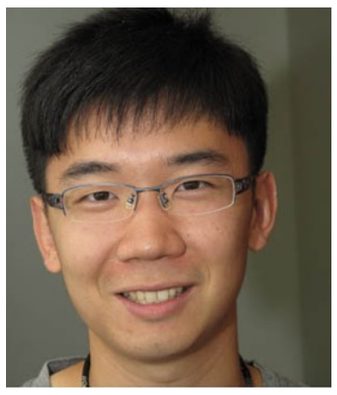
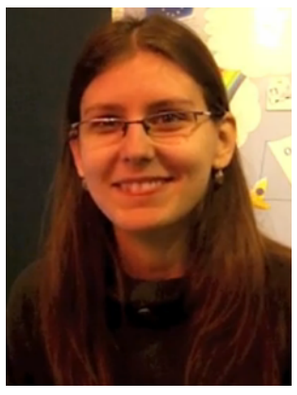
Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering

Jialin Li, Ellis Michael, Naveen Kr. Sharma, Adriana Szekeres, Dan R. K. Ports

UNIVERSITY of WASHINGTON




Server failures are the common case in data centers

Server failures are the common case in data centers

Cloud News Daily

Lightning Strikes Disrupt Google Data Center

Server failures are the common case in data centers

Cloud News Daily

Lightning Strikes Disrupt Google Data Center

BUSINESS INSIDER

Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data

Server failures are the common case in data centers

Cloud News Daily

Lightning Strikes Disrupt Google Data Center

Technology News

Microsoft and Google cloud users suffer service outages

INSIDER

Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data

State Machine Replication

Operation A

Operation B

Operation C

State Machine Replication

Operation A

Operation B

Operation C

Operation A

Operation B

Operation C

Operation A

Operation B

Operation C

State Machine Replication

Operation A

Operation B

Operation C

Operation A

Operation B

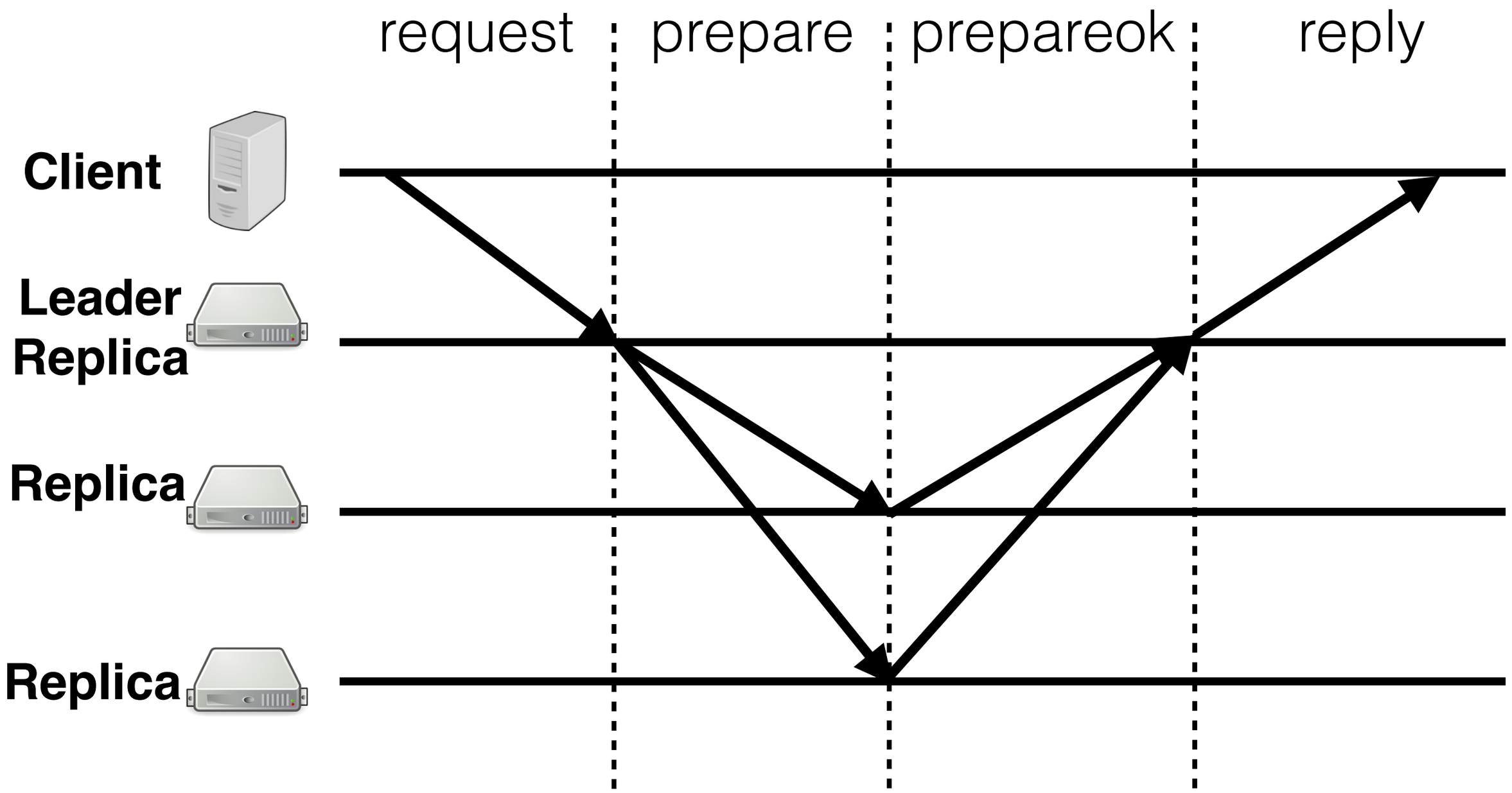
Operation C

Operation A

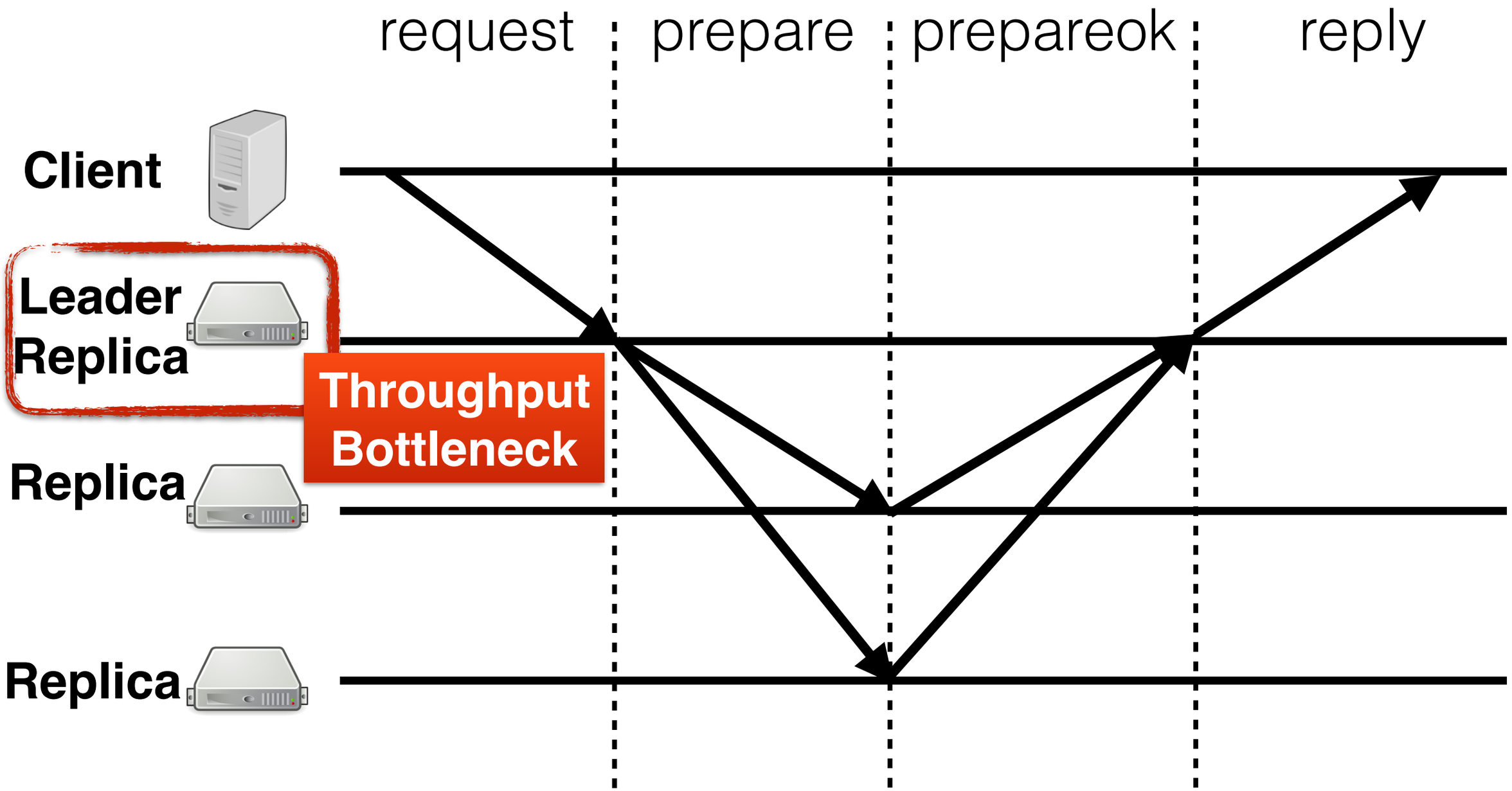
Operation B

Operation C

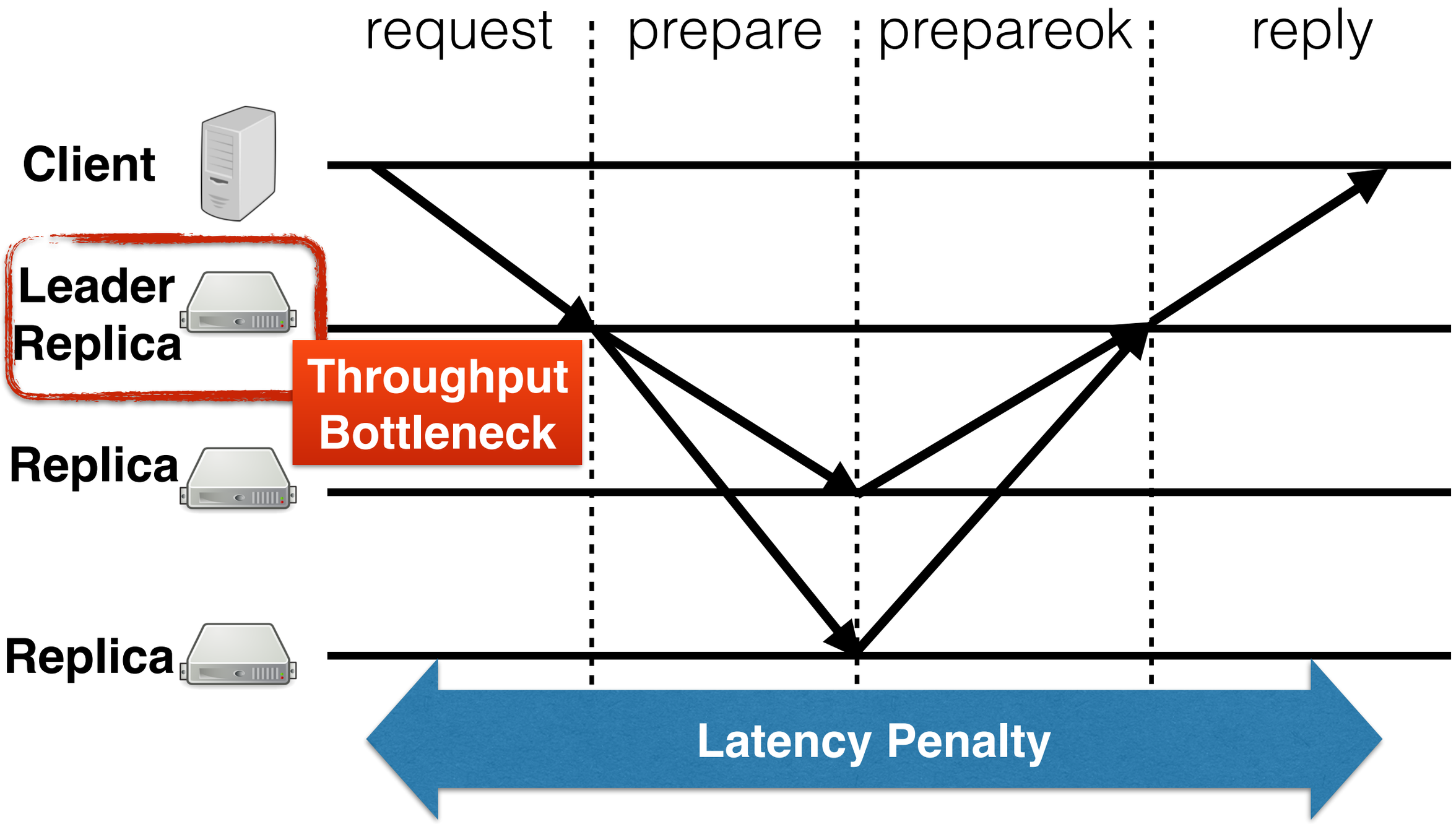
Paxos for state machine replication



Paxos for state machine replication



Paxos for state machine replication



Can we eliminate Paxos overhead?

Performance overhead due to **worst-case** network assumptions

- valid assumptions for the Internet
- data center networks are different

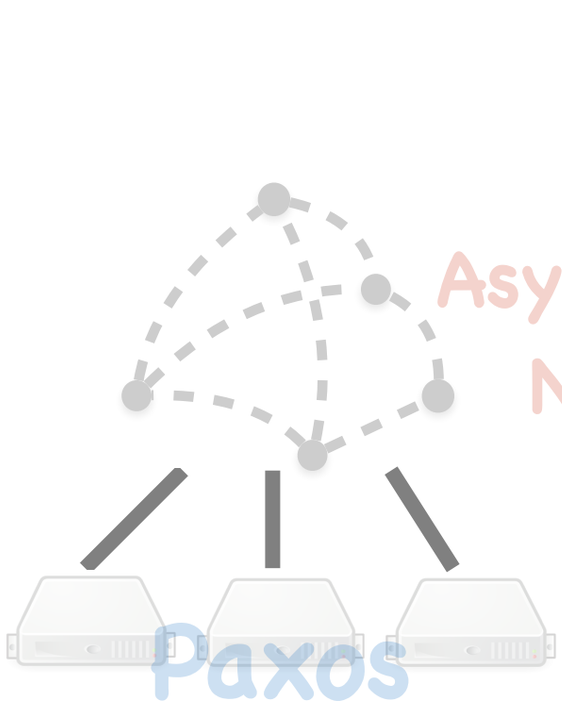
What properties should the network have to enable faster replication?

