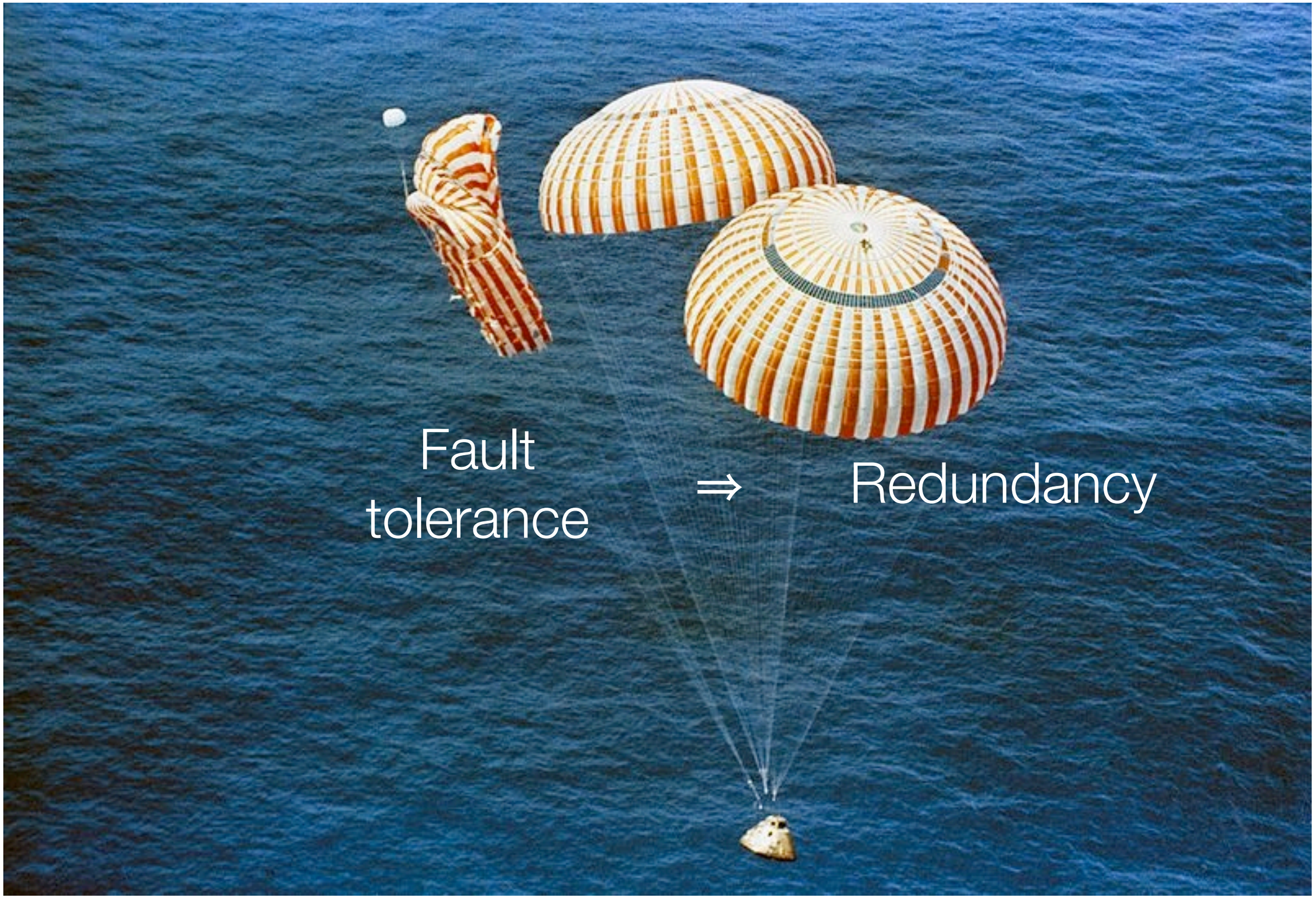
There Is More Consensus in Egalitarian Parliaments

Iulian Moraru, David Andersen, Michael Kaminsky

Carnegie Mellon University Intel Labs


Fault tolerance
⇒
Redundancy

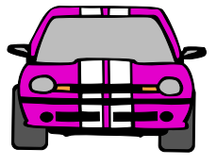
State Machine Replication

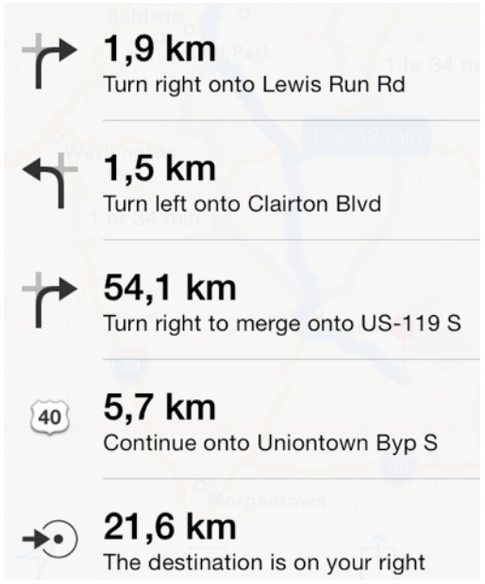
State Machine Replication

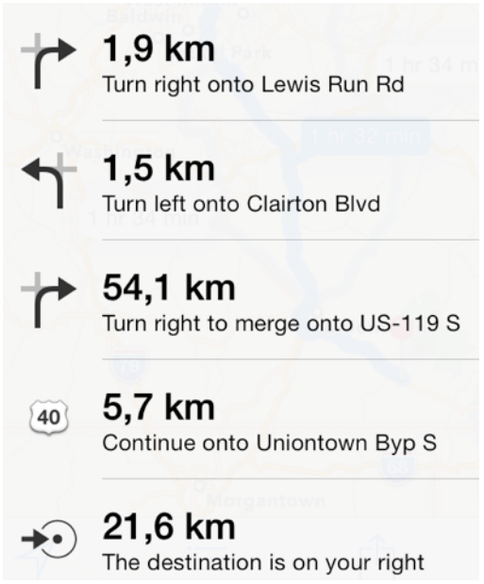
Execute the same commands in the same order

State Machine Replication

Execute the same commands in the same order

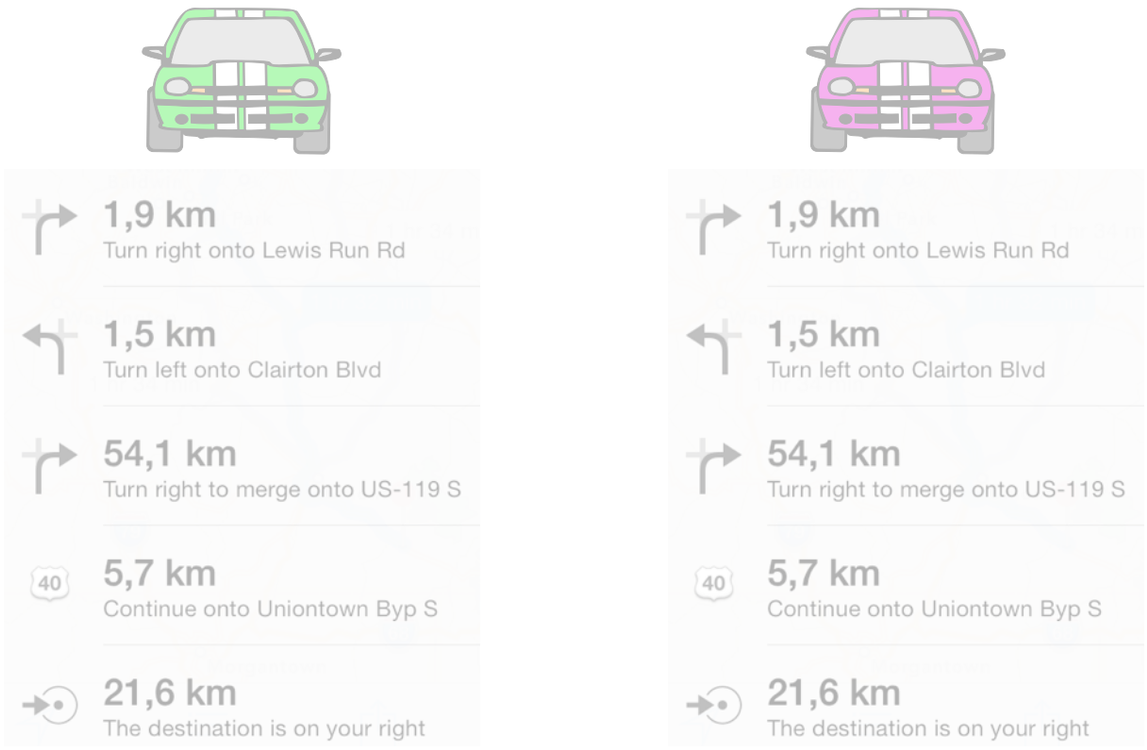






State Machine Replication

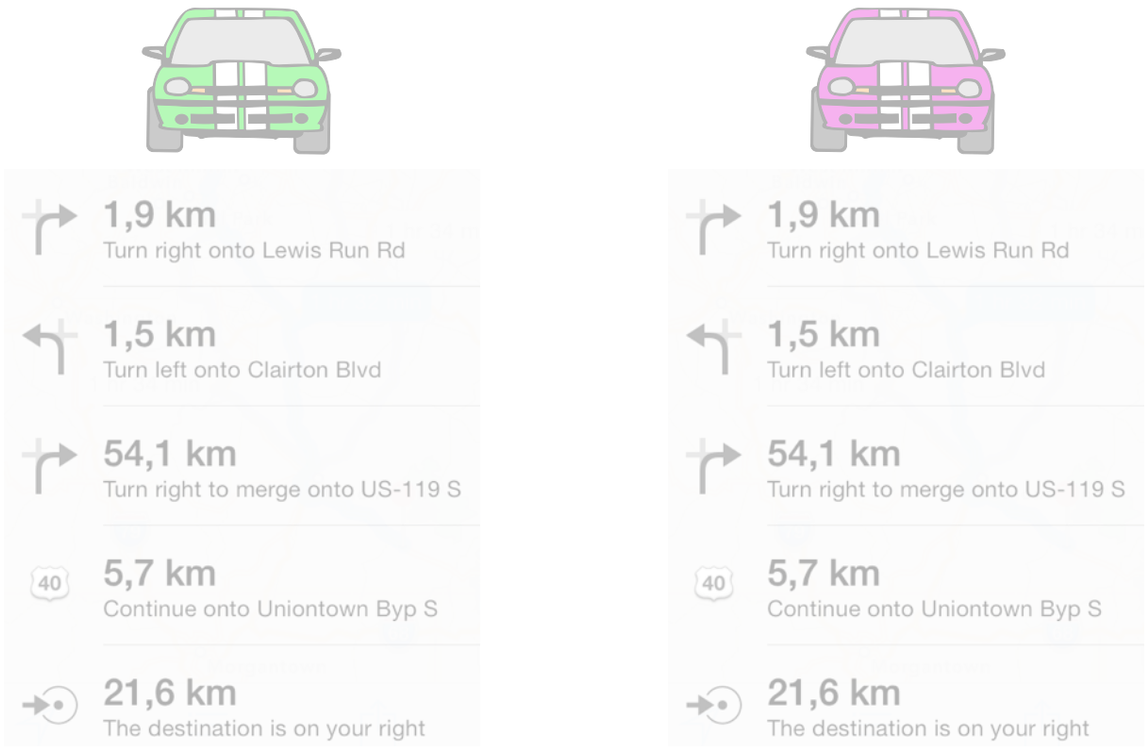
Execute the same commands in the same order



Paxos

State Machine Replication

Execute the same commands in the same order

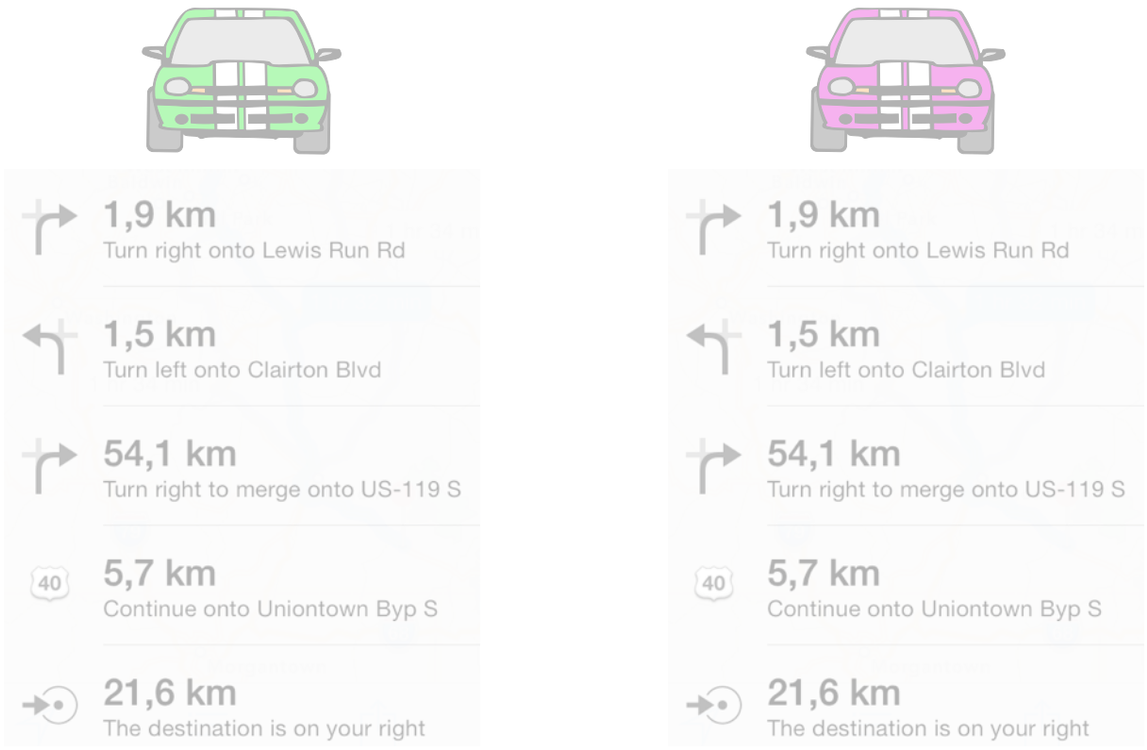


Paxos

- No external failure detector required

State Machine Replication

Execute the same commands in the same order



Paxos

- No external failure detector required
- Fast fail-over (high availability)

Paxos is important in clusters

Chubby, Boxwood, SMARTER, ZooKeeper

- Synchronization
- Resource discovery
- Data replication

High throughput
High availability

Paxos is important in the wide-area

Spanner, Megastore

- Bring data closer to clients
- Tolerate datacenter outages

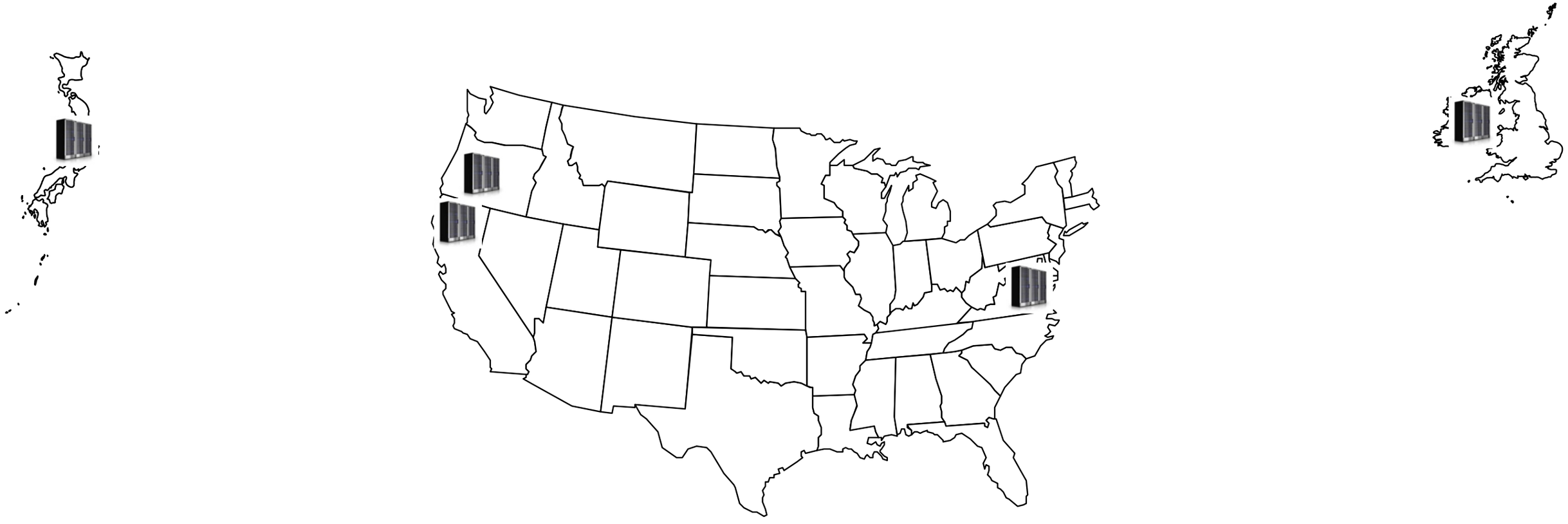
Low latency

Paxos is important in the wide-area

Spanner, Megastore

- Bring data closer to clients
- Tolerate datacenter outages

Low latency

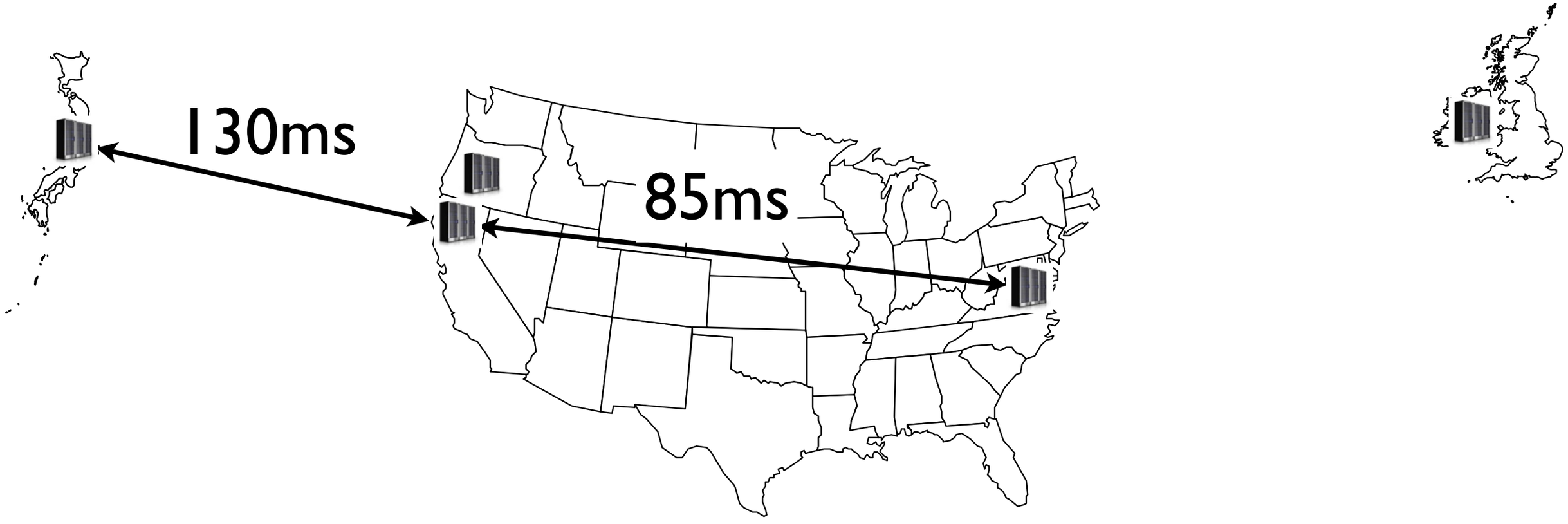


Paxos is important in the wide-area

Spanner, Megastore

- Bring data closer to clients
- Tolerate datacenter outages

Low latency



Paxos overview

Paxos overview

- Agreement protocol

Paxos overview

- Agreement protocol
- Tolerates F failures with 2F+1 replicas (optimal)
  - No external failure detector required

Paxos overview

- Agreement protocol
- Tolerates F failures with 2F+1 replicas (optimal)
  - No external failure detector required
- Replicas can fail by crashing (non-Byzantine)

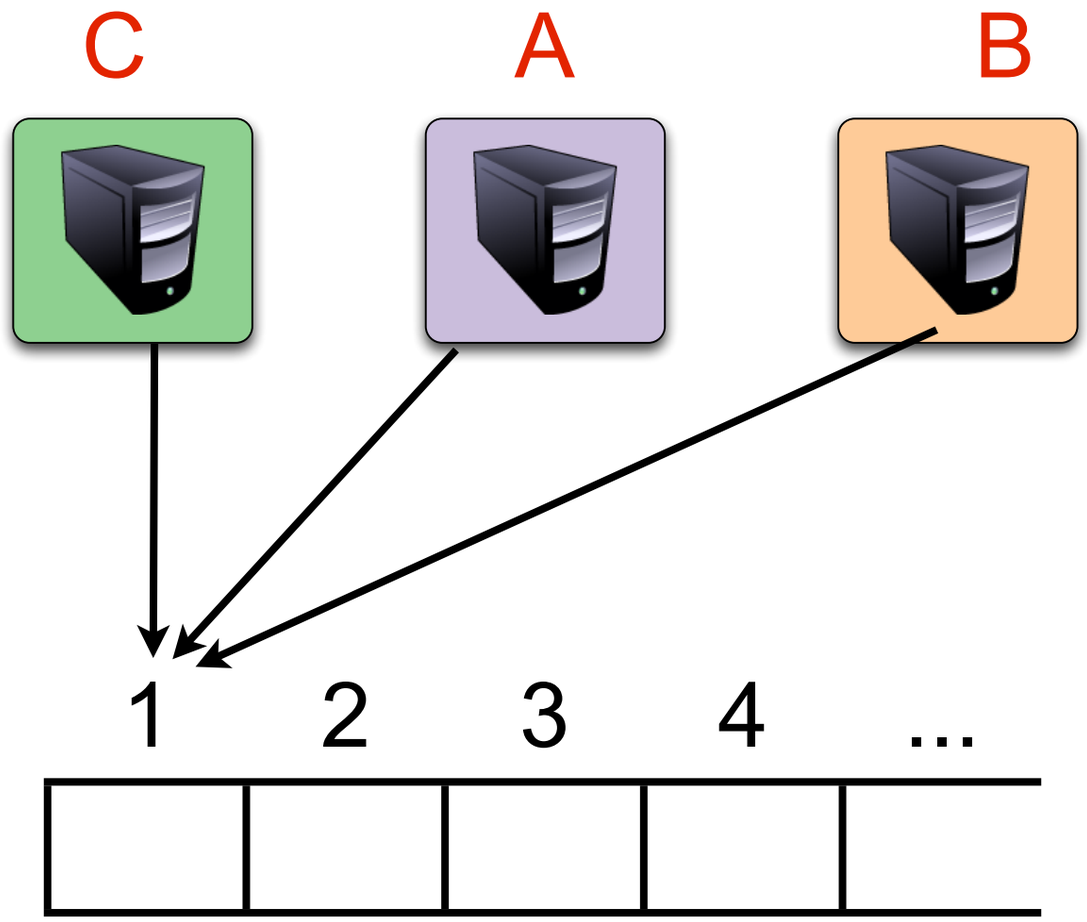
Paxos overview

- Agreement protocol
- Tolerates F failures with 2F+1 replicas (optimal)
  - No external failure detector required
- Replicas can fail by crashing (non-Byzantine)
- Asynchronous communication

