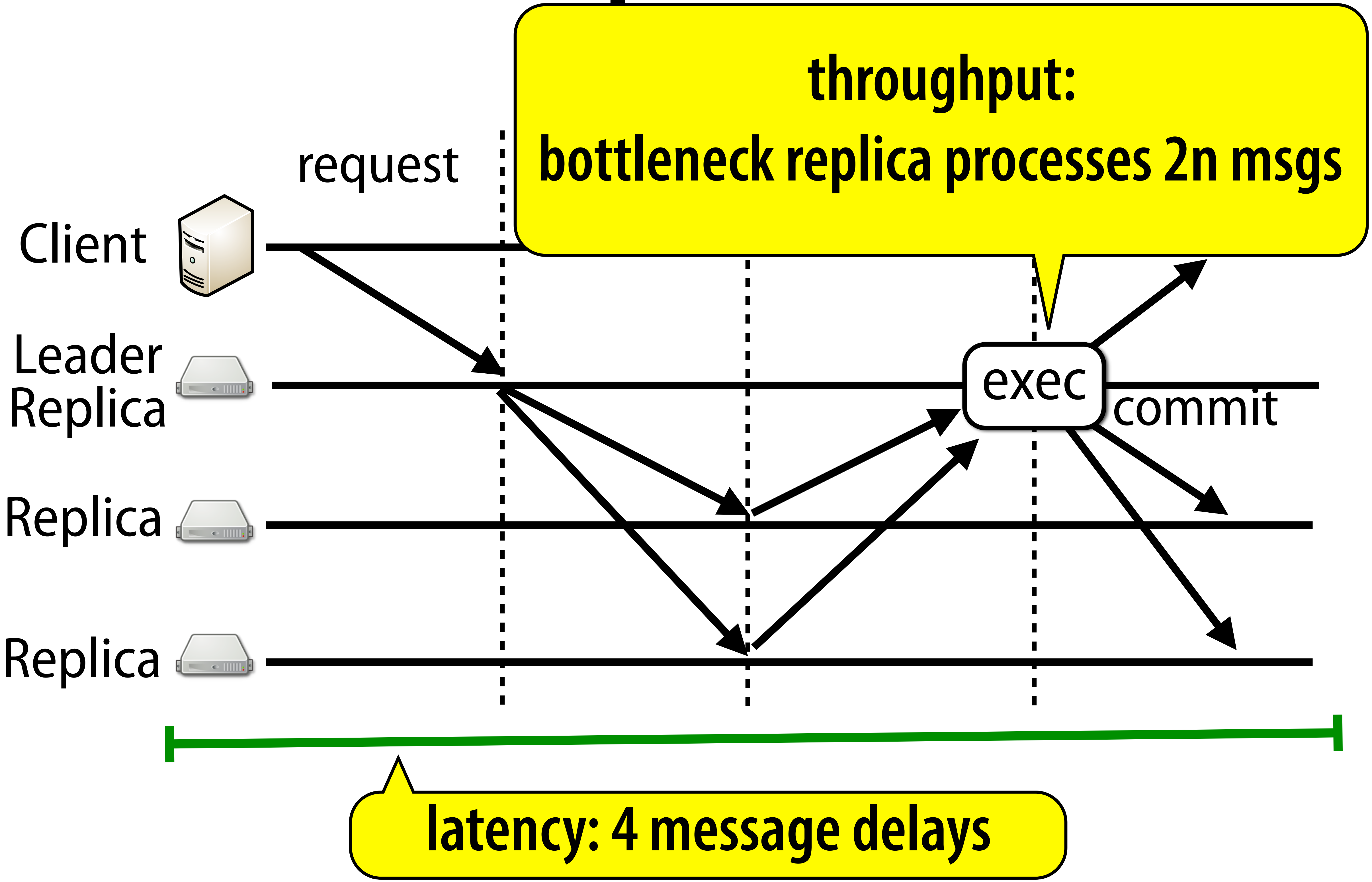
Example: Paxos



Example: Paxos



Example: Paxos



Outline

1. Co-designing Distributed Systems and Data Center Networks
2. Background: State Machine Replication & Paxos
3. **Mostly-Ordered Multicast and Speculative Paxos**
4. Evaluation

Improving Paxos Performance

Paxos requires a leader replica to order requests

Can we use the network instead?

Improving Paxos Performance

Paxos requires a leader replica to order requests

Can we use the network instead?

Engineer the network to provide
Mostly-Ordered Multicast (MOM)

- best-effort ordering of multicasts

New replication protocol: **Speculative Paxos**

- commits most operations in a single round trip

Mostly-Ordered Multicast

Concurrent messages are ordered:

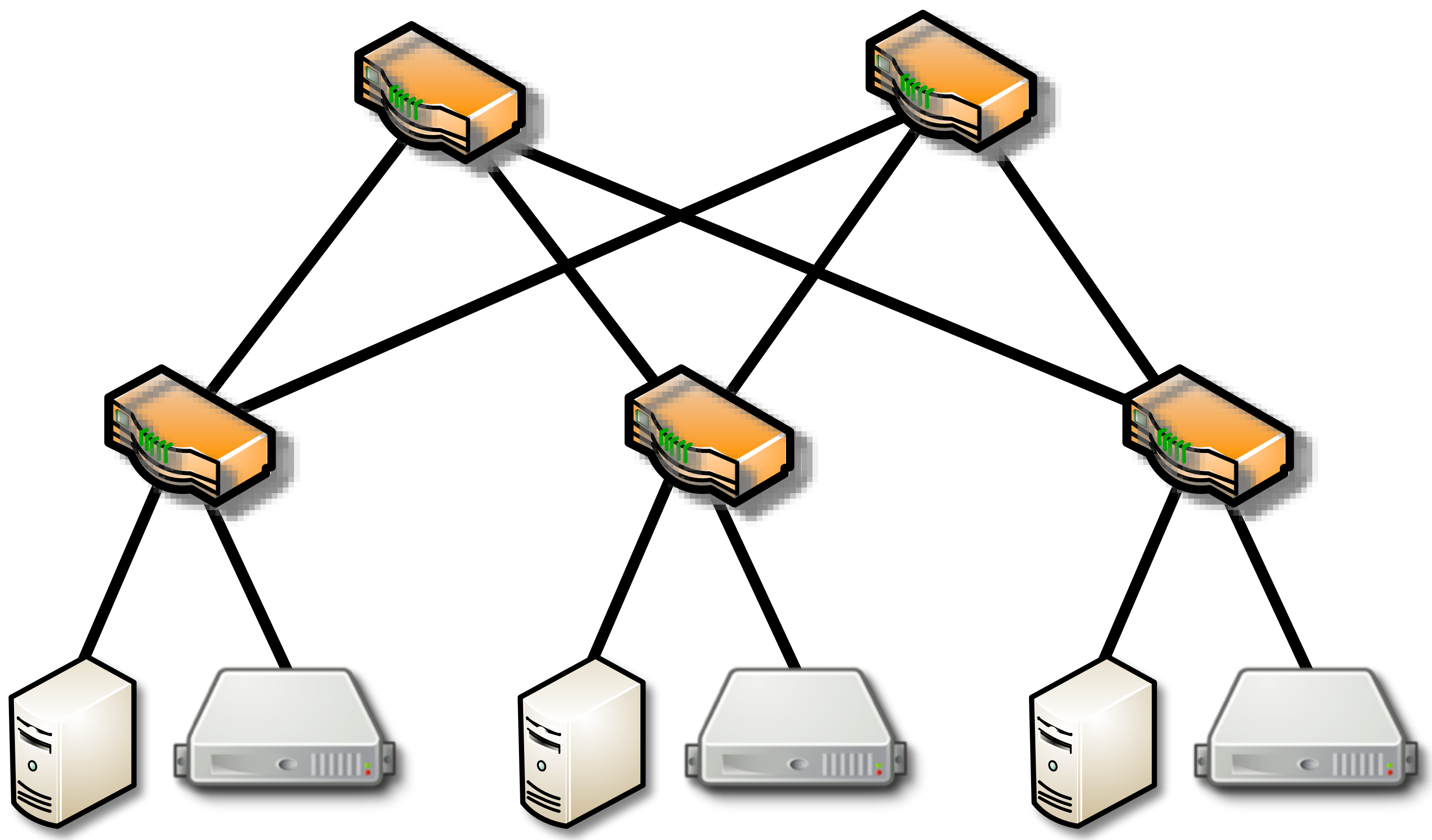
If any node receives message A then B, then all other receivers process them ***in the same order***