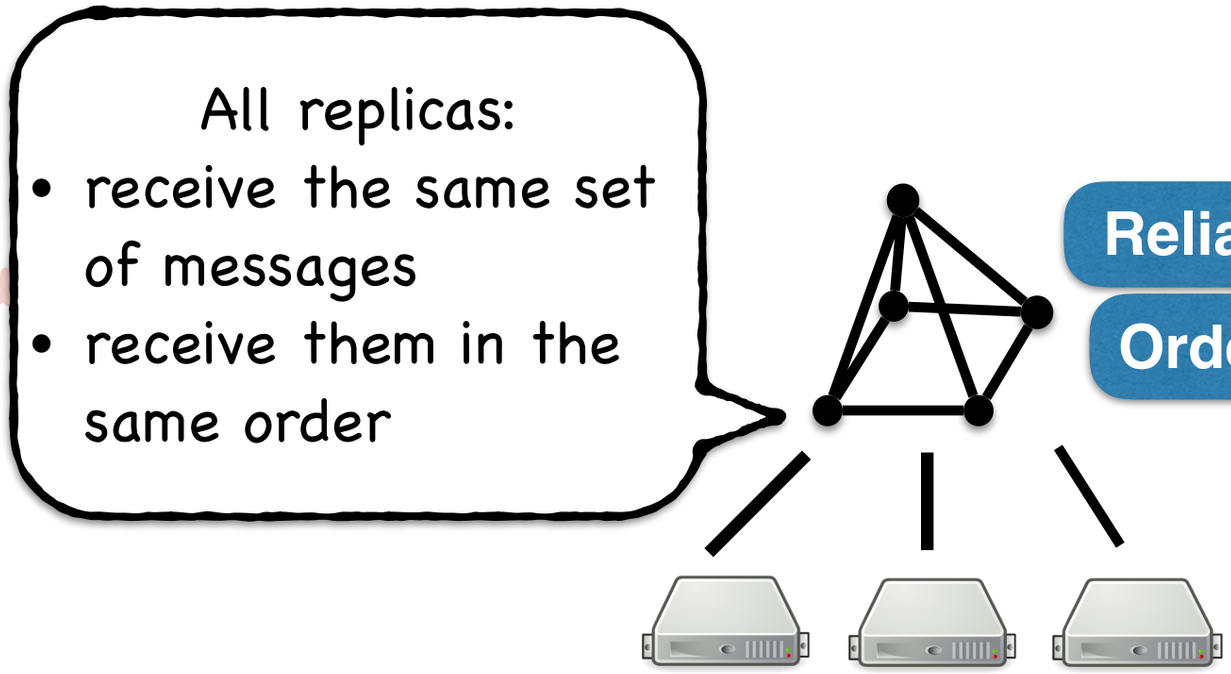
Network properties determine replication complexity

Messages may be:
- dropped
- reordered
- delivered with arbitrary latency

- Paxos protocol on every operation
- High performance cost

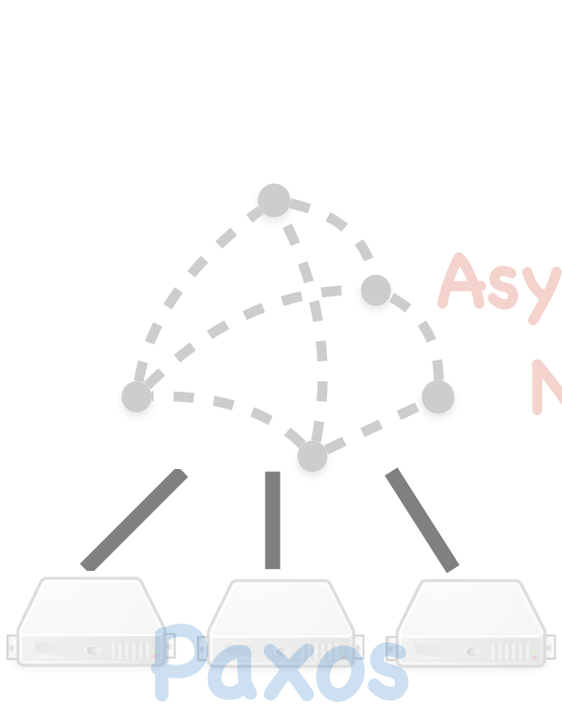
Network properties determine replication complexity

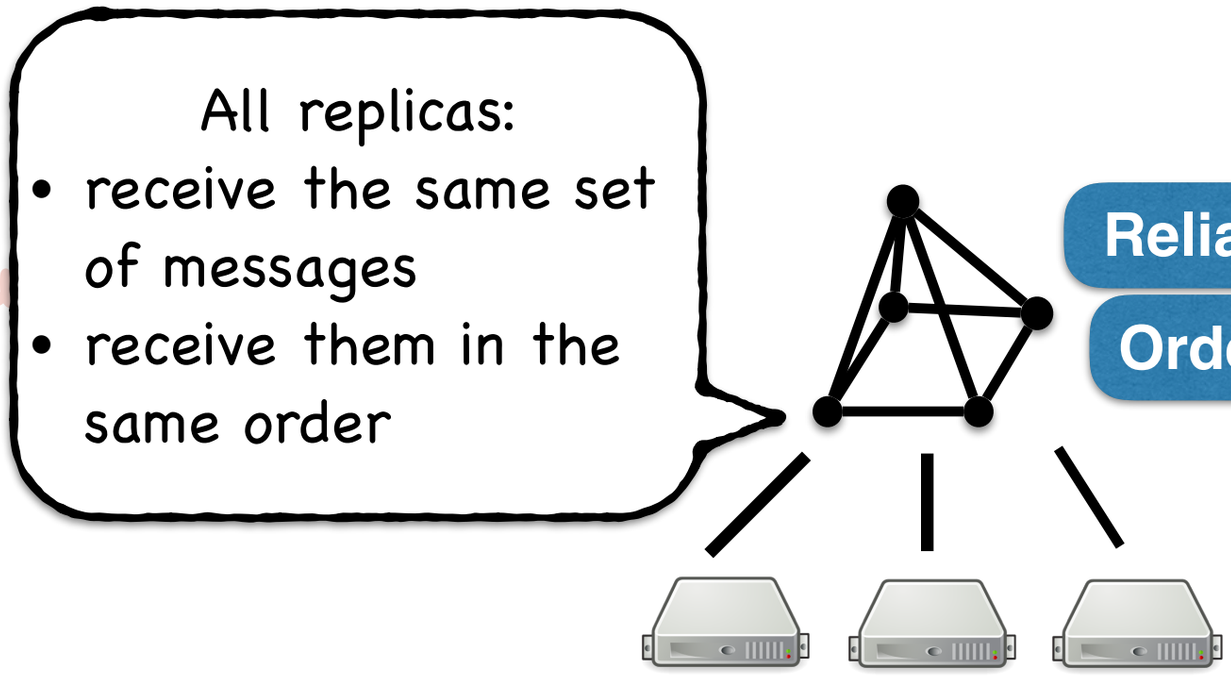




- Paxos protocol on every operation
- High performance cost

Network properties determine replication complexity

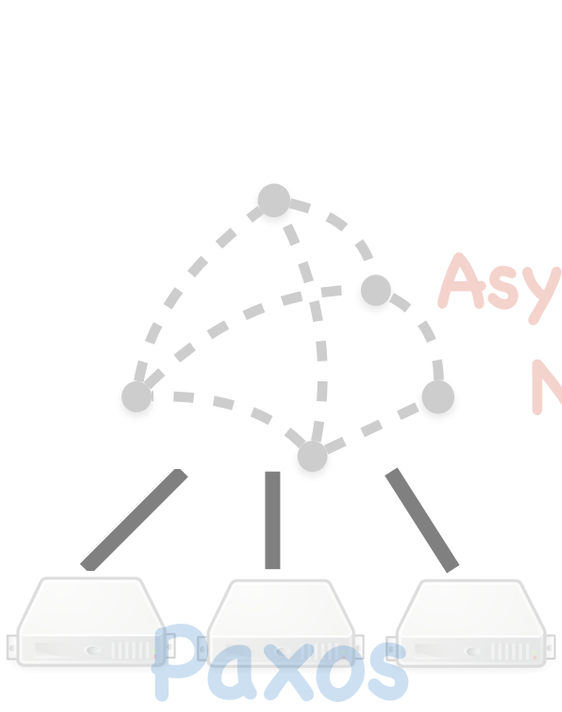




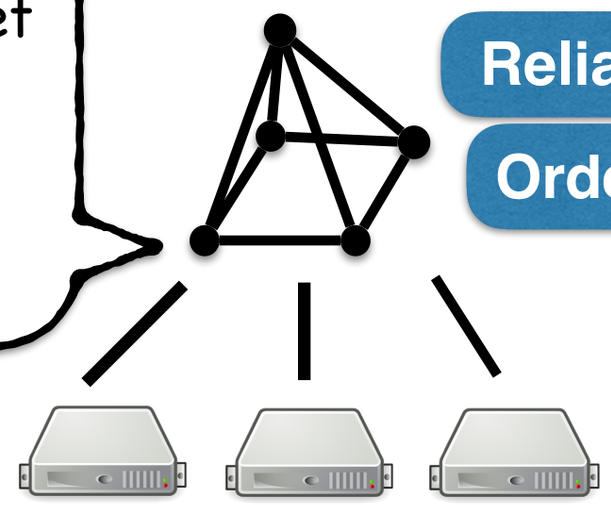
- Paxos protocol on every operation
- High performance cost

- Replication is trivial

Network properties determine replication complexity



All replicas:
- receive the same set of messages
- receive them in the same order

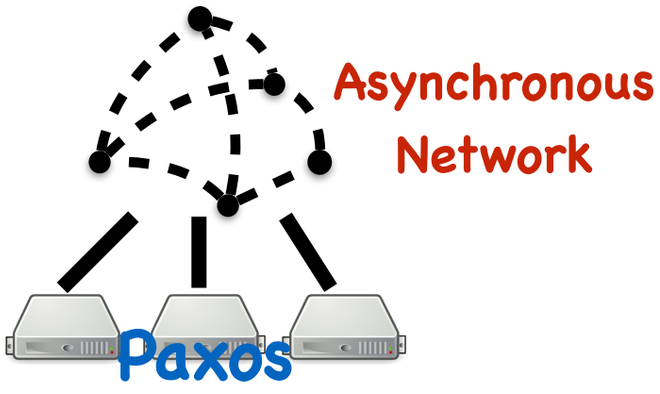


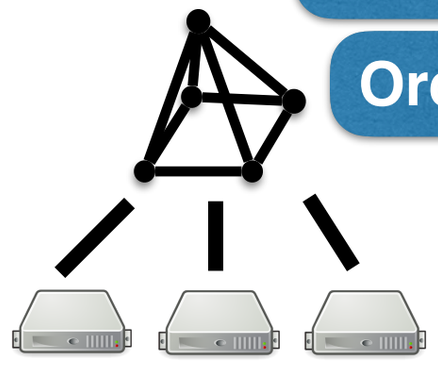
Reliability

Ordering

- Paxos protocol on every operation
- High performance cost

- Replication is trivial
- Network implementation has the same complexity as Paxos





Reliability

Ordering

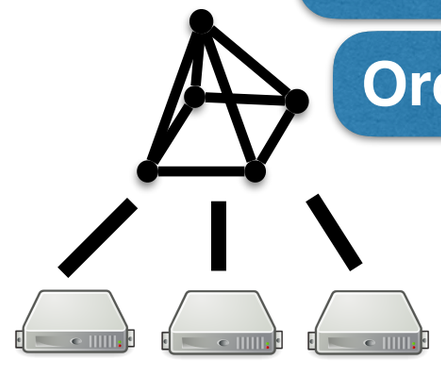
Weak

Strong

Network Guarantee

Reliability

Ordering



Weak

Strong

Network Guarantee

Can we build a network model that:

- provides **performance benefits**
- can be implemented more **efficiently**

This Talk

This Talk

A new network model with ***near-zero-cost*** implementation:
Ordered Unreliable Multicast

This Talk

A new network model with ***near-zero-cost*** implementation:
Ordered Unreliable Multicast

+

This Talk

A new network model with ***near-zero-cost*** implementation:
Ordered Unreliable Multicast

+

A ***coordination-free*** replication protocol:
Network-Ordered Paxos

This Talk

A new network model with ***near-zero-cost*** implementation:
Ordered Unreliable Multicast

+

A ***coordination-free*** replication protocol:
Network-Ordered Paxos

=

This Talk

A new network model with ***near-zero-cost*** implementation:
Ordered Unreliable Multicast

+

A ***coordination-free*** replication protocol:
Network-Ordered Paxos

=

replication within **2%** throughput overhead

Outline

1. Background on state machine replication and data center network
2. Ordered Unreliable Multicast
3. Network-Ordered Paxos
4. Evaluation

Towards an ordered but unreliable network

Key Idea: Separate **ordering** from **reliable delivery** in state machine replication

Network provides **ordering**

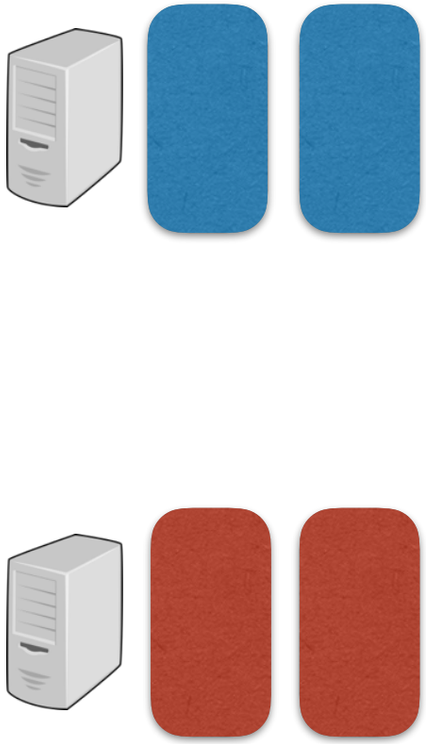
Replication protocol handles **reliability**

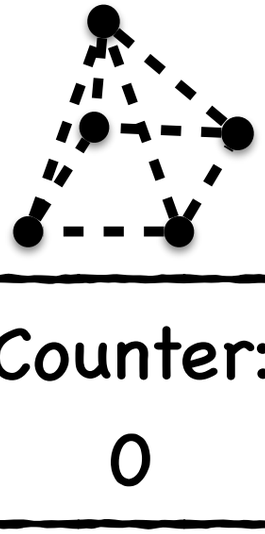
OUM Approach

- Designate one **sequencer** in the network
- Sequencer maintains a counter for each OUM group