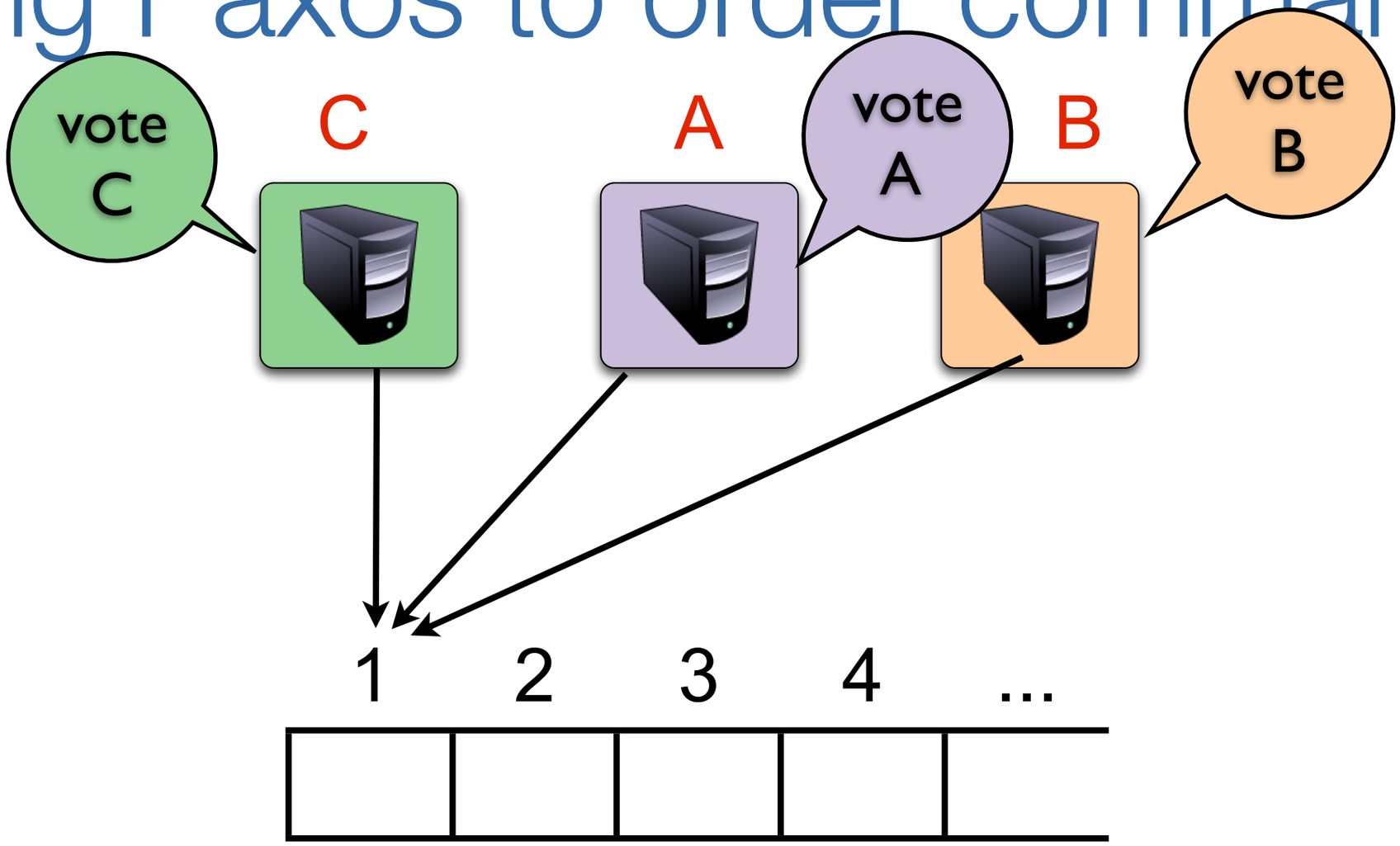
Using Paxos to order commands

Using Paxos to order commands

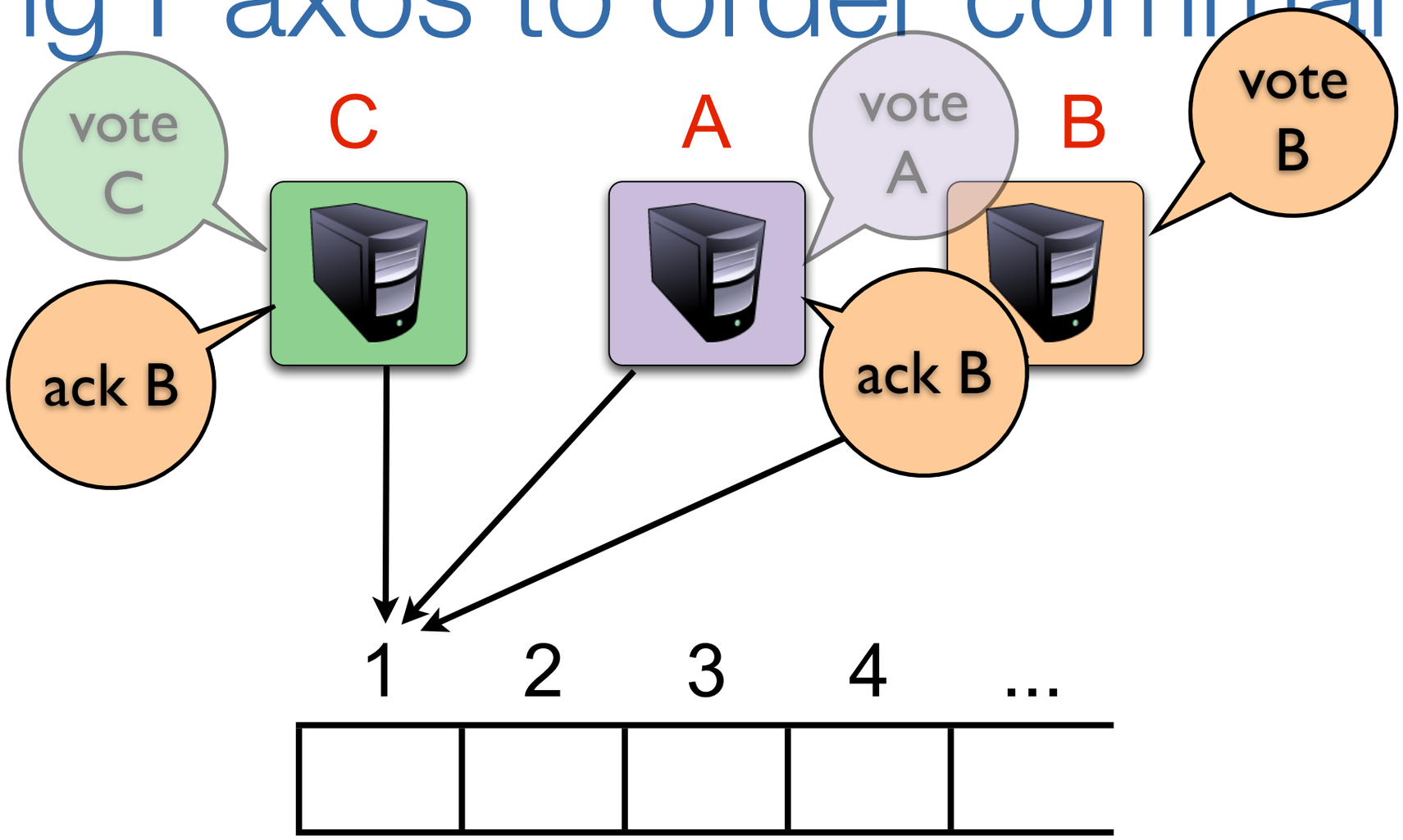
Using Paxos to order commands



Using Paxos to order commands

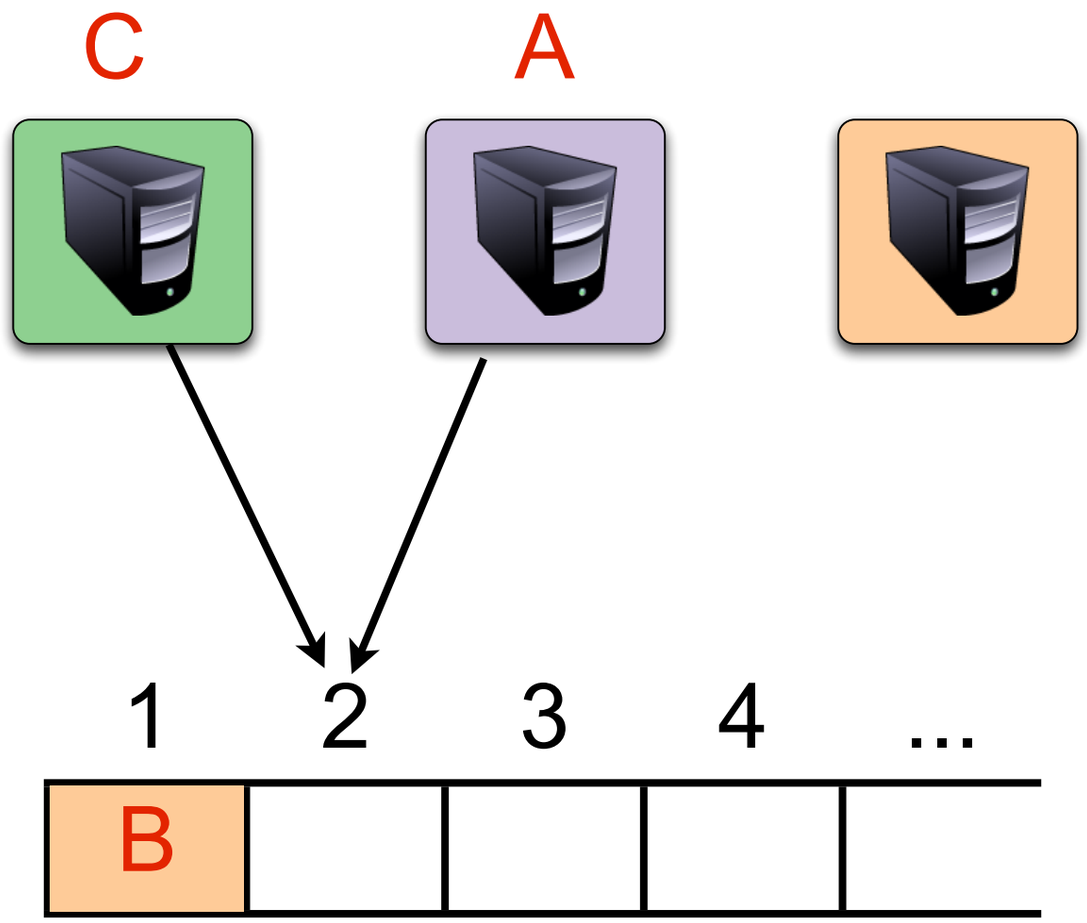


Using Paxos to order commands



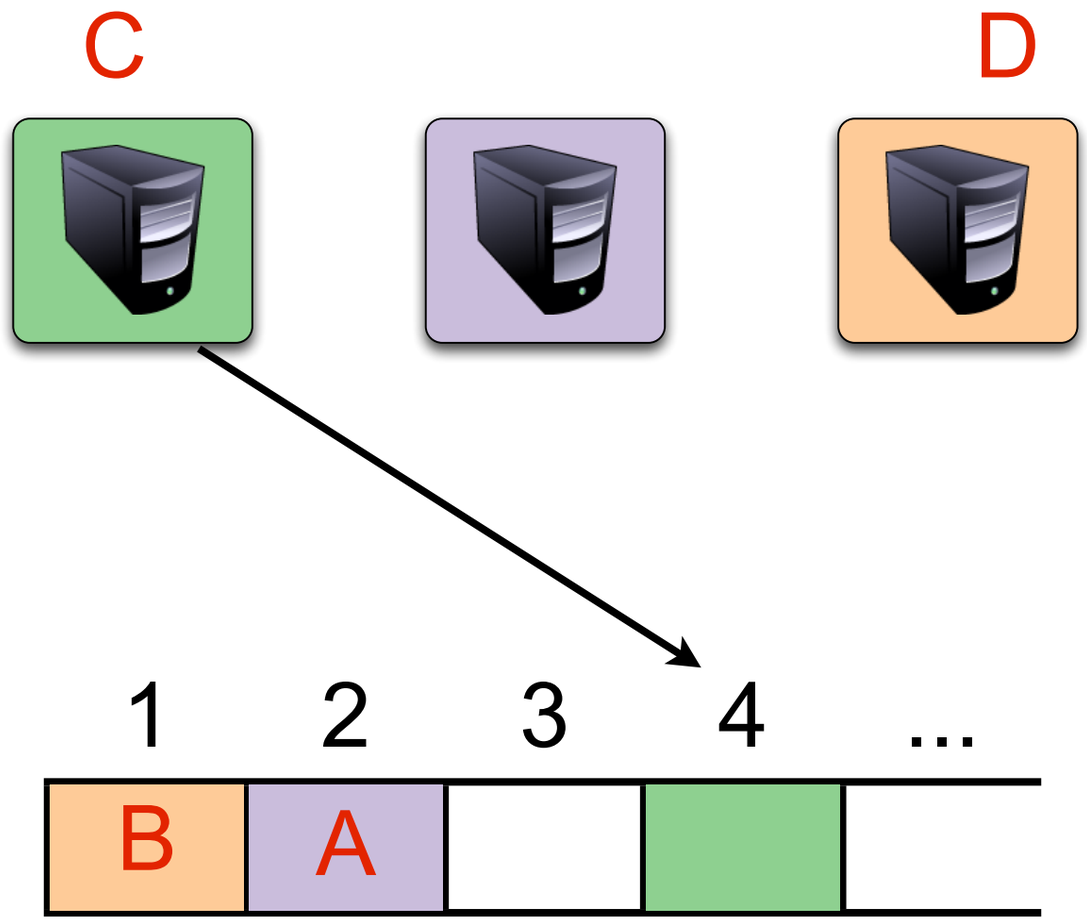
Using Paxos to order commands

Using Paxos to order commands

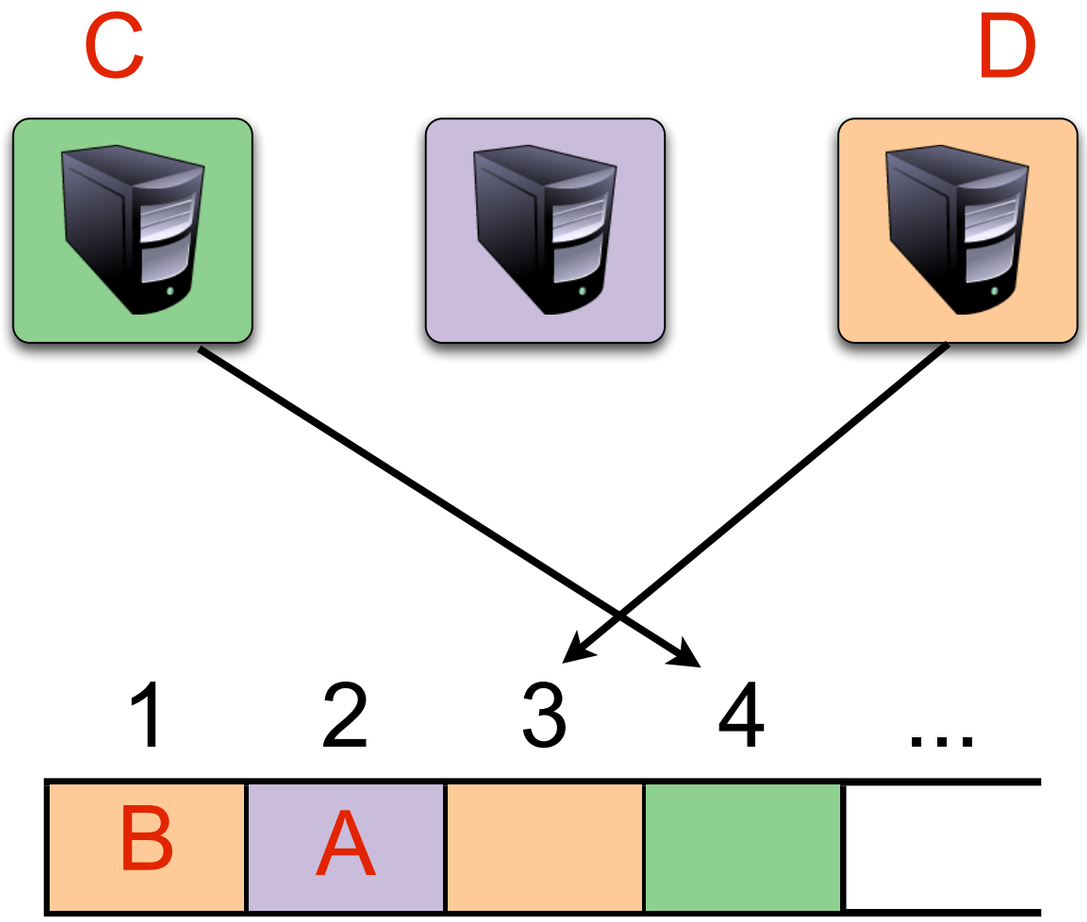


Using Paxos to order commands

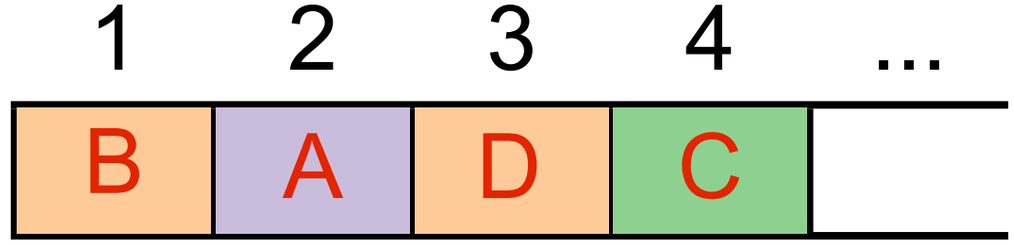
Using Paxos to order commands



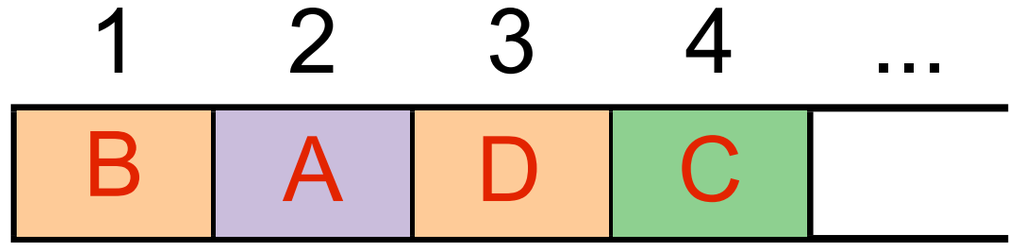
Using Paxos to order commands



Using Paxos to order commands

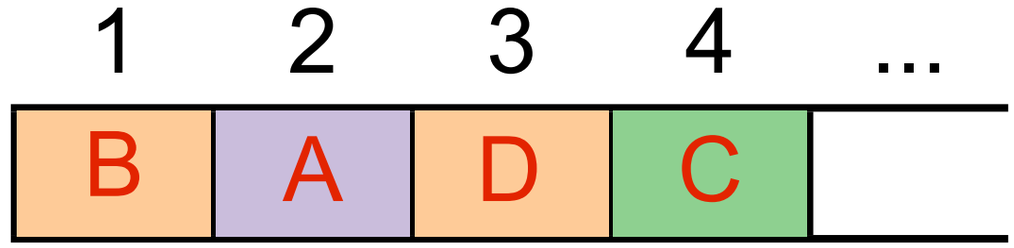


Using Paxos to order commands



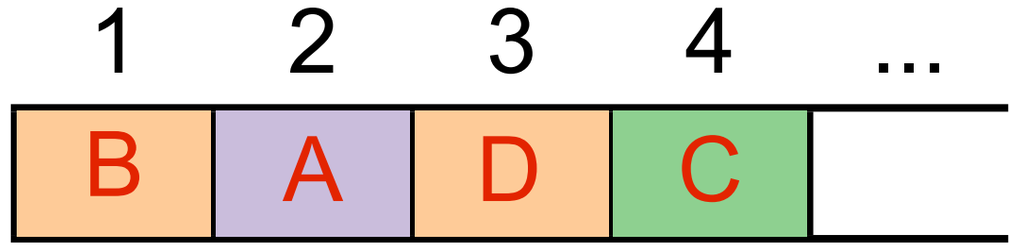
- Choose commands *independently* for each slot

Using Paxos to order commands



- Choose commands *independently* for each slot
- At least 2 RTTs per slot:

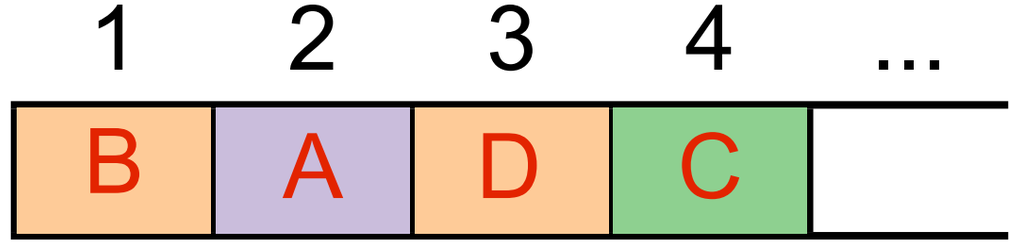
Using Paxos to order commands



- Choose commands *independently* for each slot
- At least 2 RTTs per slot:

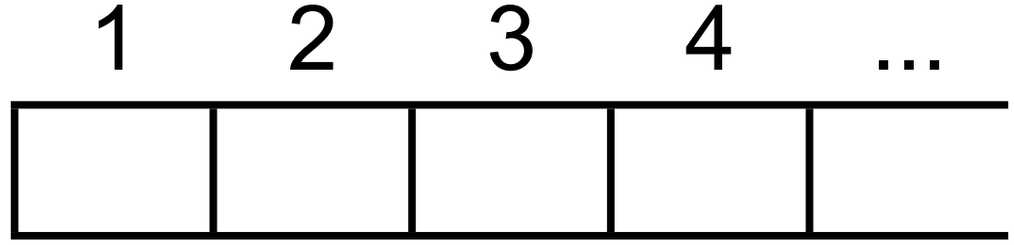
1. Take ownership of a slot

Using Paxos to order commands

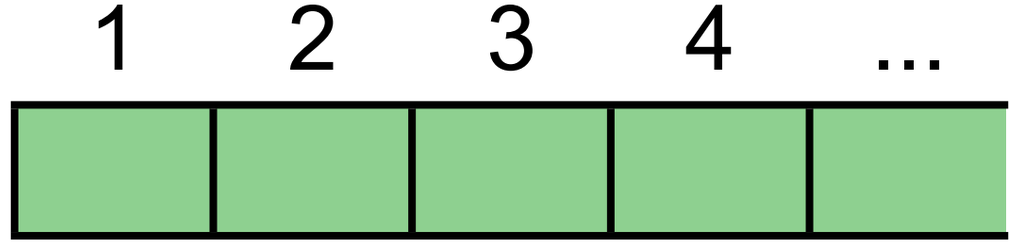


- Choose commands *independently* for each slot
- At least 2 RTTs per slot:
  1. Take ownership of a slot
  2. Propose command

Multi-Paxos

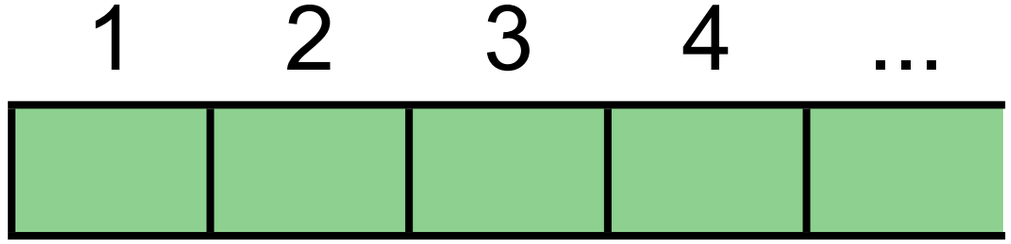


Multi-Paxos

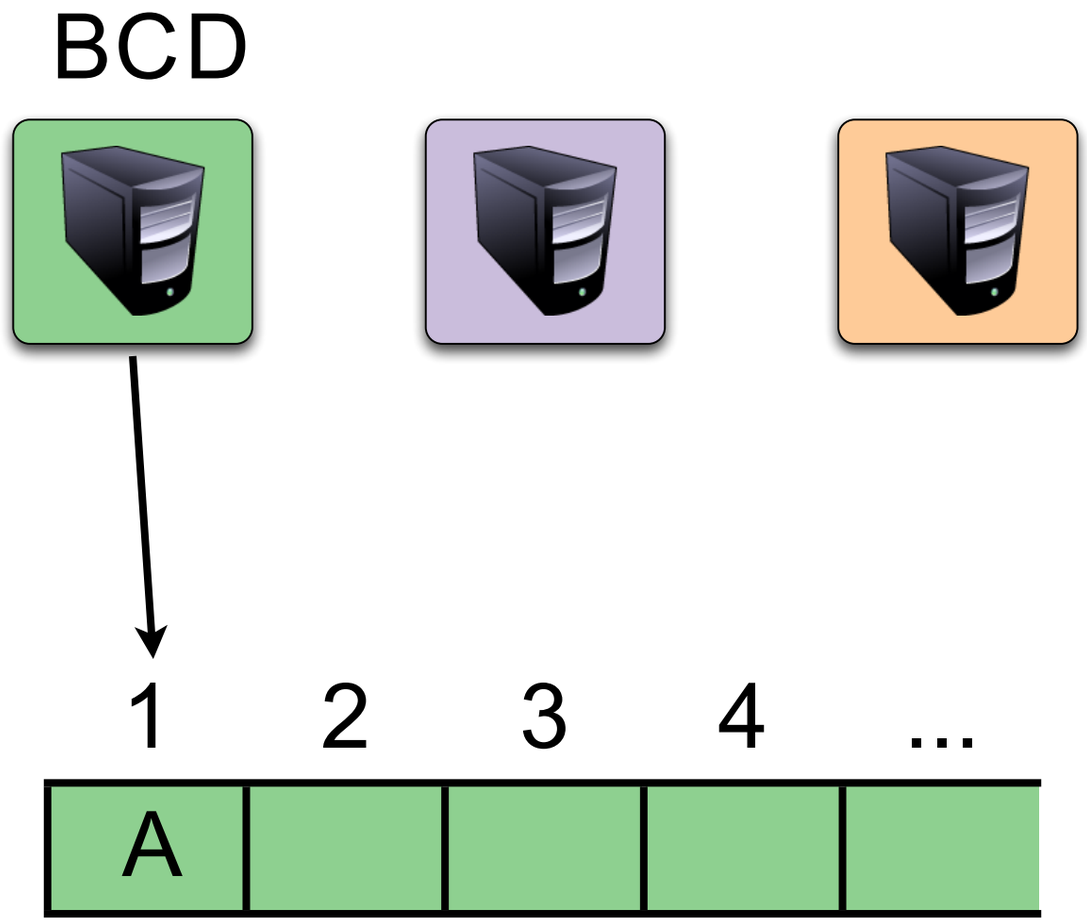


Multi-Paxos

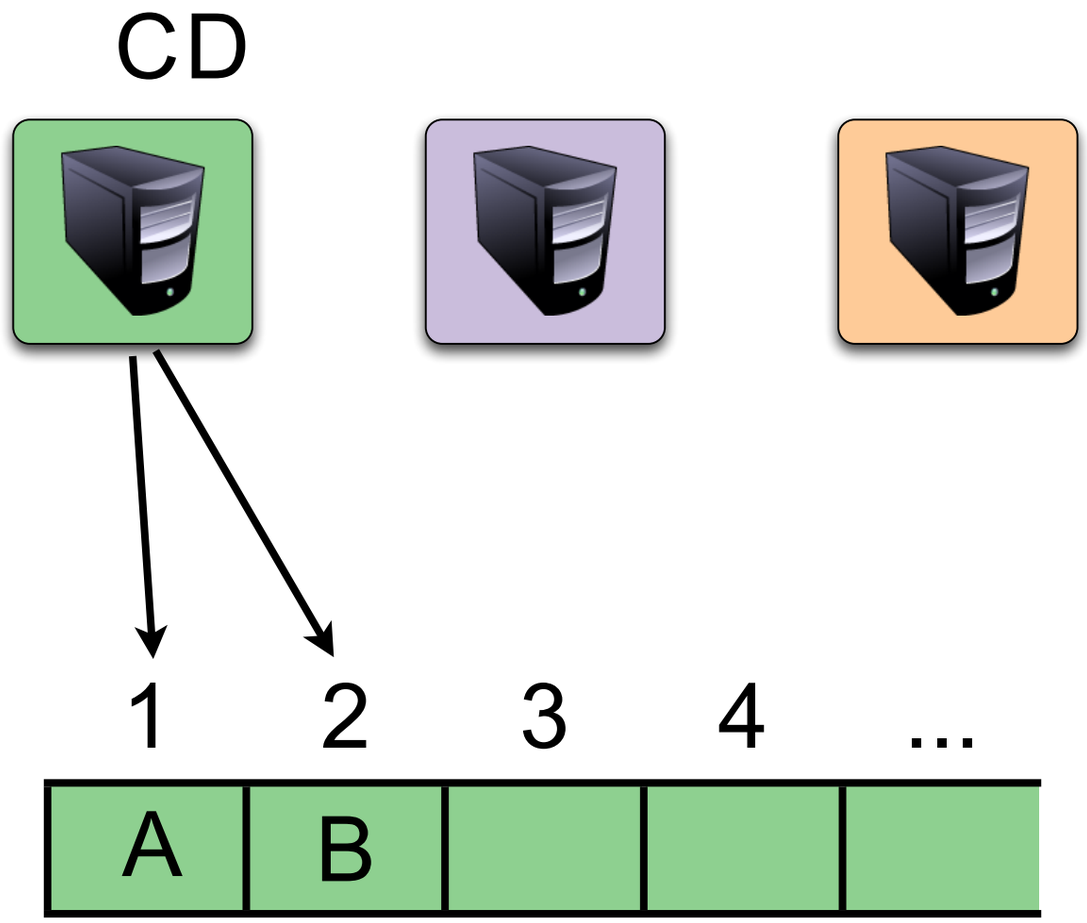
ABCD



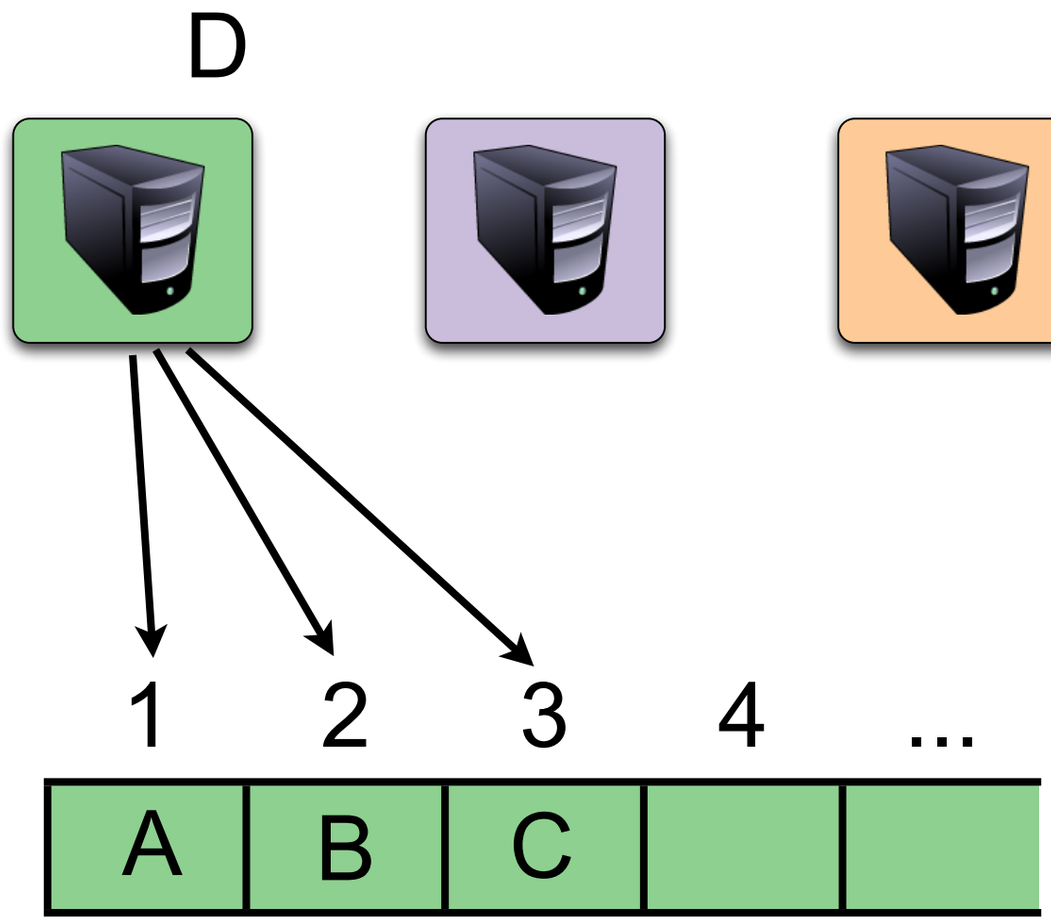
Multi-Paxos



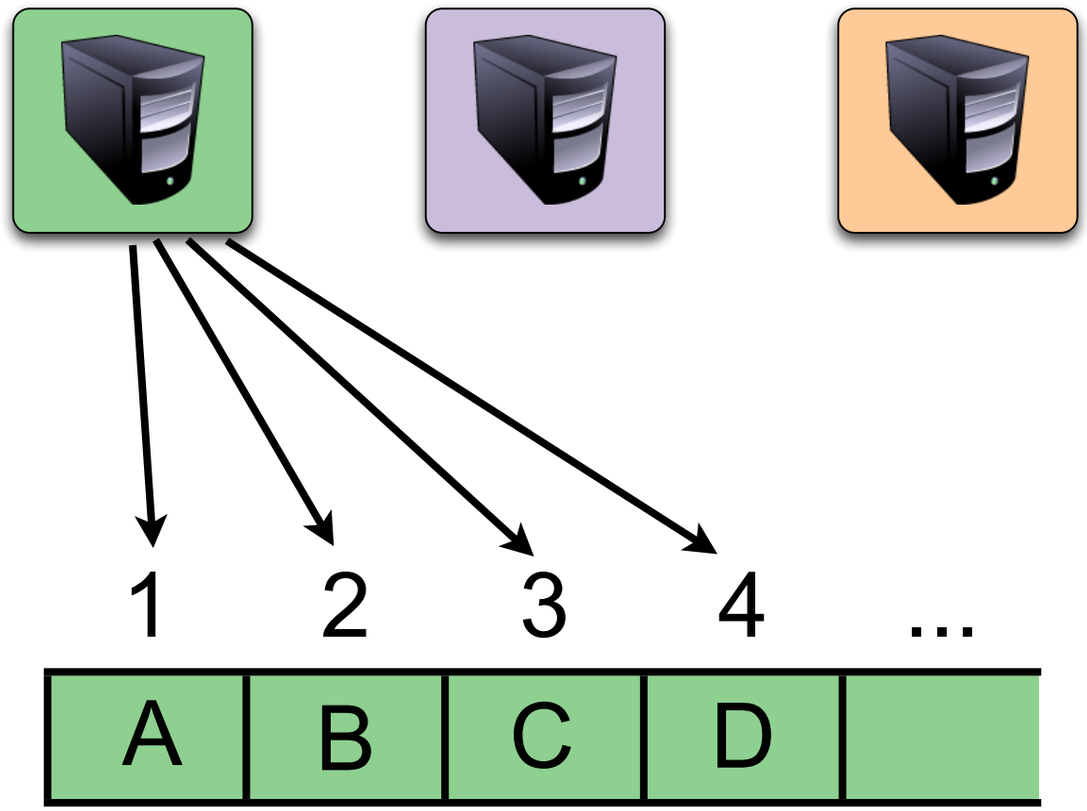
Multi-Paxos



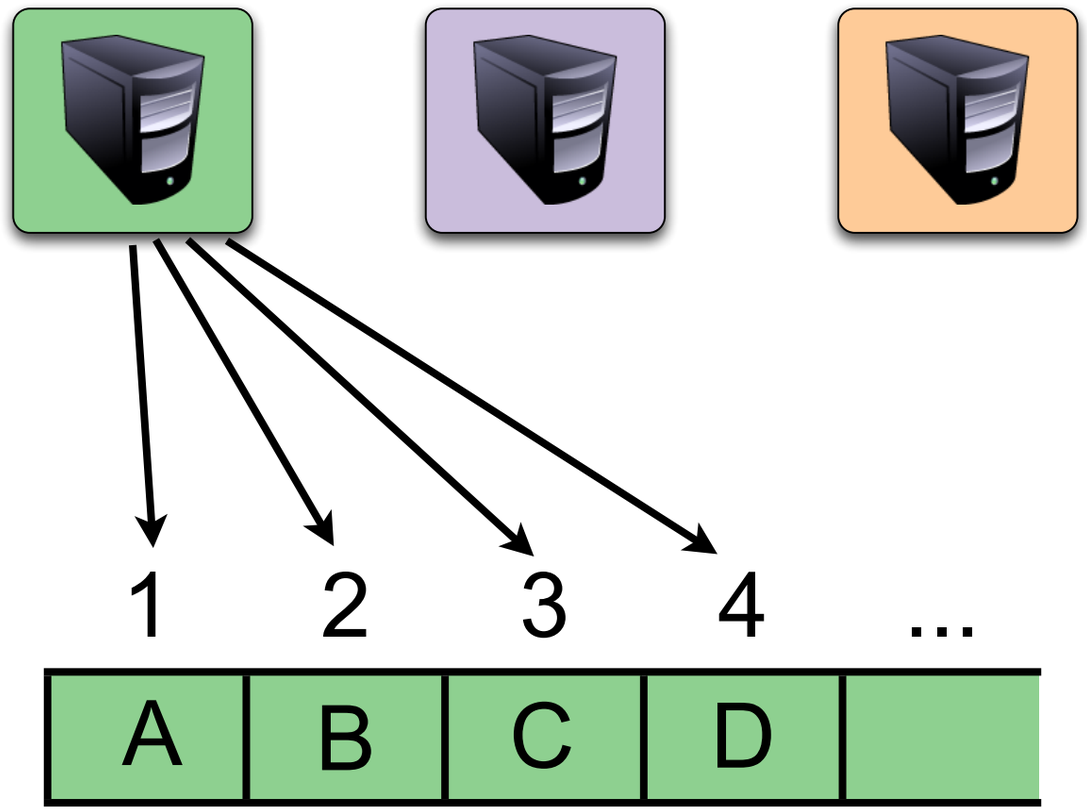
Multi-Paxos



Multi-Paxos

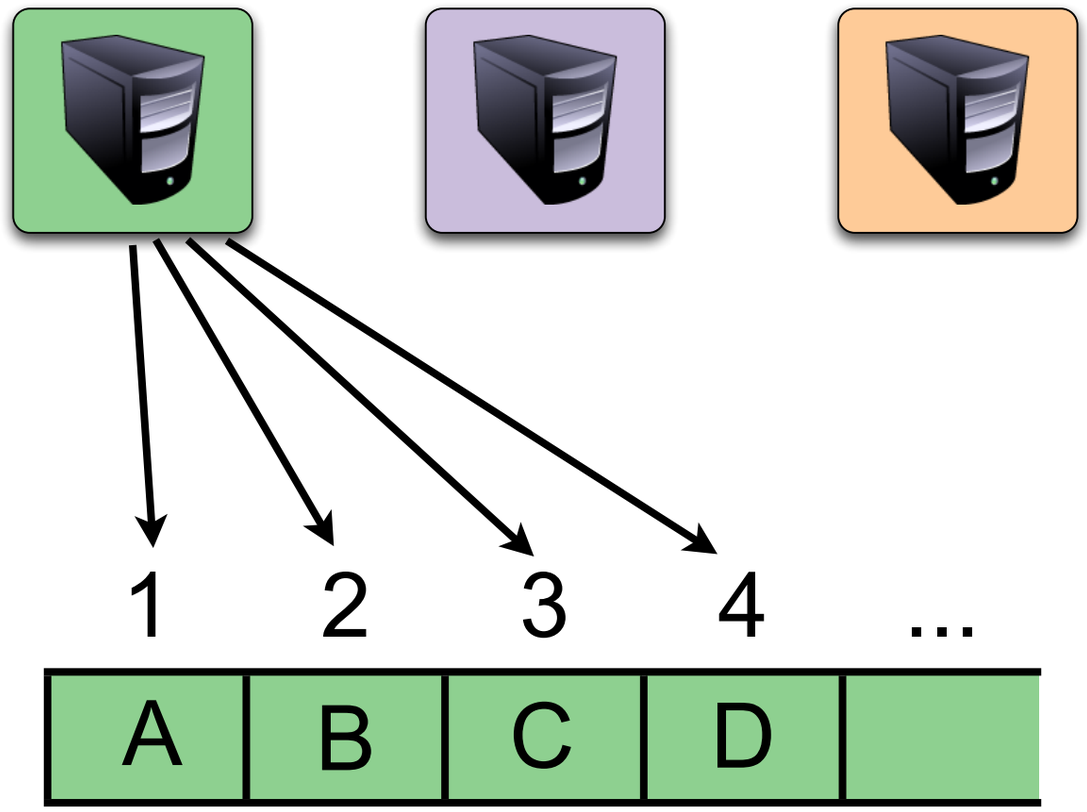


Multi-Paxos



- 1 RTT to commit

Multi-Paxos



- 1 RTT to commit
- Bottleneck for performance and availabilty

Can we have it all?