- best effort — not guaranteed

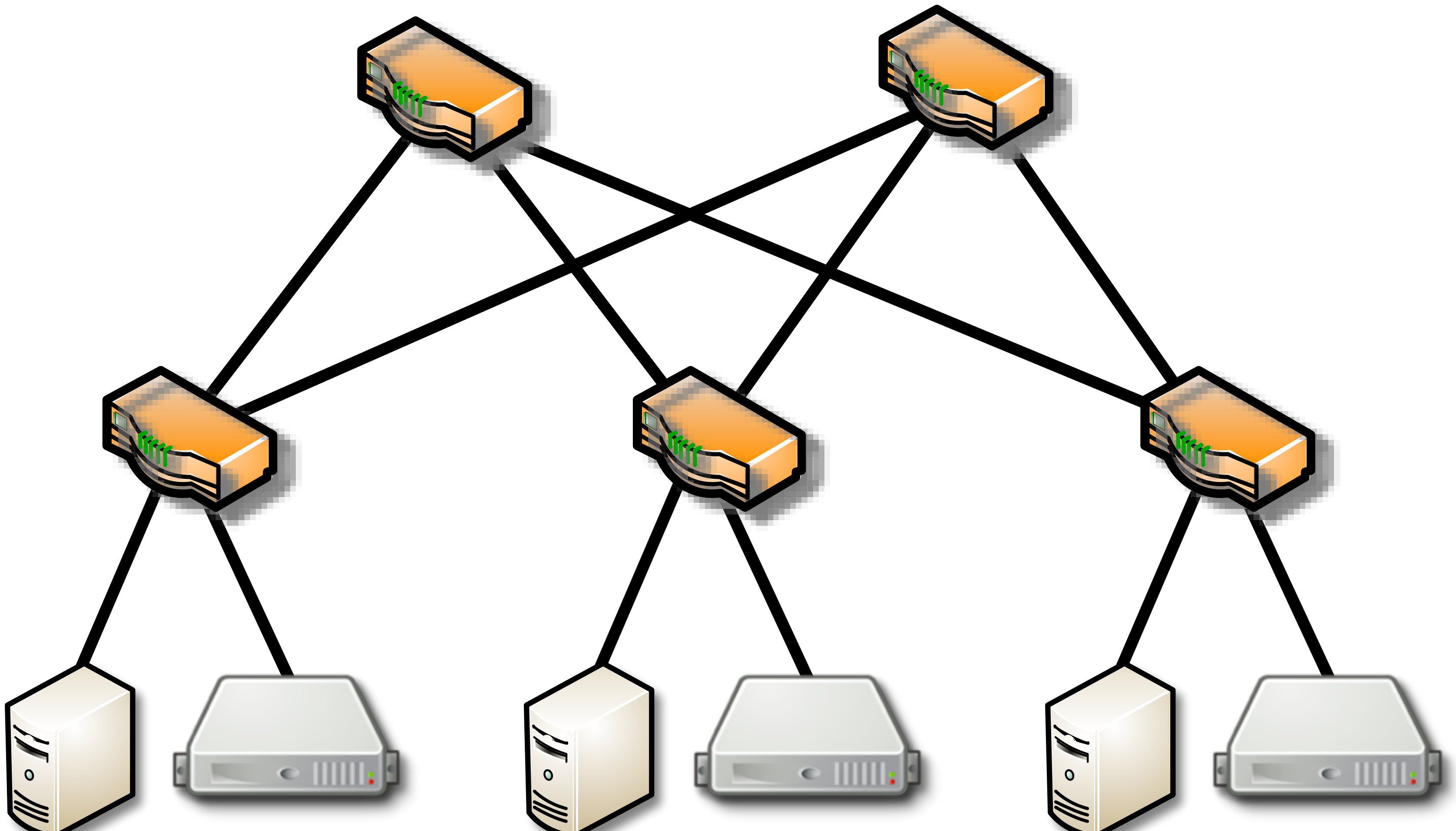
Practical to implement

- can be violated in event of network failure
- but not satisfied by existing multicast protocols!

Mostly-Ordered Multicast

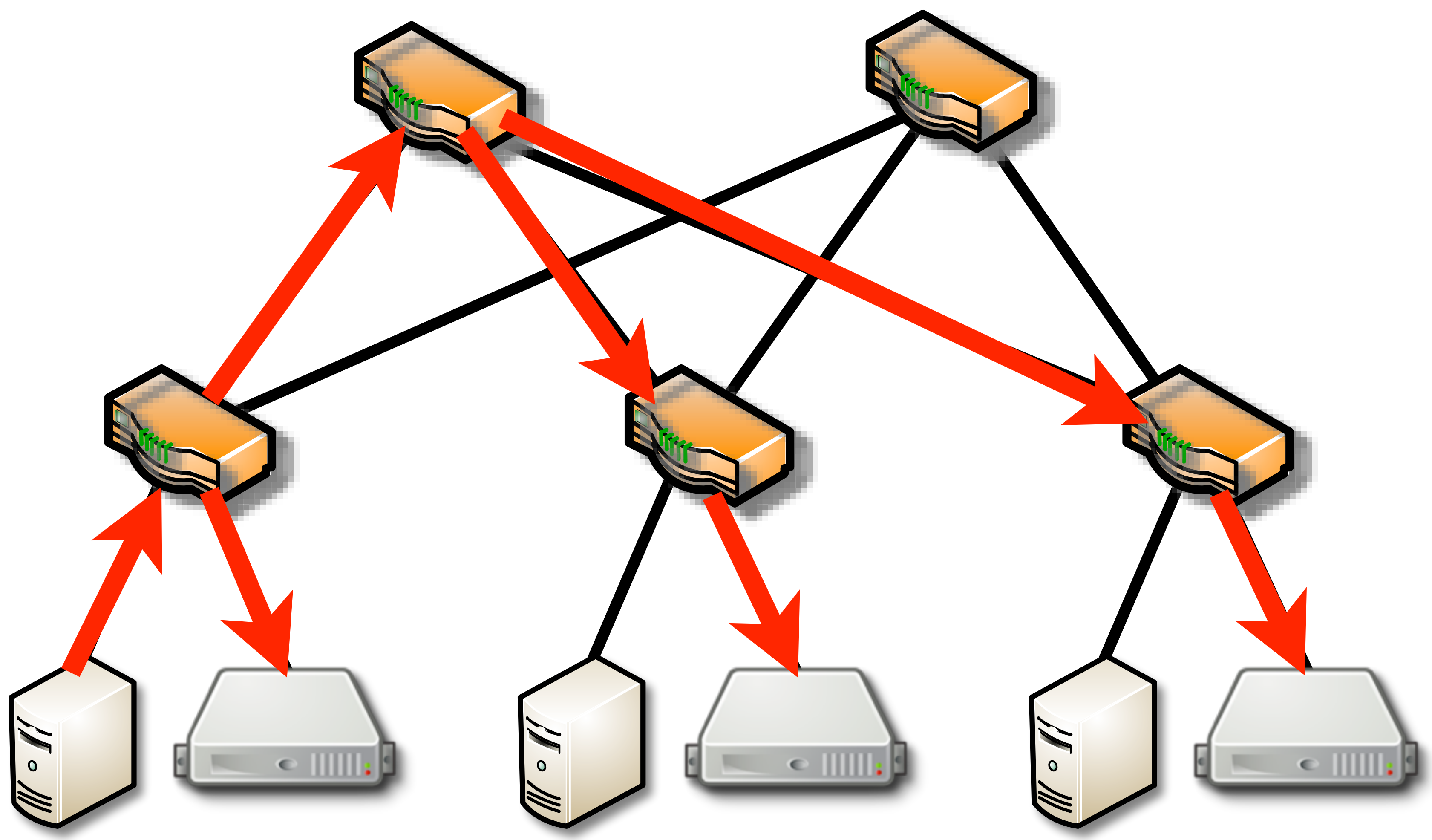


Mostly-Ordered Multicast



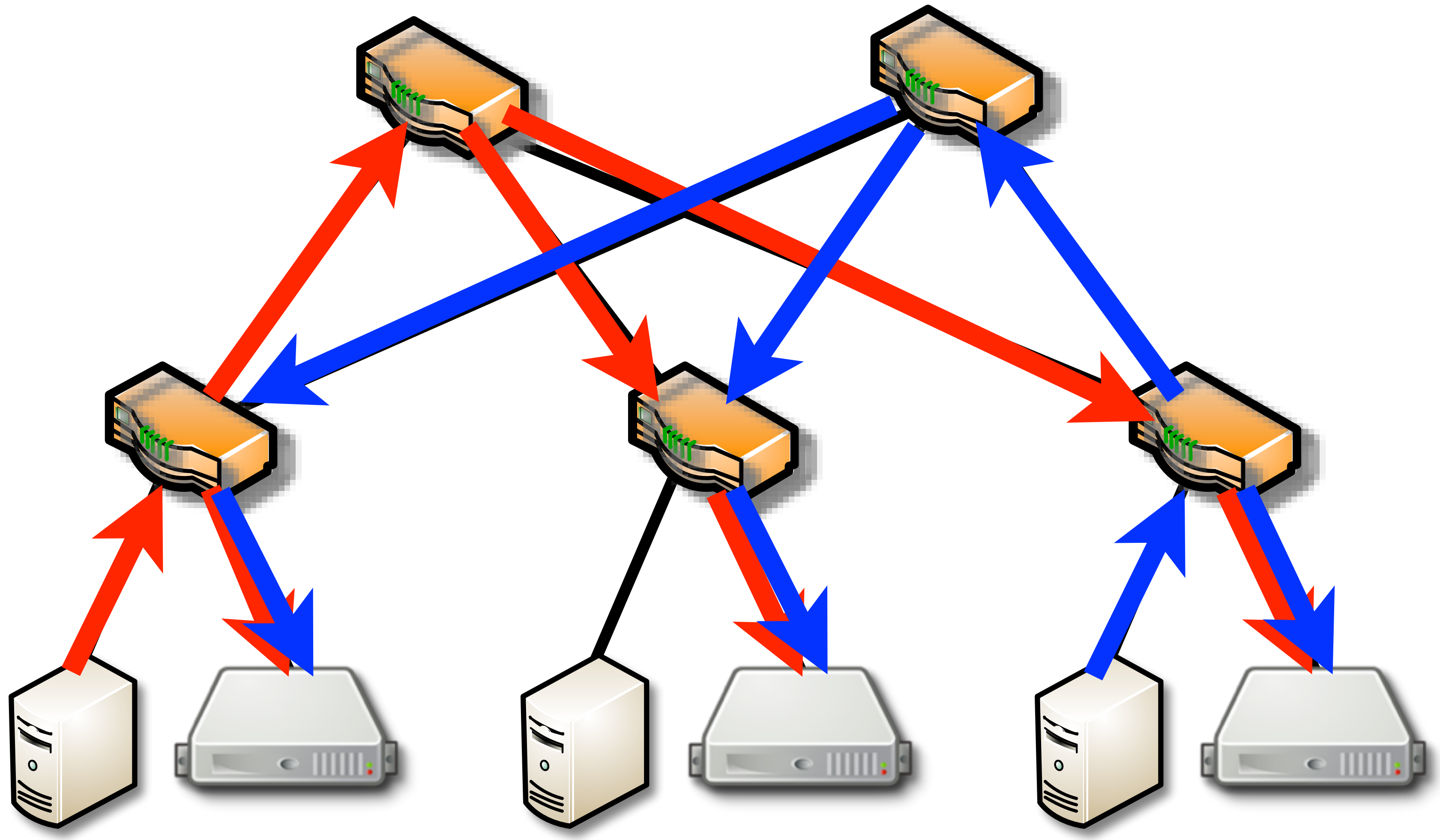
- Different path lengths, congestion cause reordering