1. Forward OUM messages to the sequencer
2. Sequencer increments counter and writes counter value into packet headers
3. Receivers use sequence numbers to detect **reordering** and **message drops**

Ordered Unreliable Multicast



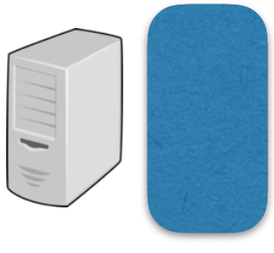


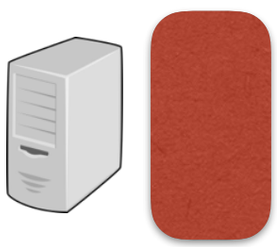
Counter:
0

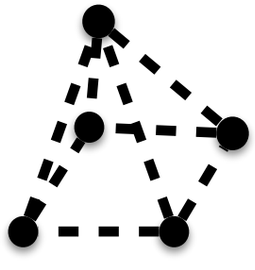
Senders

Receivers

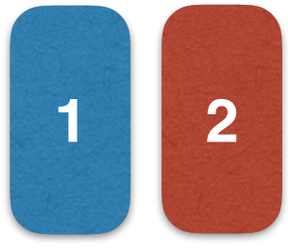
Ordered Unreliable Multicast

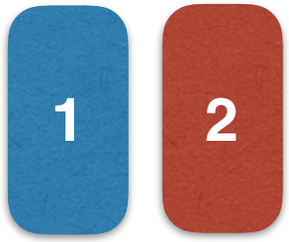


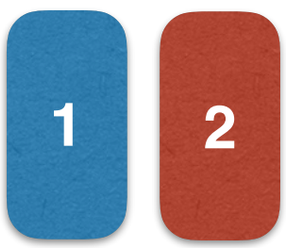




Counter:
2



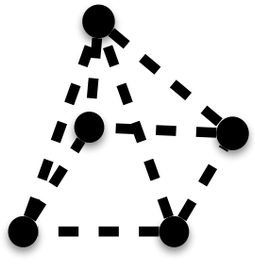




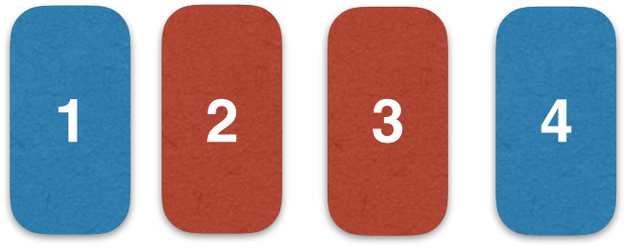
Senders

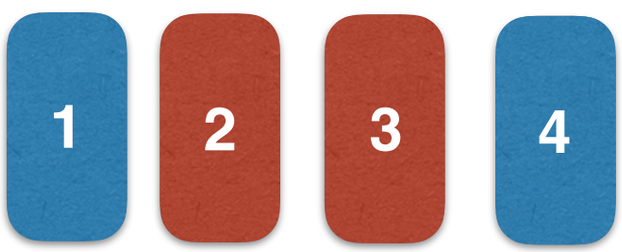
Receivers

Ordered Unreliable Multicast



Counter:
4

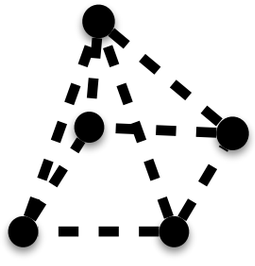




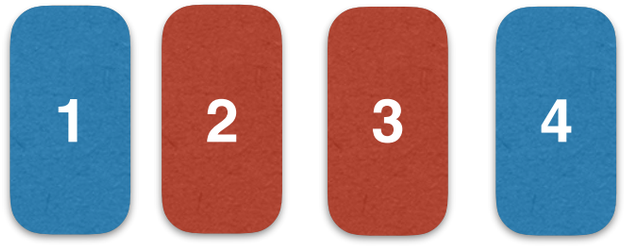
Senders

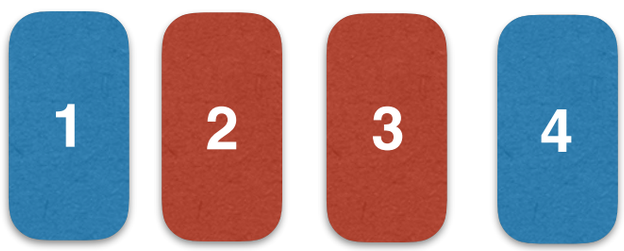
Receivers

Ordered Unreliable Multicast



Counter:
4





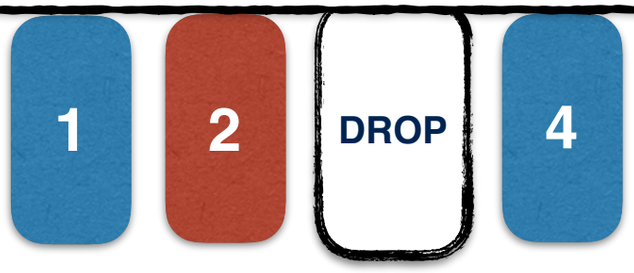
Senders

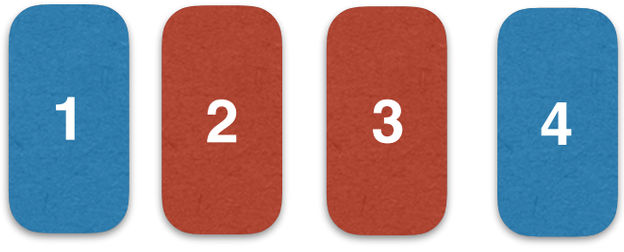
Receivers

Ordered Unreliable

Ordered Multicast:
no coordination required to determine order of messages

Counter:
4



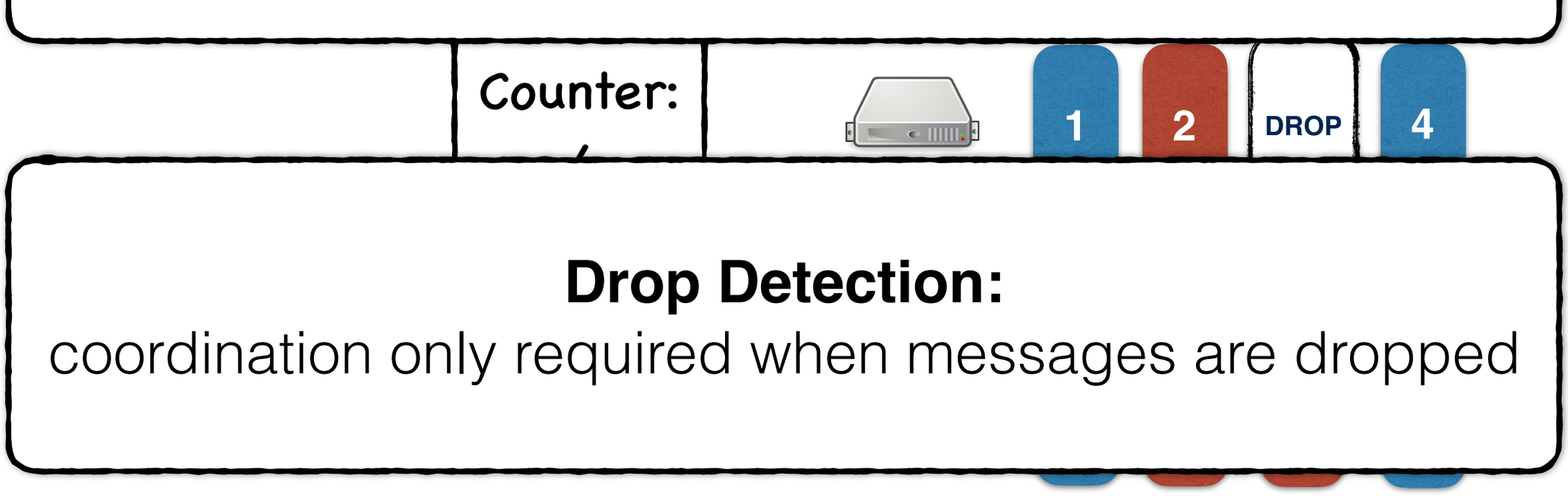


Senders

Receivers

Ordered Unreliable

Ordered Multicast:
no coordination required to determine order of messages



Drop Detection:
coordination only required when messages are dropped

Senders

Receivers

Sequencer Implementations

In-switch sequencing

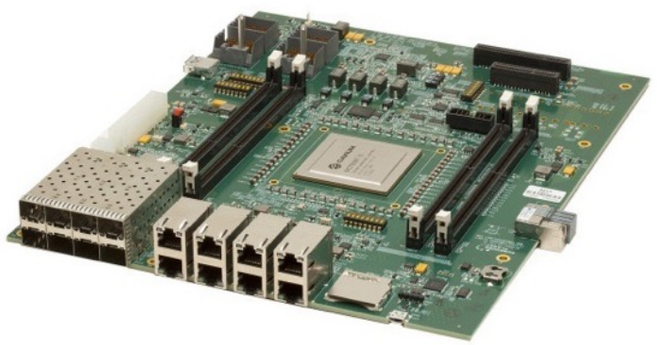
- next generation programmable switches
- implemented in P4
- nearly **zero cost**

Middlebox prototype

- Cavium Octeon network processor
- connects to root switches
- adds 8 us latency

End-host sequencing

- no specialized hardware required
- incurs higher latency penalties
- similar throughput benefits



Sequencer Implementations

In-switch sequencing

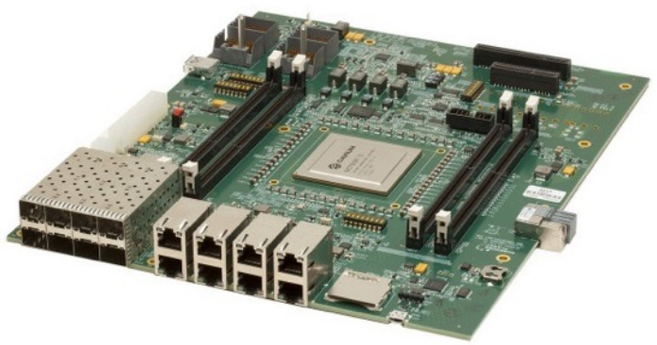
- next generation programmable switches
- implemented in P4
- nearly **zero cost**

Middlebox prototype

- Cavium Octeon network processor
- connects to root switches
- adds 8 us latency

End-host sequencing

- no specialized hardware required
- incurs higher latency penalties
- similar throughput benefits



Sequencer Implementations

In-switch sequencing

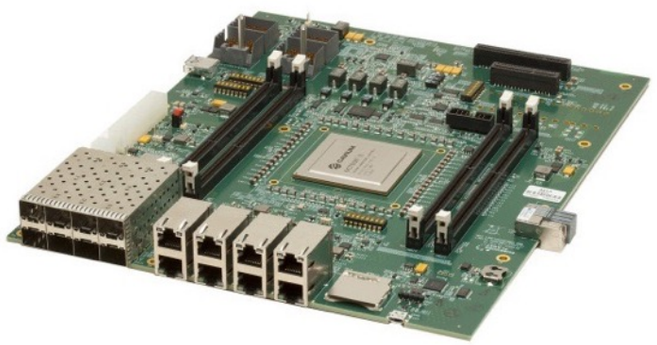
- next generation programmable switches
- implemented in P4
- nearly **zero cost**

Middlebox prototype

- Cavium Octeon network processor
- connects to root switches
- adds 8 us latency

End-host sequencing

- no specialized hardware required
- incurs higher latency penalties
- similar throughput benefits



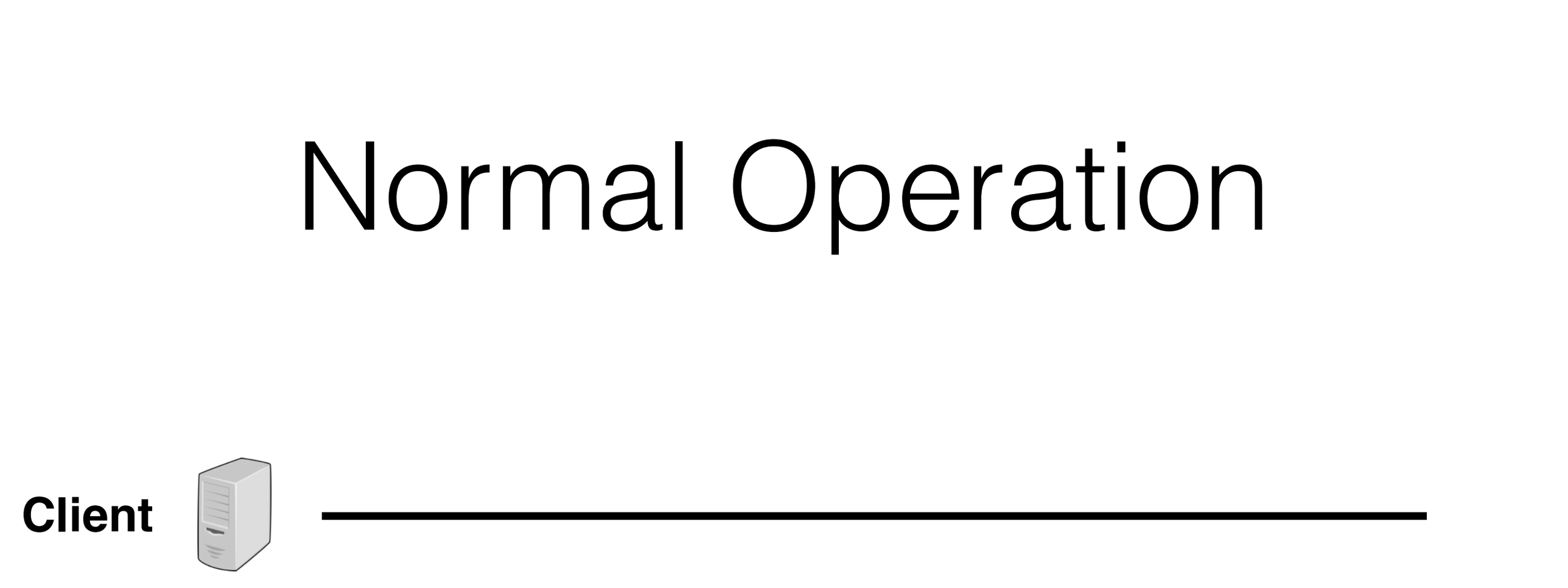
Outline

1. Background on state machine replication and data center network
2. Ordered Unreliable Multicast
3. Network-Ordered Paxos
4. Evaluation