Can we have it all?

- High throughput, low latency

Can we have it all?

- High throughput, low latency
- Constant availability

Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas

Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas

Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas

Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas

Paxos

Can we have it all?

- High throughput, low latency

Multi-Paxos

- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas

Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas

Multi-Paxos

Egalitarian Paxos (EPaxos)

EPaxos is all about ordering

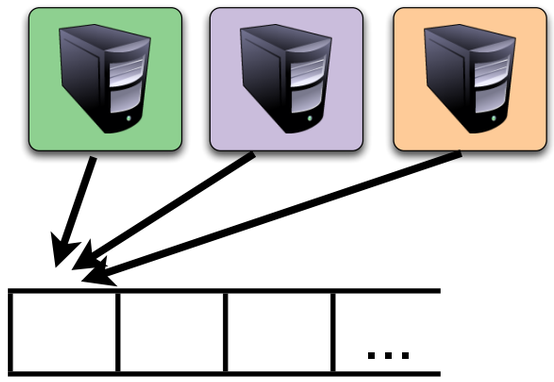
EPaxos is all about ordering

Previous strategies:

EPaxos is all about ordering

Previous strategies:

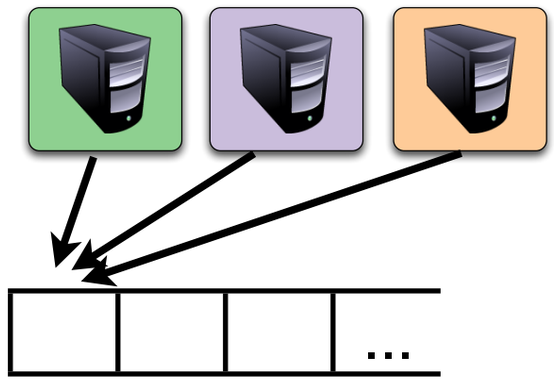
Contend for slots



EPaxos is all about ordering

Previous strategies:

Contend for slots

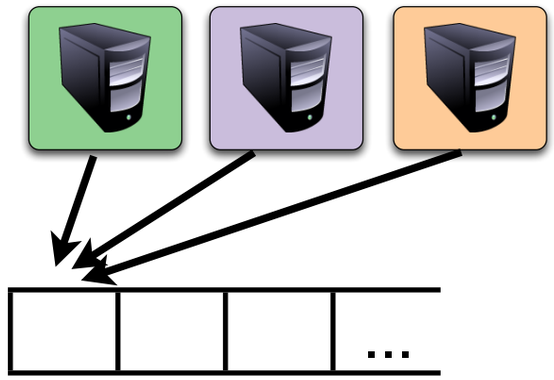


Paxos

EPaxos is all about ordering

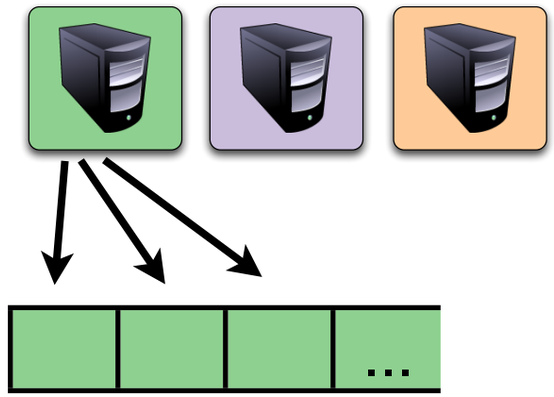
Previous strategies:

Contend for slots



Paxos

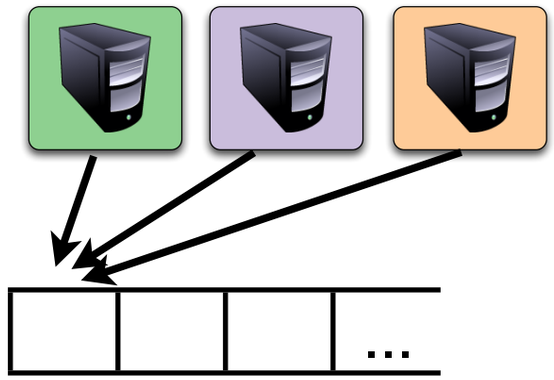
One replica decides



EPaxos is all about ordering

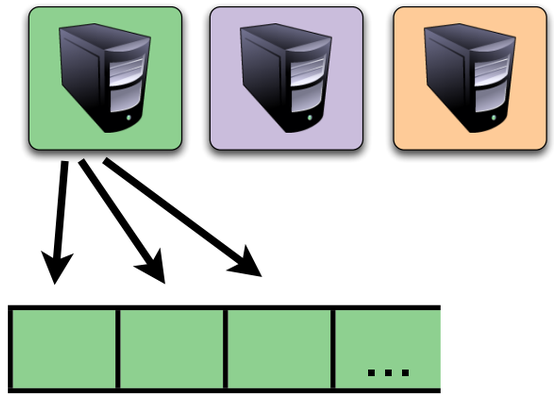
Previous strategies:

Contend for slots



Paxos

One replica decides

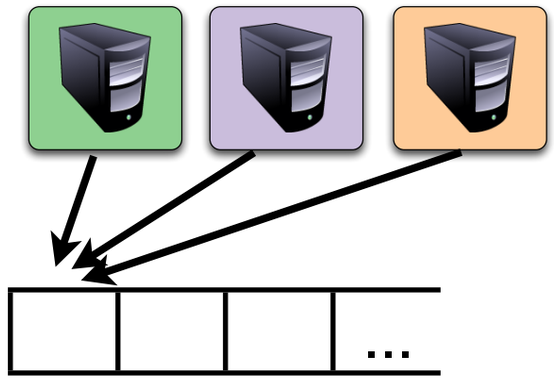


Multi-Paxos, Fast Paxos, Generalized Paxos

EPaxos is all about ordering

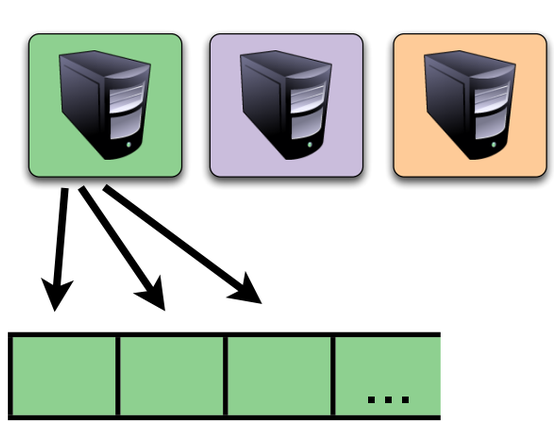
Previous strategies:

Contend for slots



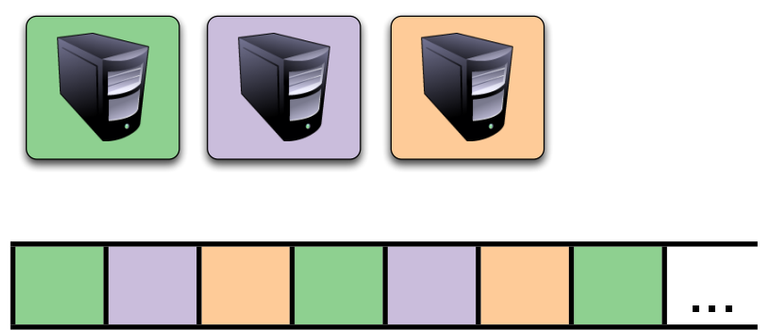
Paxos

One replica decides



Multi-Paxos, Fast Paxos, Generalized Paxos

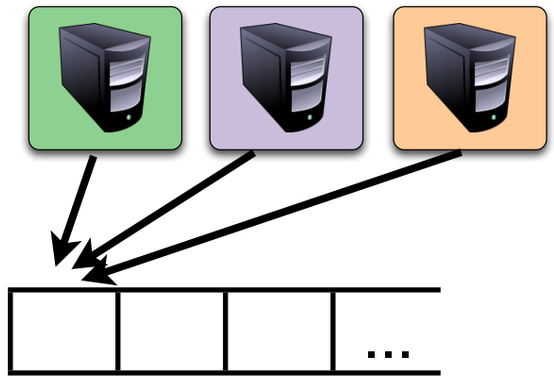
Take turns round-robin



EPaxos is all about ordering

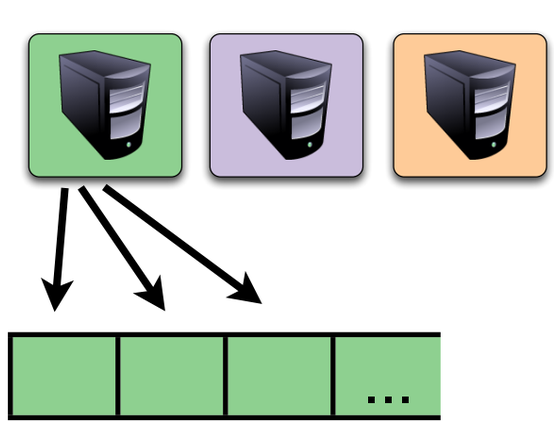
Previous strategies:

Contend for slots



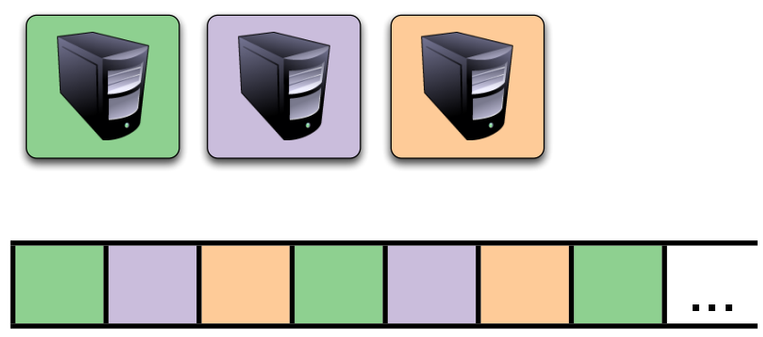
Paxos

One replica decides



Multi-Paxos, Fast Paxos, Generalized Paxos

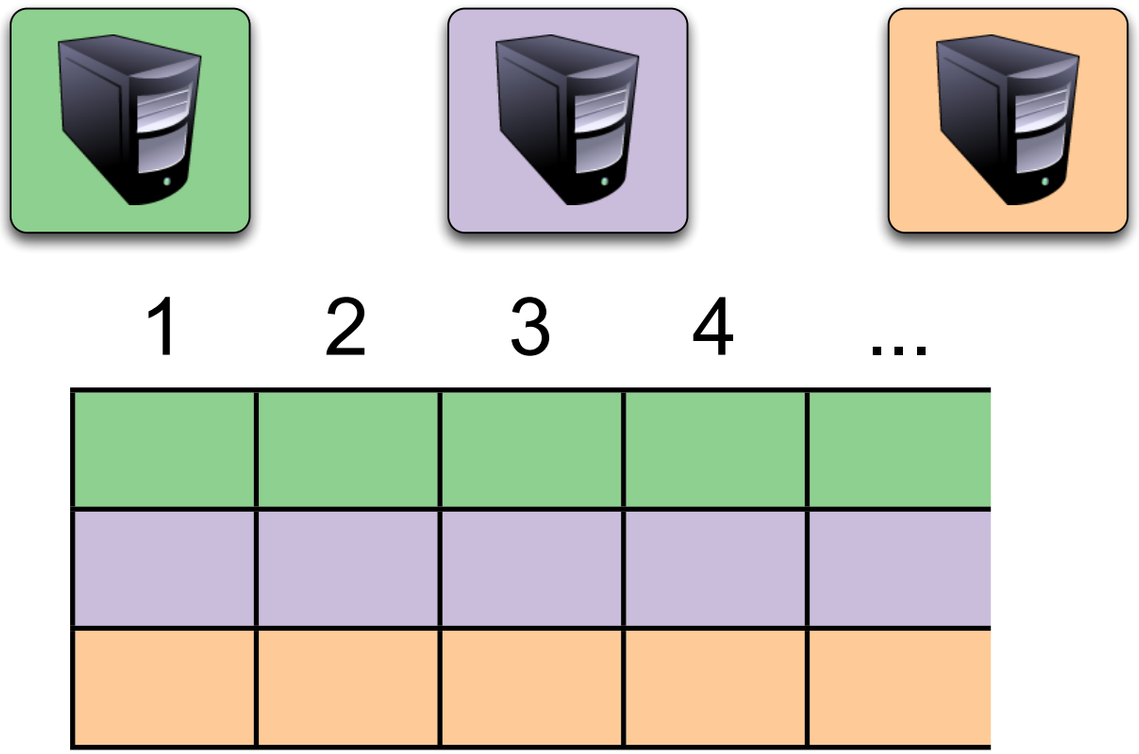
Take turns round-robin



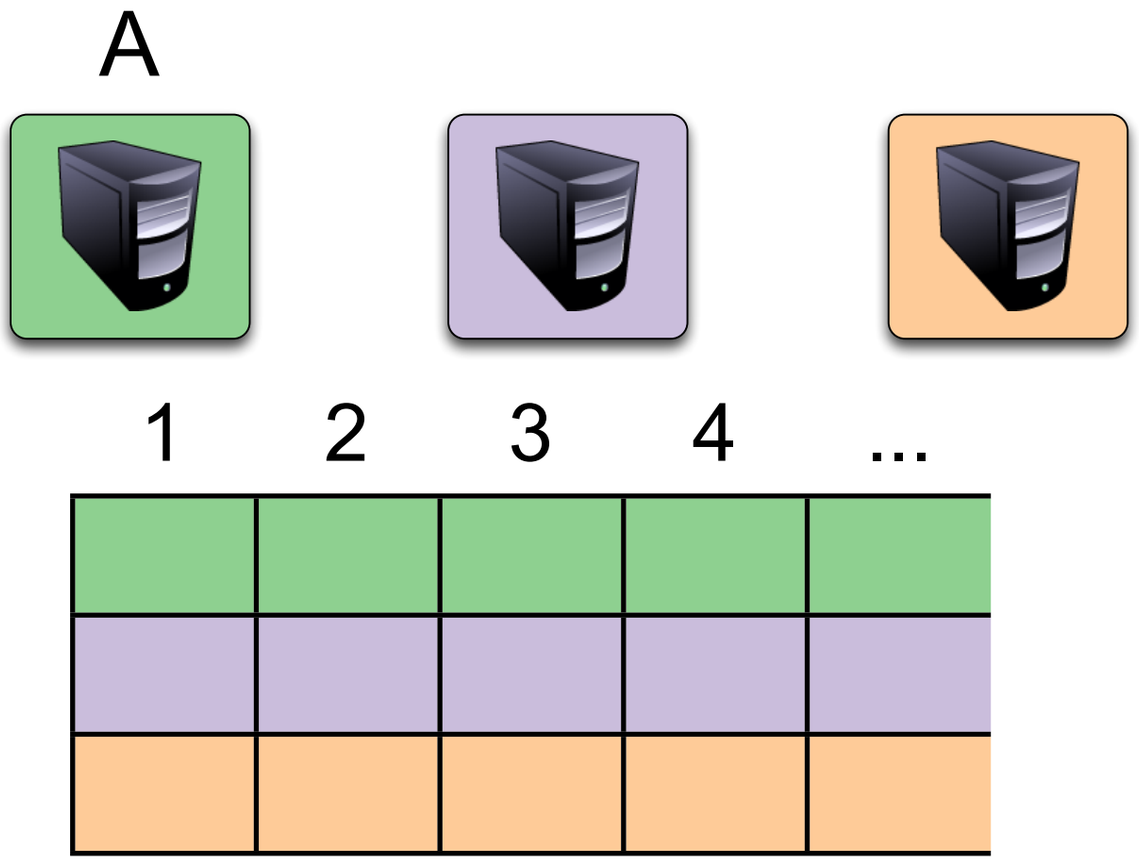
Mencius

EPaxos intuition

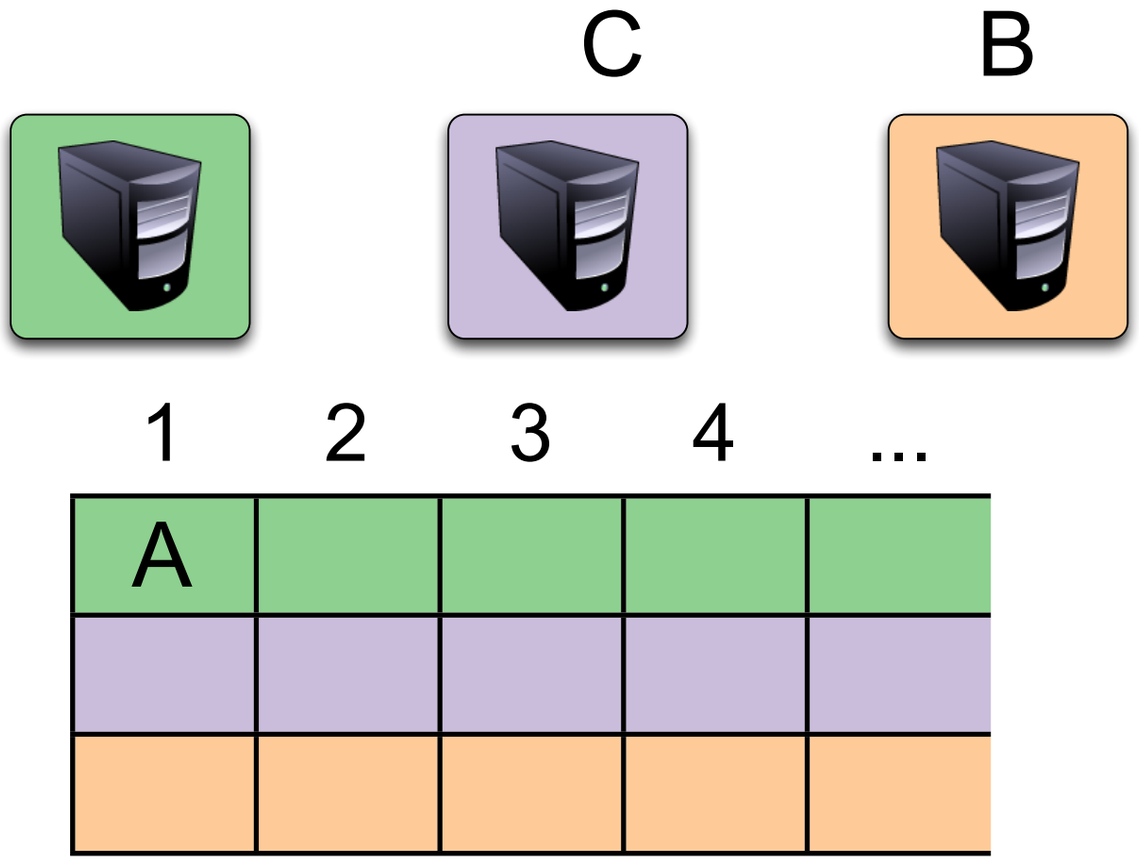
EPaxos intuition



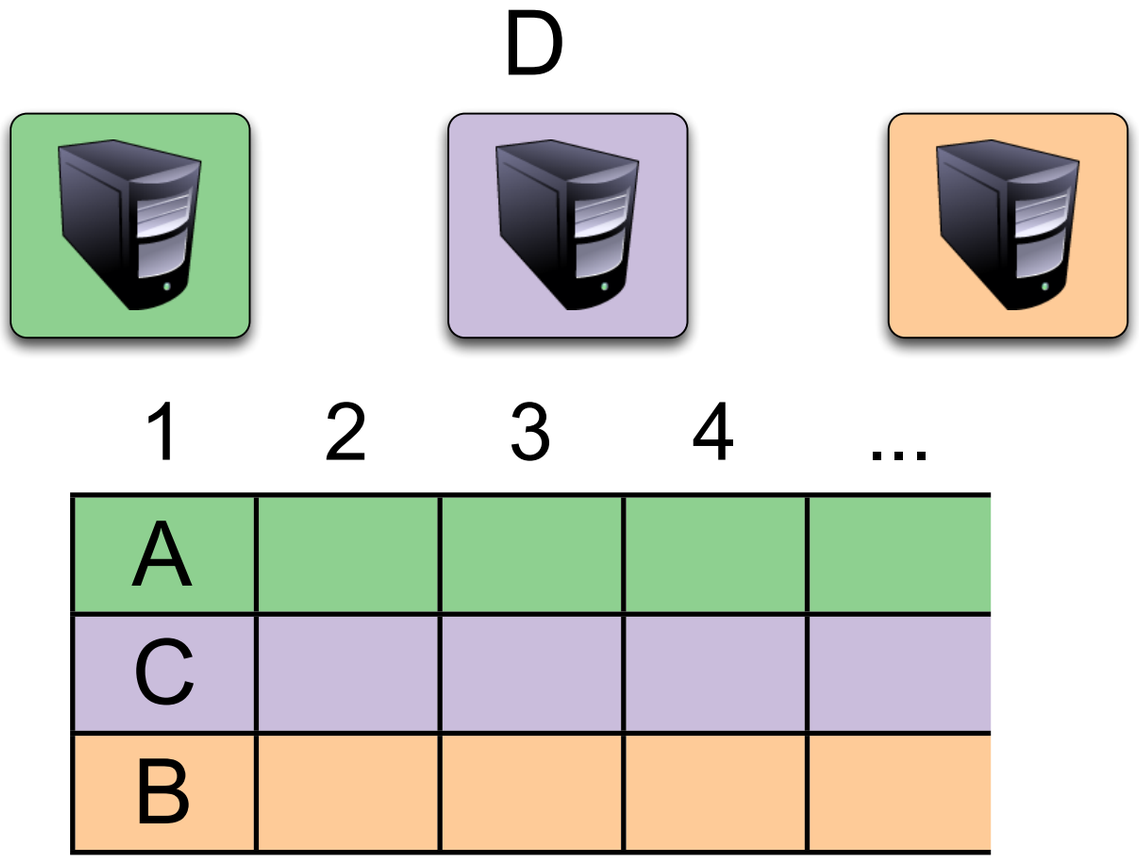
EPaxos intuition



EPaxos intuition



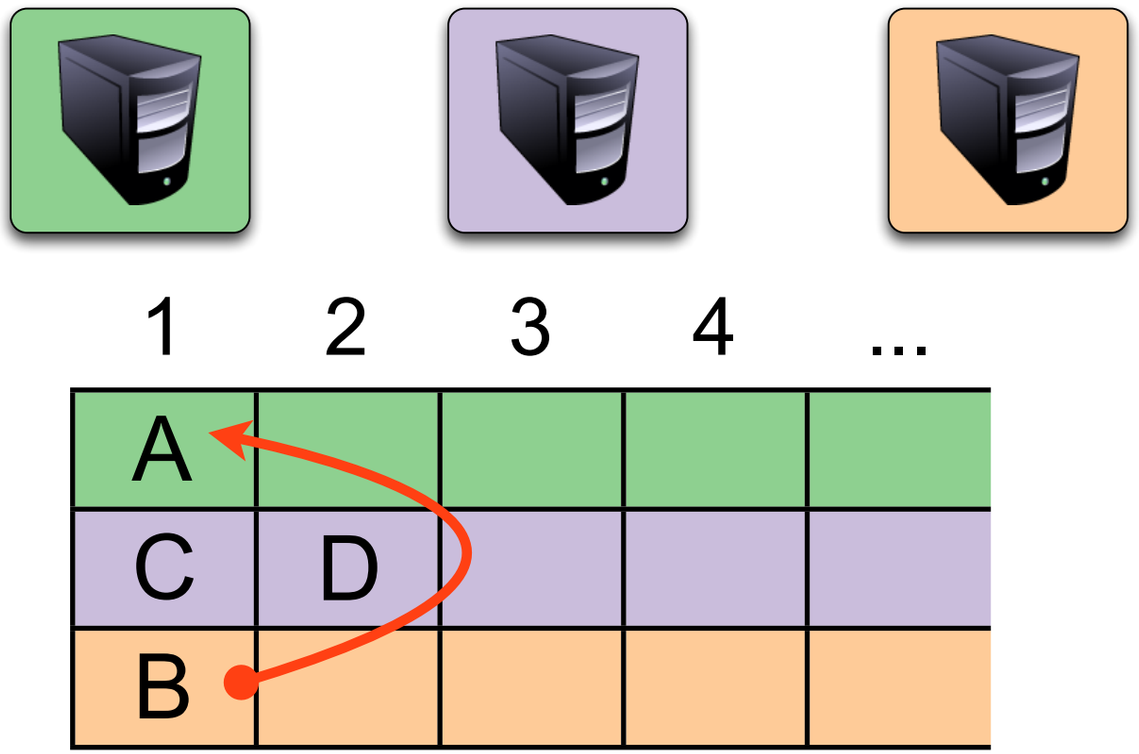
EPaxos intuition



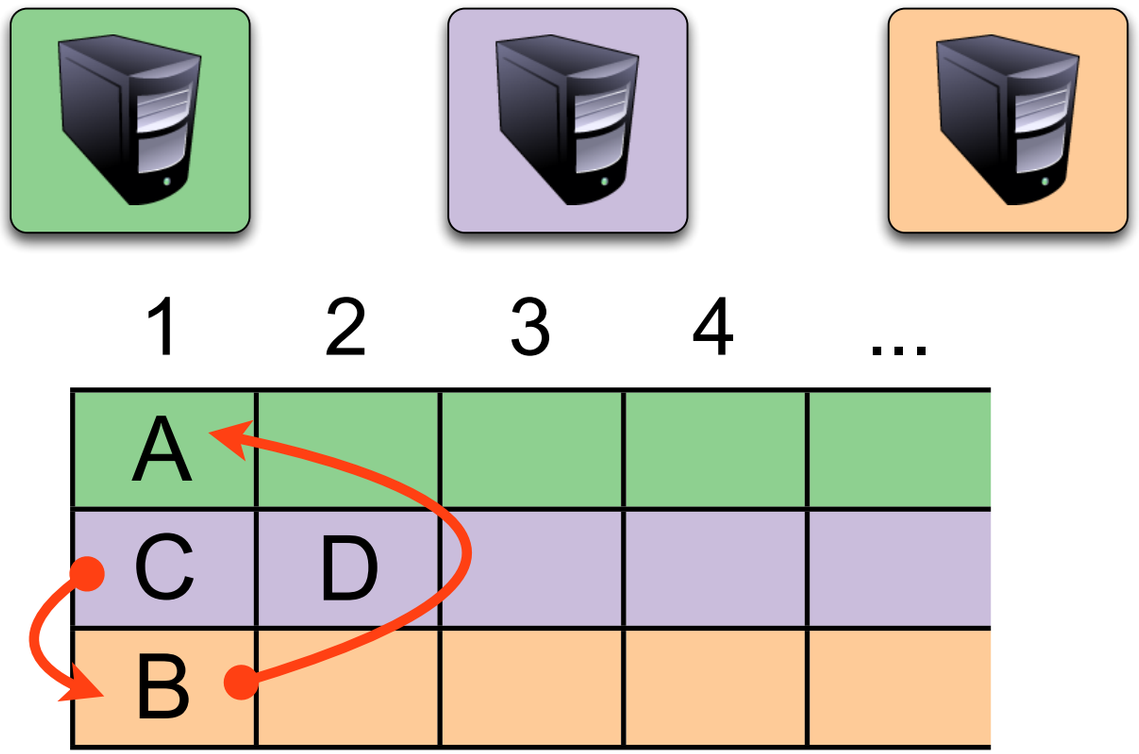
EPaxos intuition

	1	2	3	4	...
A					
C	D				
B					

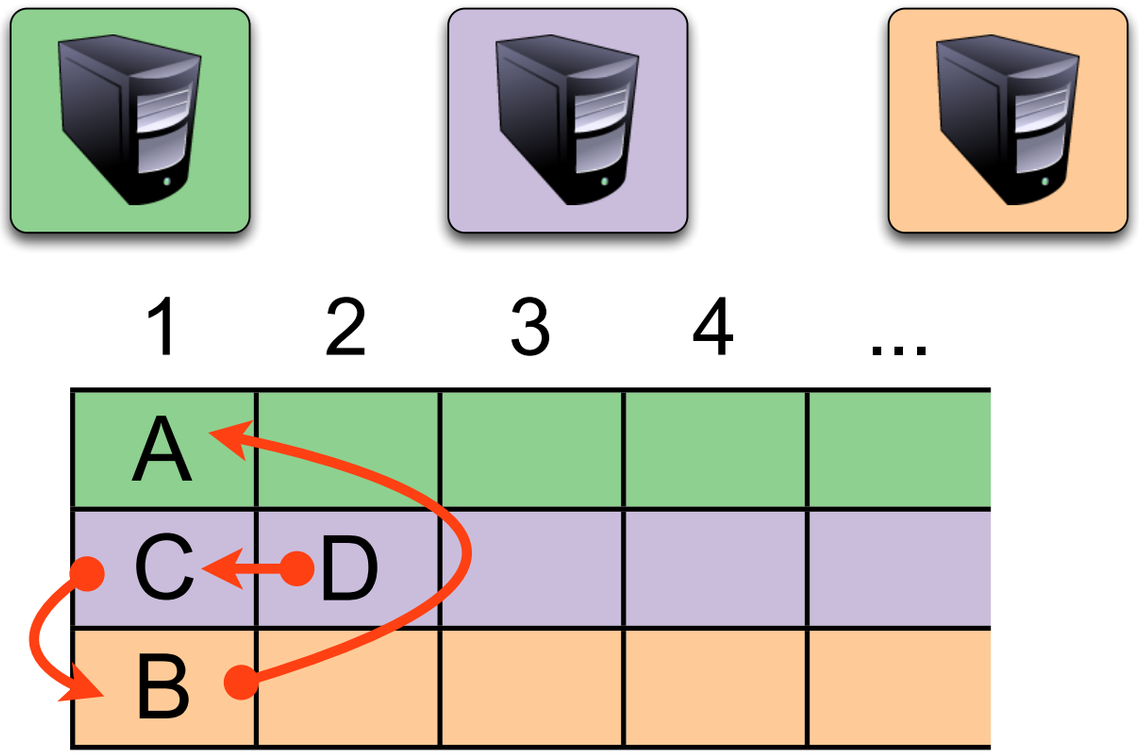
EPaxos intuition



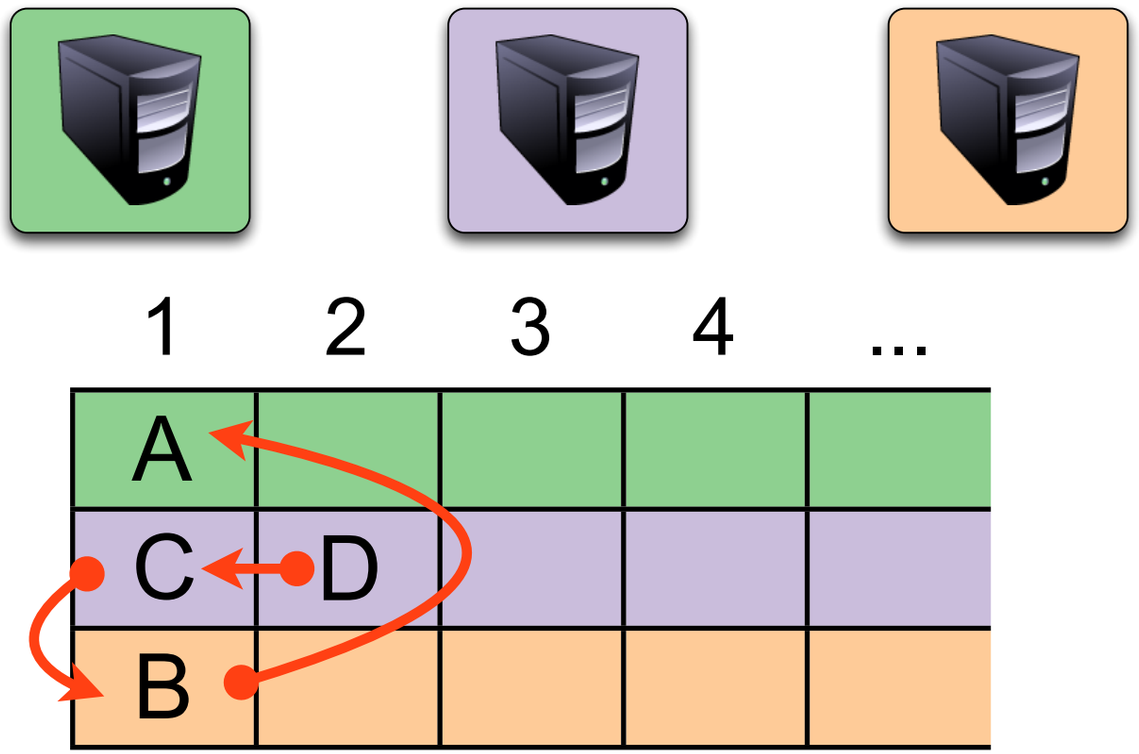
EPaxos intuition



EPaxos intuition

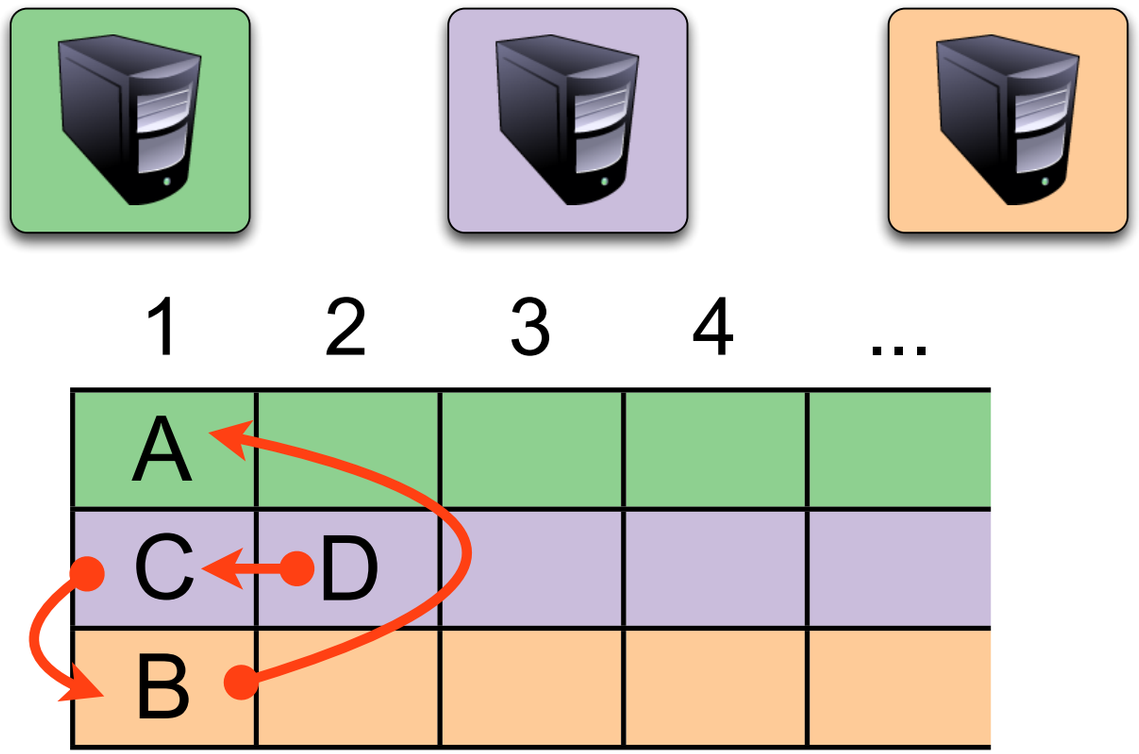


EPaxos intuition



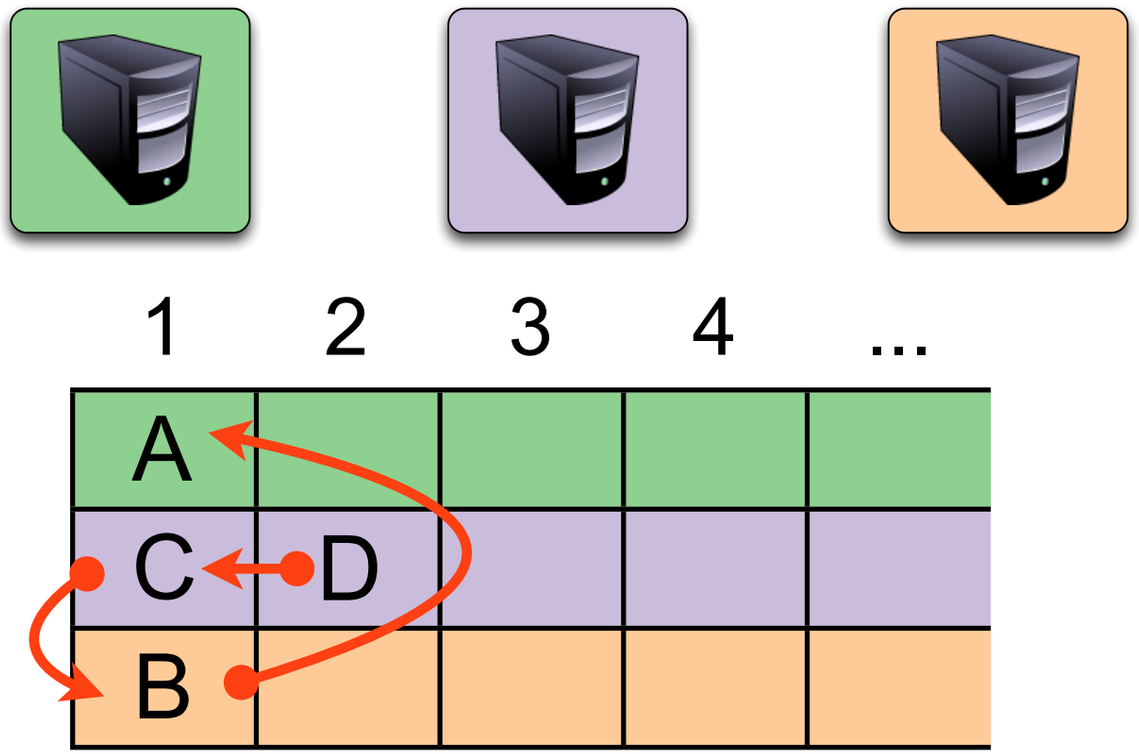
After commit
@ each replica

EPaxos intuition



After commit
@ each replica

EPaxos intuition

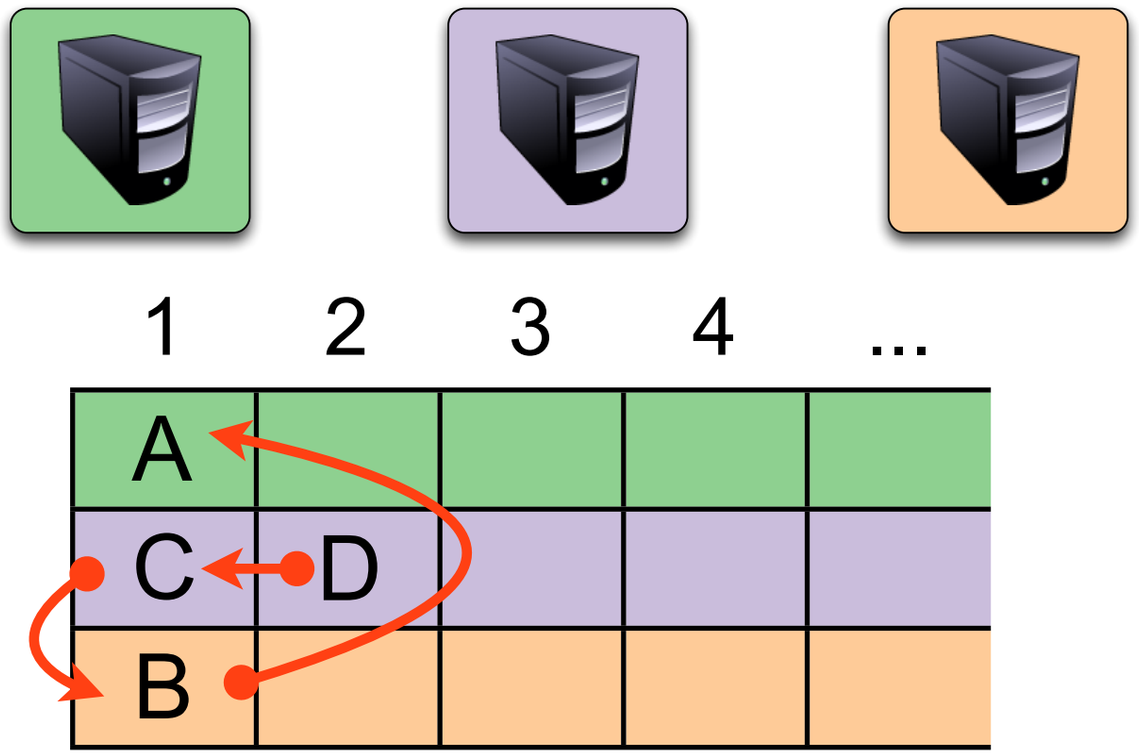


After commit
@ each replica

A ← B ← C ← D

- Load balance (every replica is a leader)

EPaxos intuition



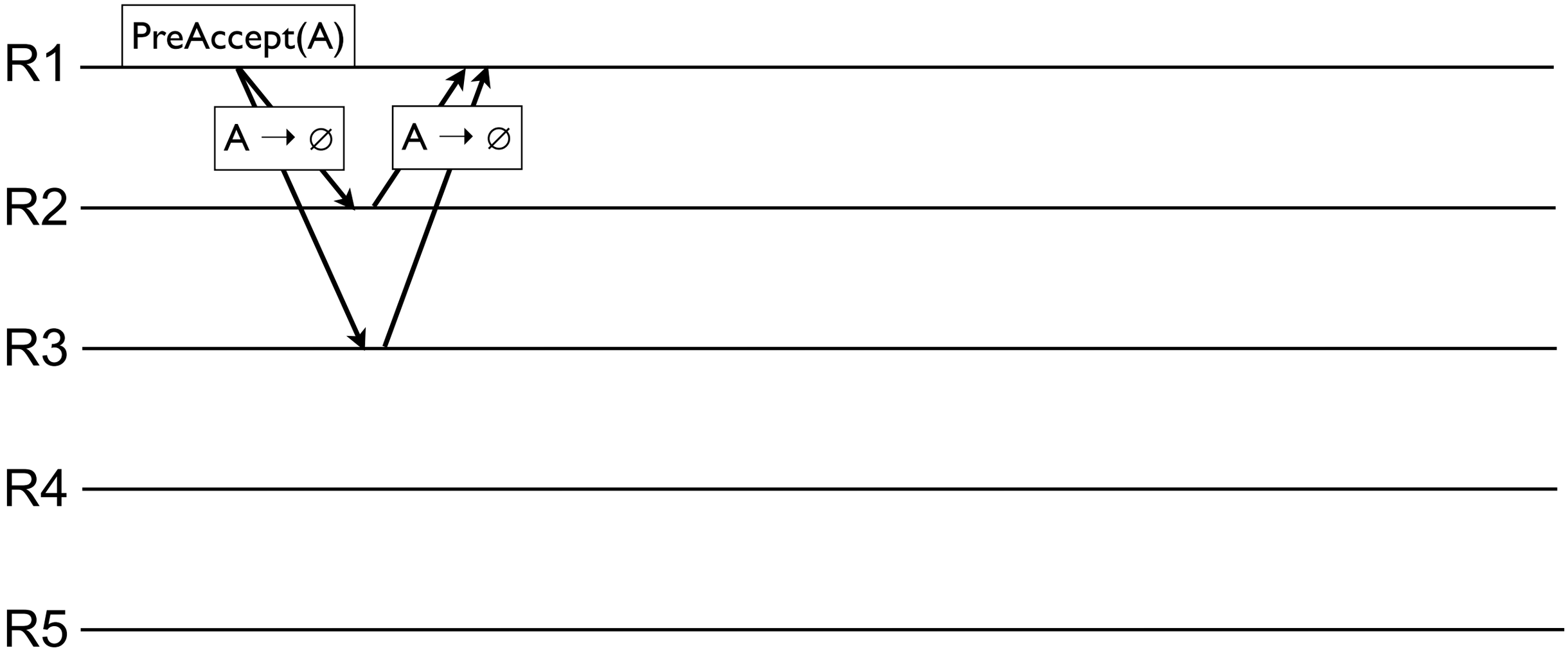
After commit
@ each replica

- Load balance (every replica is a leader)
- EPaxos can choose *any* quorum for each command