Mostly-Ordered Multicast



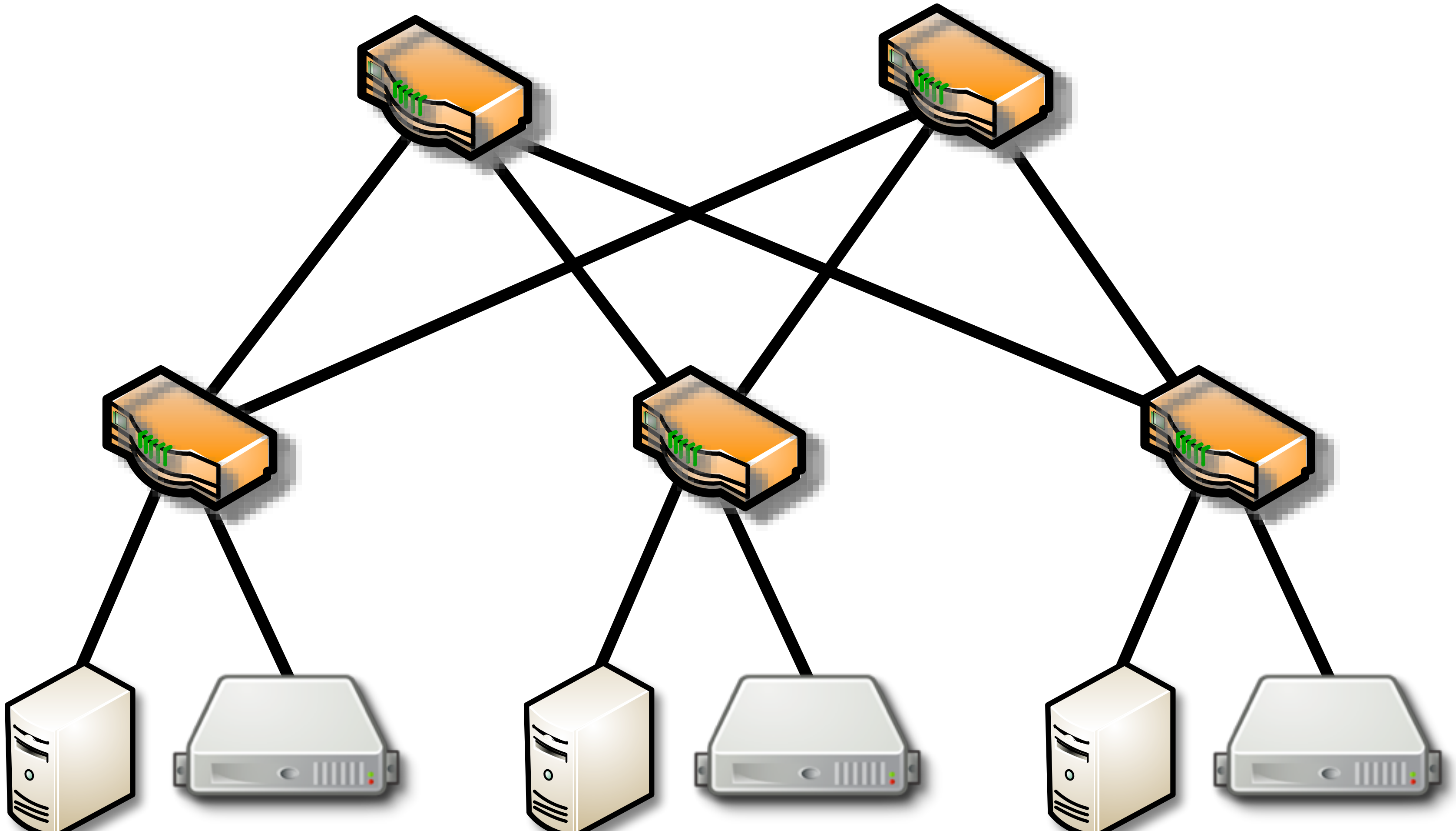
- Different path lengths, congestion cause reordering

Mostly-Ordered Multicast



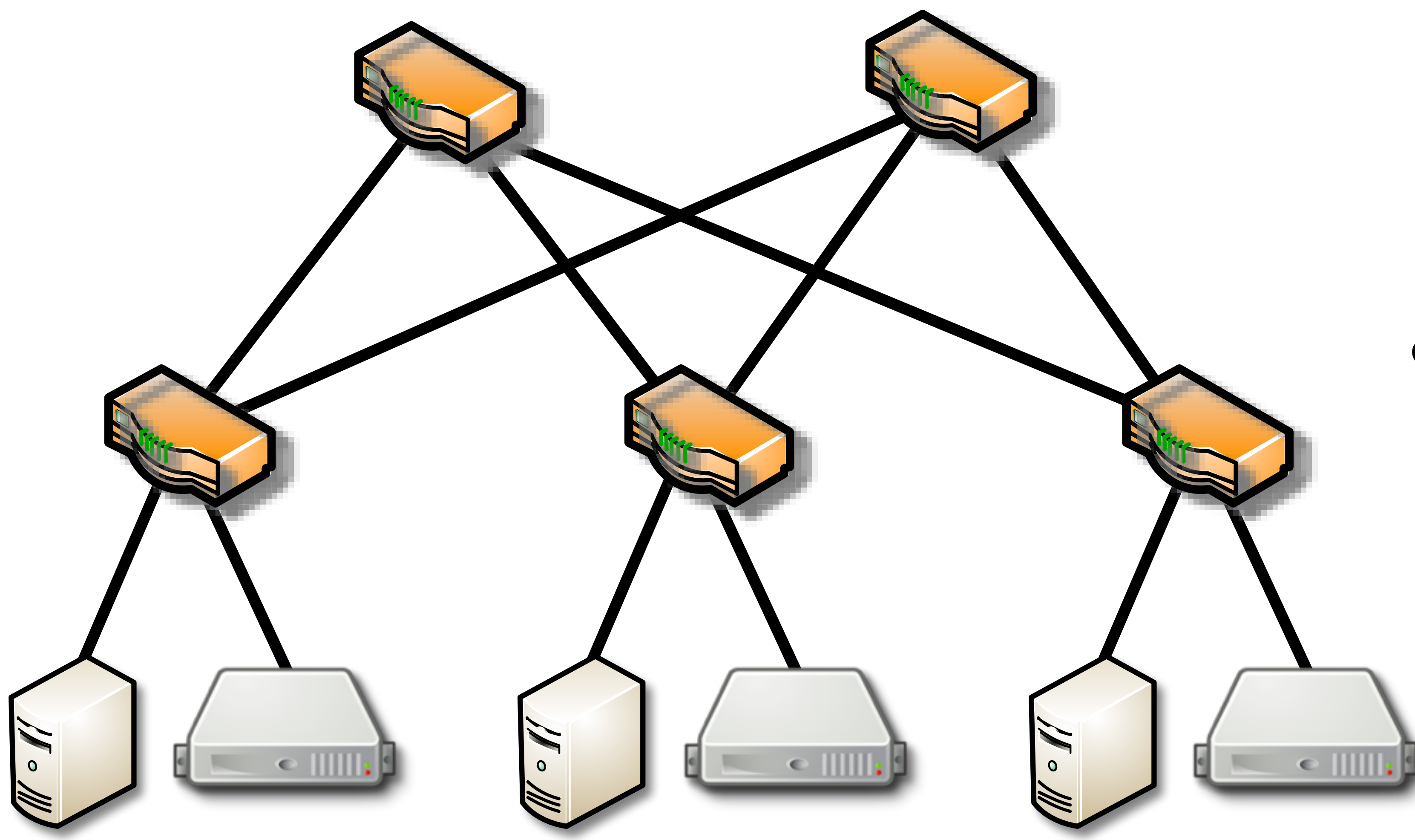
- Different path lengths, congestion cause reordering

Mostly-Ordered Multicast



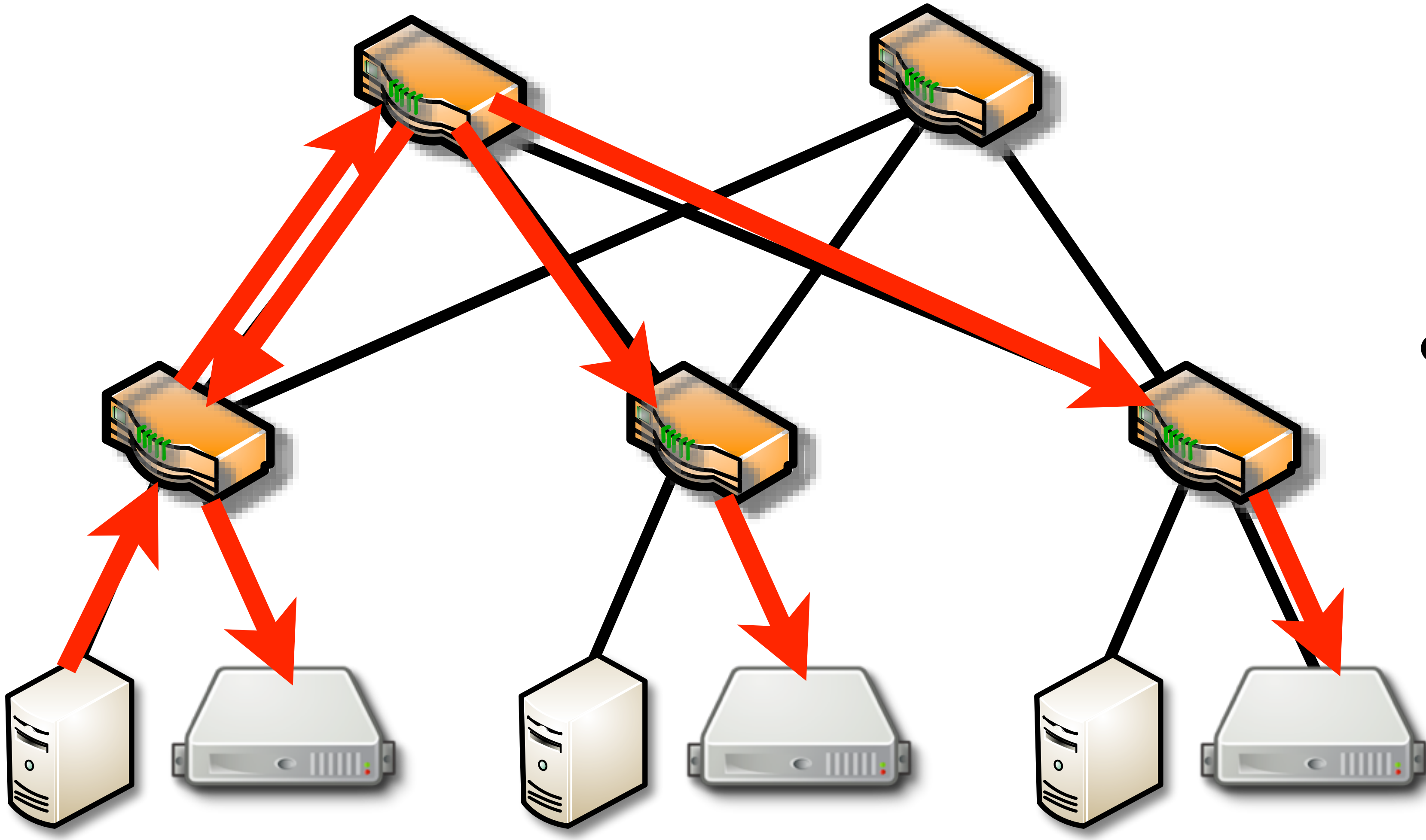
- Different path lengths, congestion cause reordering