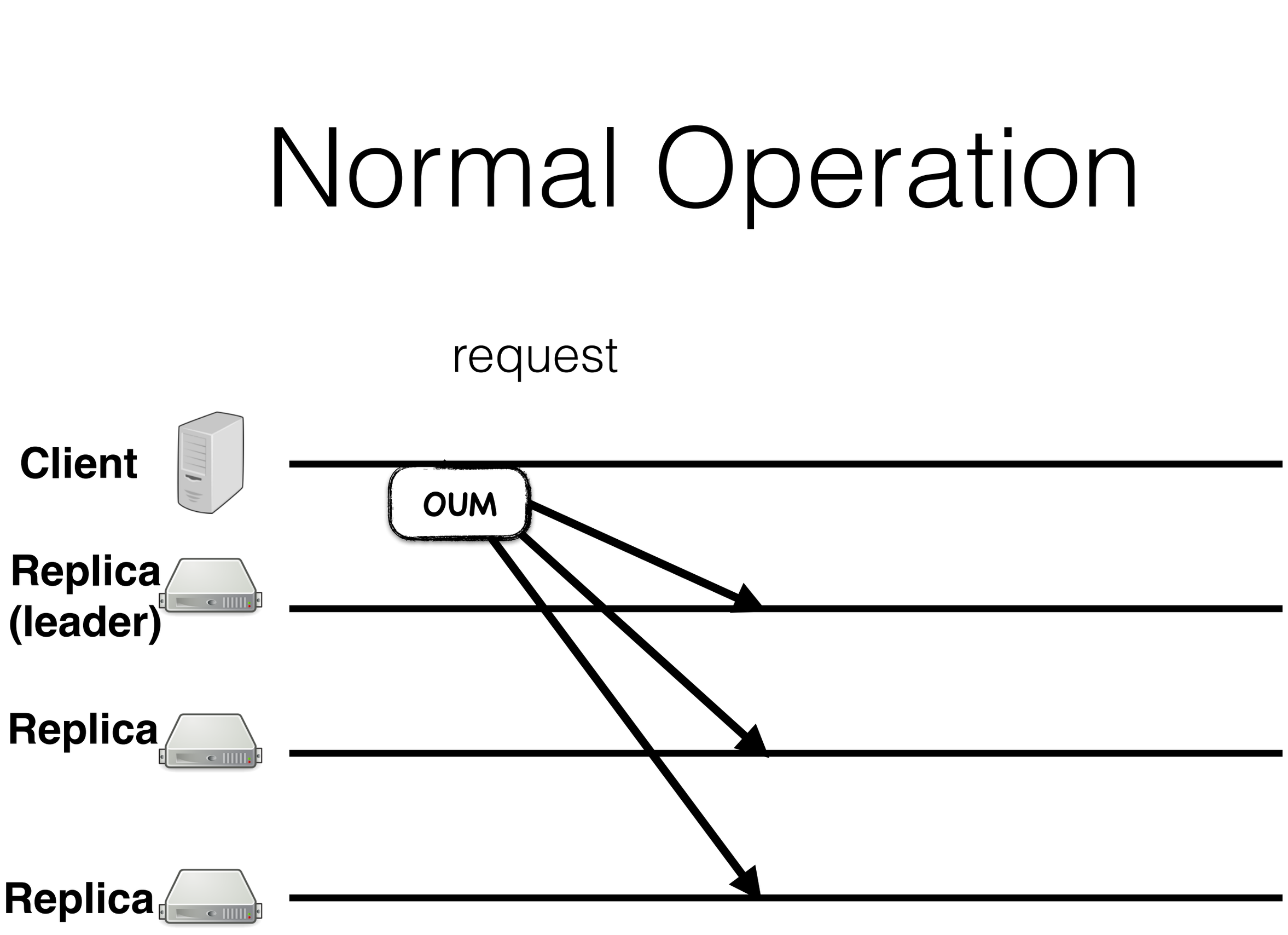
NOPaxos Overview

- Built on top of the guarantees of OUM
- Client requests are **totally ordered** but can be dropped
- **No coordination** in the common case
- Replicas run agreement on drop detection
- View change protocol for leader or sequencer failure