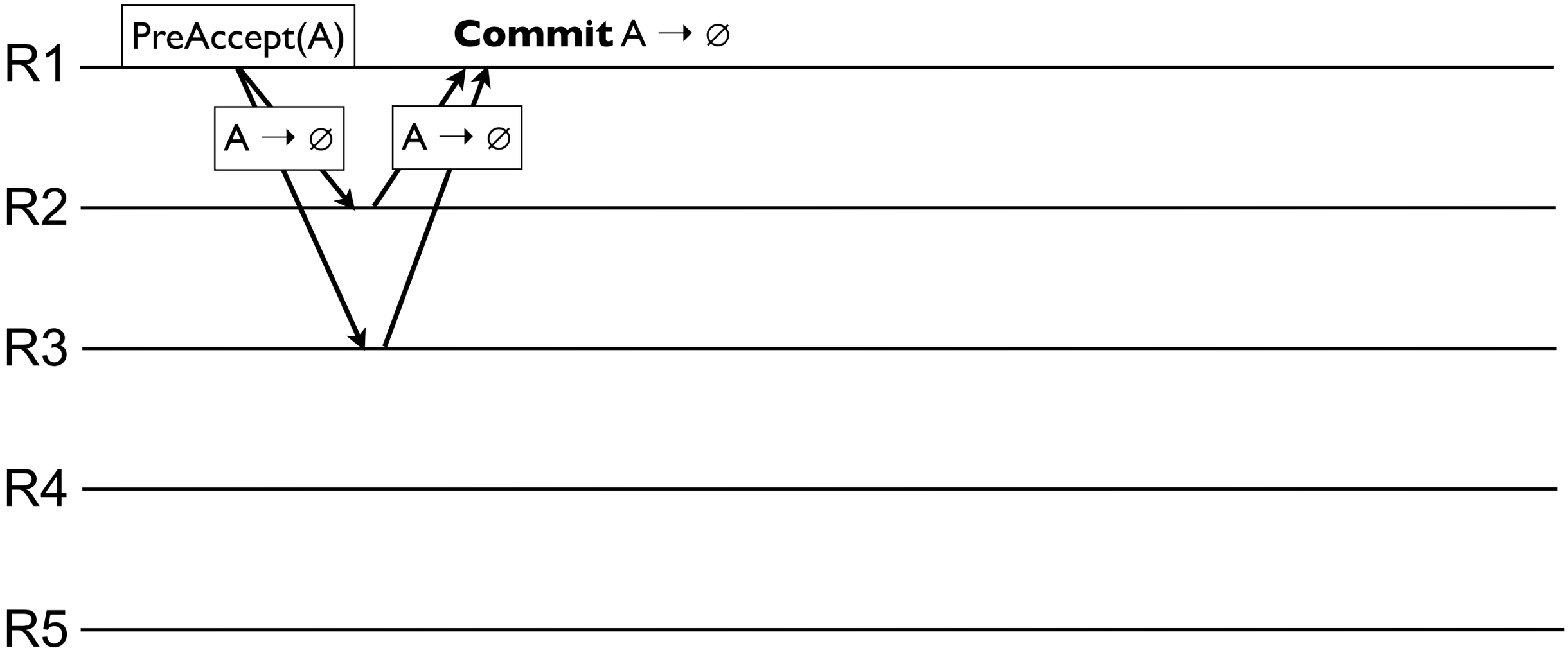
EPaxos commit protocol

EPaxos commit protocol

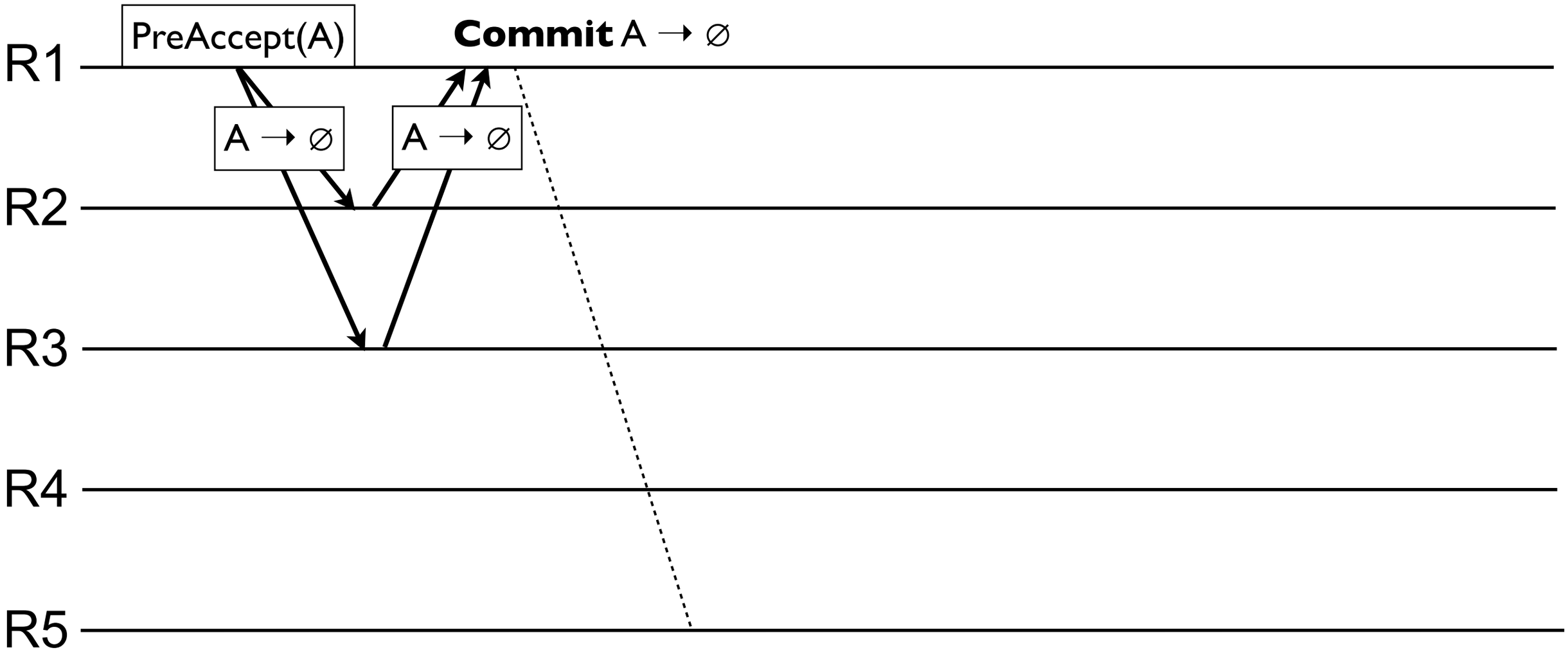
EPaxos commit protocol



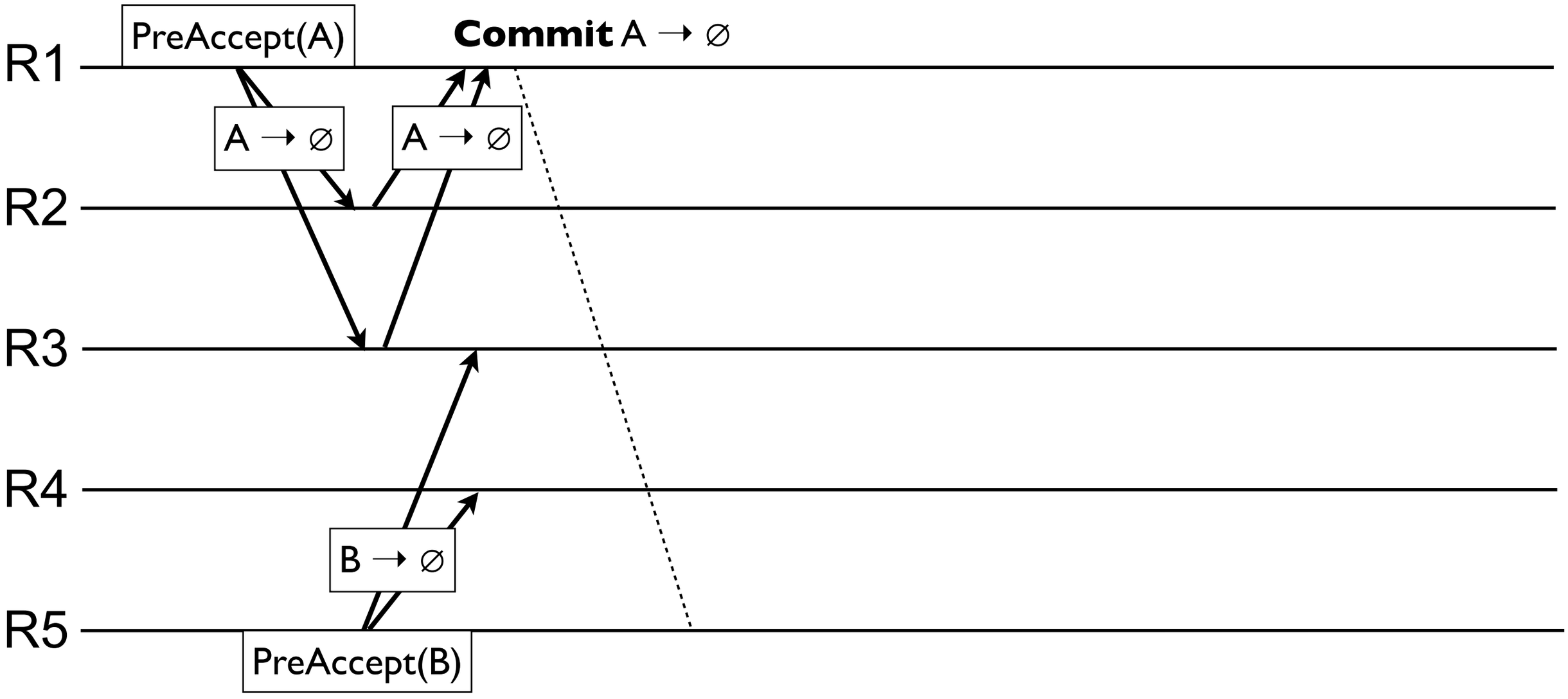
EPaxos commit protocol



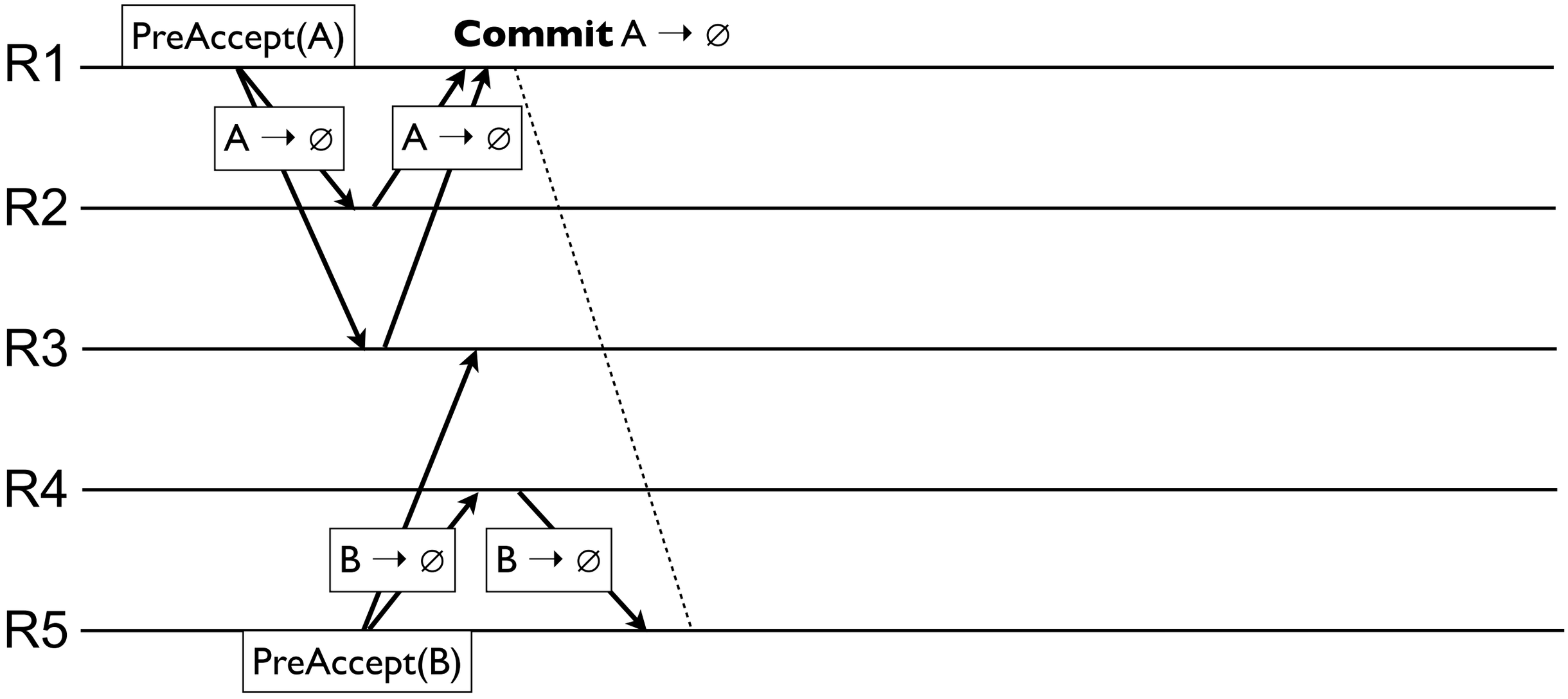
EPaxos commit protocol



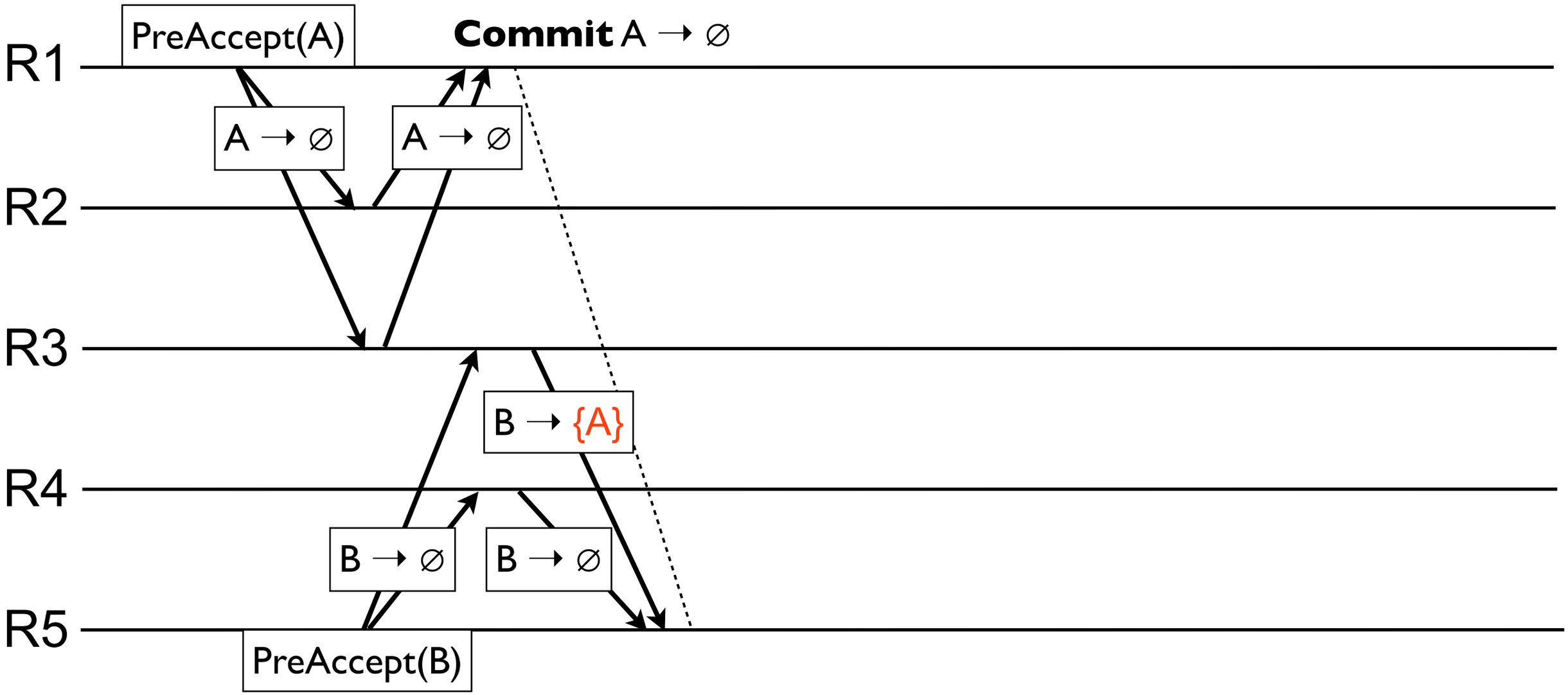
EPaxos commit protocol



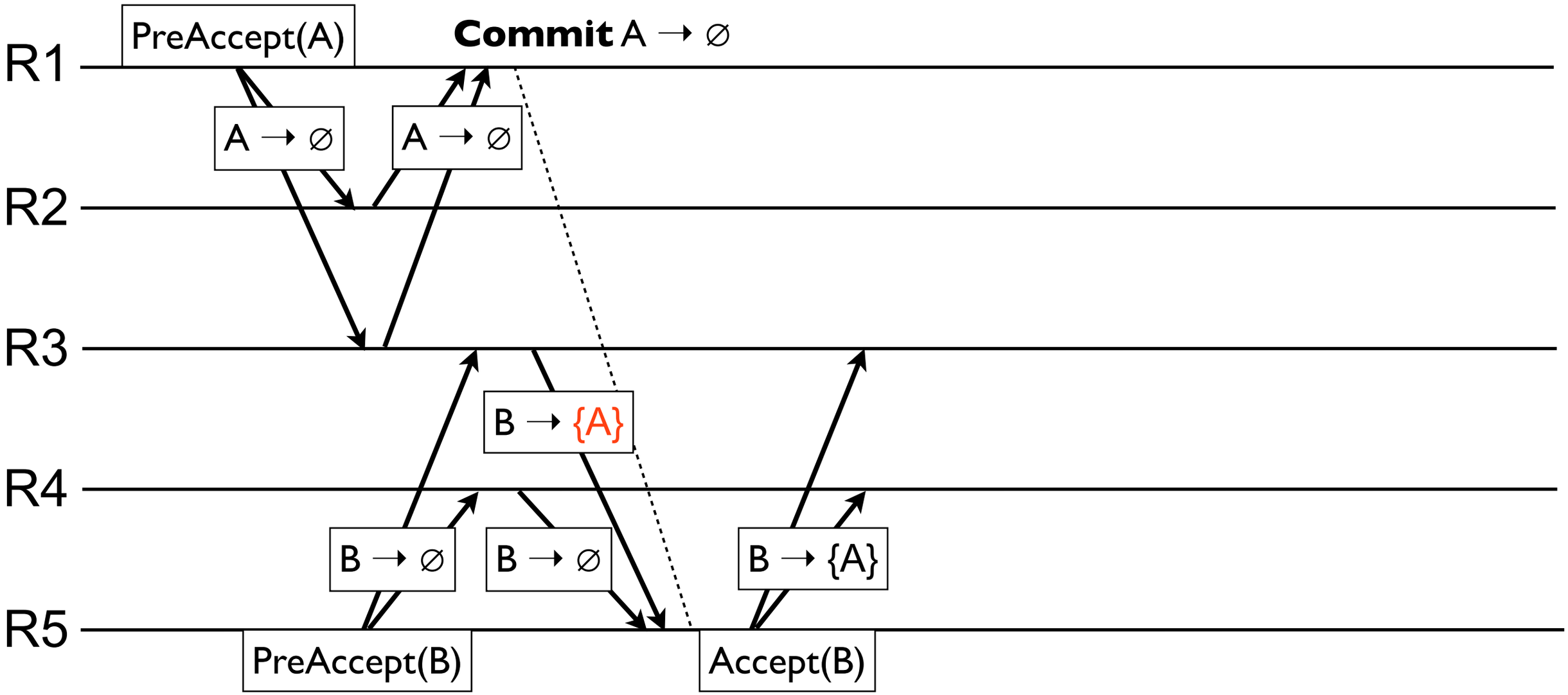
EPaxos commit protocol



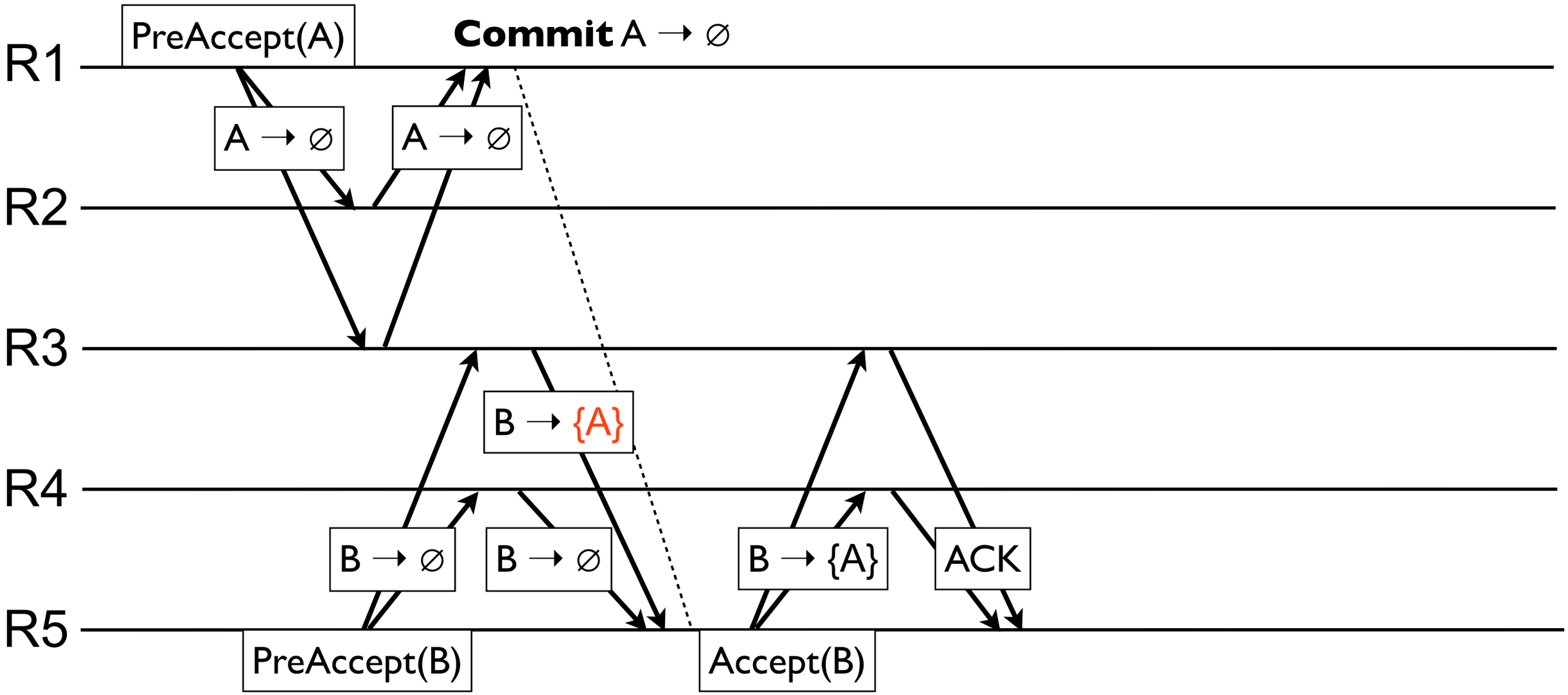
EPaxos commit protocol



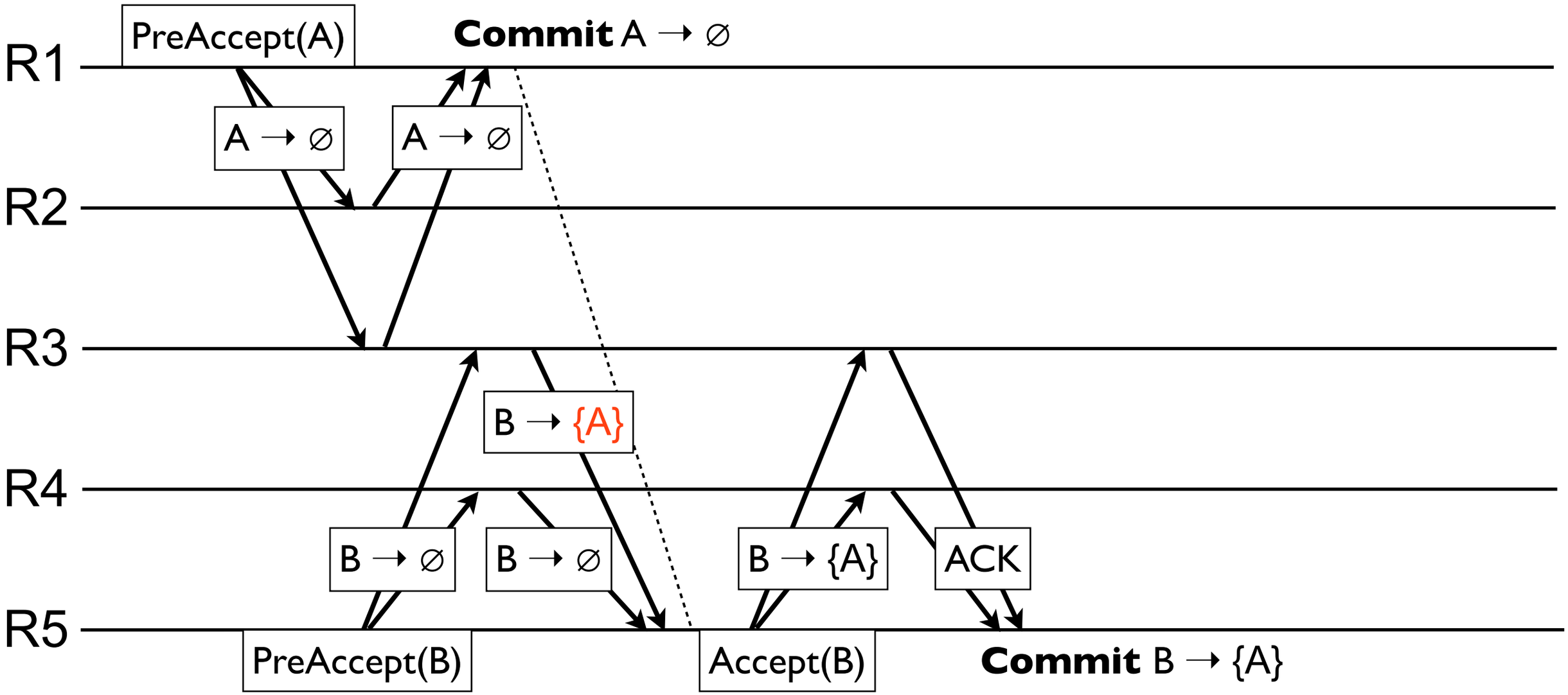
EPaxos commit protocol



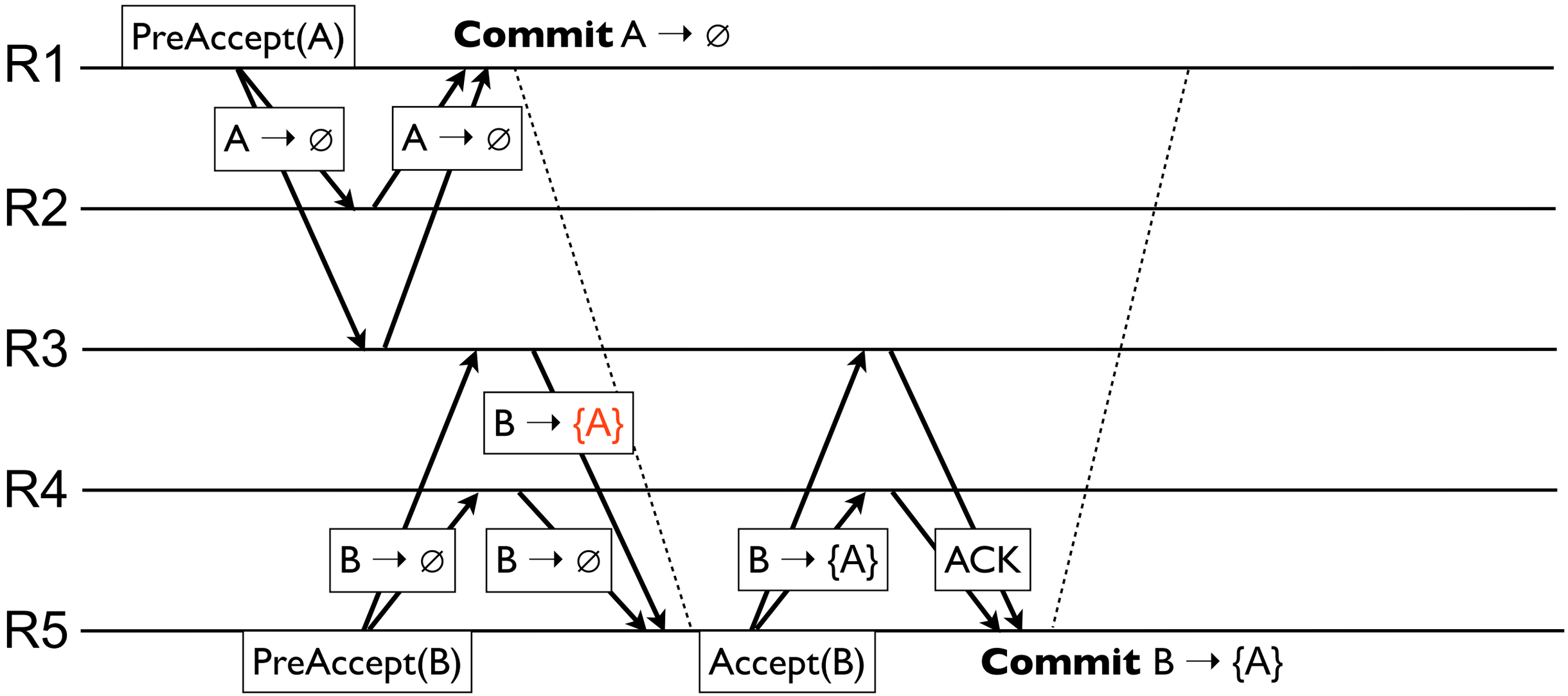
EPaxos commit protocol



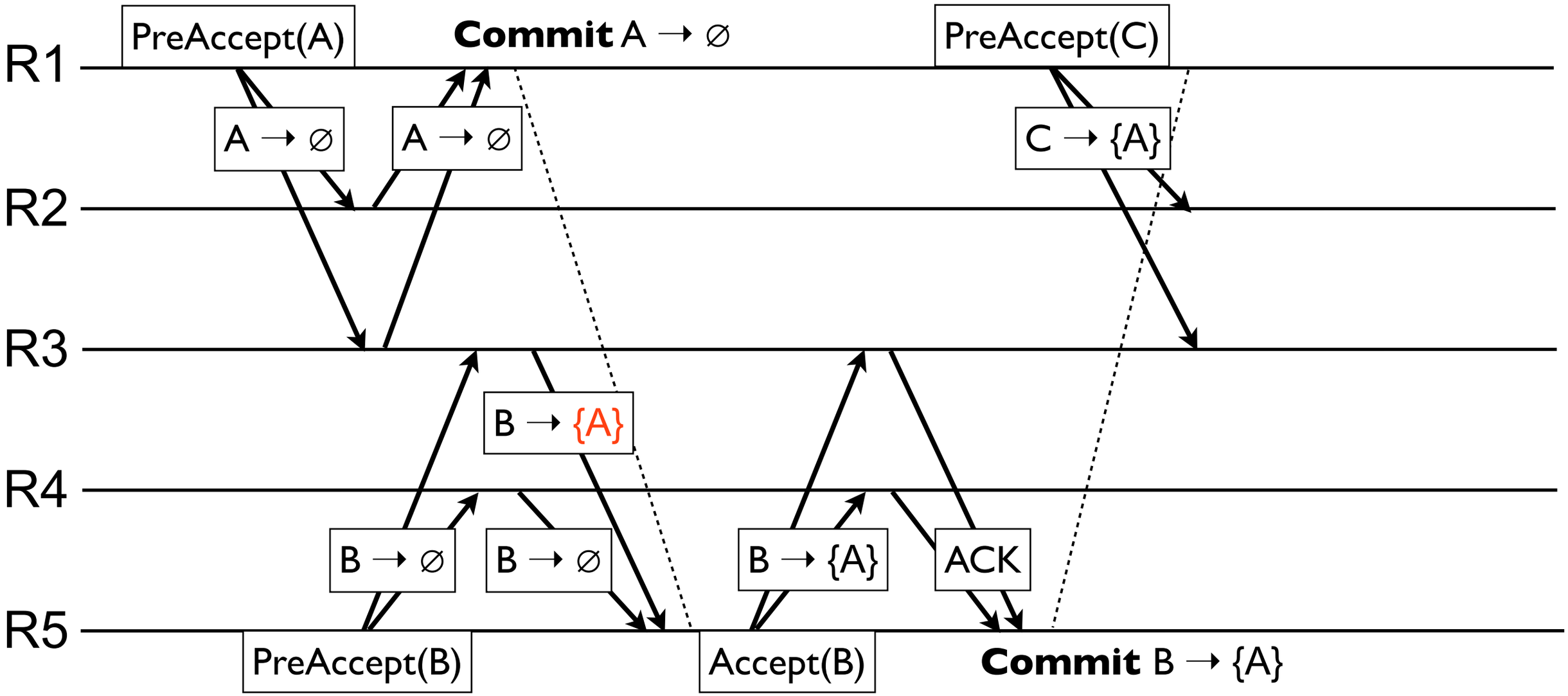
EPaxos commit protocol



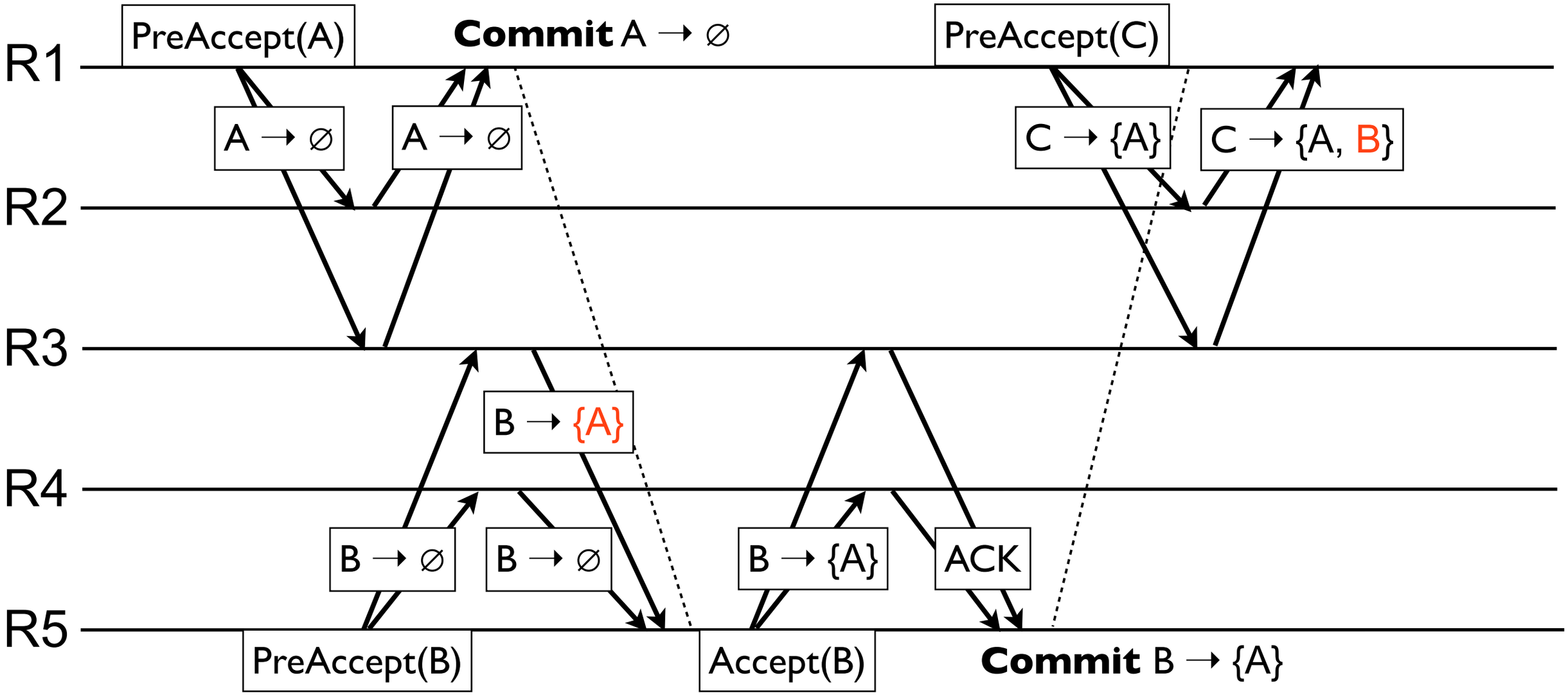
EPaxos commit protocol



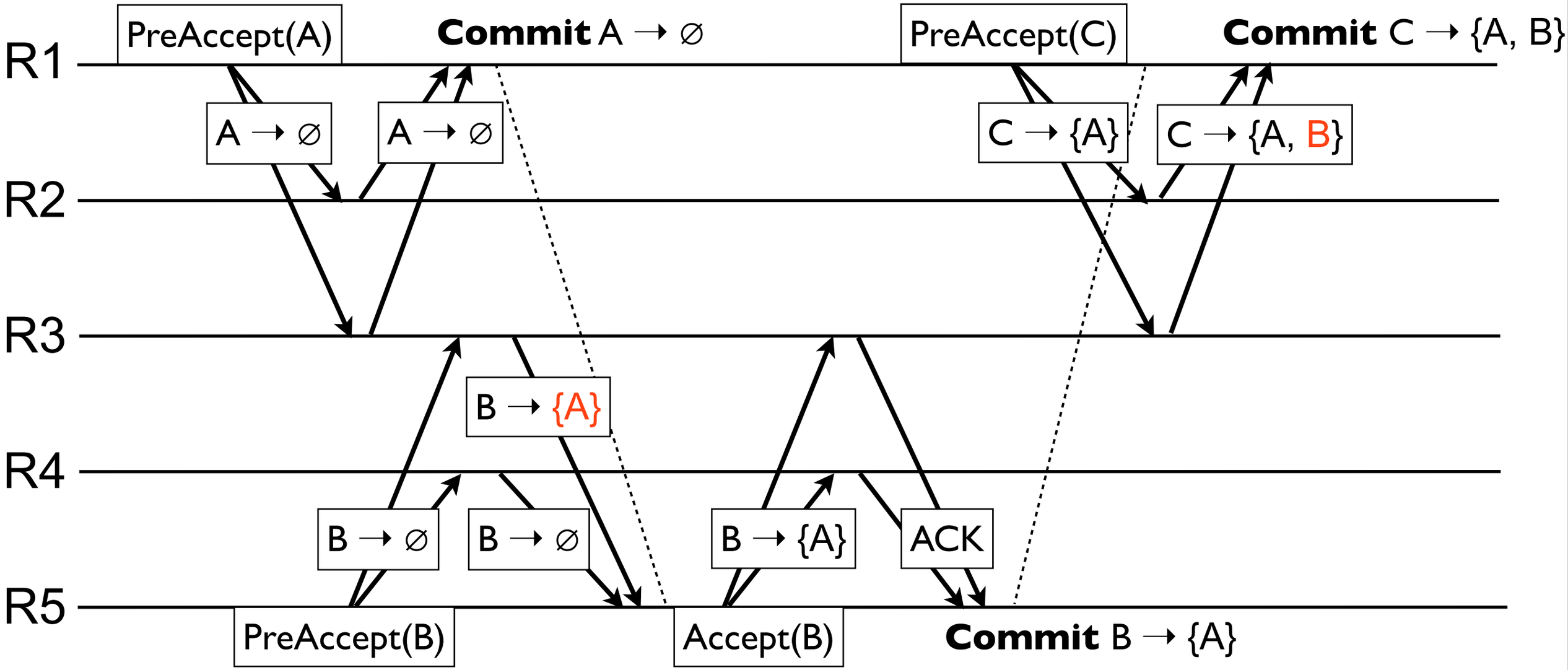
EPaxos commit protocol



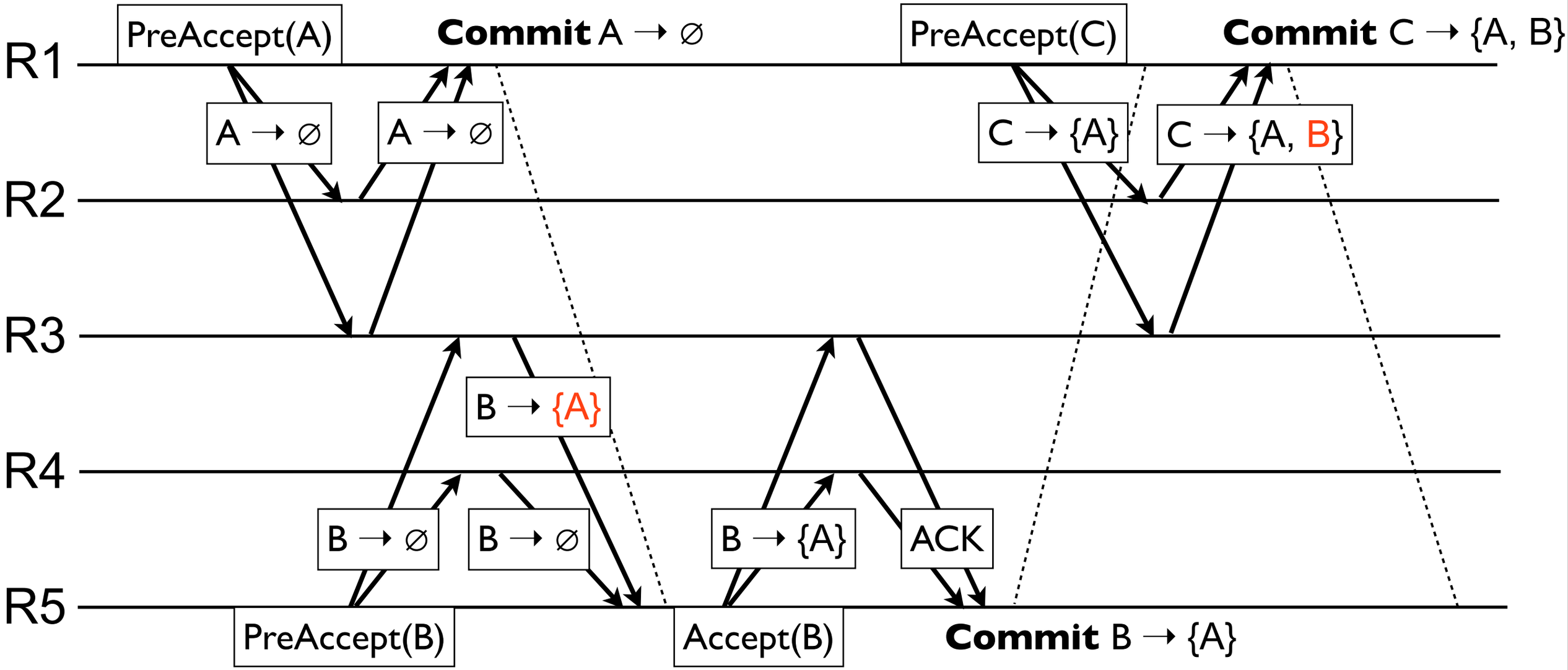
EPaxos commit protocol



EPaxos commit protocol



EPaxos commit protocol



Order only interfering commands

- 1 RTT
  - Non-concurrent commands
  - OR non-interfering commands

- 2 RTTs
  - Concurrent AND interfering

Interference is application-specific

Interference is application-specific

KV store

- Infer from operation key

Interference is application-specific

KV store

- Infer from operation key

Google App Engine

- Programmer-specified

Interference is application-specific

KV store

- Infer from operation key

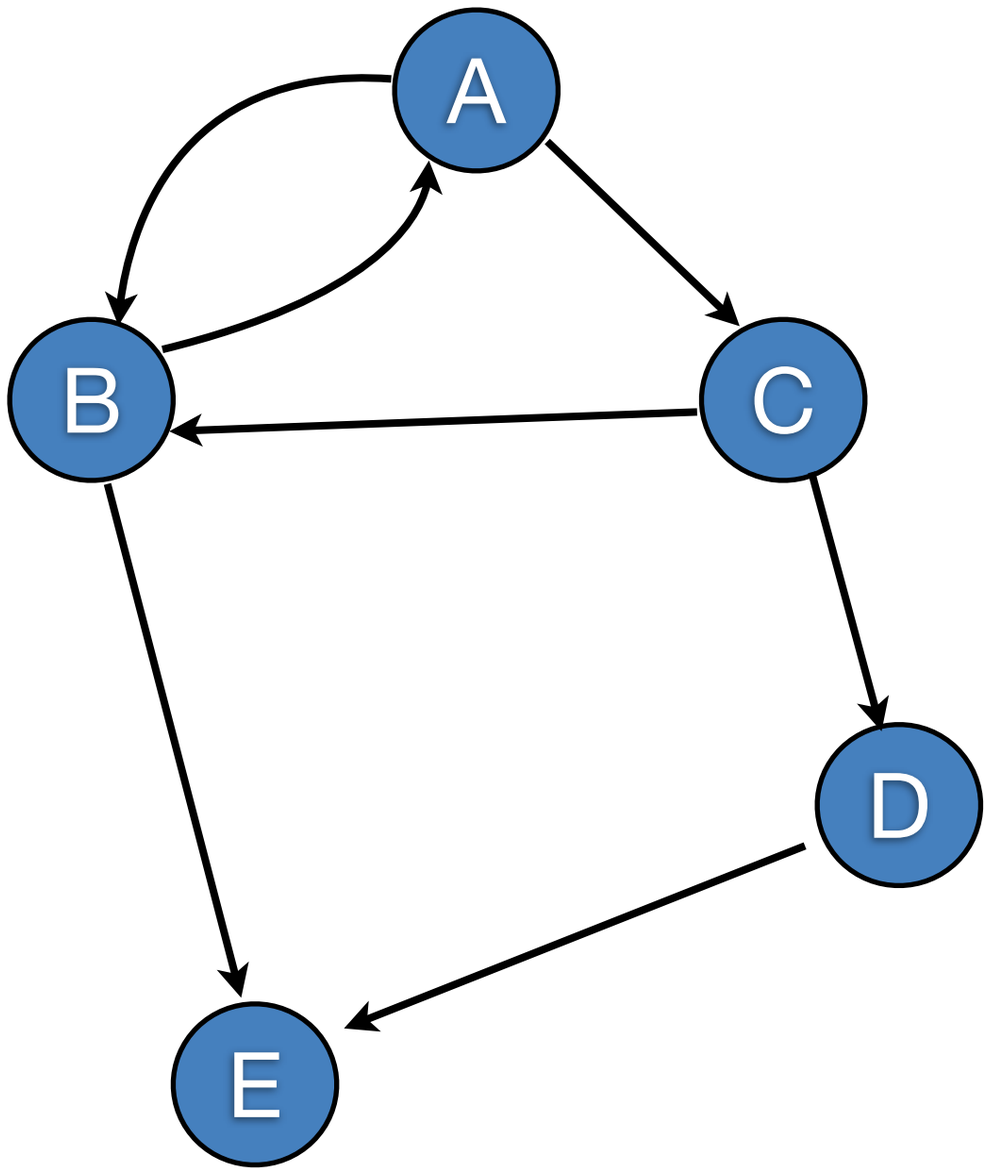
Google App Engine

- Programmer-specified

Relational databases

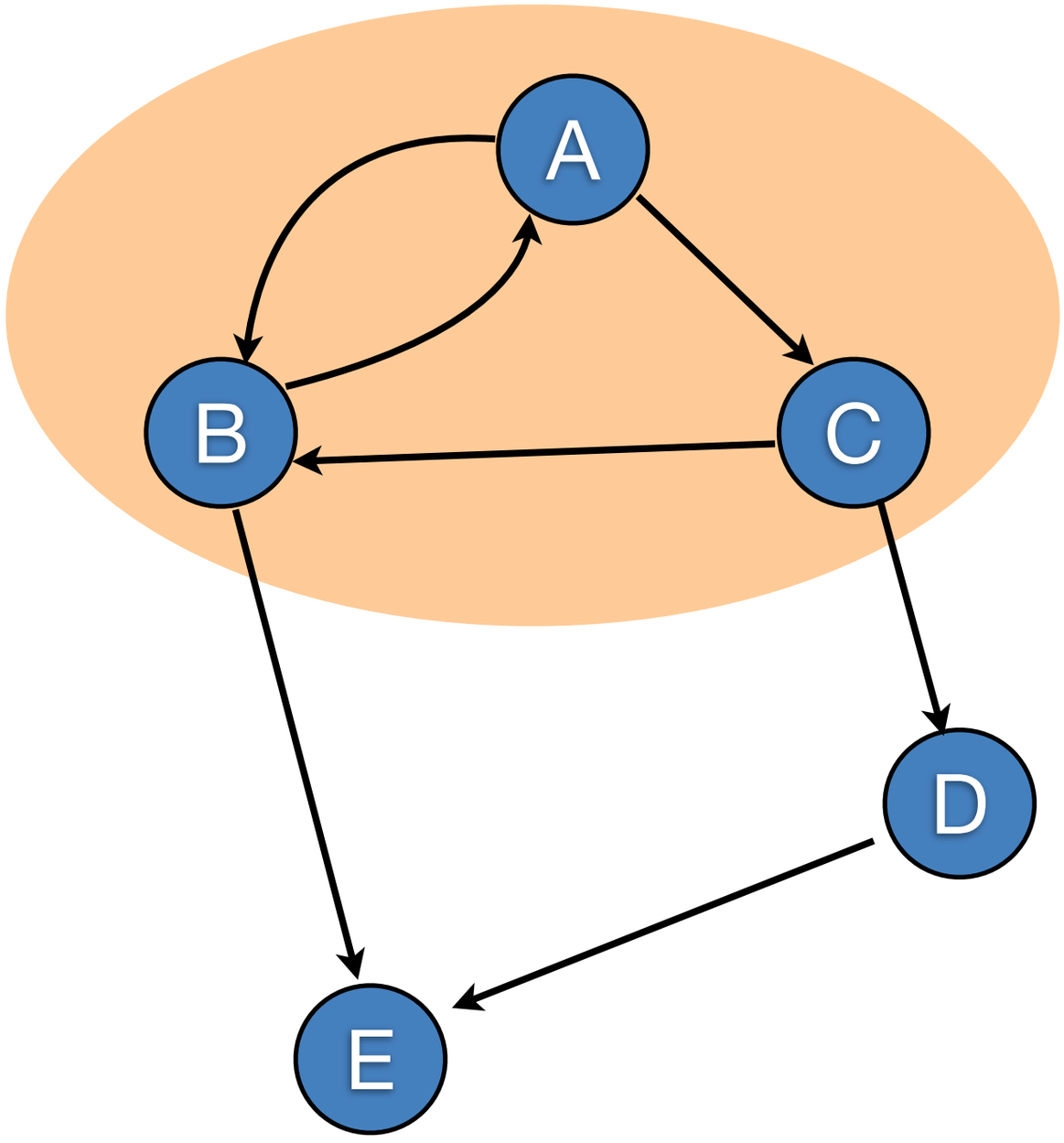
- Most transactions are simple, can be analyzed
- Few remaining transactions interfere w/ everything