Mostly-Ordered Multicast



- Different path lengths, congestion cause reordering
- **MOM approach**: Route multicast messages to a root switch equidistant from receivers

Mostly-Ordered Multicast



- Different path lengths, congestion cause reordering
- **MOM approach**: Route multicast messages to a root switch equidistant from receivers

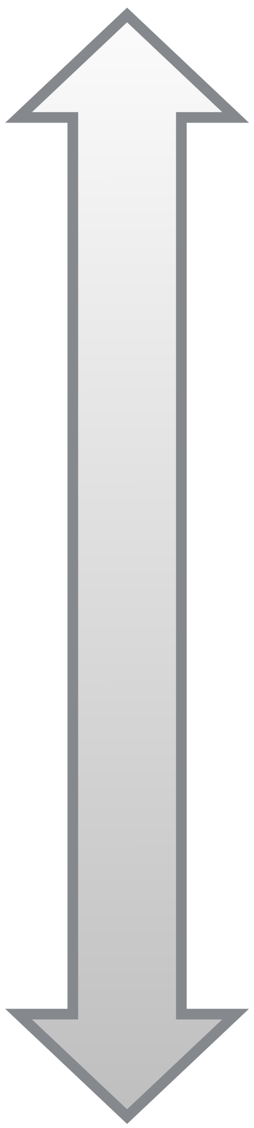
MOM Design Options

less
network support

better ordering

MOM Design Options

less network support



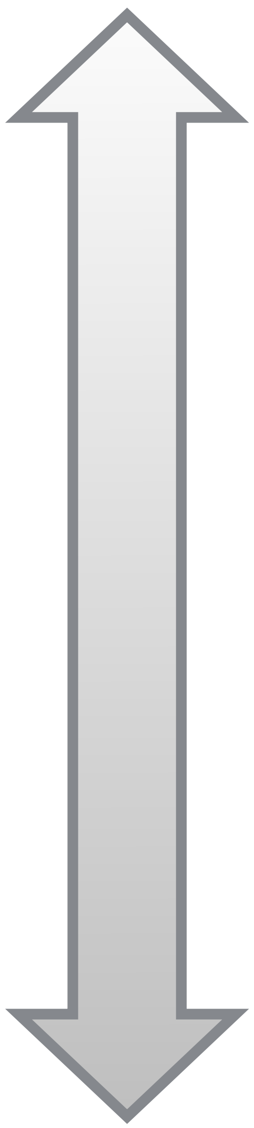
better ordering

1. Topology-Aware Multicast

   route packets to a randomly-chosen root switch

MOM Design Options

less network support

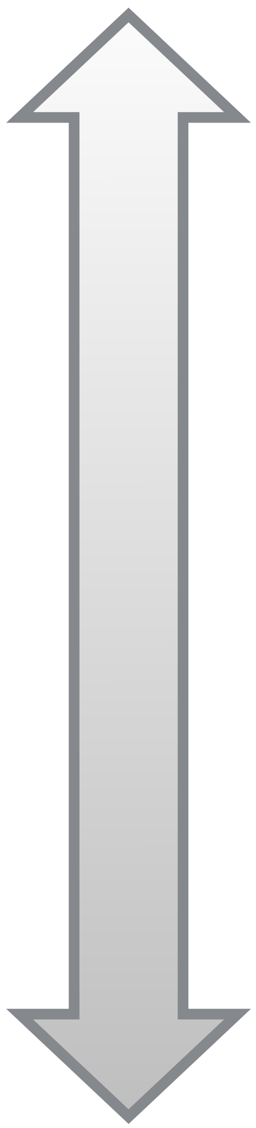


better ordering

1. Topology-Aware Multicast
   route packets to a randomly-chosen root switch
2. High-Priority Multicast
   use higher QoS priority to avoid link congestion

MOM Design Options

less network support



better ordering

1. Topology-Aware Multicast
   route packets to a randomly-chosen root switch
2. High-Priority Multicast
   use higher QoS priority to avoid link congestion
3. Network Serialization
   route packets through a *single* root switch

Speculative Paxos

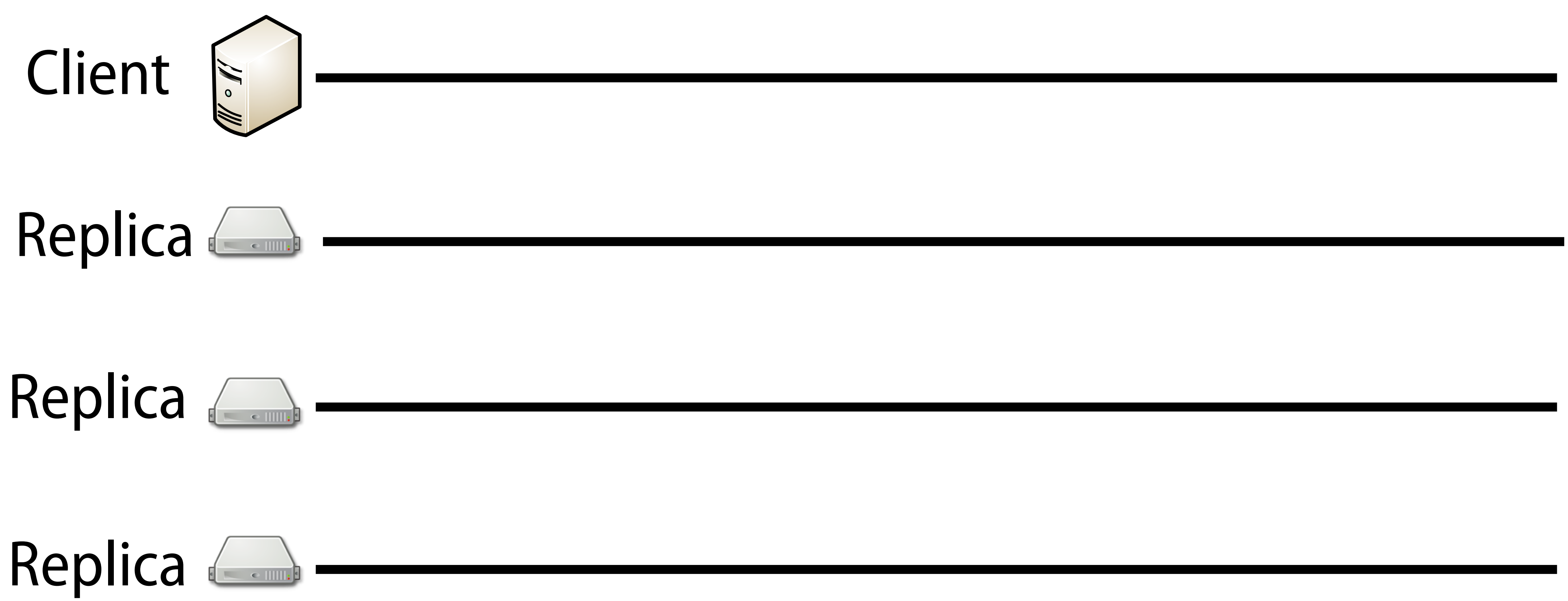
New state machine replication protocol

Relies on MOM to order requests
in the normal case

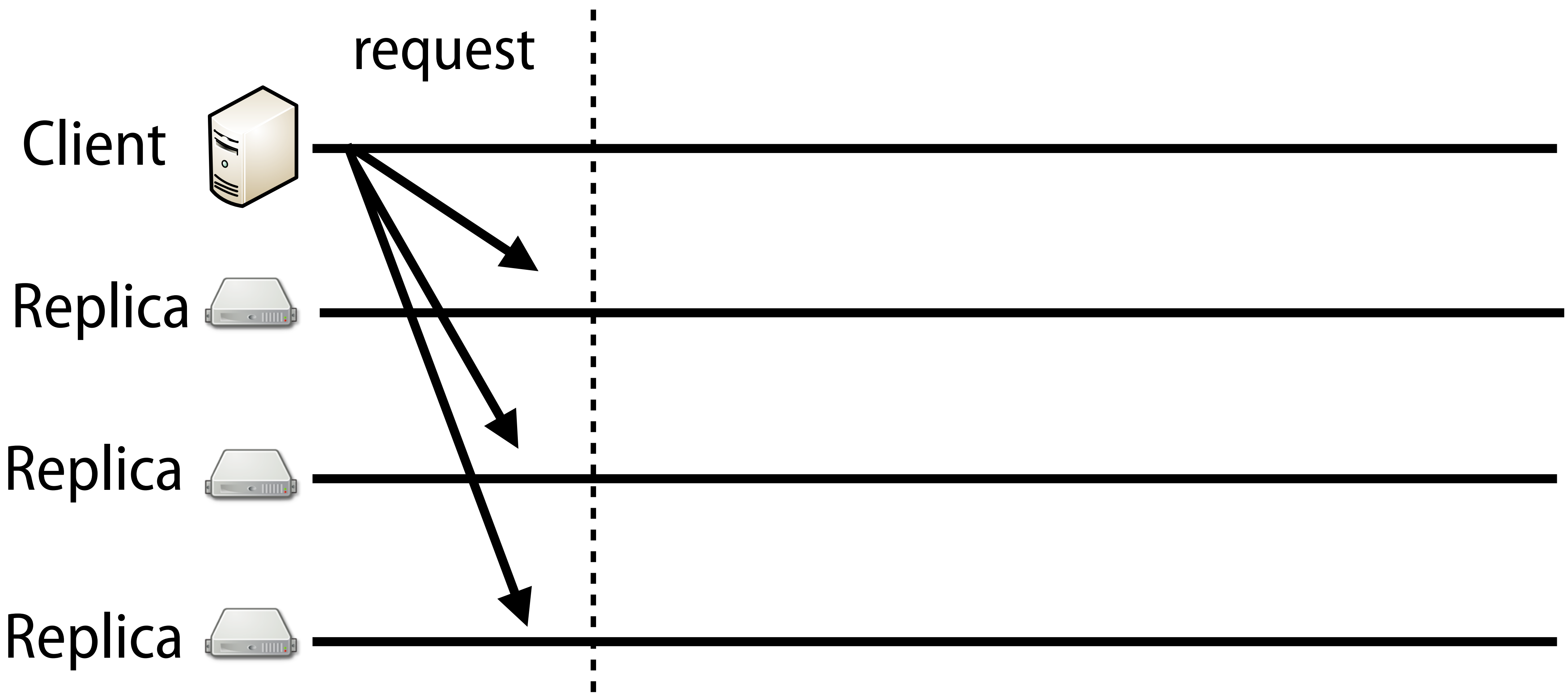
But not required:

- remains correct even with reorderings:
  safety + liveness under usual conditions

Speculative Paxos

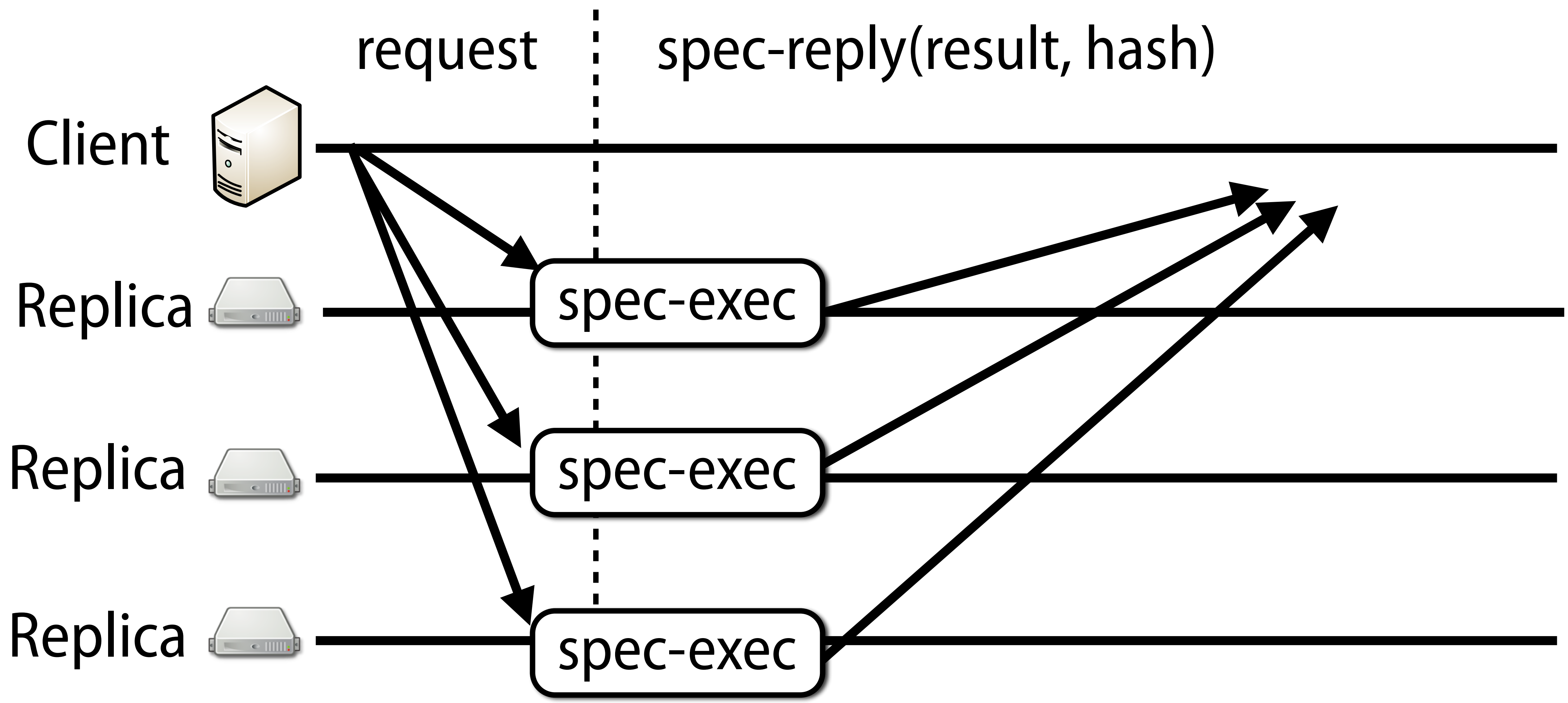


Speculative Paxos



Speculative Paxos

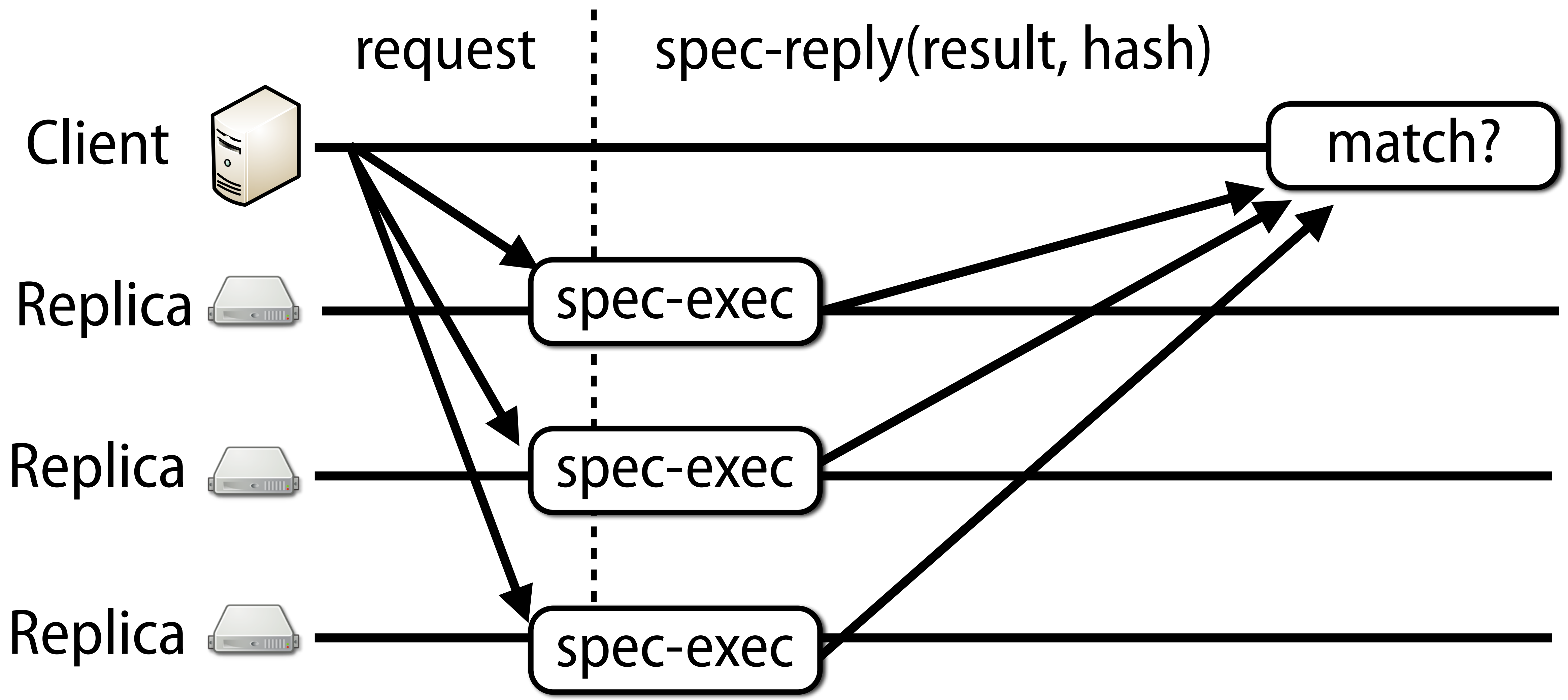
replicas immediately speculatively execute request & reply!



Speculative Paxos

replicas immediately speculatively execute request & reply!

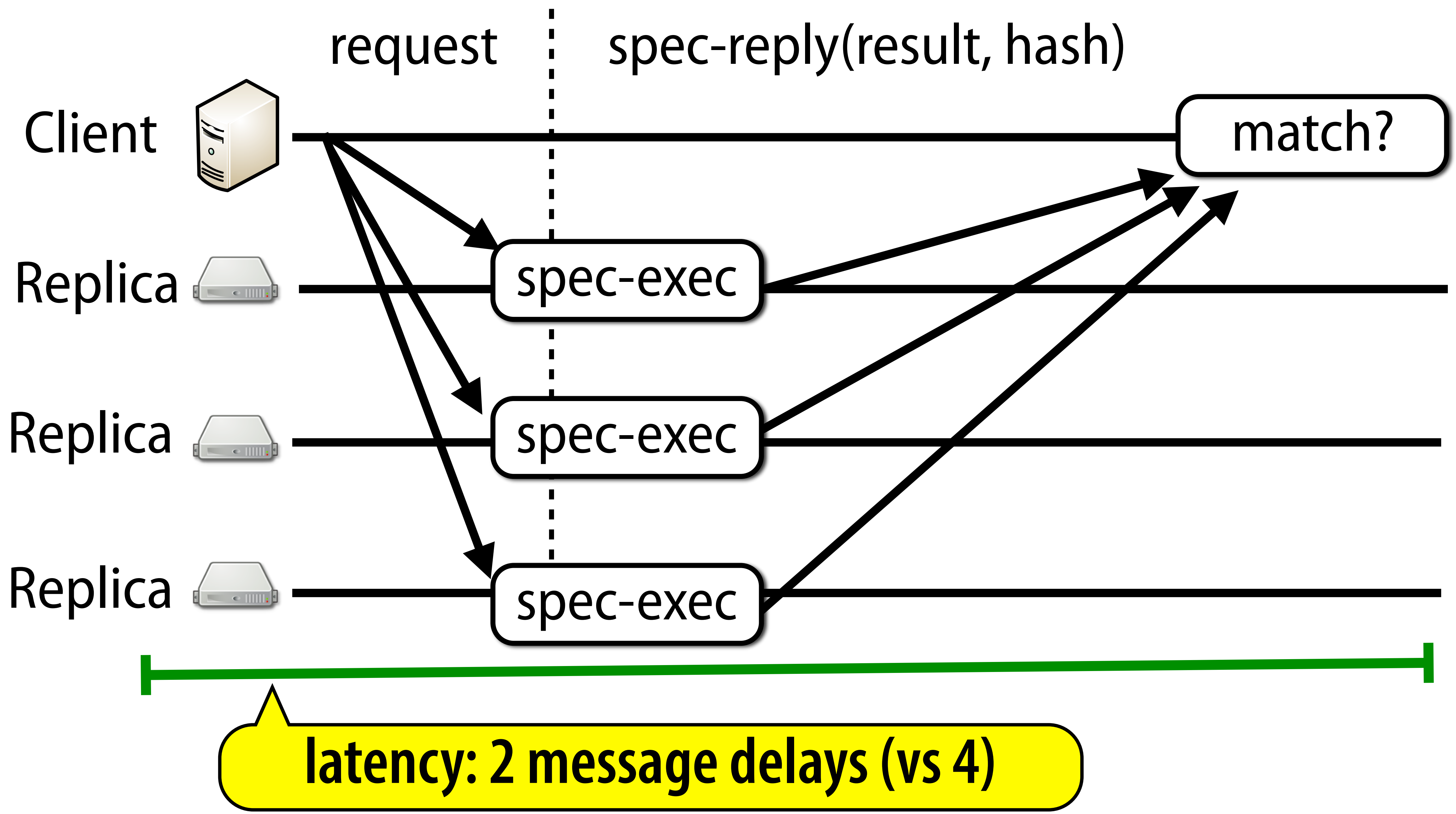
client checks for matching responses from 3/4 superquorum



Speculative Paxos

replicas immediately speculatively execute request & reply!