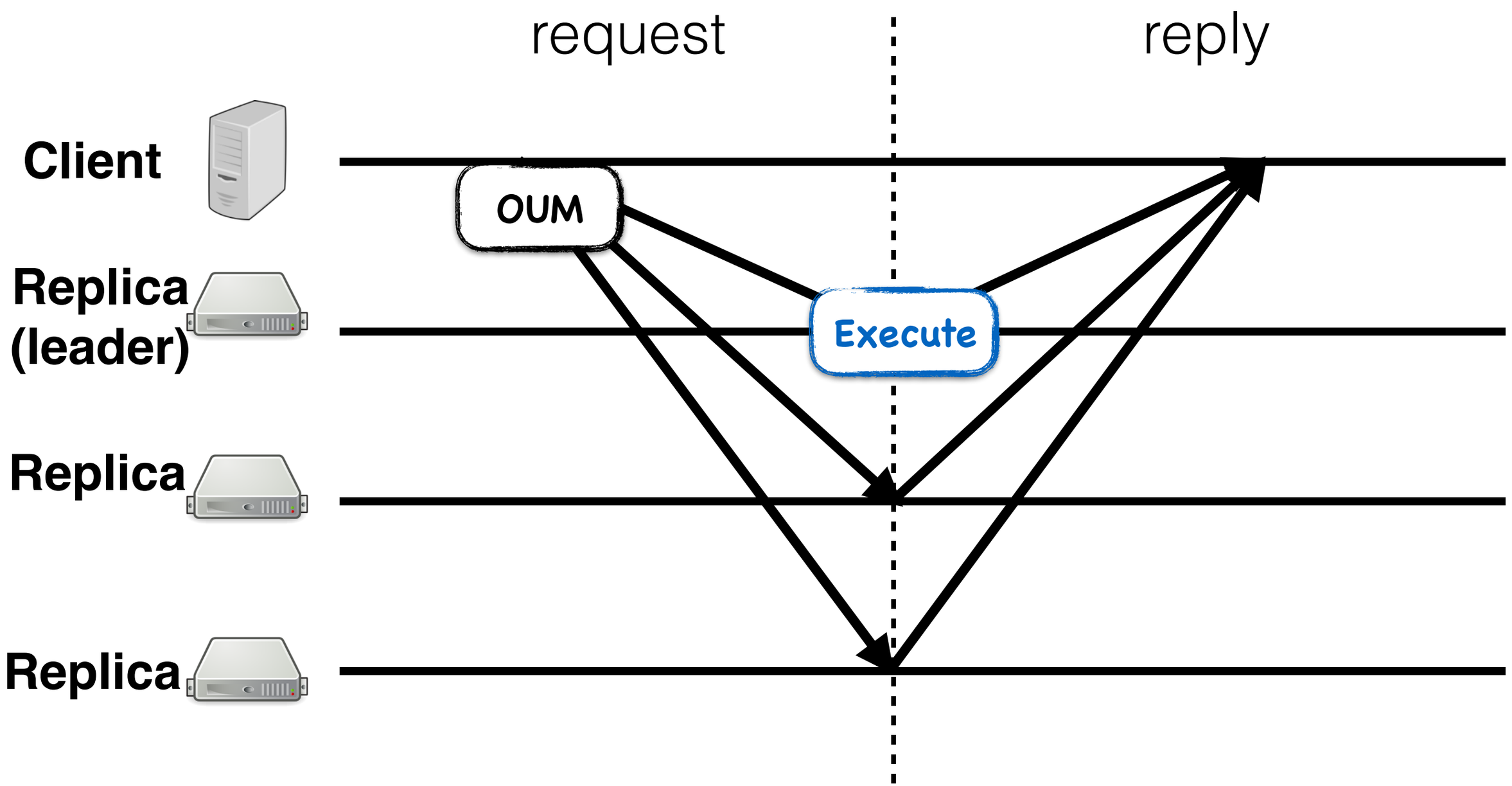
Normal Operation



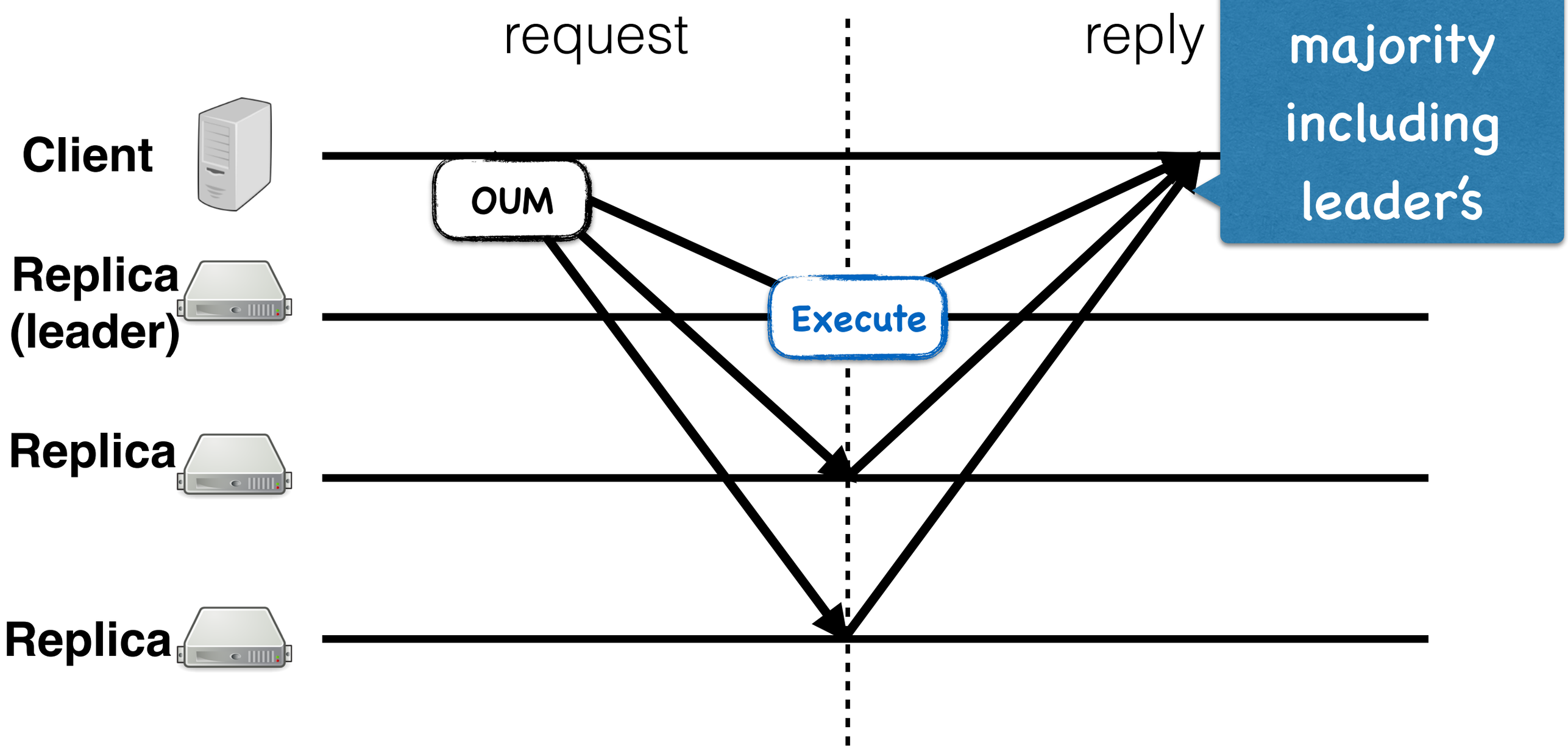
Normal Operation



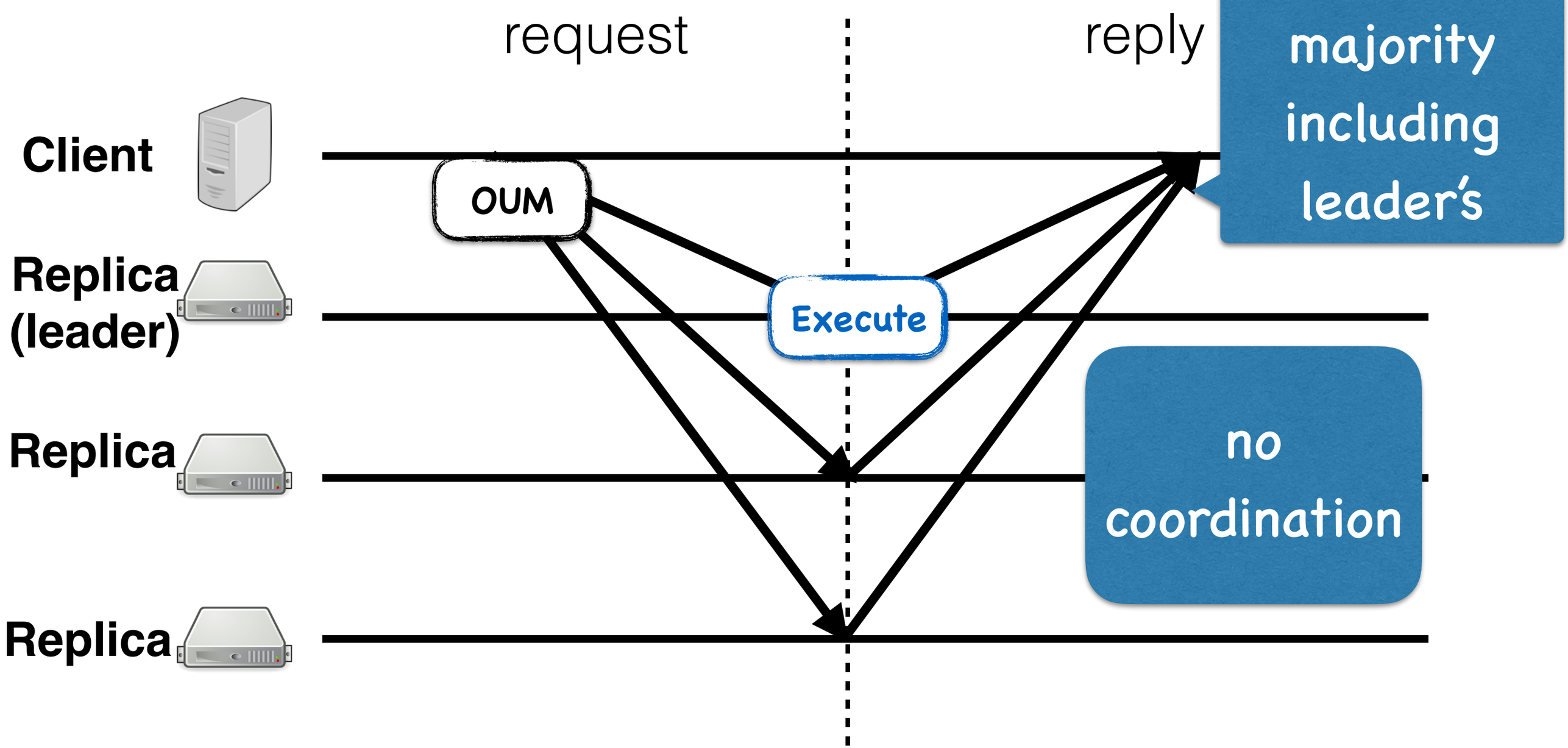
Normal Operation



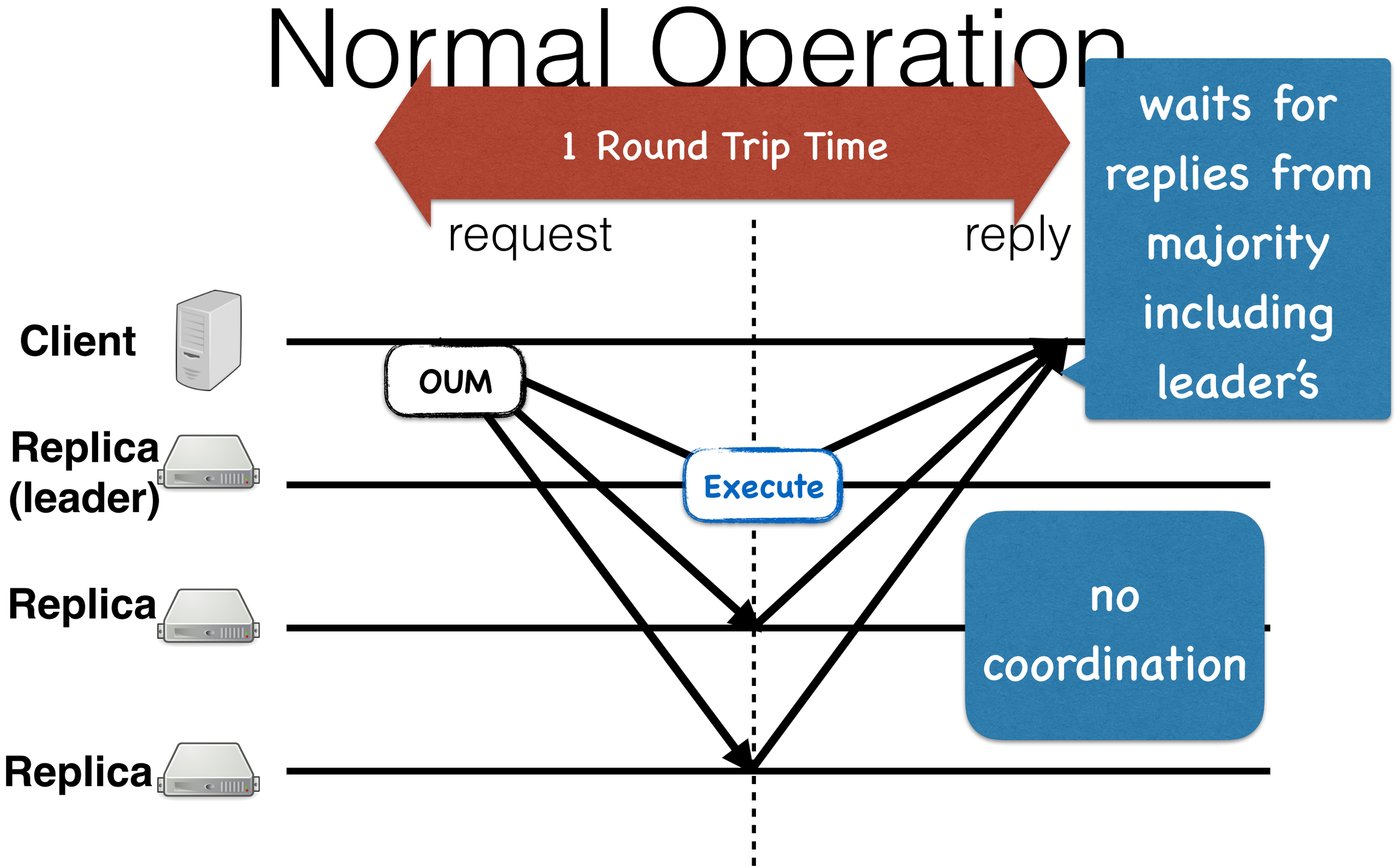
Normal Operation



Normal Operation



Normal Operation



Gap Agreement

Replicas detect message drops

- **Non-leader replicas:** recover the missing message from the leader
- **Leader replica:** coordinates to commit a NO-OP (Paxos)

- Efficient recovery from network anomalies

View Change

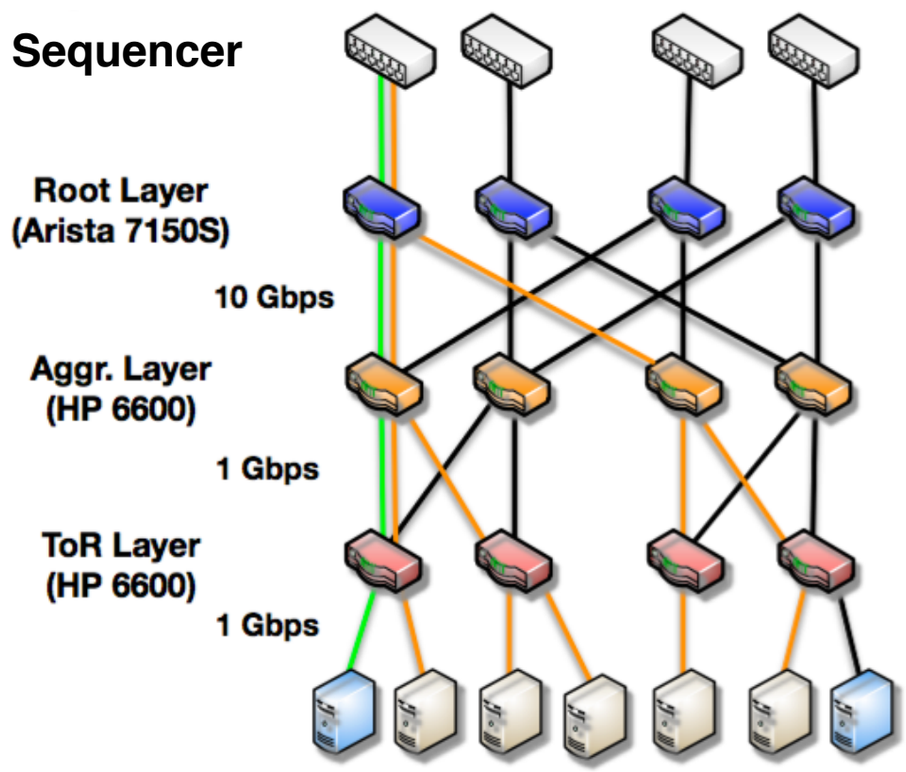
- Handles leader or sequencer failure
- Ensures that all replicas are in a **consistent state**
- Runs a view change protocol similar to VR
- *view-number* is a tuple of *<leader-number, session-number>*

Outline

1. Background on state machine replication and data center network
2. Ordered Unreliable Multicast
3. Network-Ordered Paxos
4. Evaluation

Evaluation Setup

- 3-level fat-tree network testbed
- 5 replicas with 2.5 GHz Intel Xeon E5-2680
- Middle box sequencer



NOPaxos achieves better throughput **and** latency

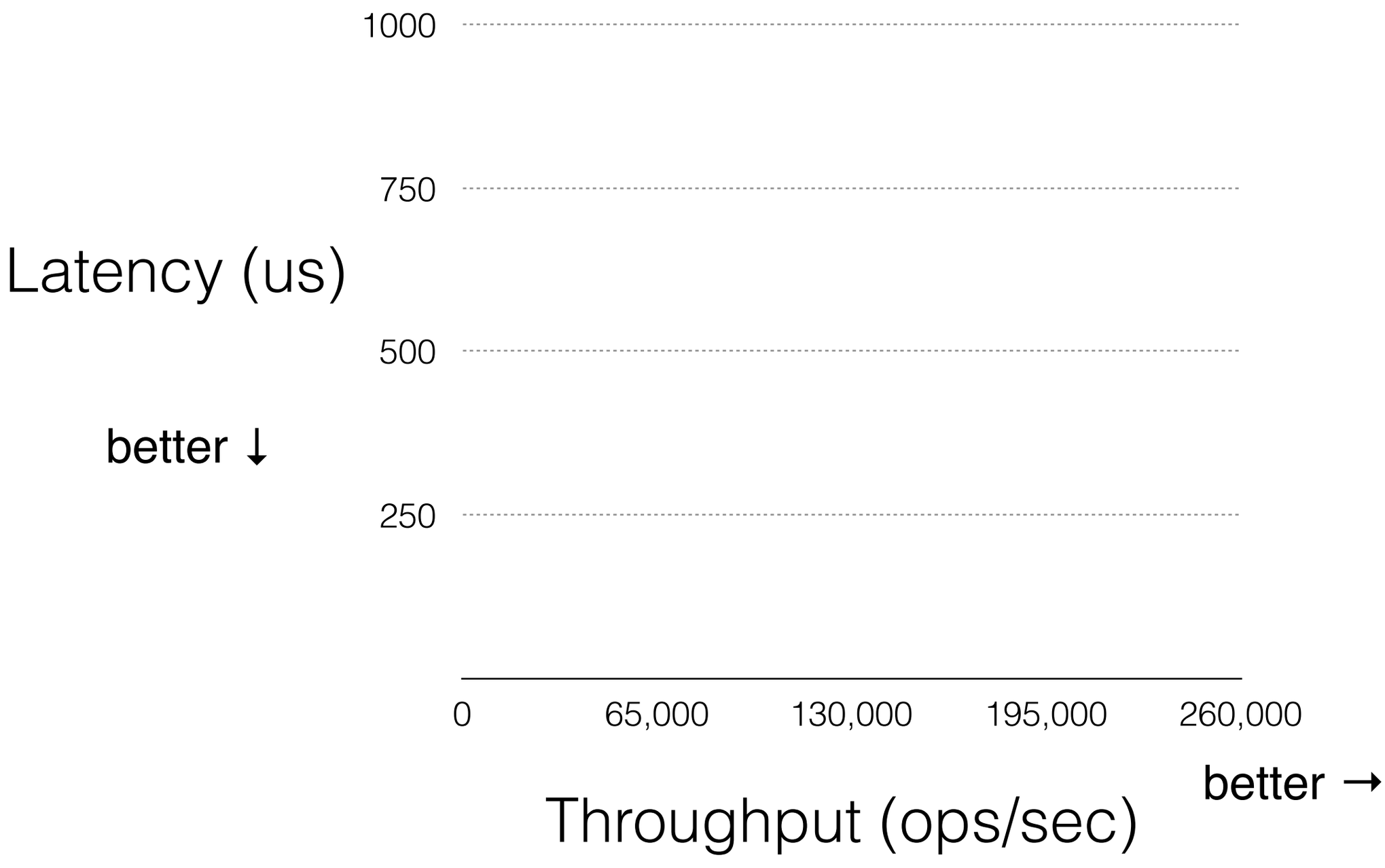
Latency (us)

better ↓

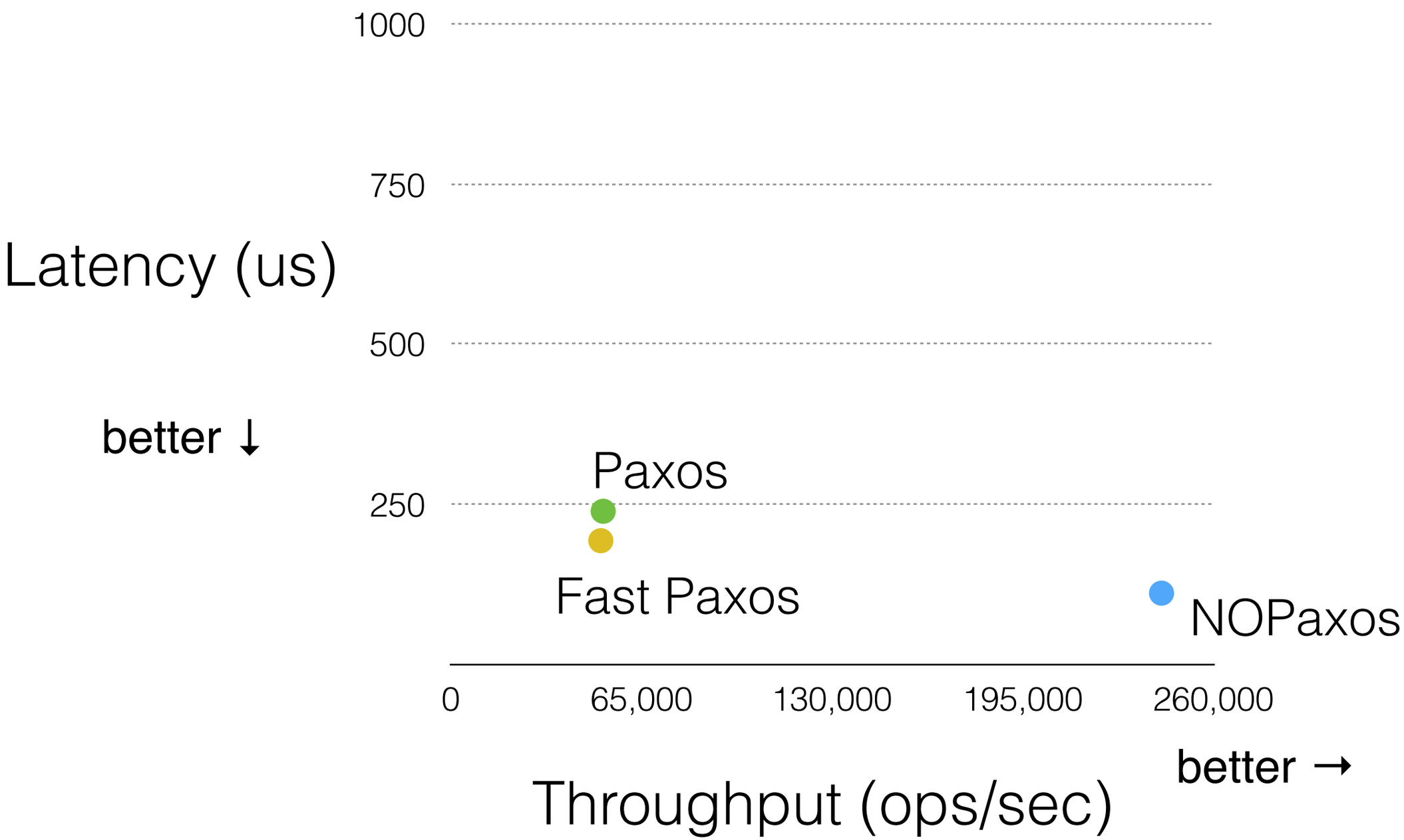
Throughput (ops/sec)