Execution



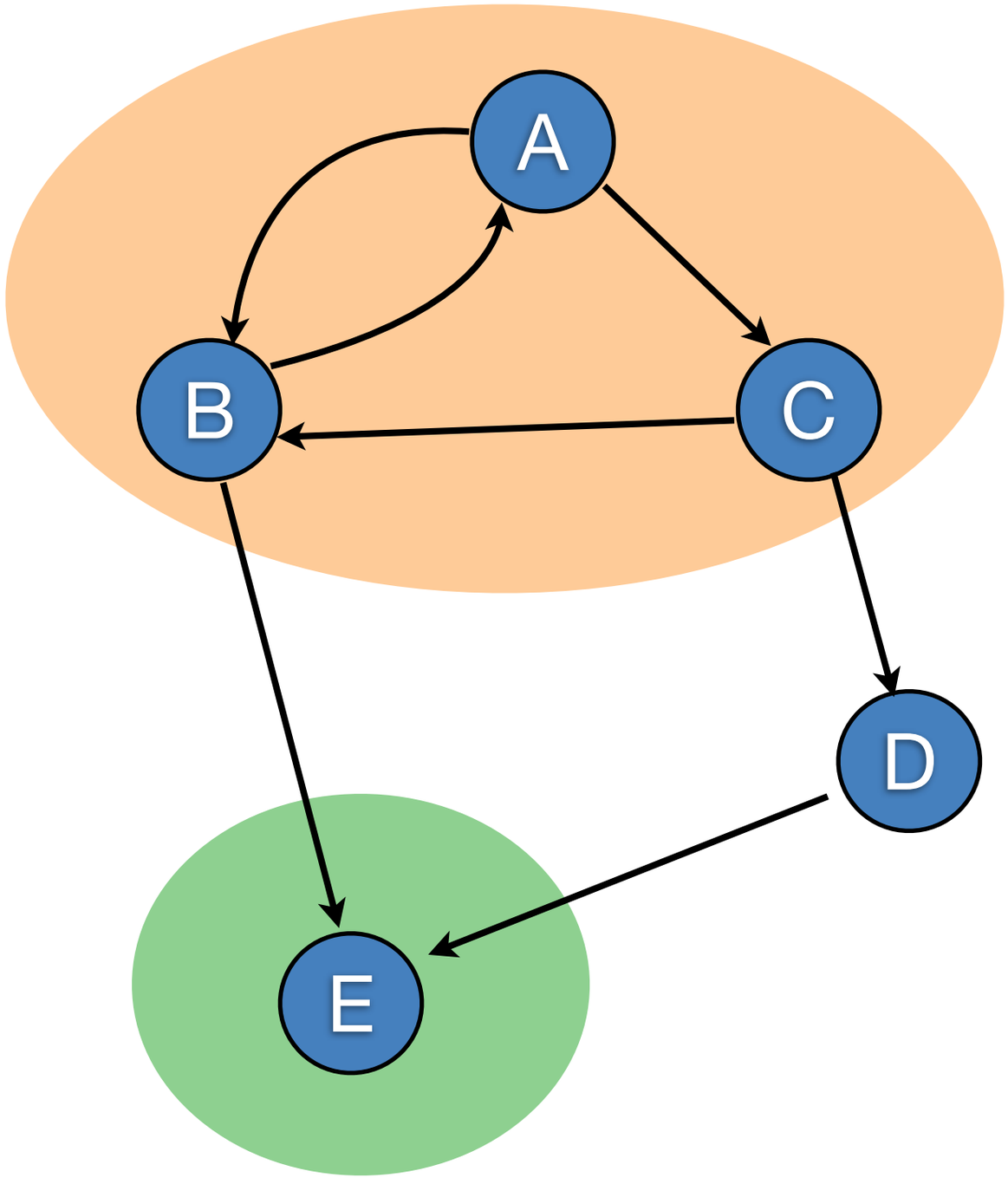
Execution

strongly-connected component



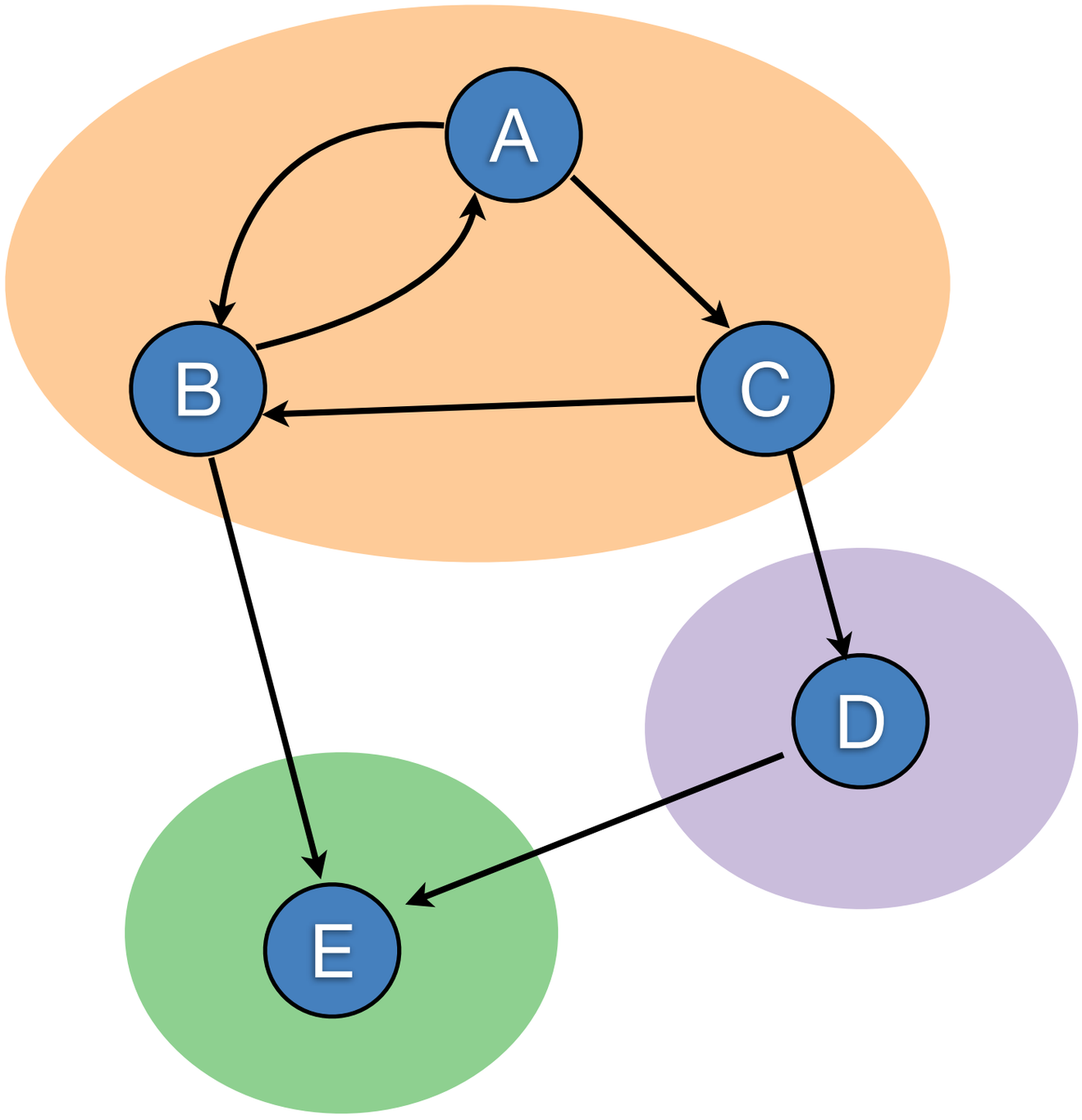
Execution

strongly-connected component



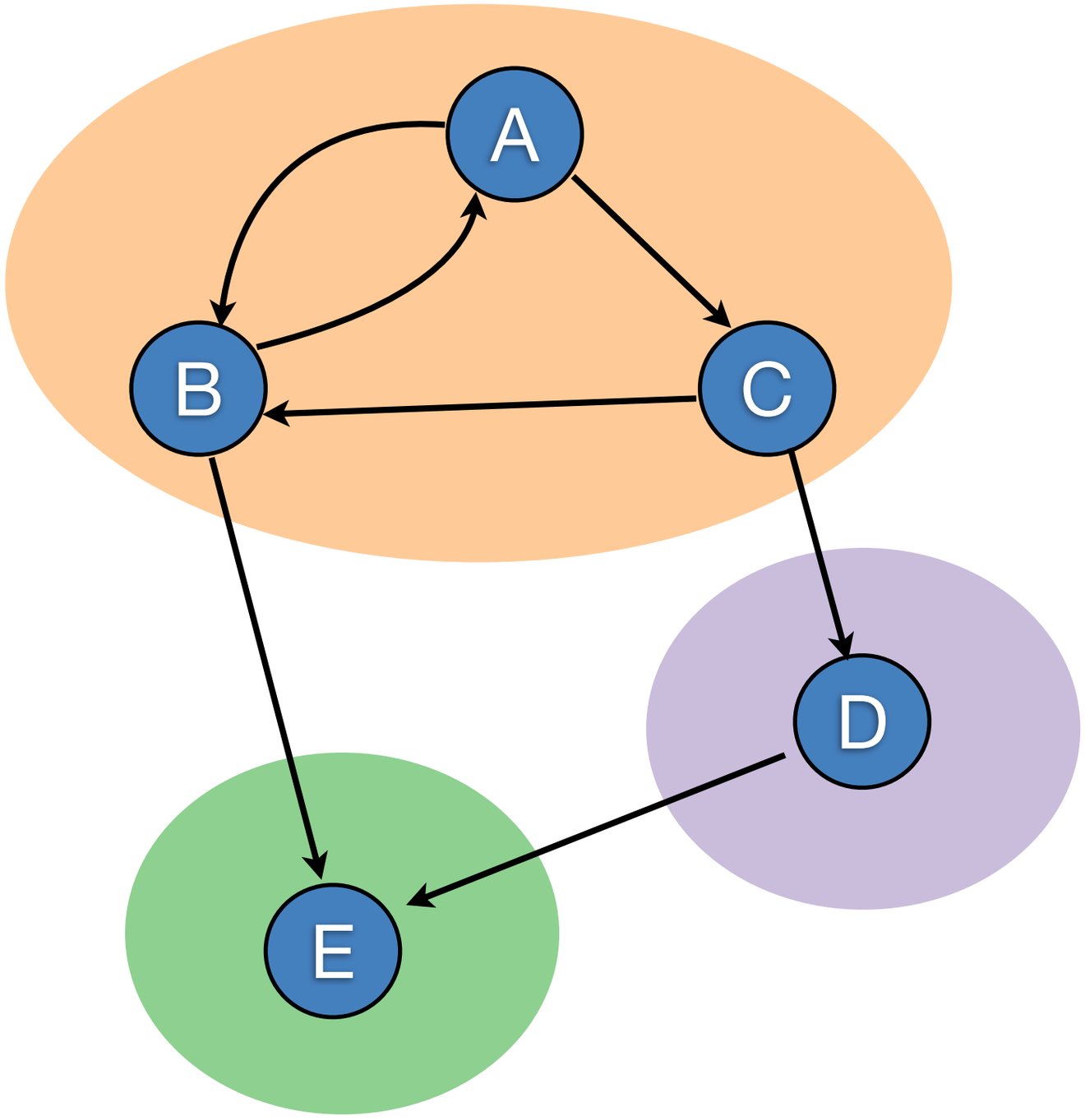
Execution

strongly-connected component

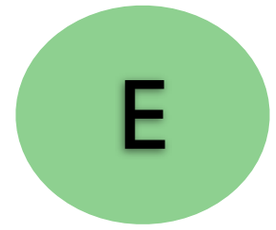


Execution

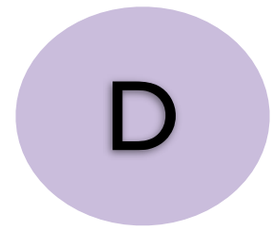
strongly-connected component



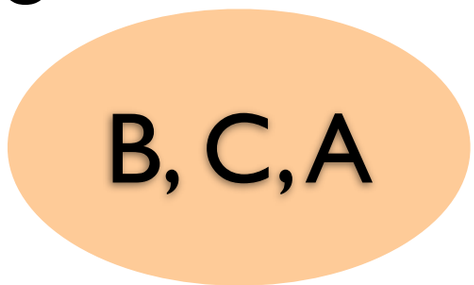
1



2

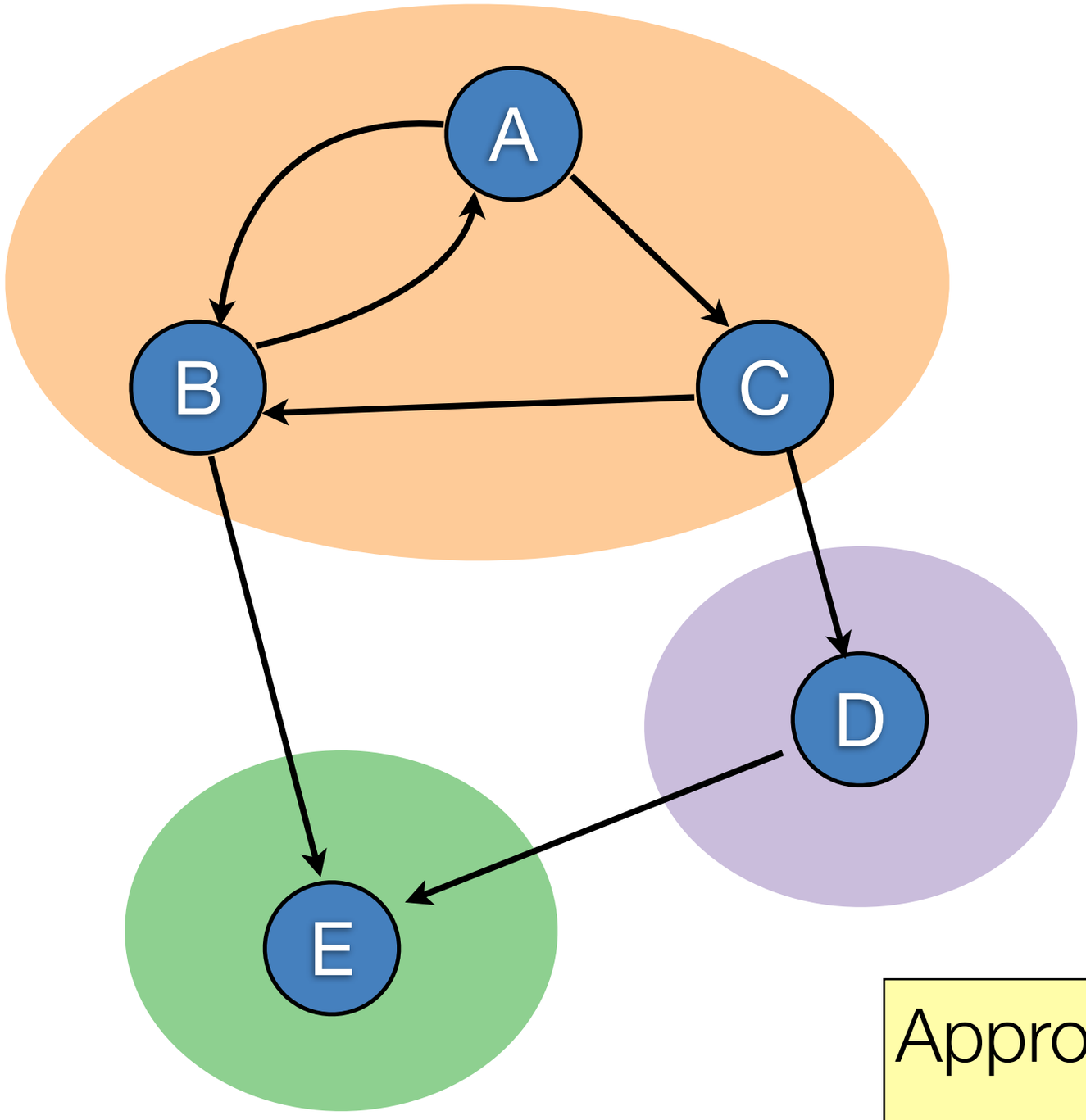


3

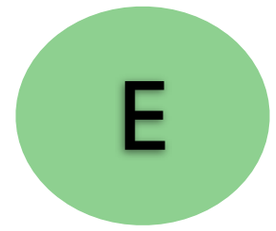


Execution

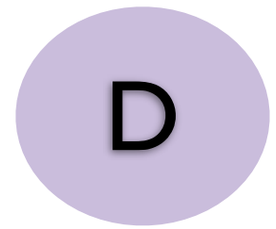
strongly-connected component



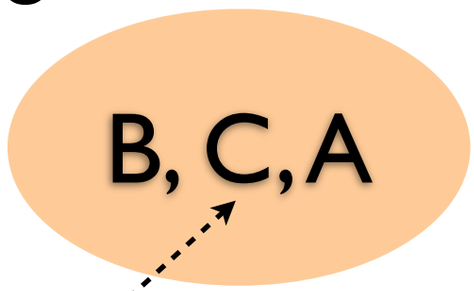
1



2



3



Approximate sequence # order (Lamport clock)

EPaxos properties

EPaxos properties

Linearizability: **If** A~B, and A committed before B proposed
then A will be executed before B.

EPaxos properties

Linearizability: **If** A~B, and A committed before B proposed
then A will be executed before B.

Fast-path quorum: $F + \lceil F / 2 \rceil$

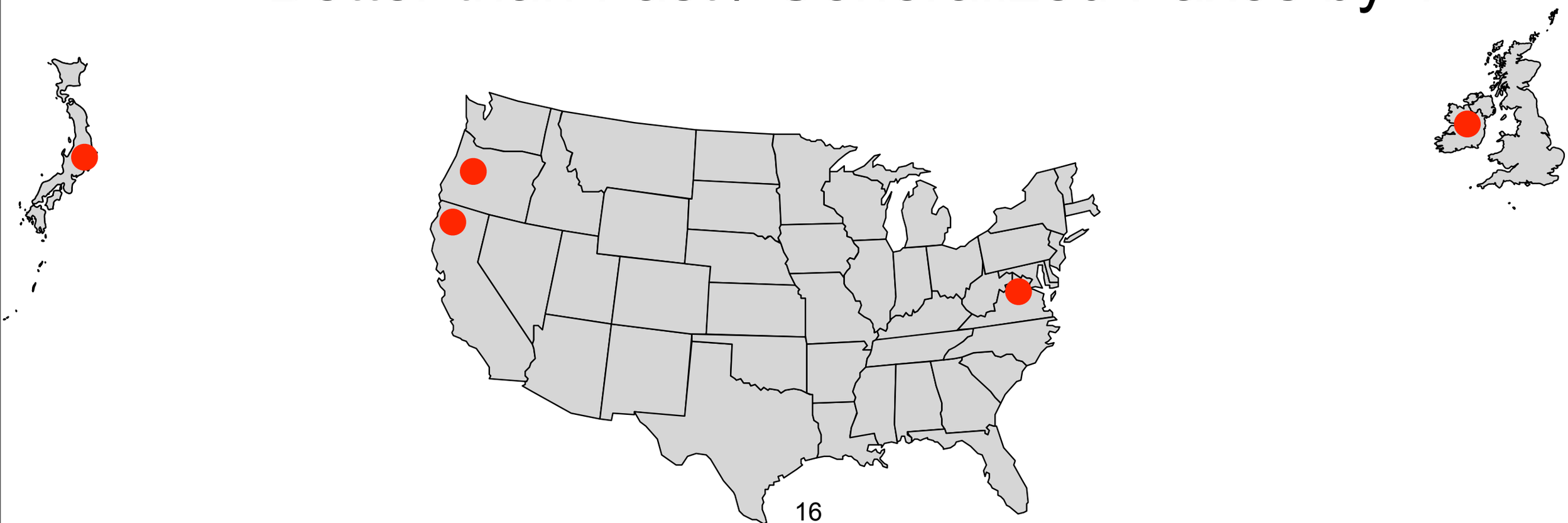
- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1

EPaxos properties

Linearizability: **If** A~B, and A committed before B proposed
then A will be executed before B.

Fast-path quorum: F + ⌈F / 2⌉

- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1

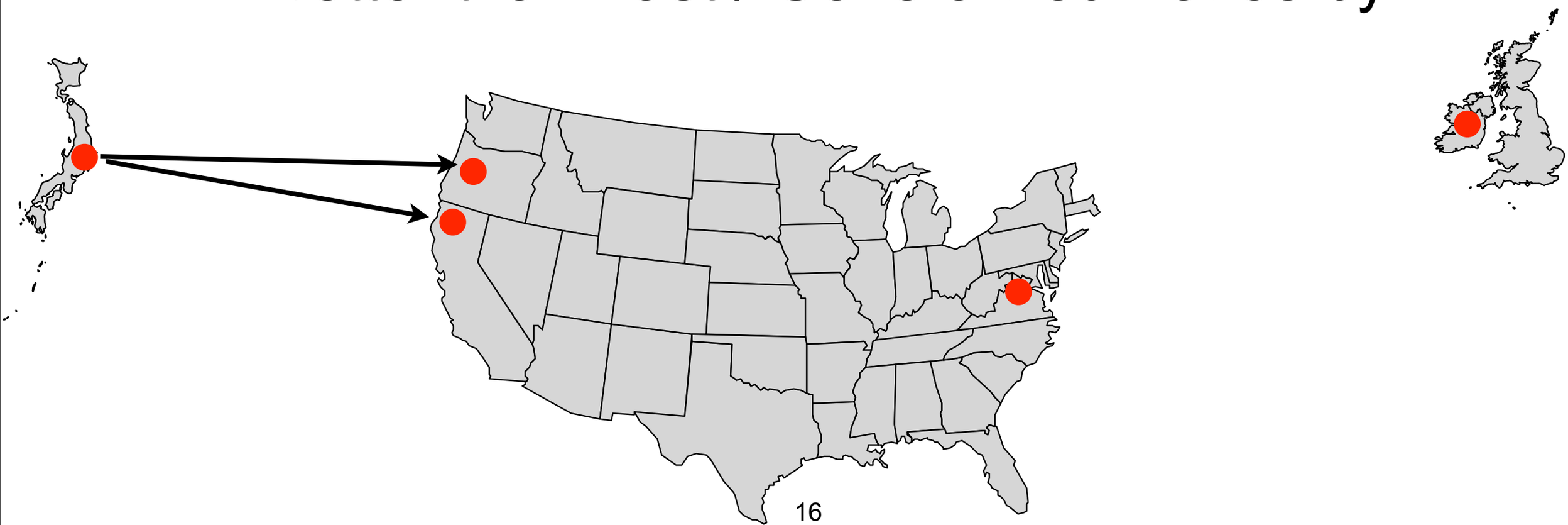


EPaxos properties

Linearizability: **If** A~B, and A committed before B proposed
then A will be executed before B.

Fast-path quorum: F + ⌈F / 2⌉

- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1

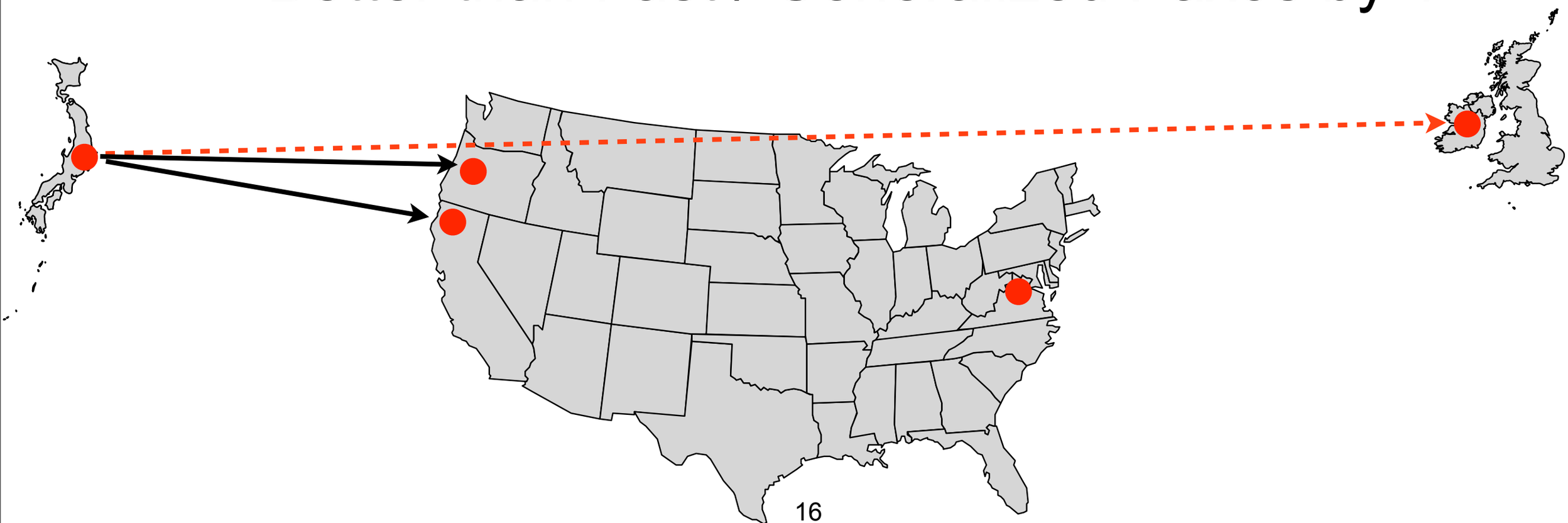


EPaxos properties

Linearizability: **If** A~B, and A committed before B proposed
then A will be executed before B.

Fast-path quorum: F + ⌈F / 2⌉

- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1

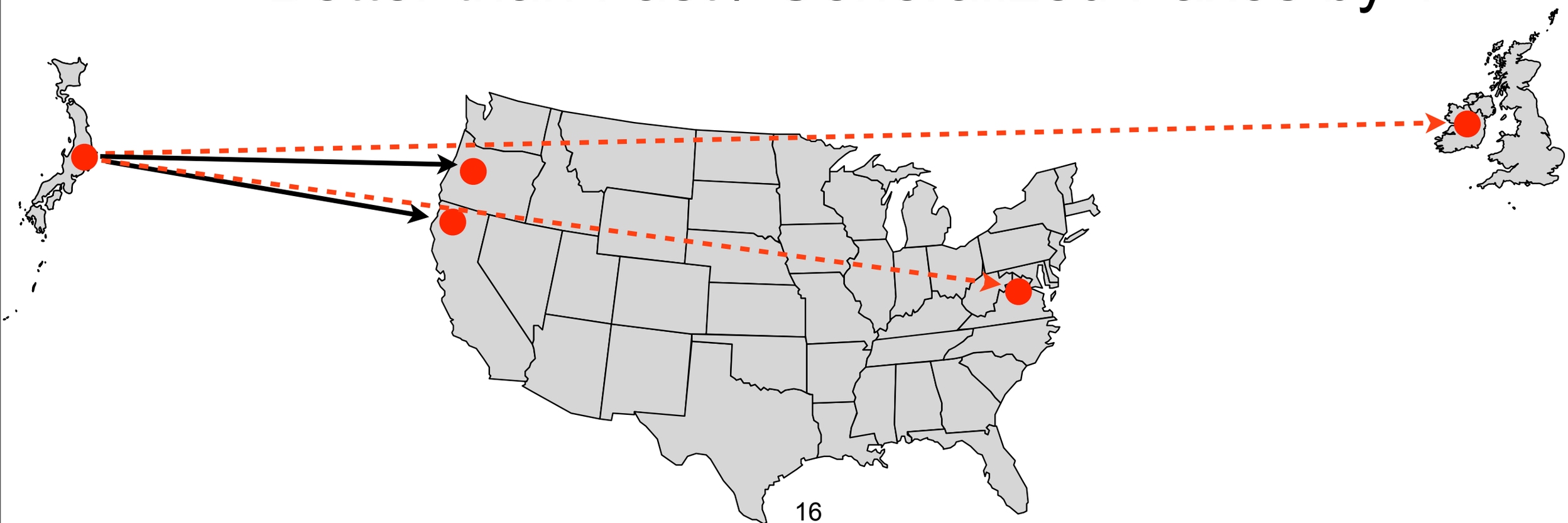


EPaxos properties

Linearizability: **If** A~B, and A committed before B proposed
then A will be executed before B.