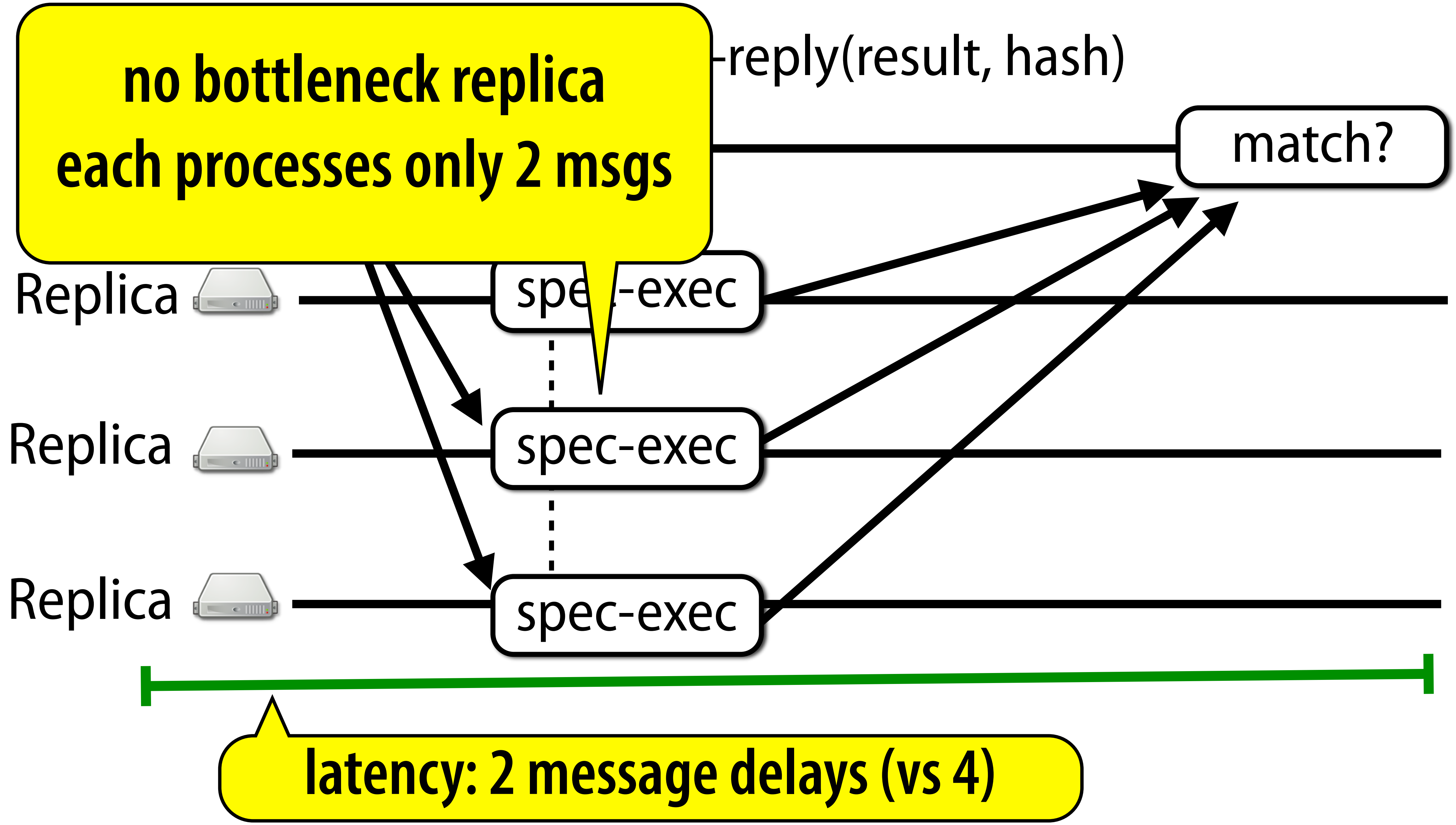
client checks for matching responses from 3/4 superquorum



Speculative Paxos

replicas immediately speculatively execute request & reply!

client checks for matching responses from 3/4 superquorum



Speculative Execution

Replicas execute requests speculatively

- might have to roll back operations

Clients know their requests succeeded

- they check for matching hashes in replies
- means clients don't need to speculate

Similar to Zyzzyva [SOSP'07]

Handling Ordering Violations

What if replicas don't execute requests in the same order?

Replicas periodically run ***synchronization*** protocol

If divergence detected: ***reconciliation***

- replicas pause execution, select leader, send logs
- leader decides ordering for operations and notifies replicas
- replicas rollback and re-execute requests in proper order

Handling Ordering Violations

What if replicas don't execute requests in the same order?

Replicas periodically run ***synchronization*** protocol

If divergence detected: ***reconciliation***

- replicas pause execution, select leader, send logs
- leader decides ordering for operations and notifies replicas
- replicas rollback and re-execute requests in proper order

Note: 3/4 superquorum requirement ensures new leader can always be sure which requests succeeded even if 1/2 fail. [cf. Fast Paxos]

Outline

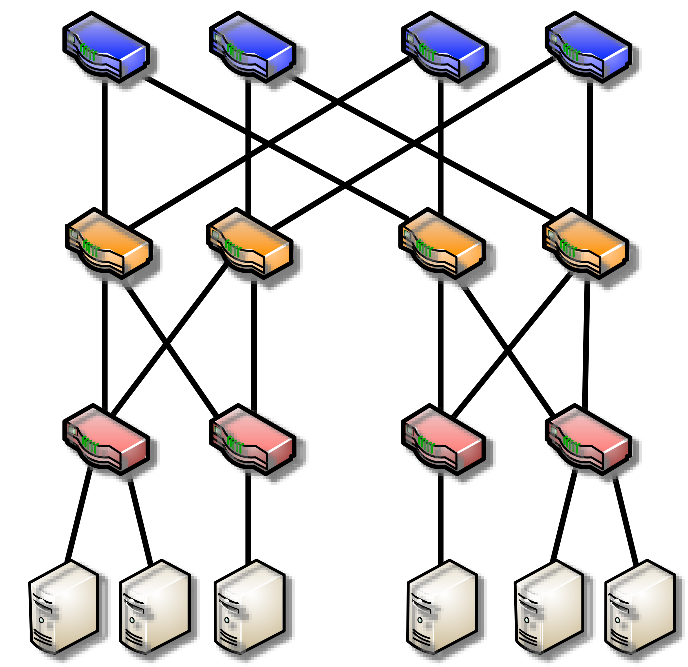
1. Co-designing Distributed Systems and Data Center Networks
2. Background: State Machine Replication & Paxos
3. Mostly-Ordered Multicast and Speculative Paxos
4. **Evaluation**

Evaluation Setup

12-switch fat tree testbed

1 Gb / 10 Gb ethernet

3 replicas (2.27 GHz Xeon L5640)



MOM scalability experiments:

2560-host simulated fat tree data center network

background traffic from Microsoft data center measurements

SpecPaxos Improves Latency and Throughput

(emulated datacenter network with MOMs)

better ↑

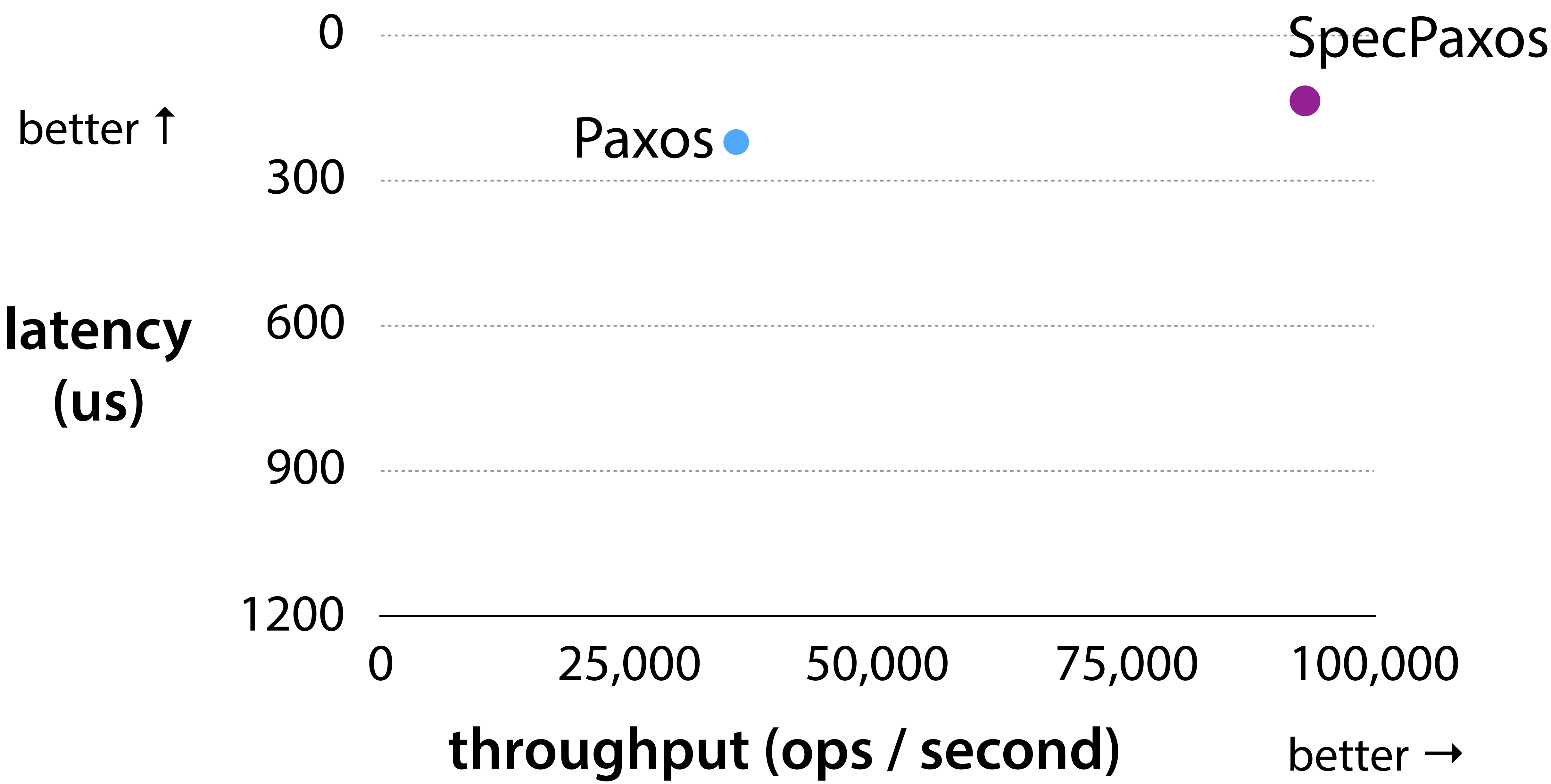
latency (us)