better →

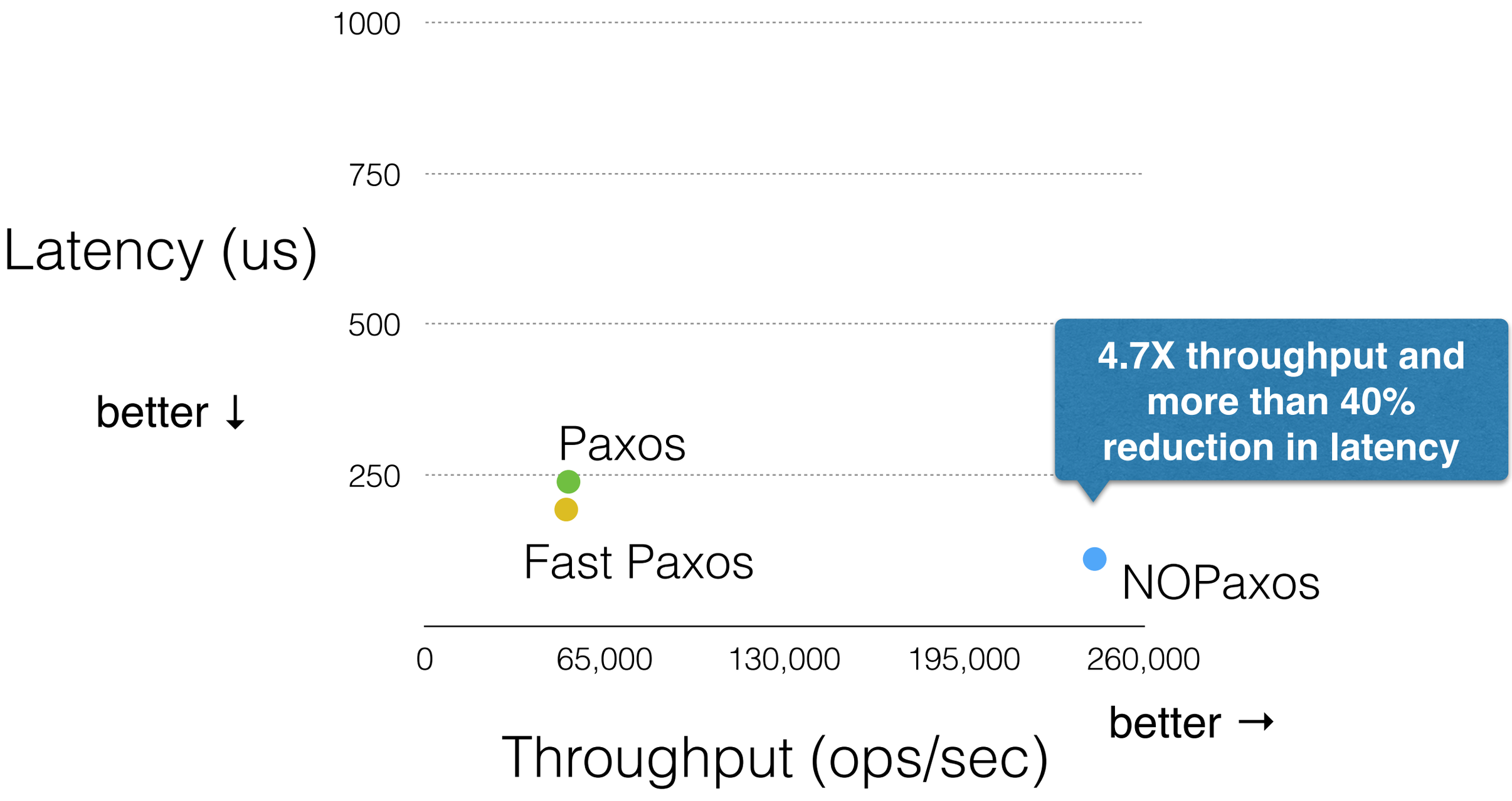
NOPaxos achieves better throughput **and** latency



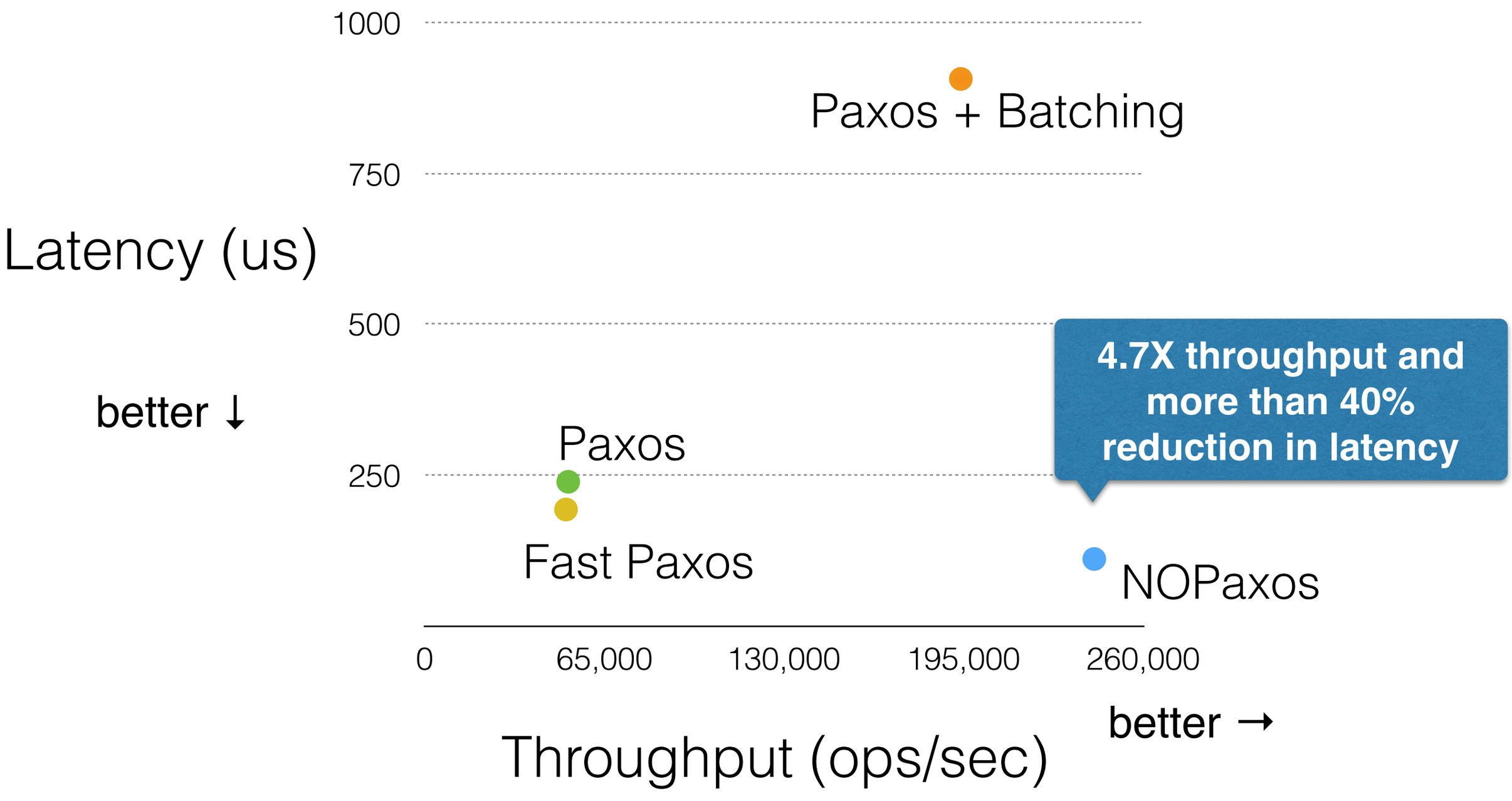
NOPaxos achieves better throughput **and** latency



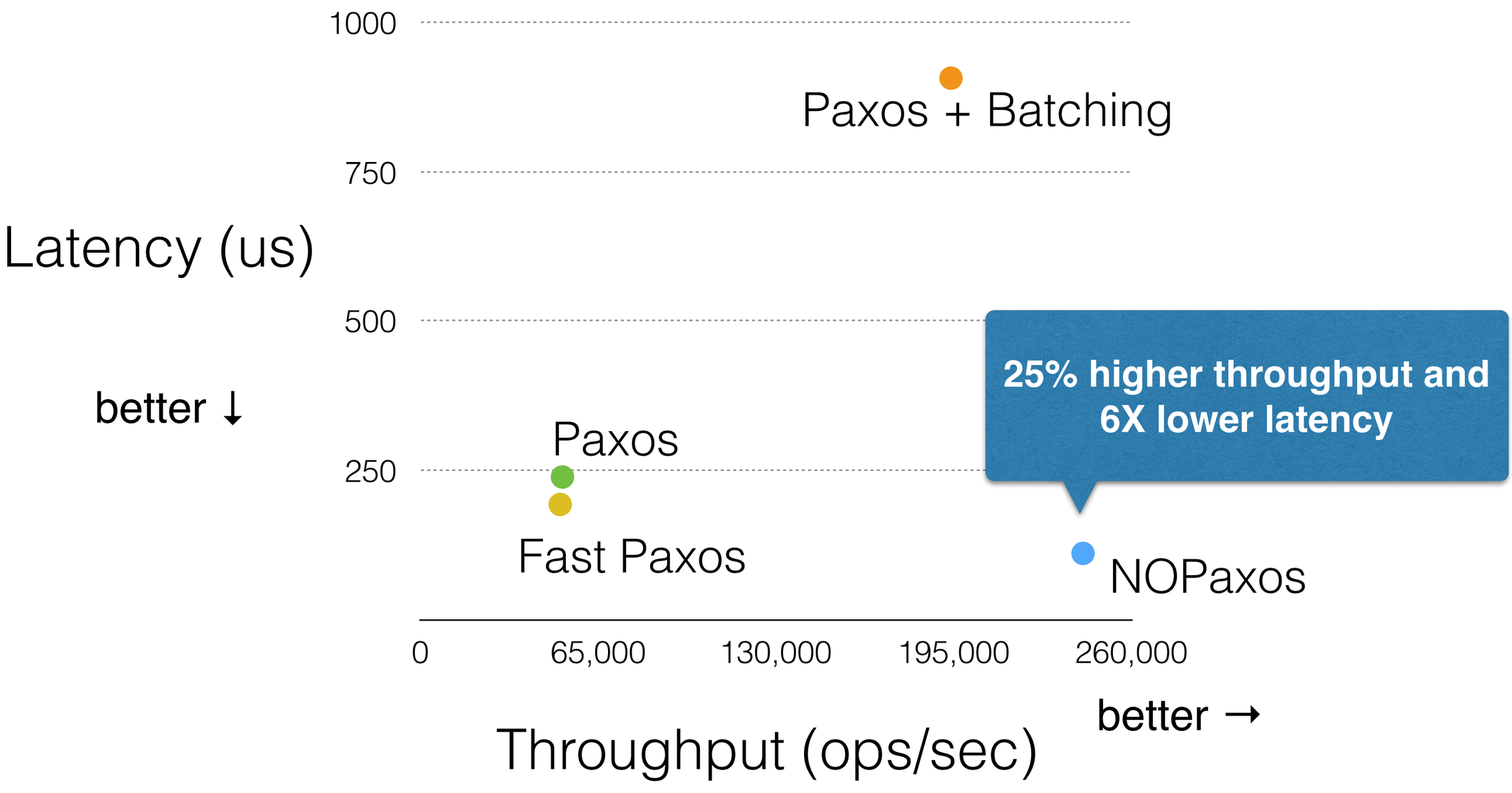
NOPaxos achieves better throughput **and** latency



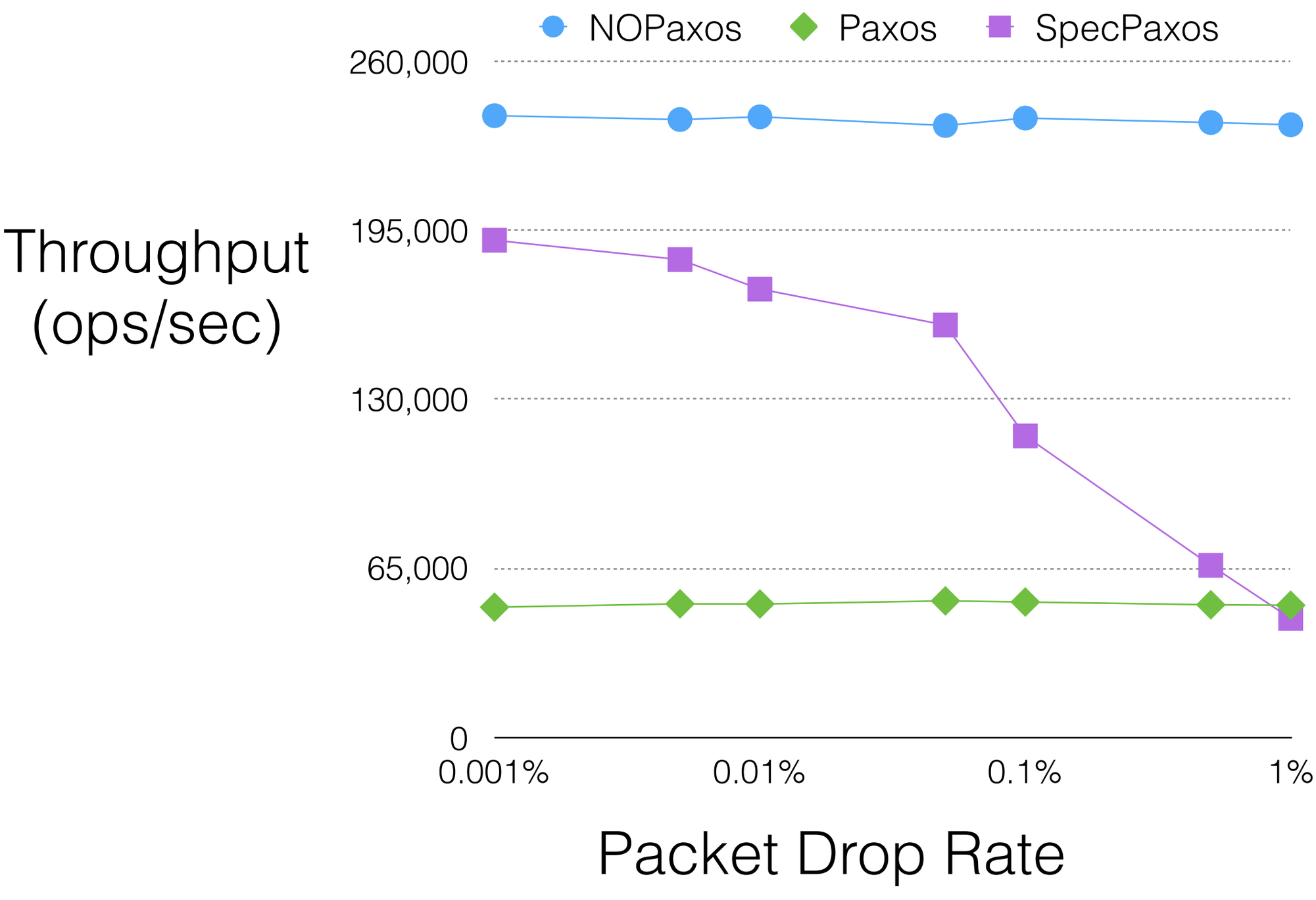
NOPaxos achieves better throughput **and** latency