Fast-path quorum: F + ⌈F / 2⌉

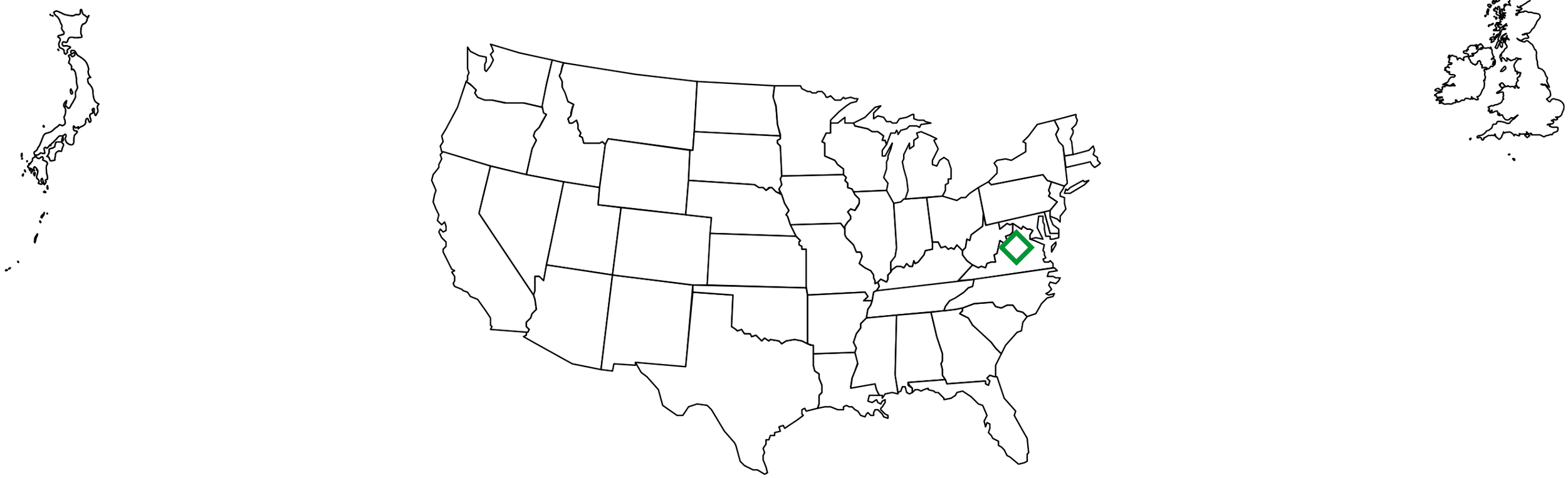
- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1