throughput (ops / second)

better →

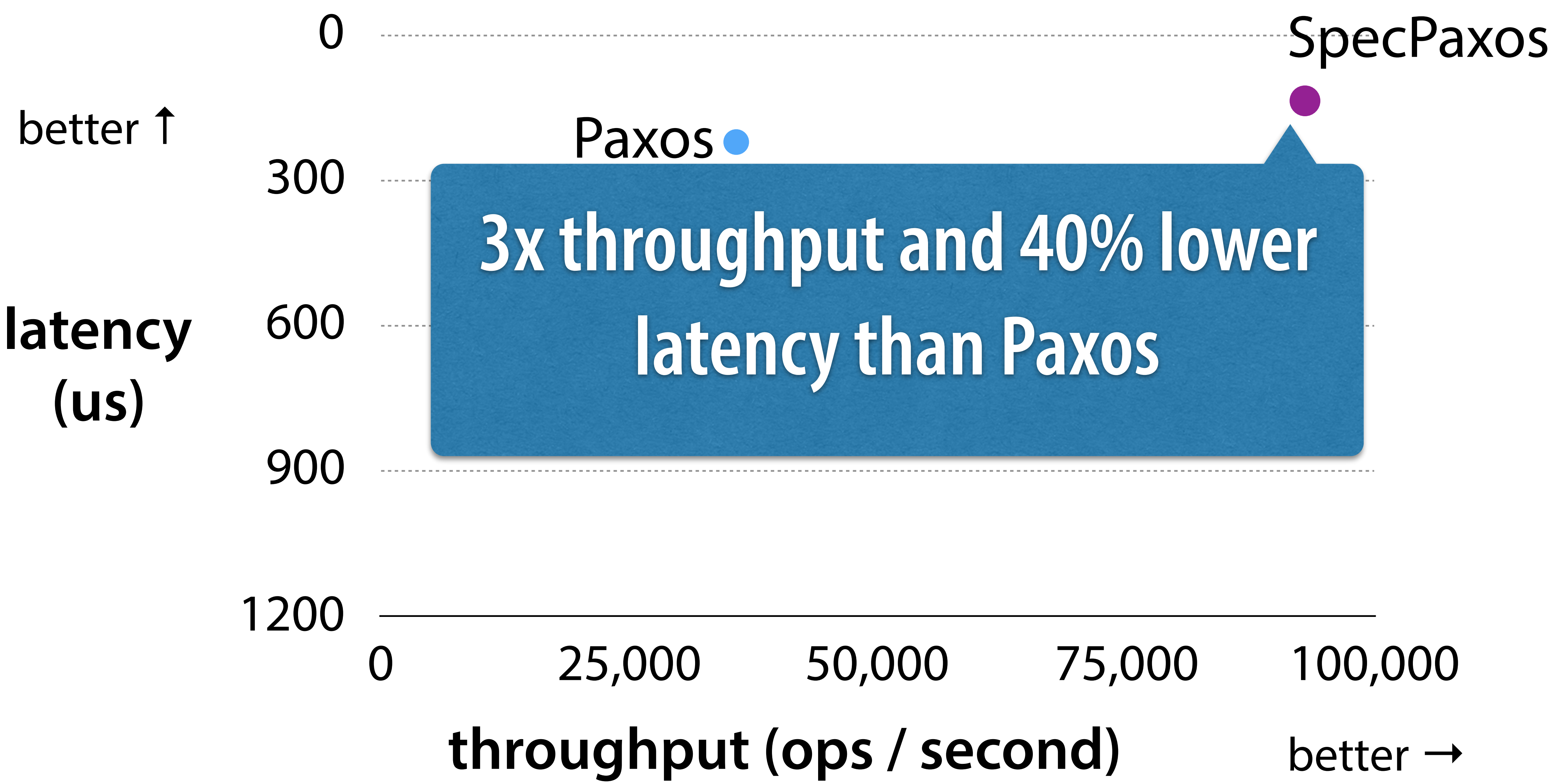
SpecPaxos Improves Latency and Throughput

(emulated datacenter network with MOMs)



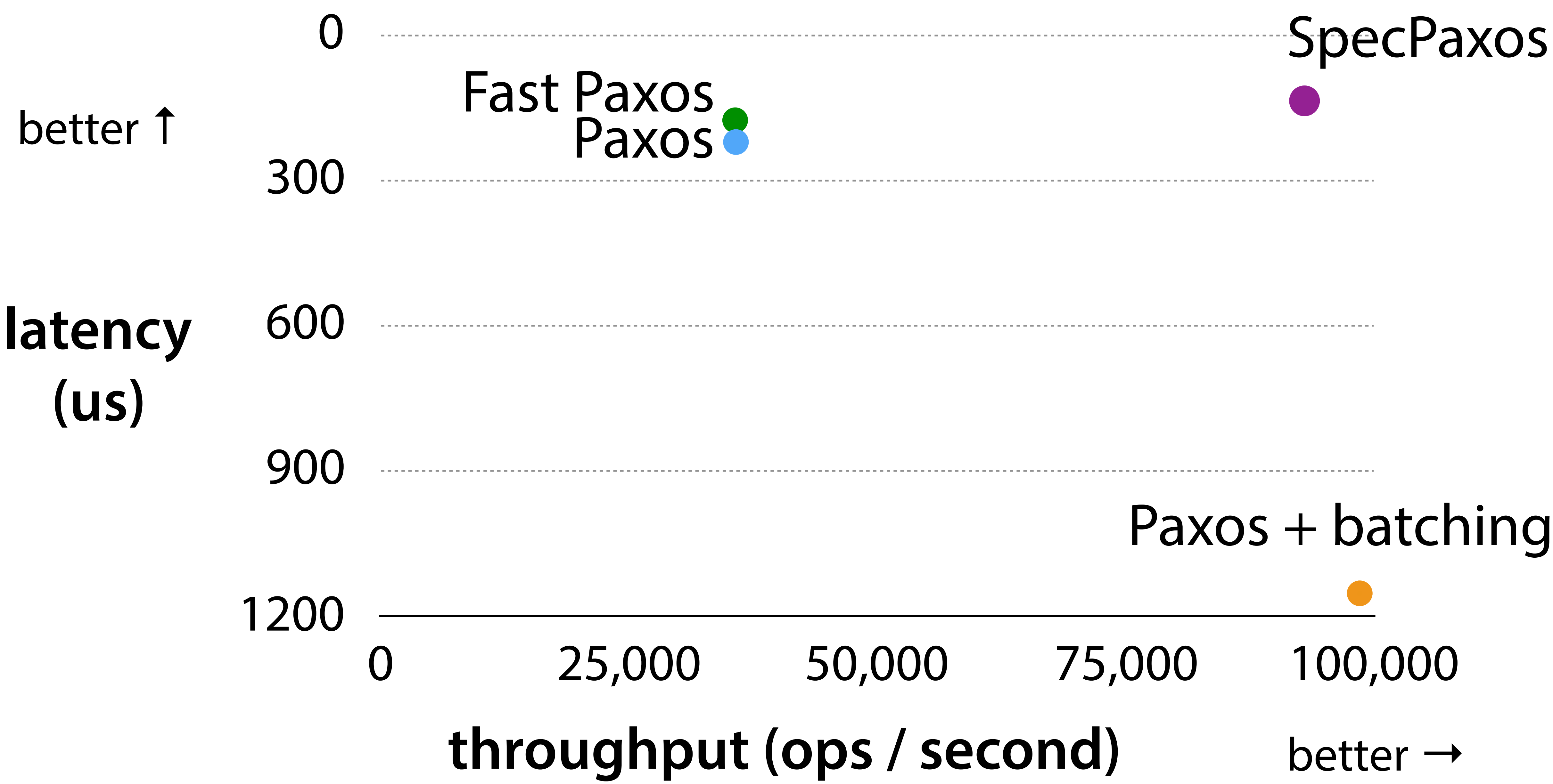
SpecPaxos Improves Latency and Throughput

(emulated datacenter network with MOMs)