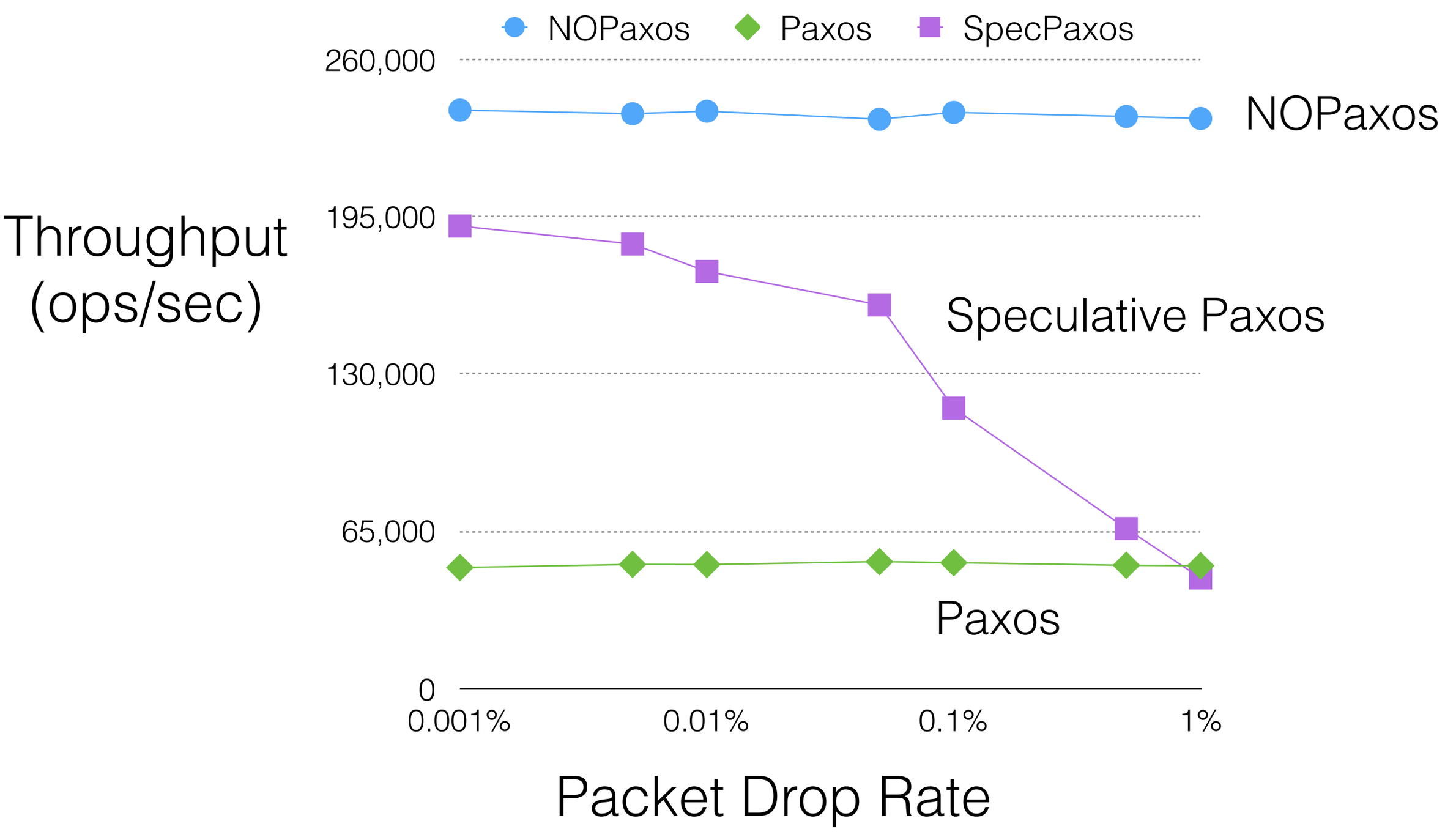
NOPaxos achieves better throughput **and** latency



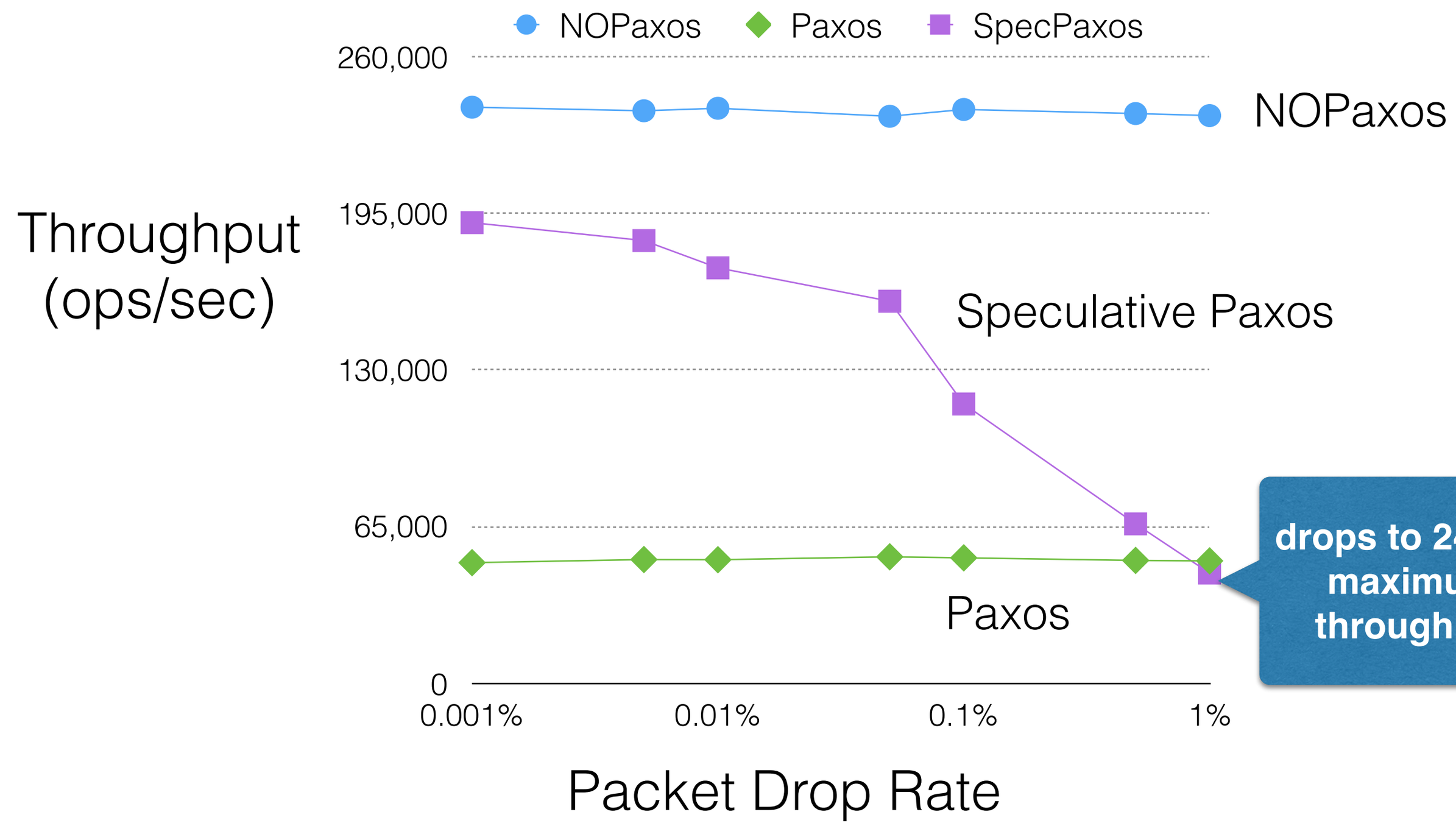
NOPaxos is resilient to network anomalies



NOPaxos is resilient to network anomalies

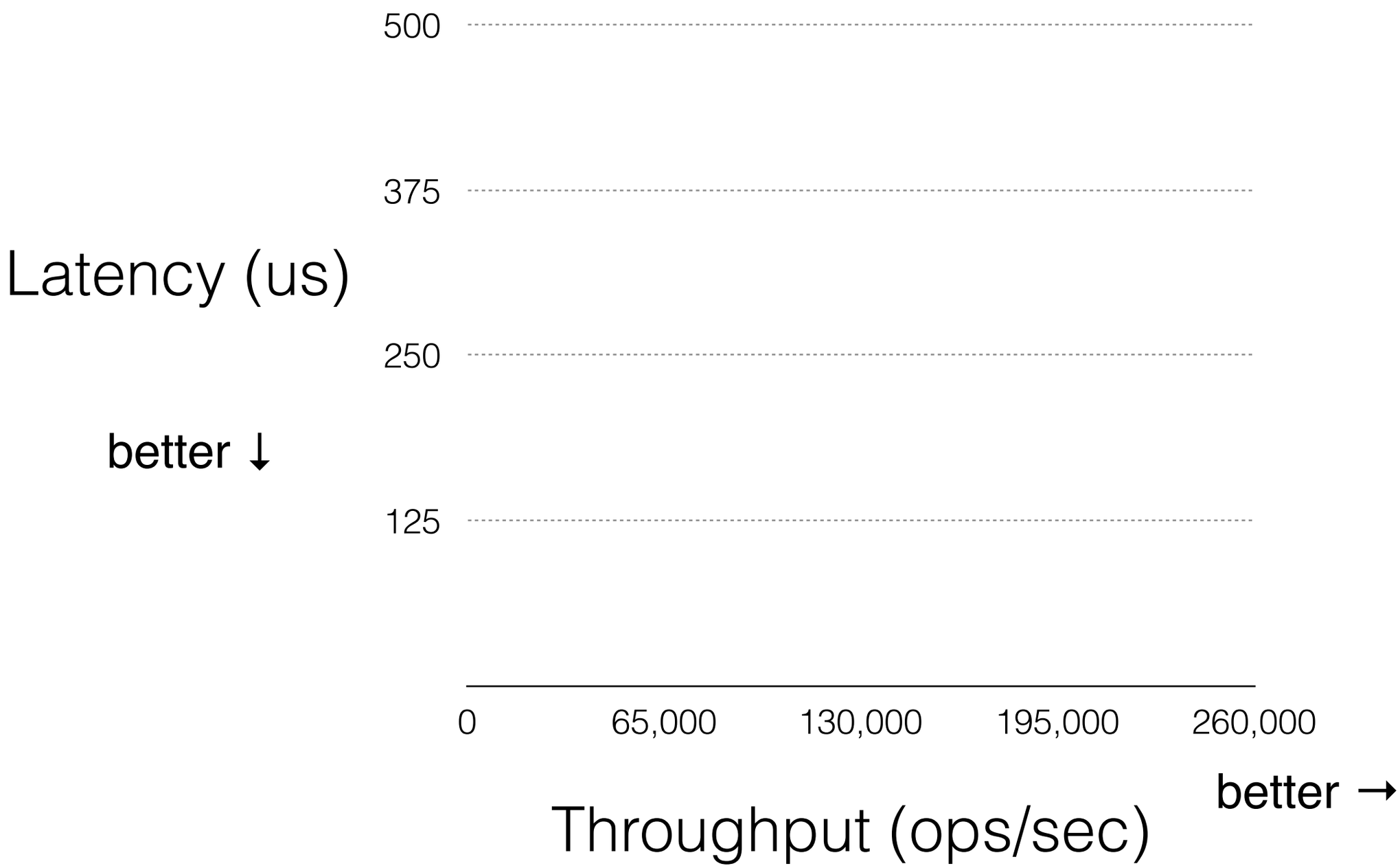


NOPaxos is resilient to network anomalies

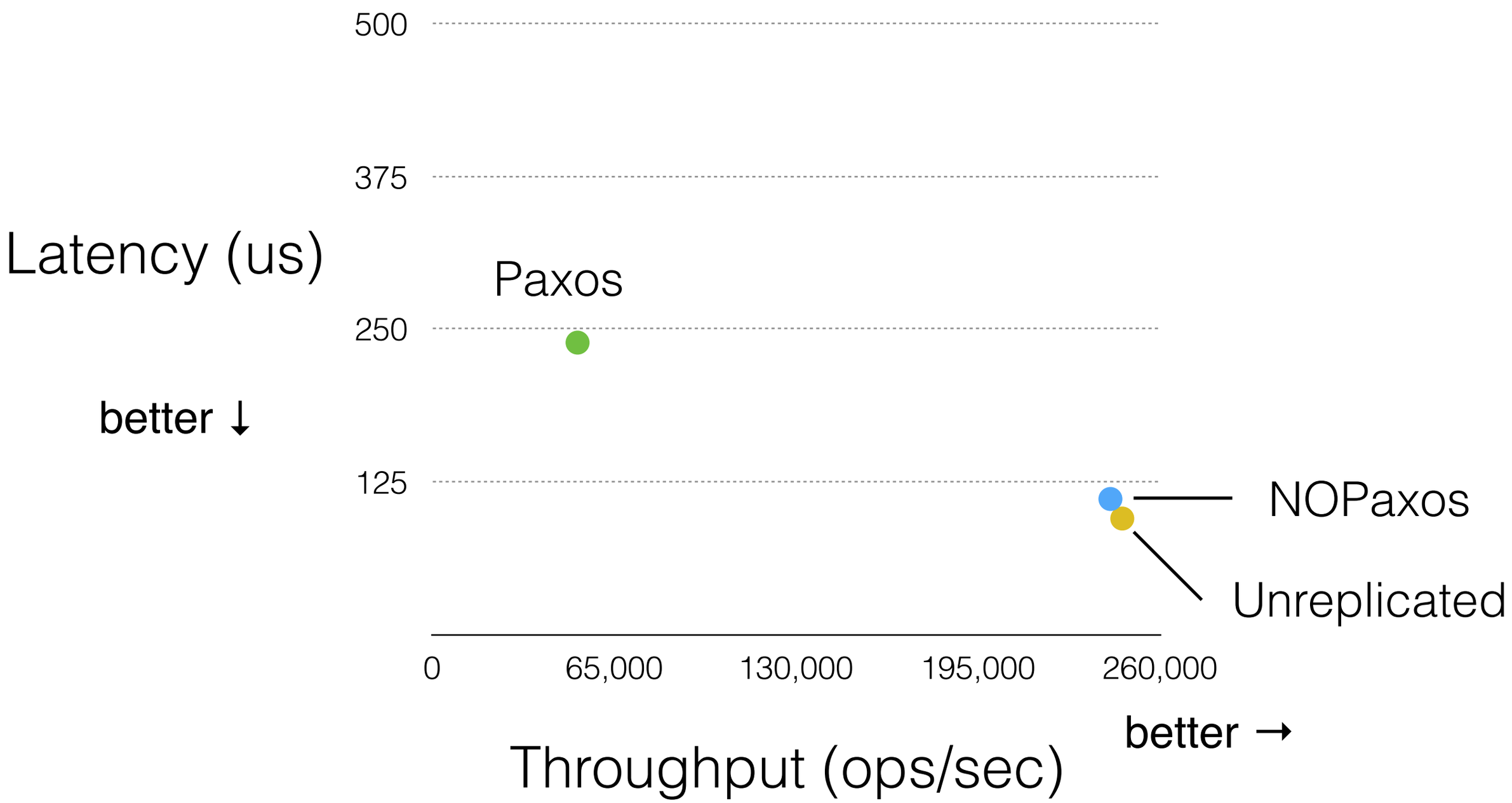


NOPaxos attains throughput within **2%** of an **unreplicated** system

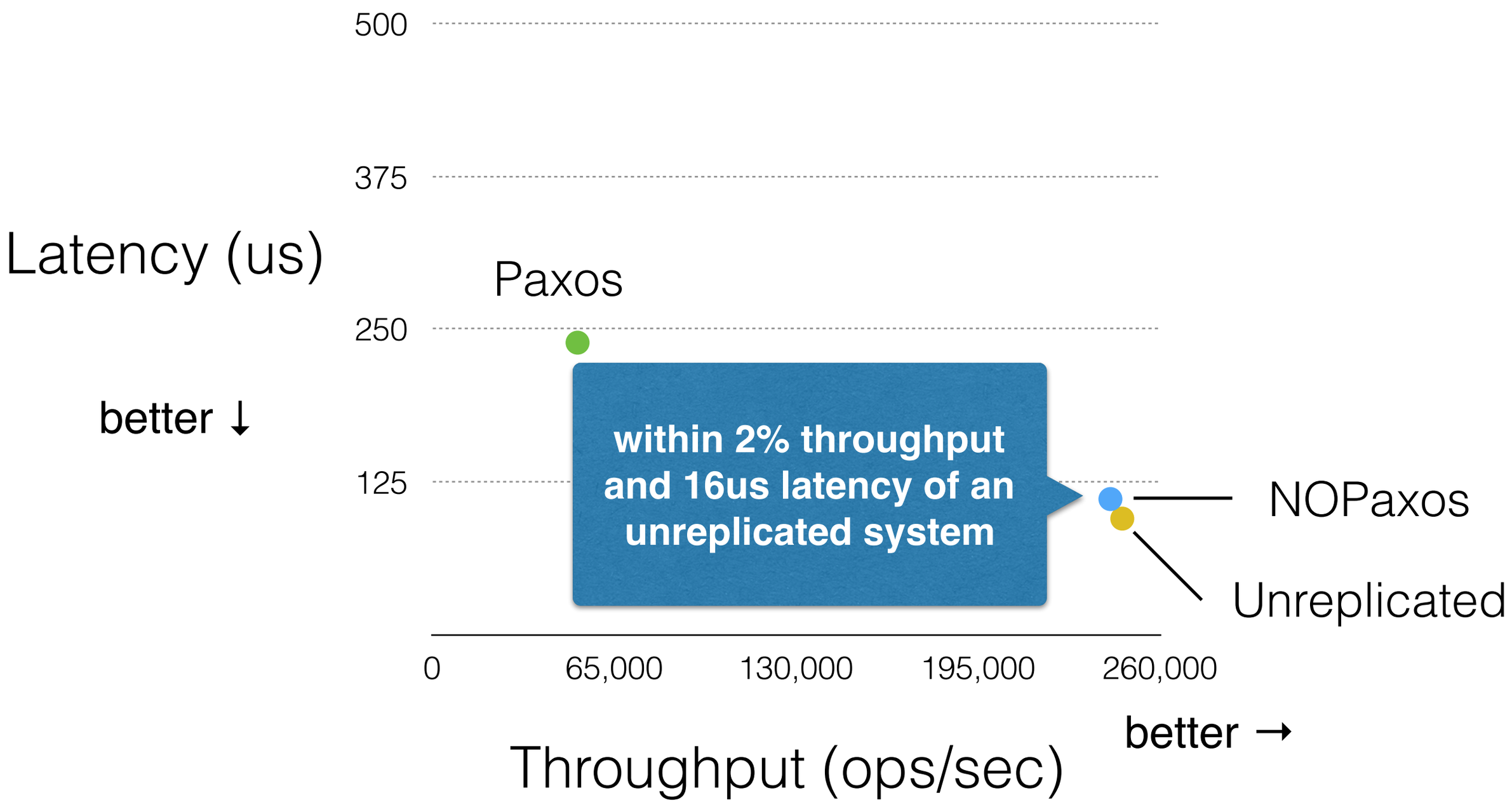
NOPaxos attains throughput within **2%** of an **unreplicated** system



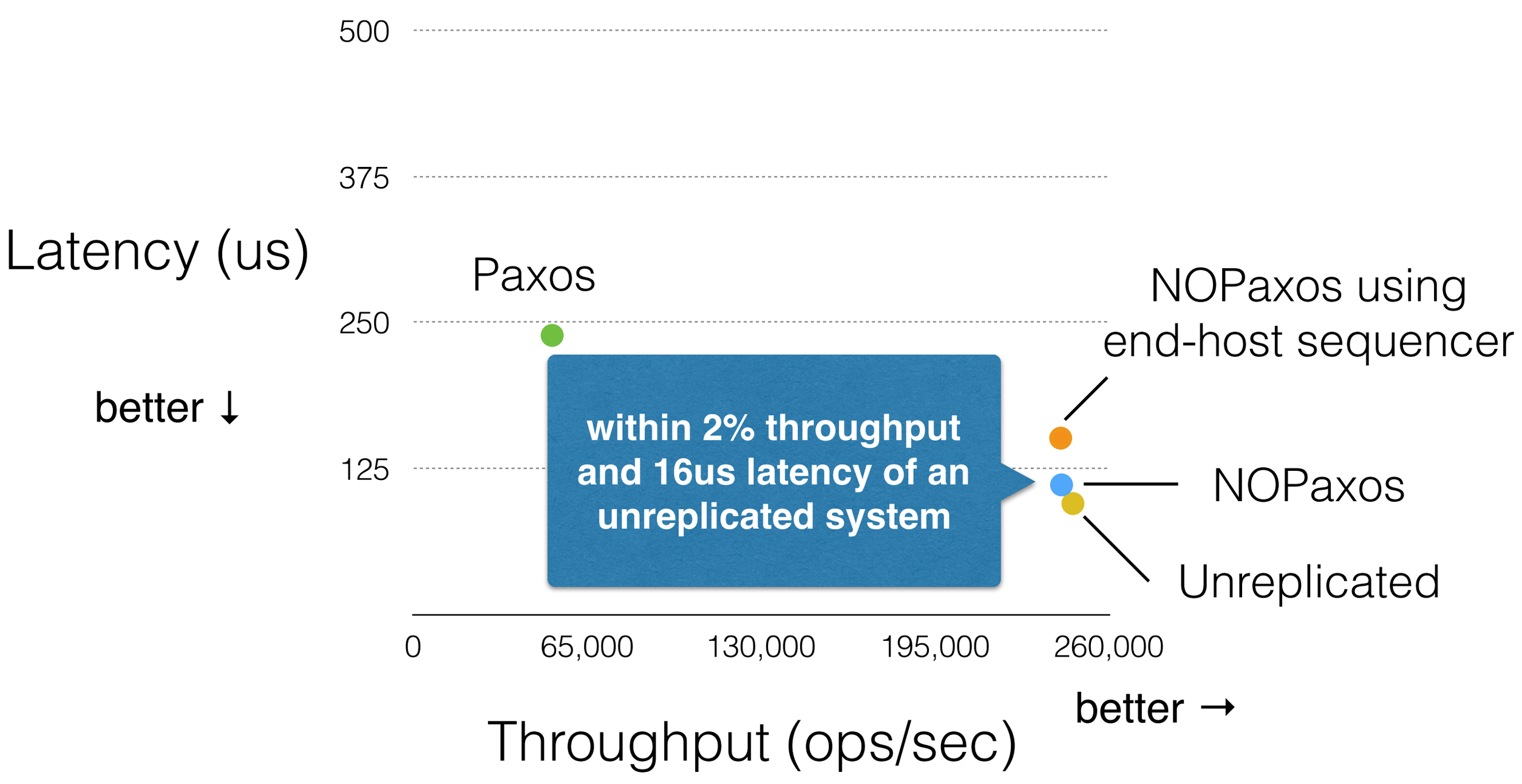
NOPaxos attains throughput within **2%** of an **unreplicated** system



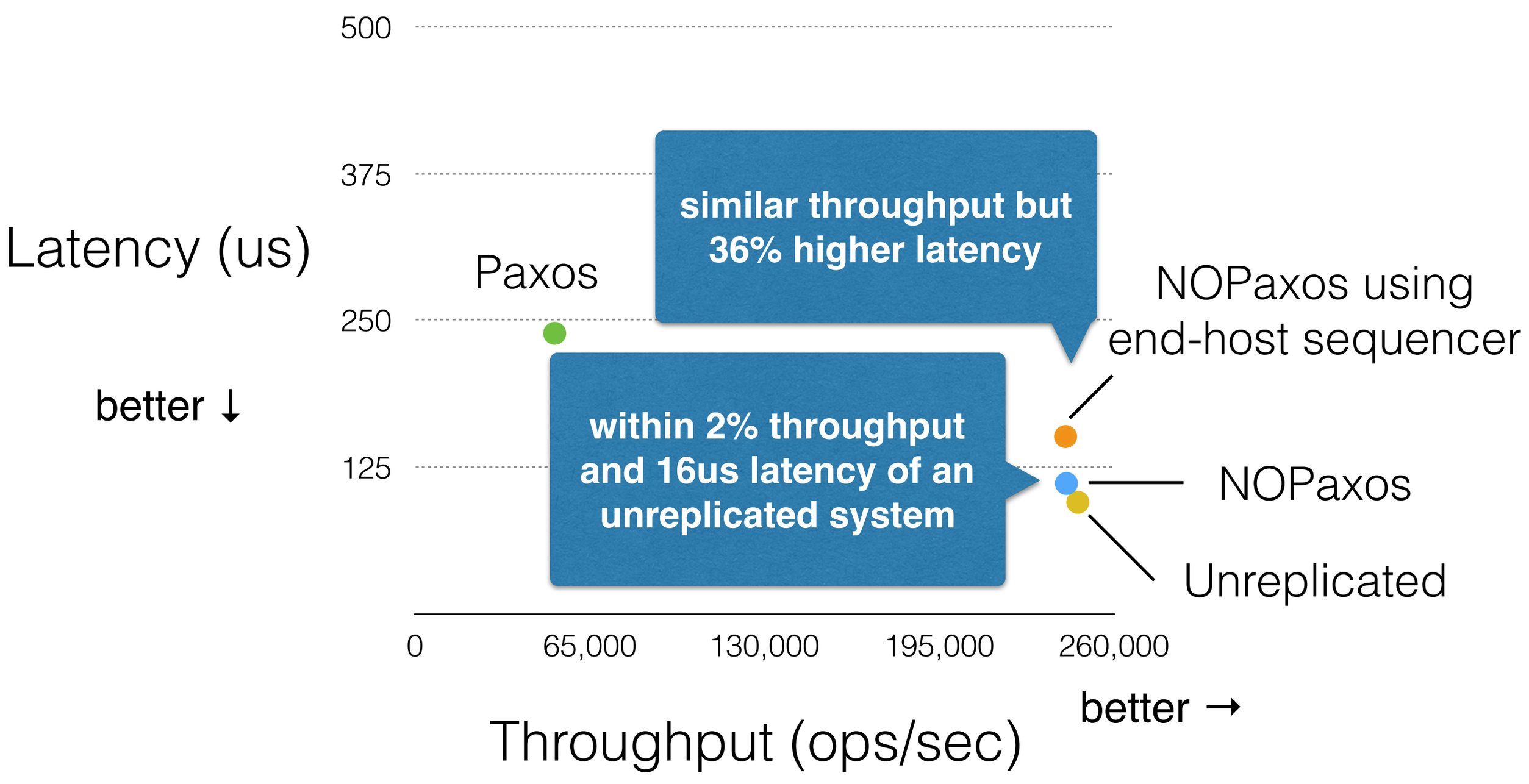
NOPaxos attains throughput within **2%** of an **unreplicated** system



NOPaxos attains throughput within **2%** of an **unreplicated** system



NOPaxos attains throughput within **2%** of an **unreplicated** system



Related Work

Group communication systems

- Virtual Synchrony [Birman, et al.], CATOCS [Cheriton, et al.], Amoeba [Kaashoek, et al.]

Consensus protocols

- Fast Paxos [Lamport], Optimistic Atomic Broadcast [Pedone, et al.], Speculative Paxos [Ports, et al.]
- Egalitarian Paxos [Moraru, et al.], Tapir [Zhang, et al.]

Network and Hardware support for distributed systems

- SwitchKV [Li, et al.], NetPaxos [Dang, et al.], FaRM [Dragojevic, et al.], Consensus in a Box [Istvan, et al.]

Summary

- Separate ordering from reliable delivery in state machine replication
- A new network model OUM that provides ordered but unreliable message delivery
- A more efficient replication protocol NOPaxos that ensures reliable delivery
- The combined system achieves performance equivalent to an unreplicated system