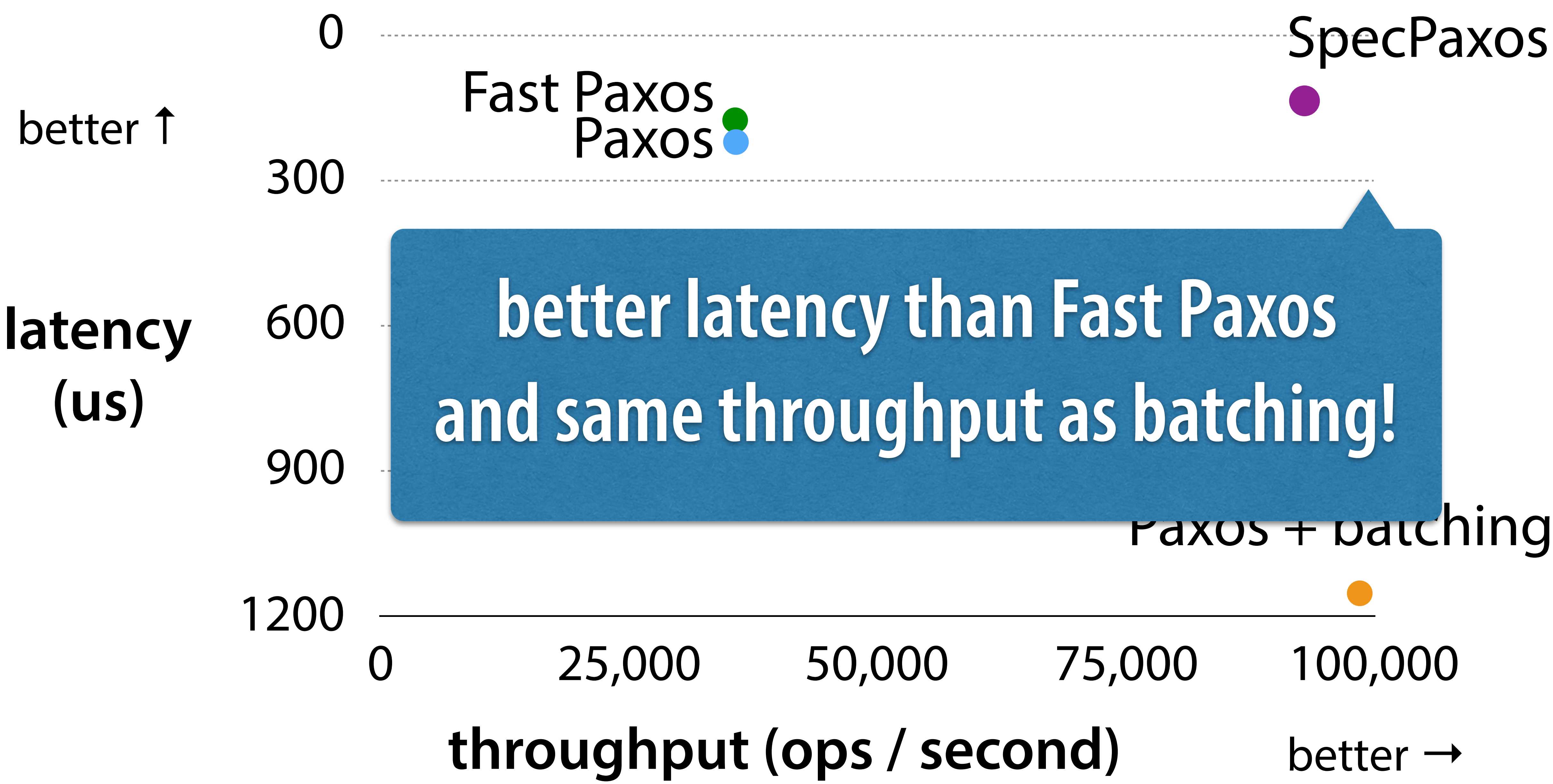
SpecPaxos Improves Latency and Throughput

(emulated datacenter network with MOMs)

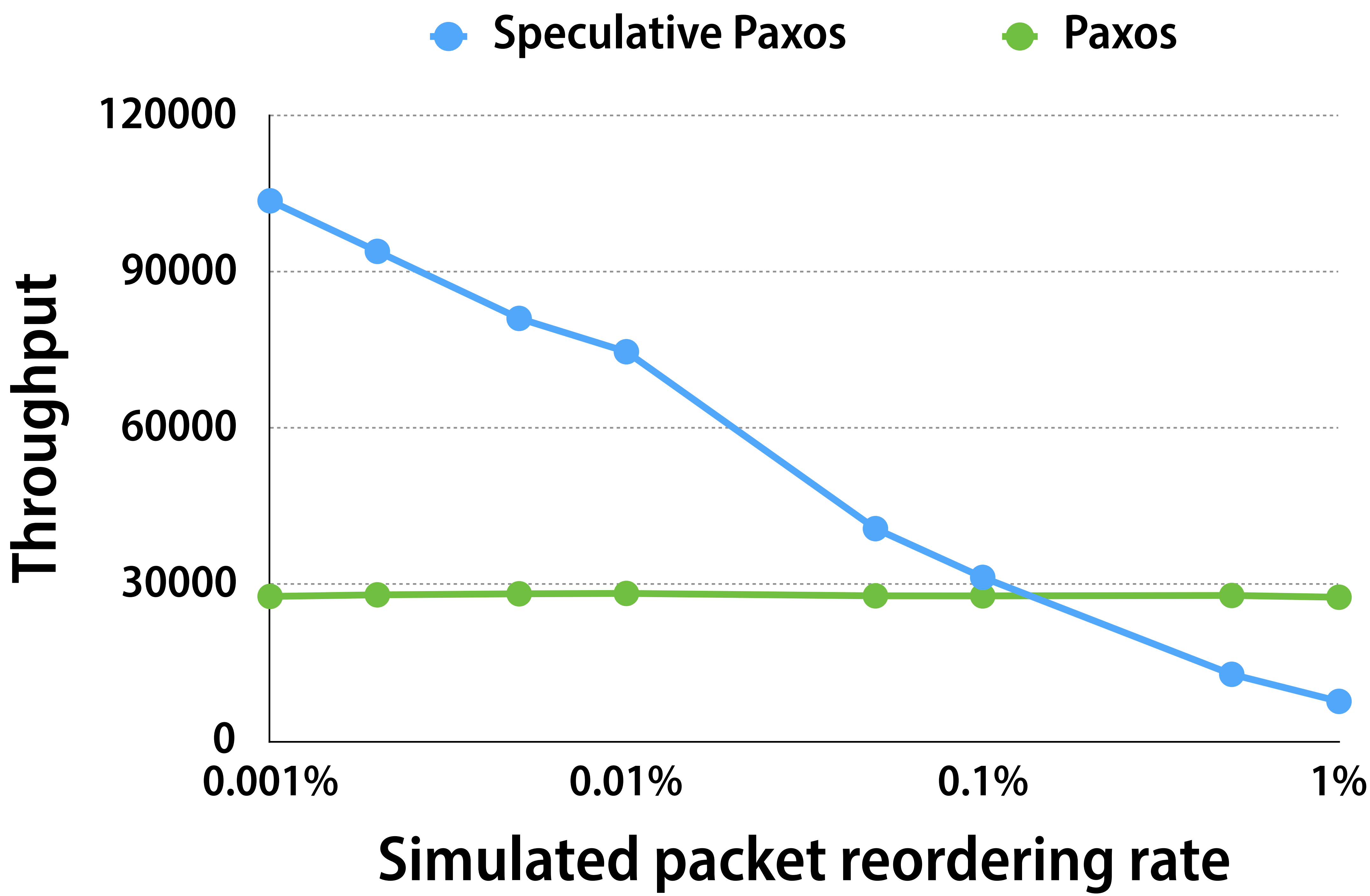


SpecPaxos Improves Latency and Throughput

(emulated datacenter network with MOMs)



MOMs Provide Necessary Support



MOM Ordering Effectiveness

Ordering Violation Rates

	Testbed (12 switches)	Simulation (119 switches, 2560 hosts)
Regular Multicast	1-10%	1-2%
Topology-Aware MOM	0.001%-0.05%	0.01%-0.1%
Network Serialization	~0%	~0%

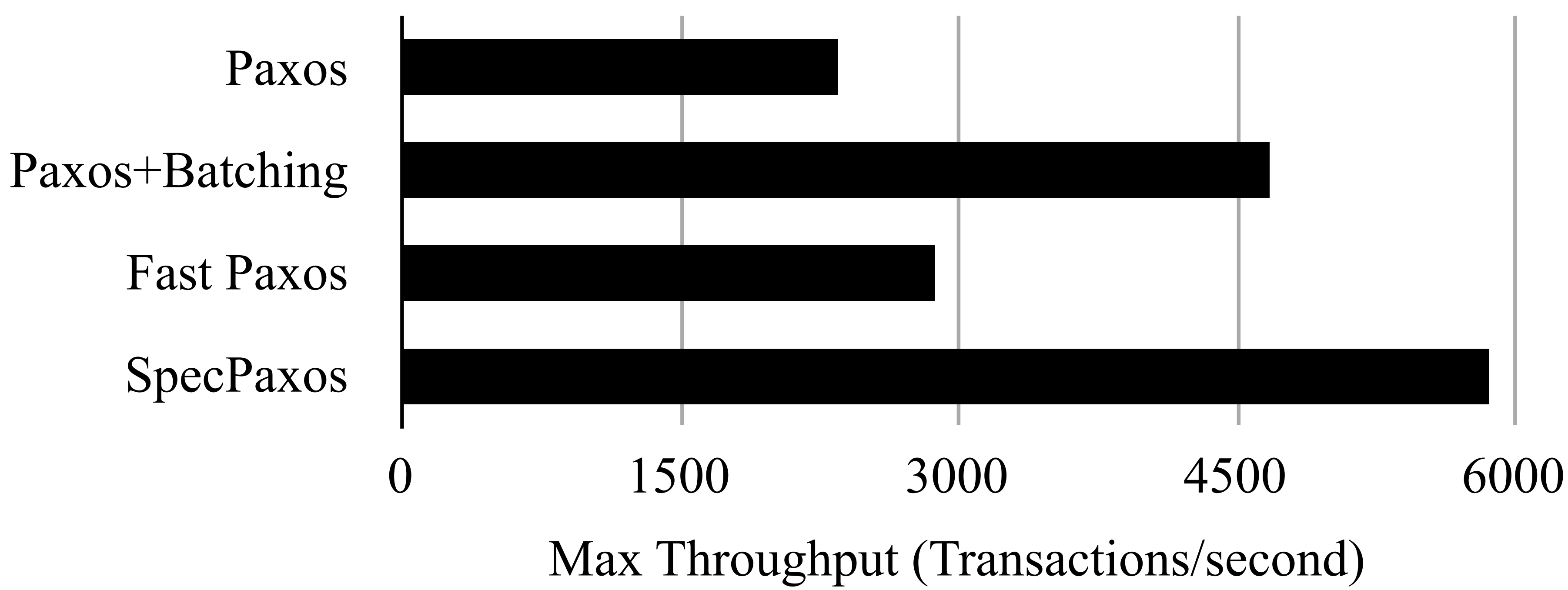
Application Performance

Transactional key-value store (2PC + OCC)

Synthetic workload based on Retwis Twitter clone

< 250 LOC required to implement rollback

Measured transactions/sec that meet 10 ms SLO



Summary

New approach to building distributed systems
based on co-designing with the data center network

Dramatic performance improvement for replication by combining

- MOM network primitive for best-effort ordering
- Speculative Paxos: efficient replication protocol

This is only the first step for co-designing distributed systems and data center networks!