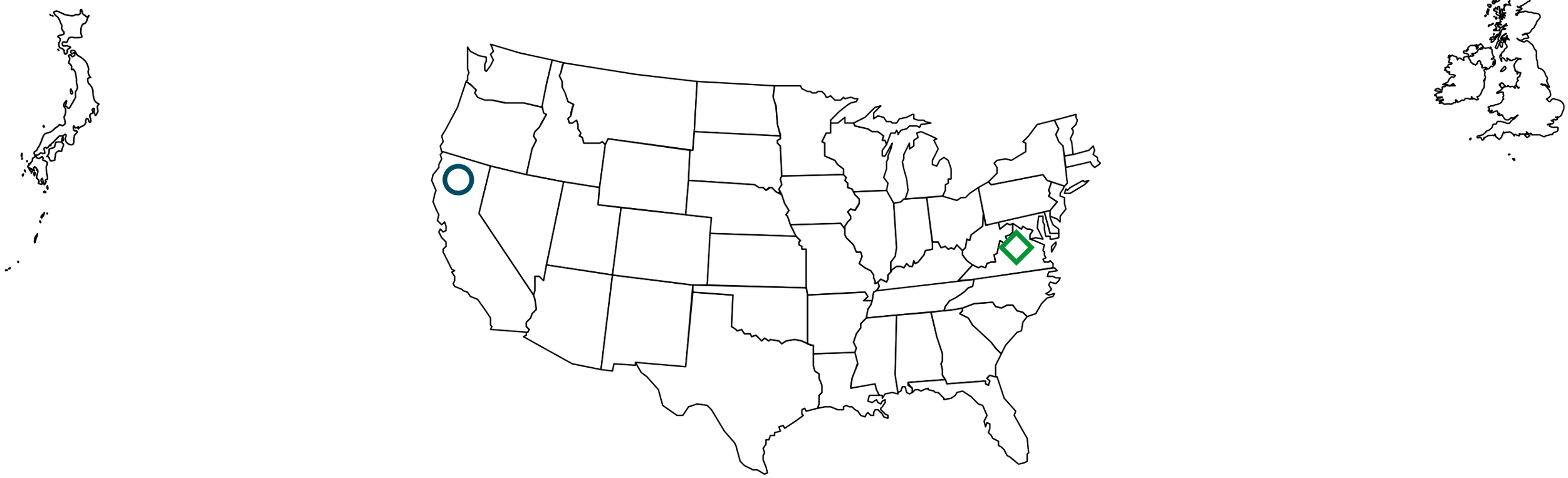
Results

Optimal wide-area commit latency

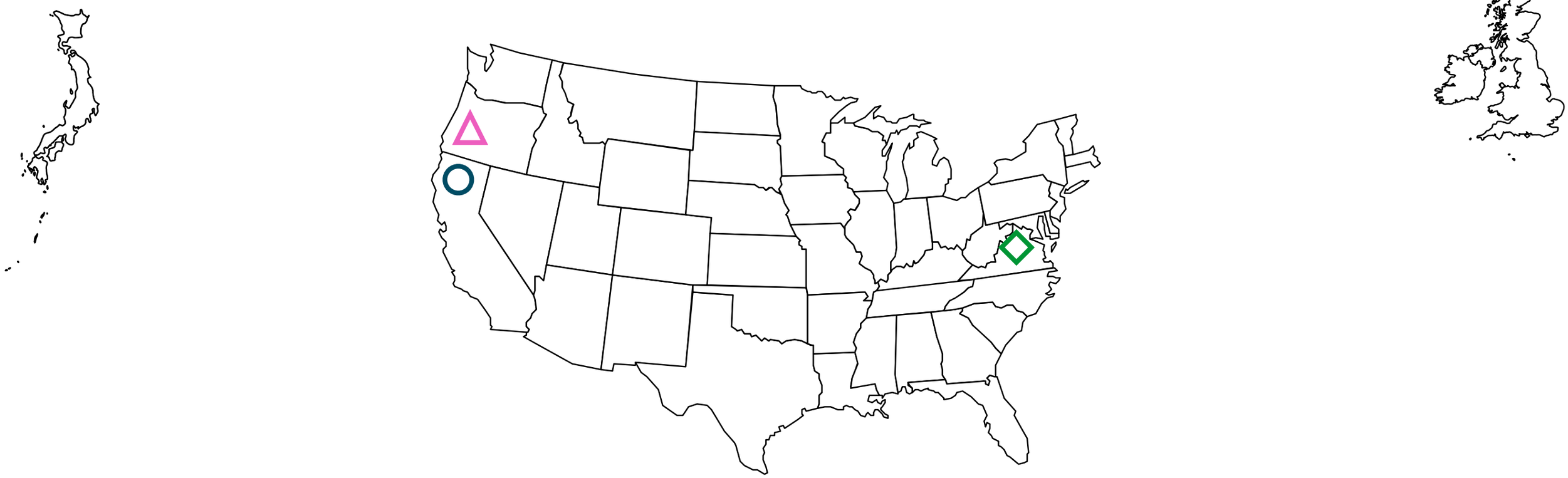
Optimal wide-area commit latency



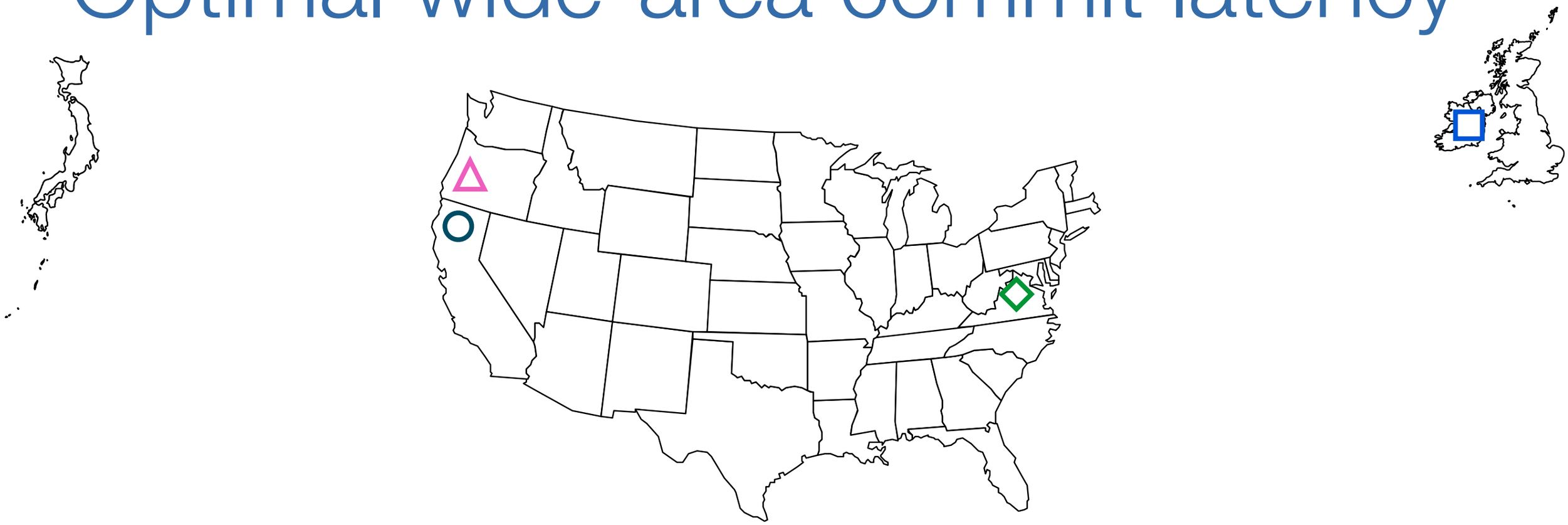
Optimal wide-area commit latency



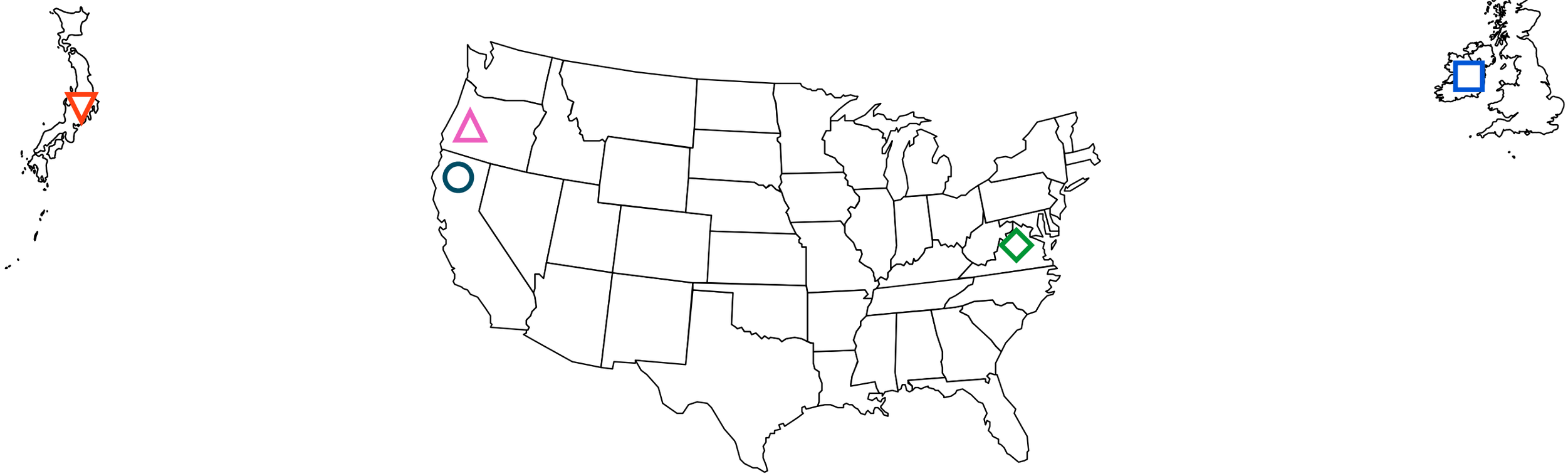
Optimal wide-area commit latency



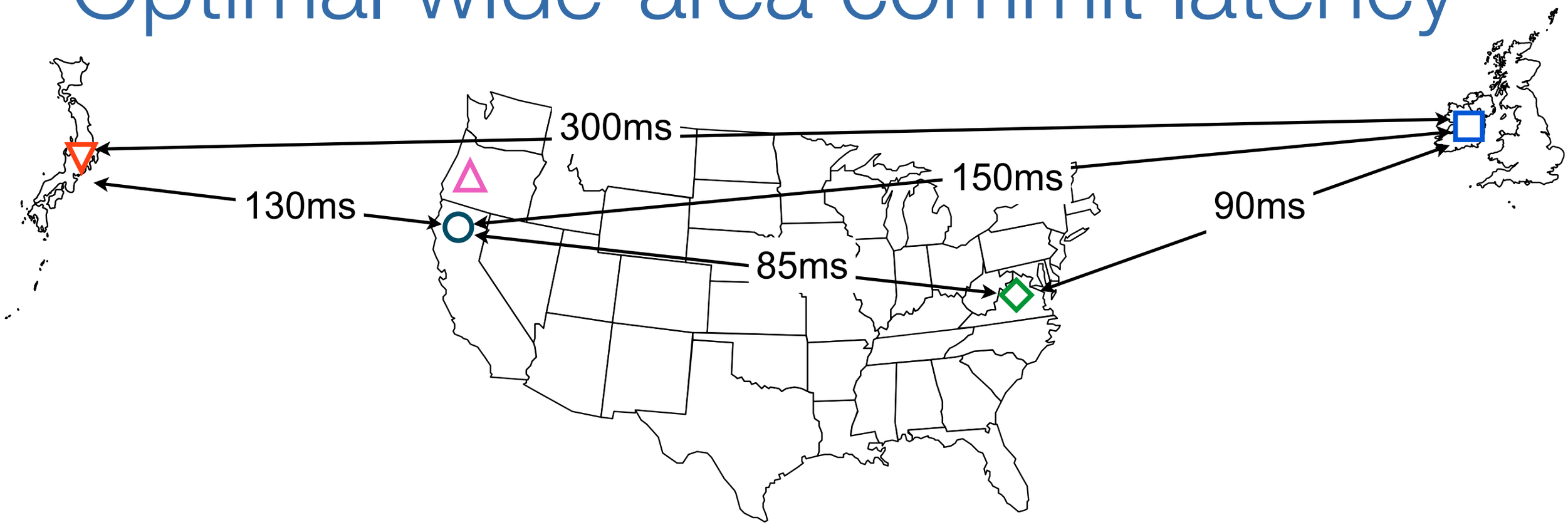
Optimal wide-area commit latency



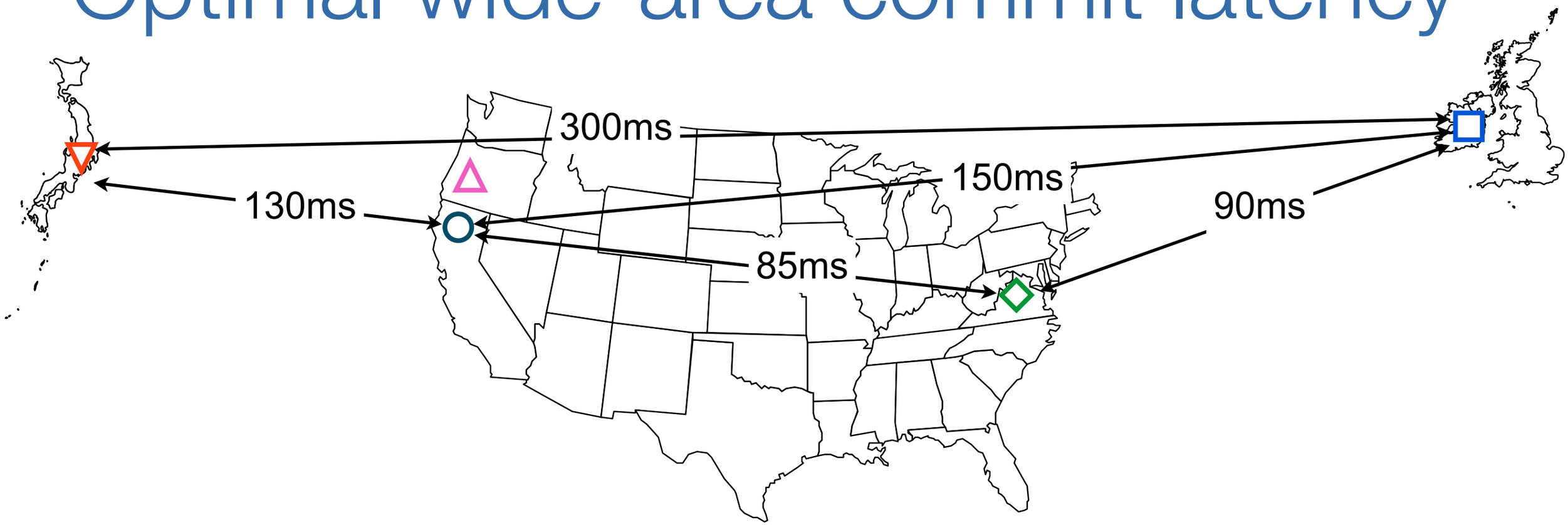
Optimal wide-area commit latency

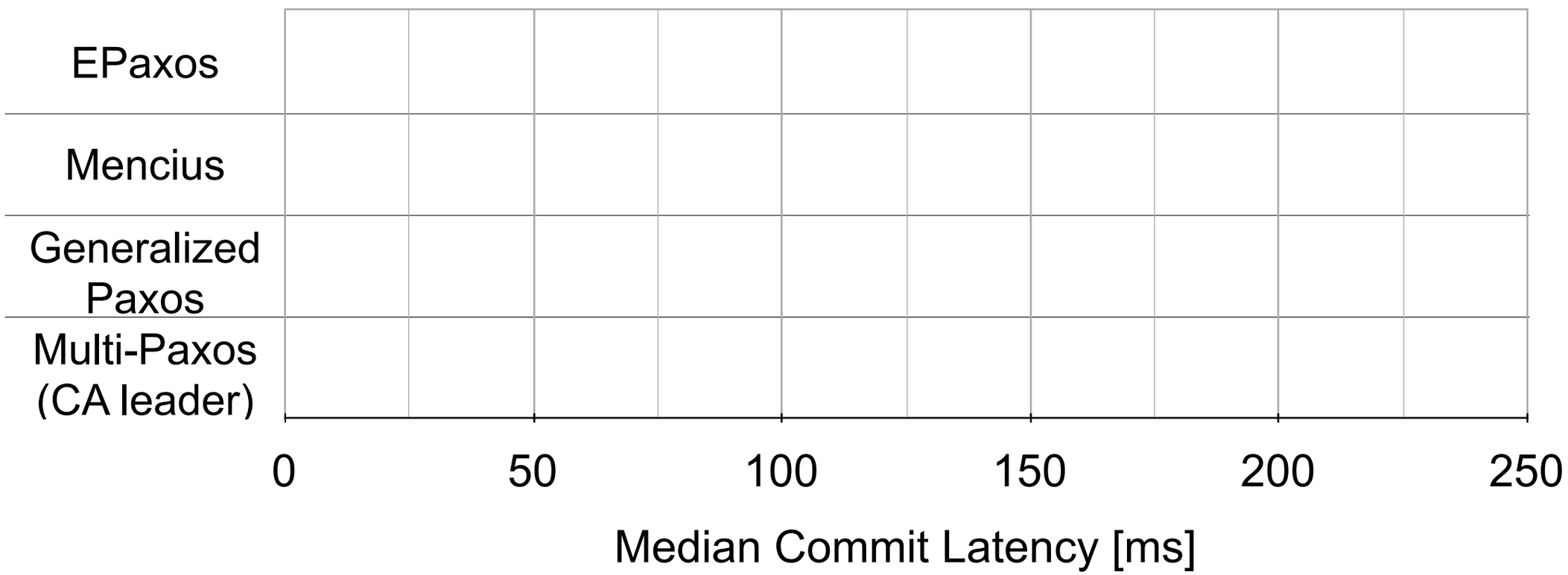


Optimal wide-area commit latency

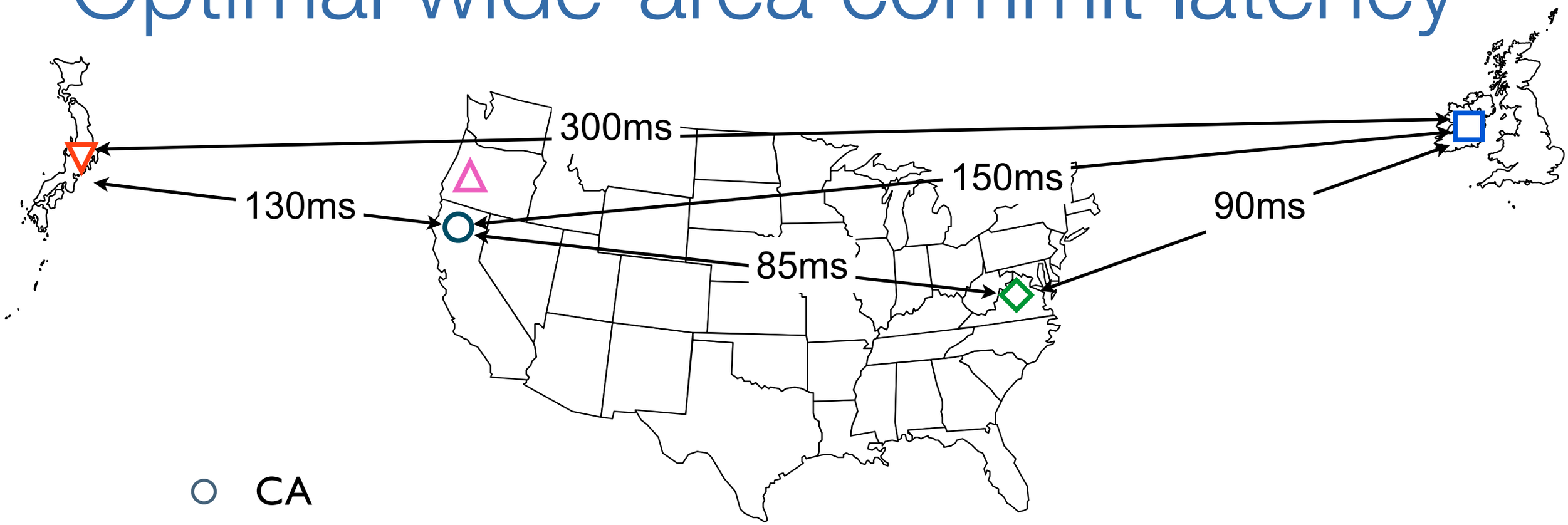


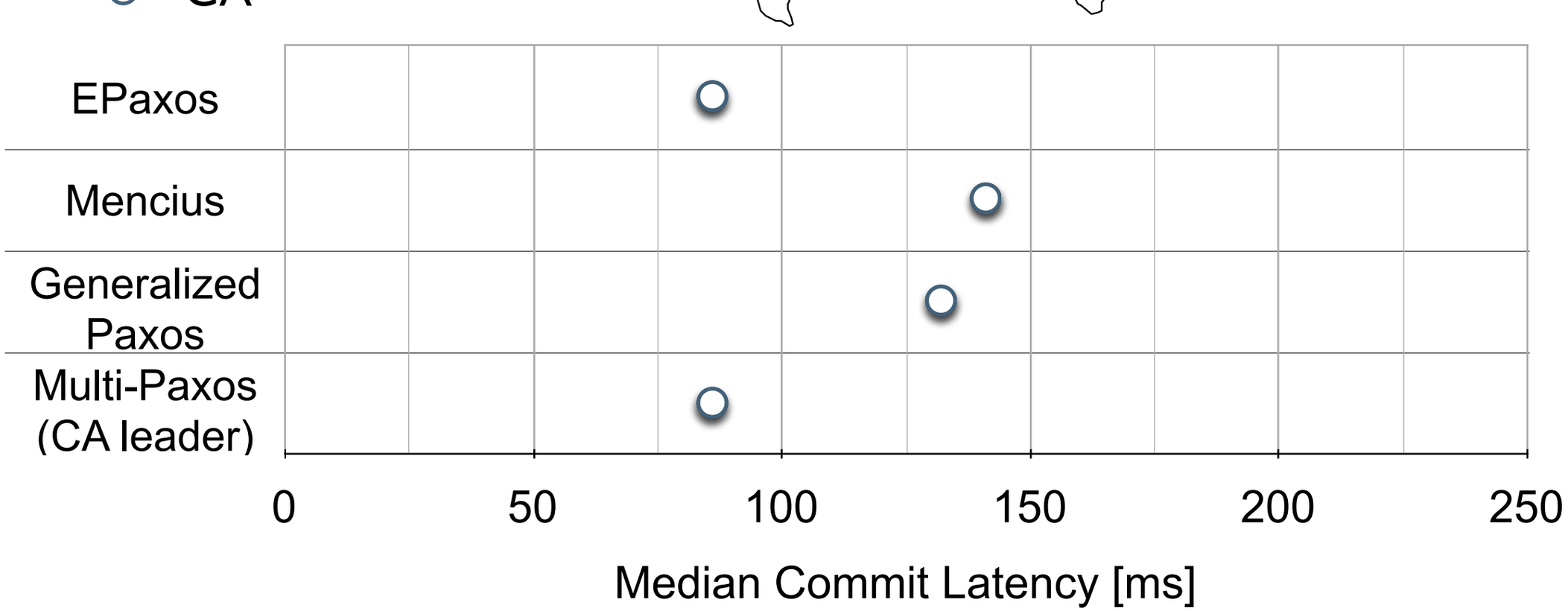
Optimal wide-area commit latency



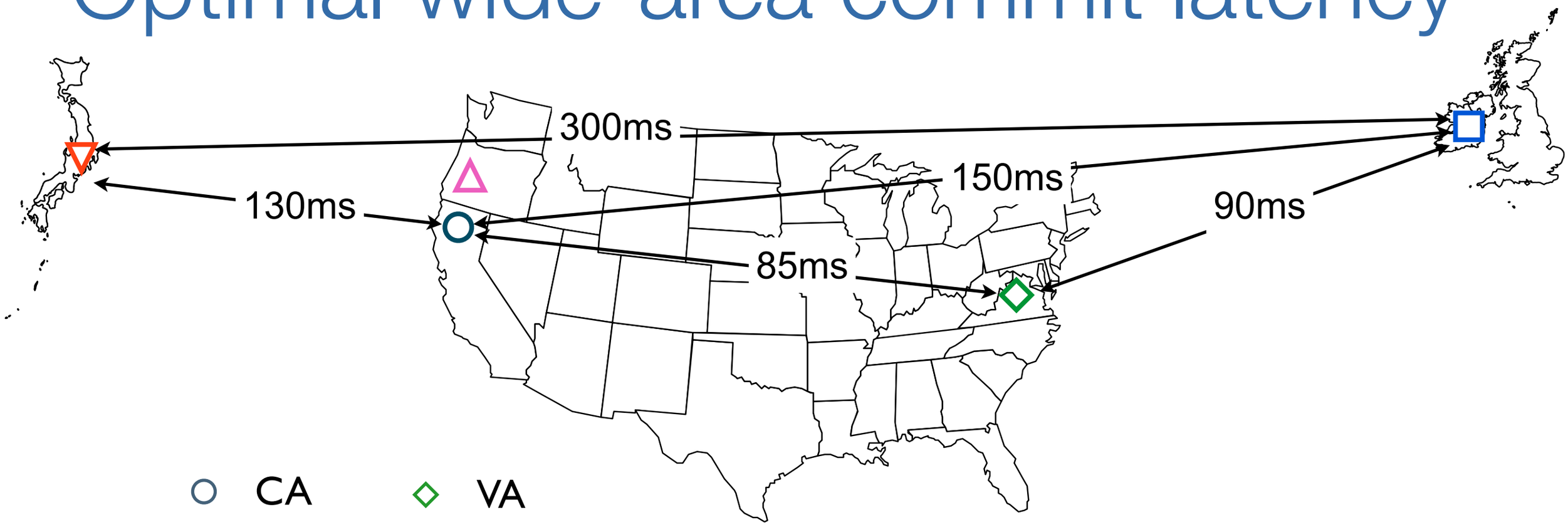


Optimal wide-area commit latency

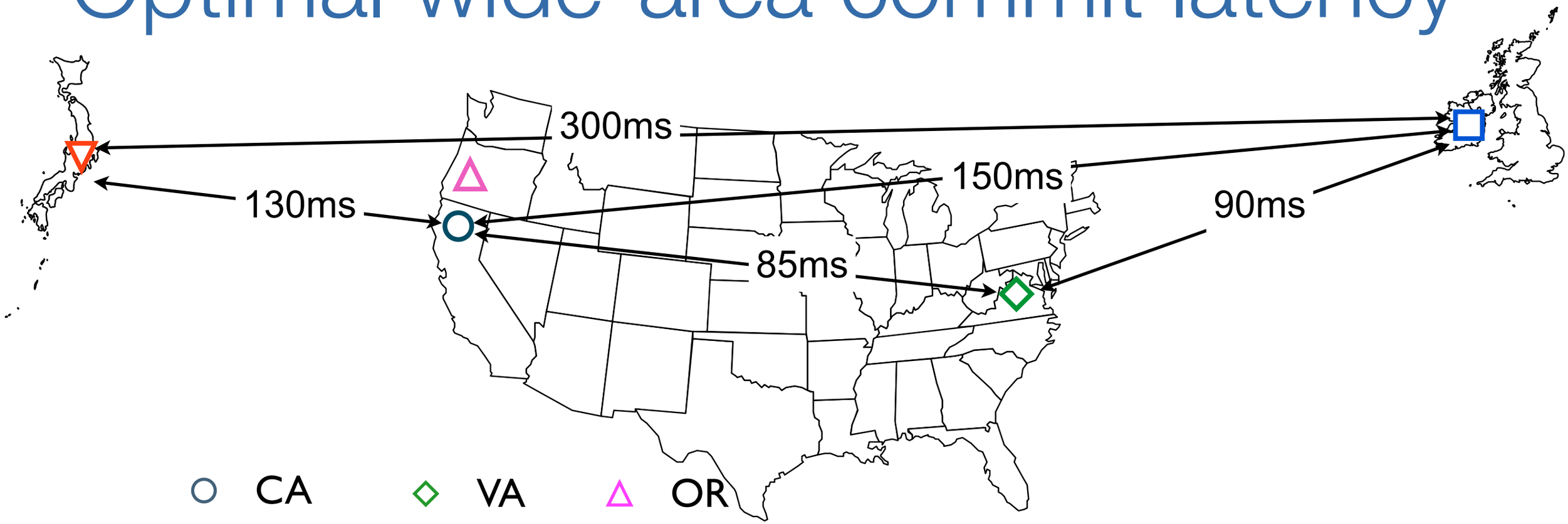




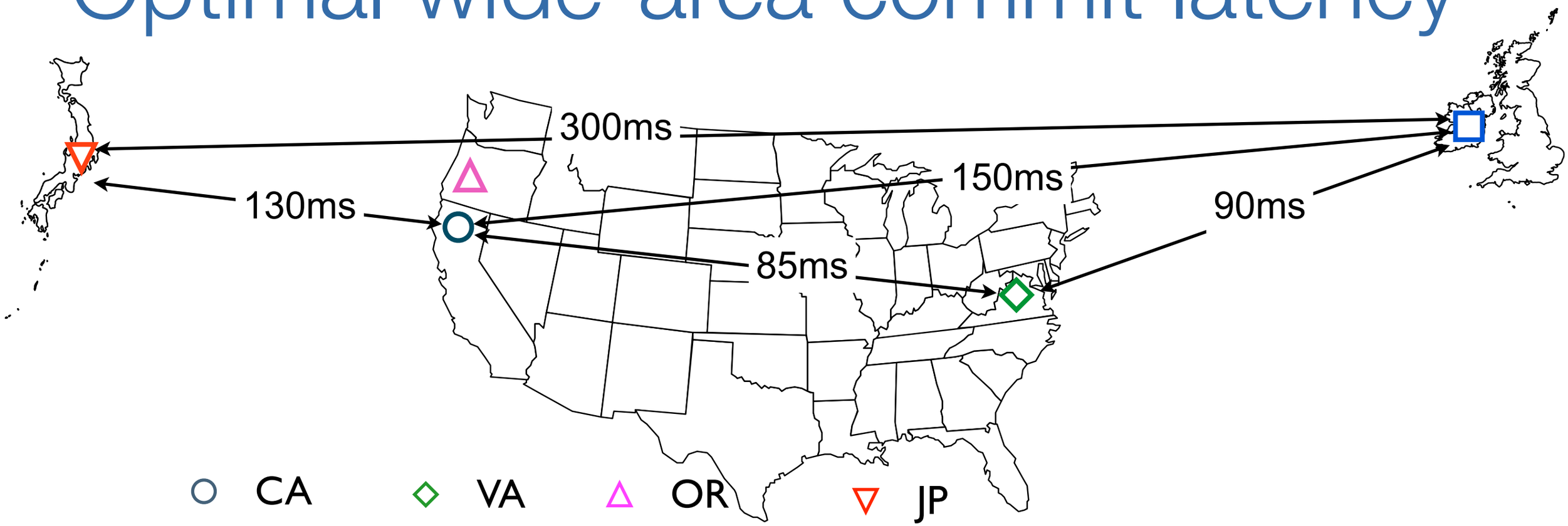
Optimal wide-area commit latency



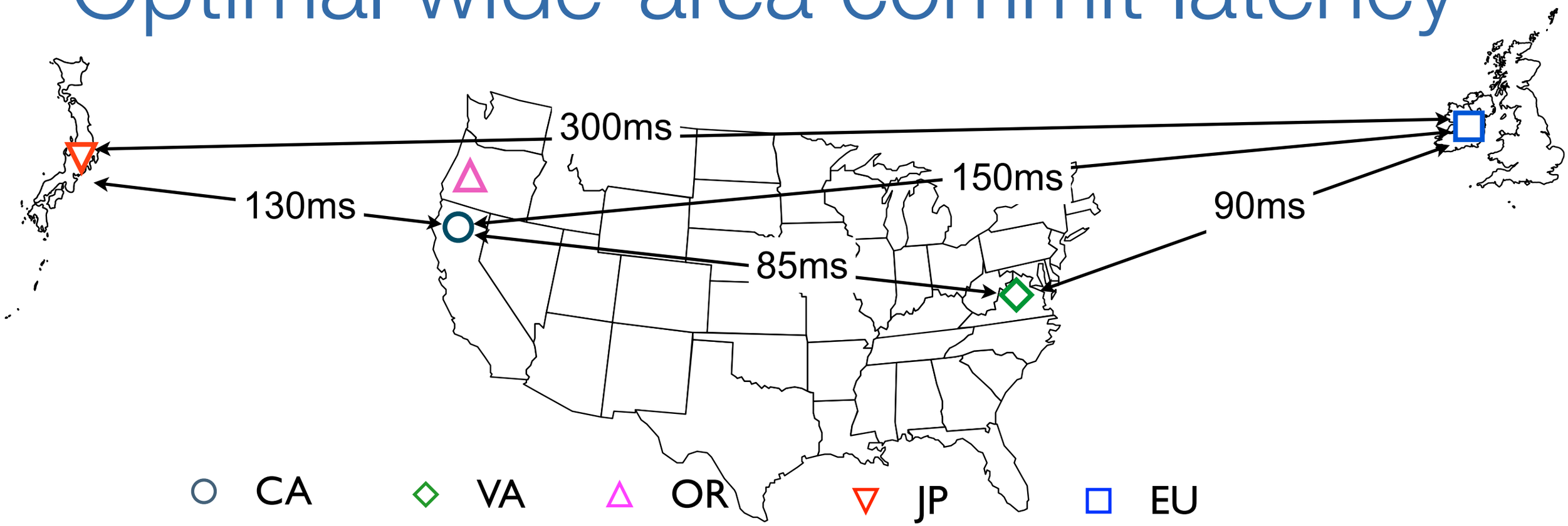
Optimal wide-area commit latency

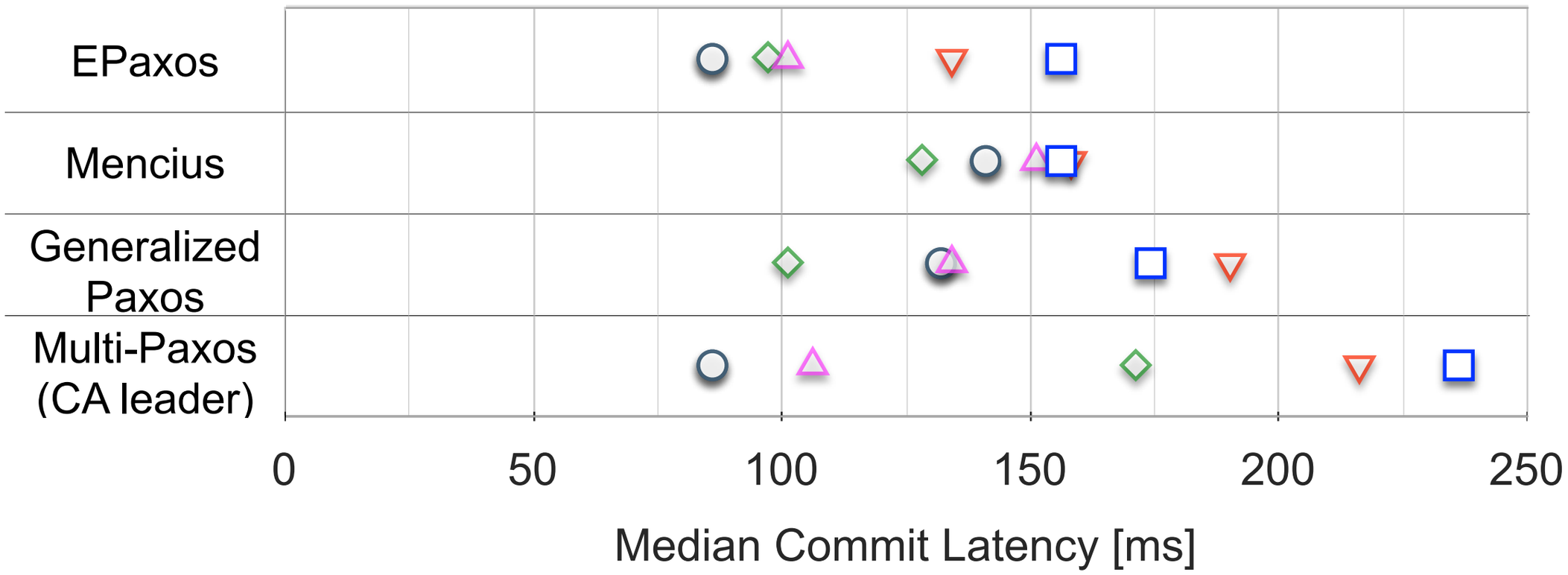


Optimal wide-area commit latency

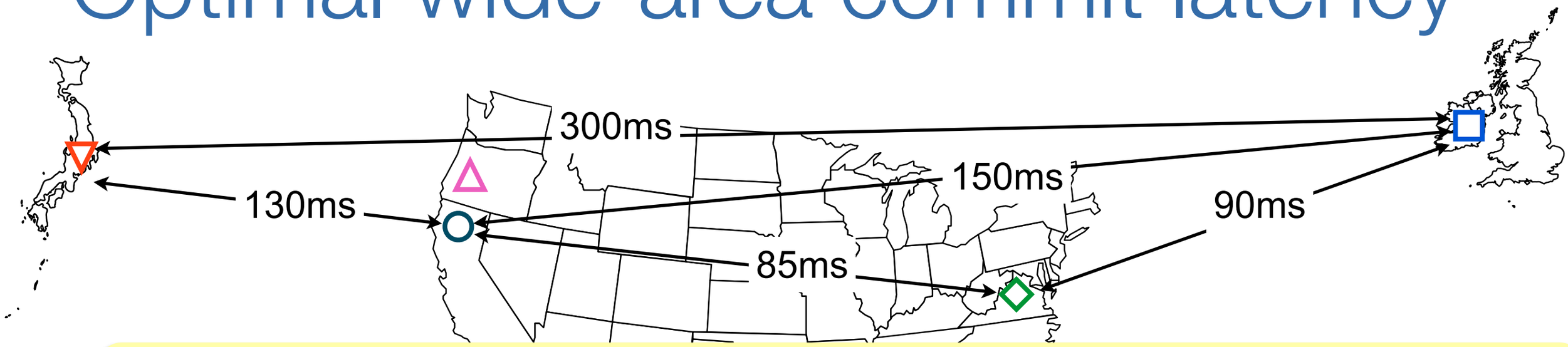


Optimal wide-area commit latency

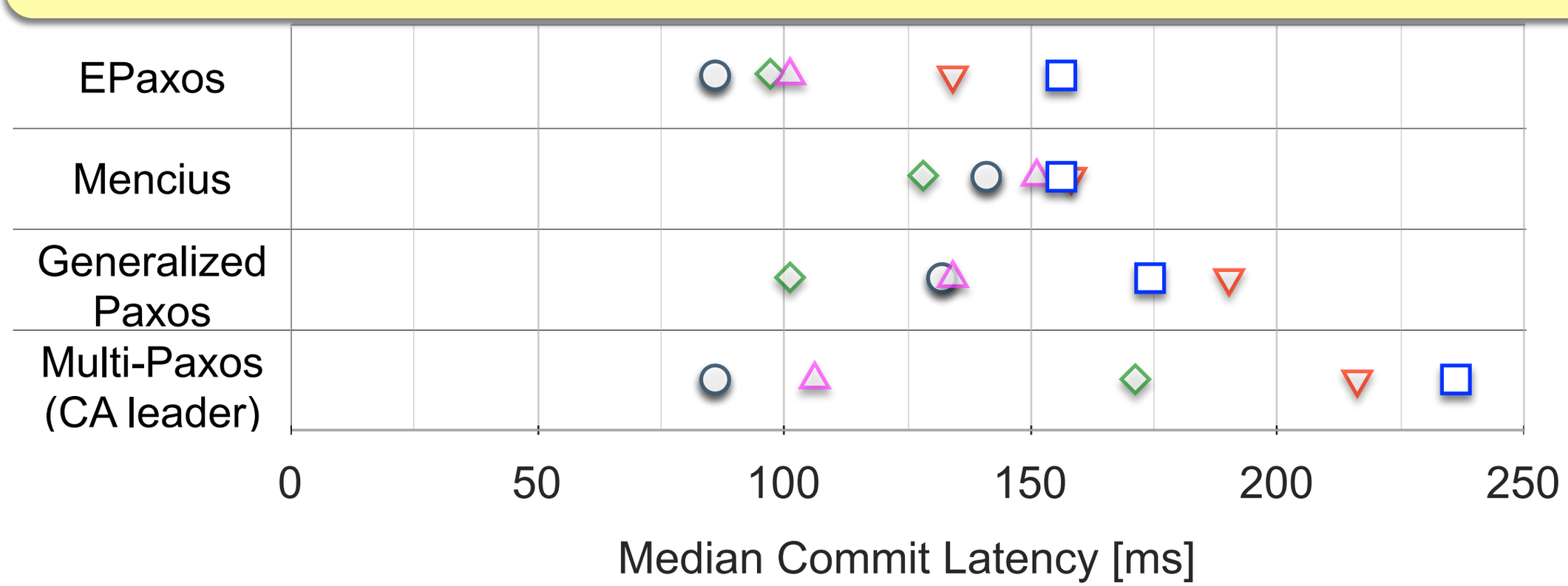




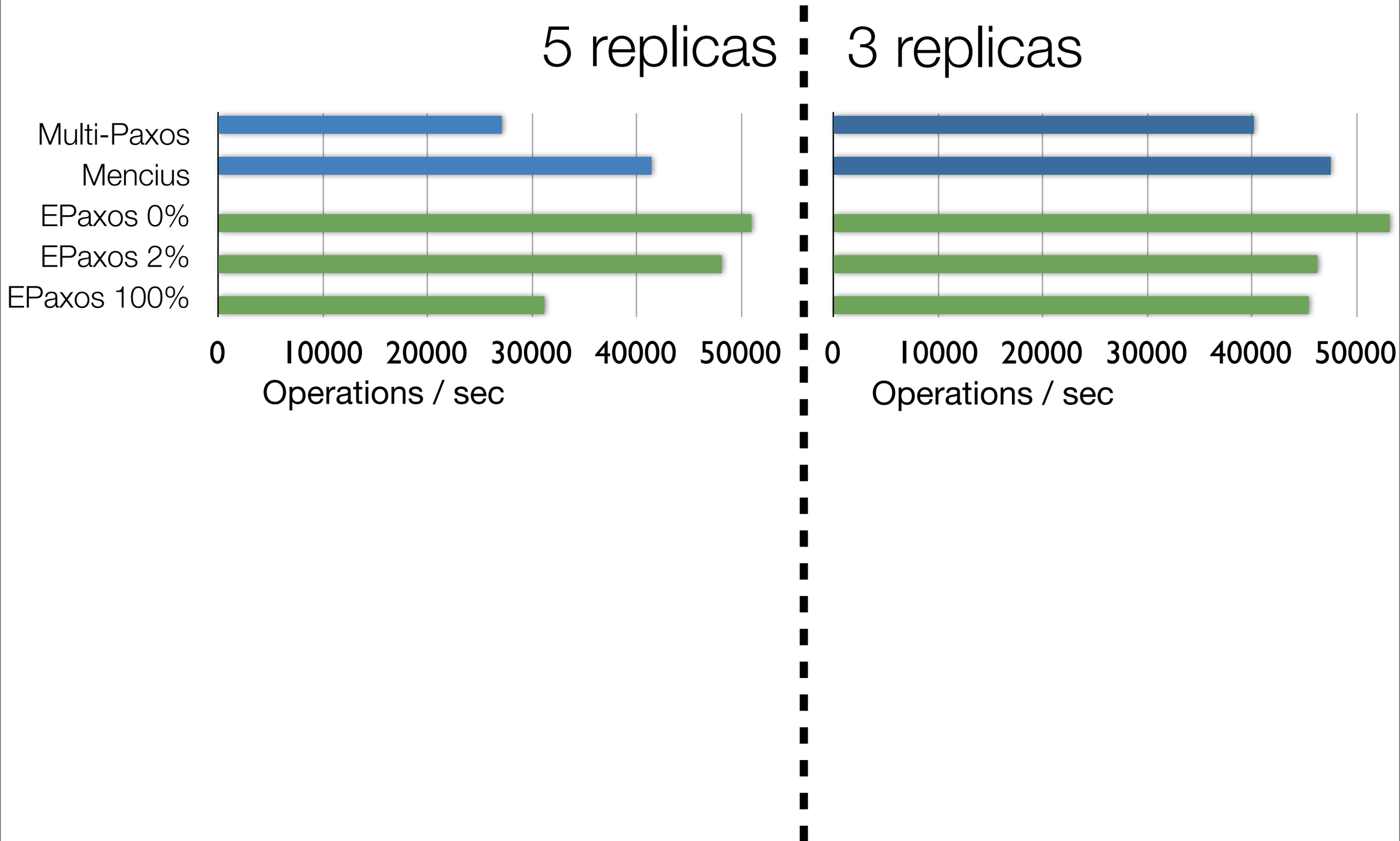
Optimal wide-area commit latency



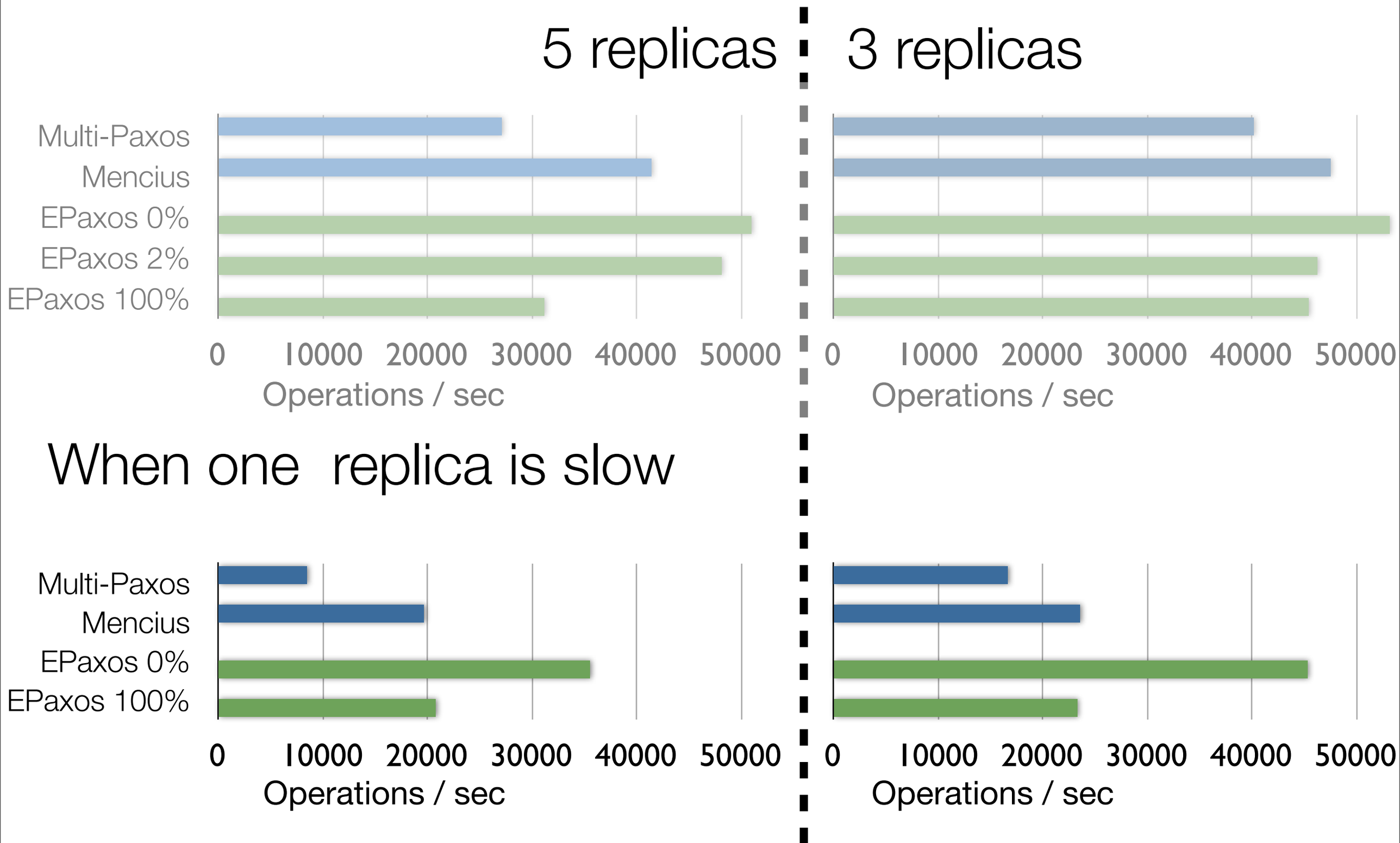
EPaxos: Optimal commit latency in wide-area for 3 and 5 replicas



Higher + more stable throughput



Higher + more stable throughput

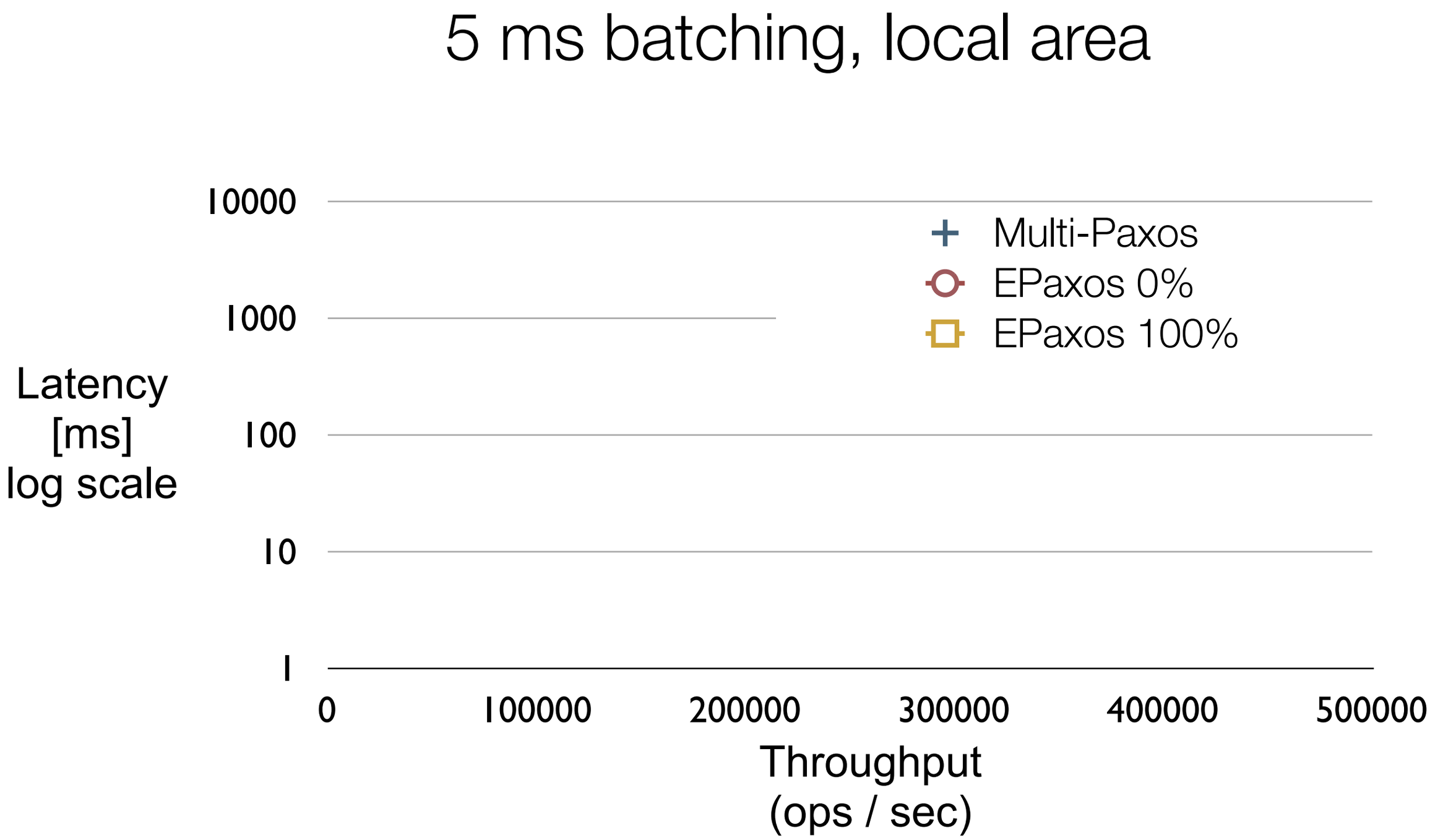


When one replica is slow

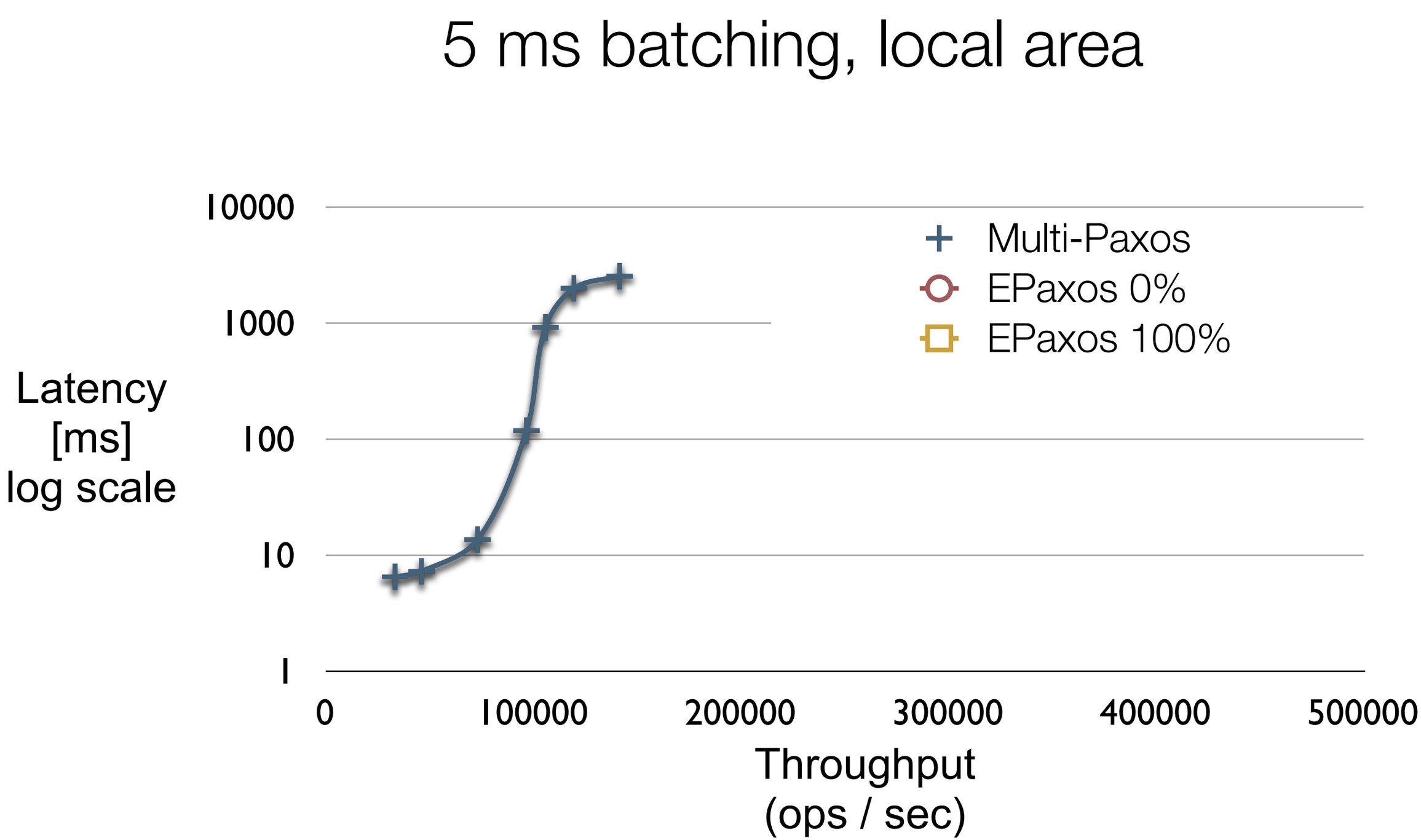
EPaxos: higher throughput w/ batching

5 ms batching, local area

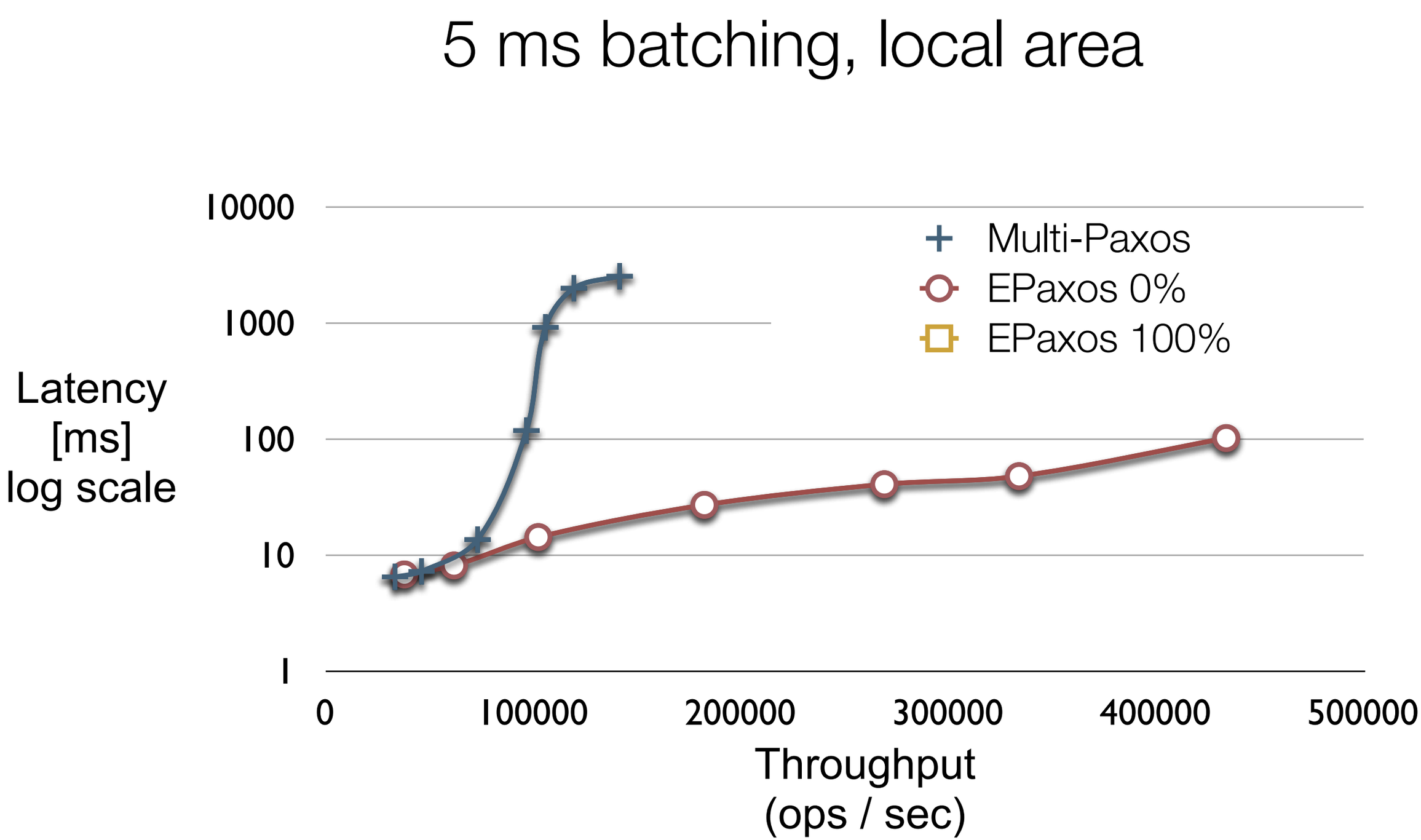
EPaxos: higher throughput w/ batching



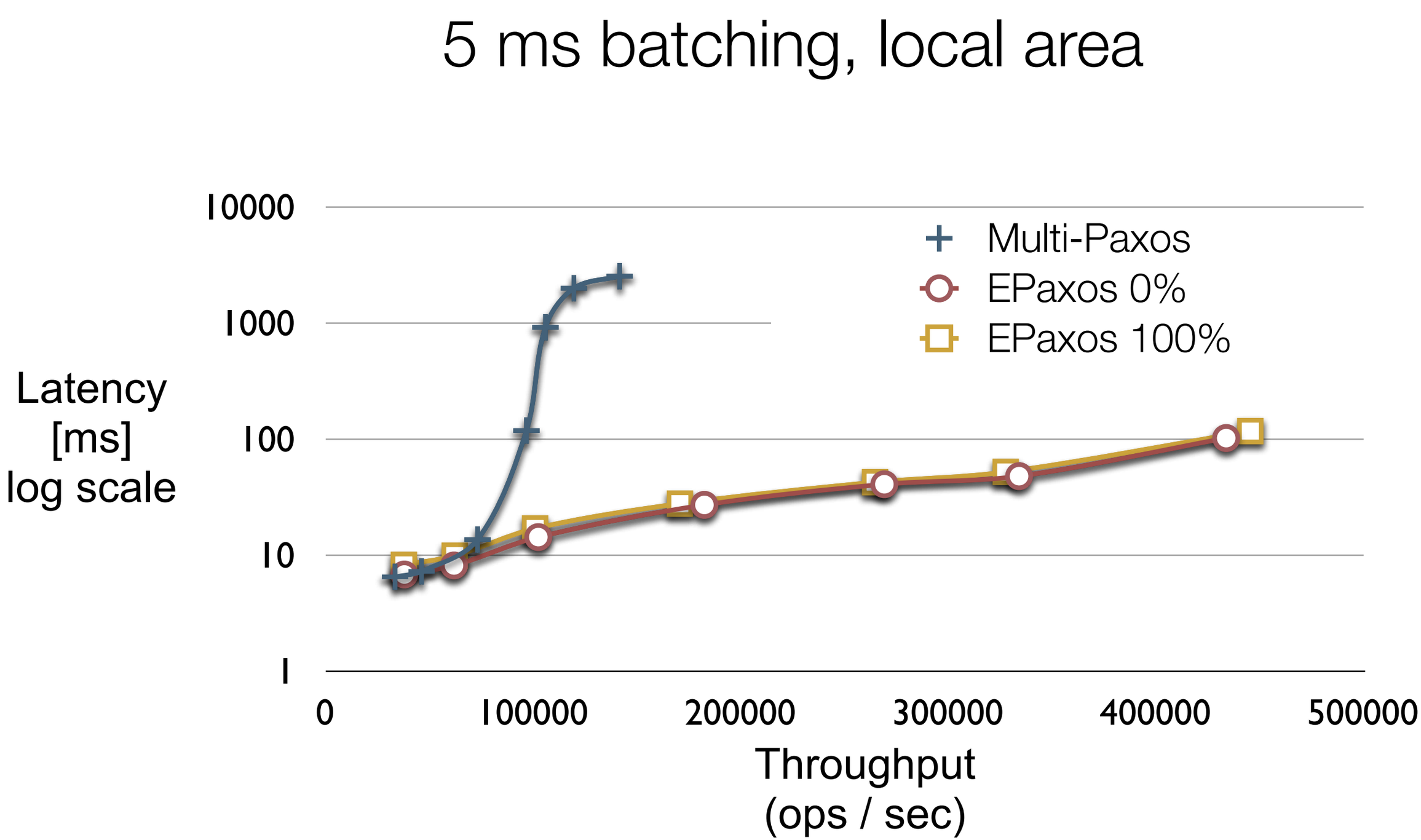
EPaxos: higher throughput w/ batching



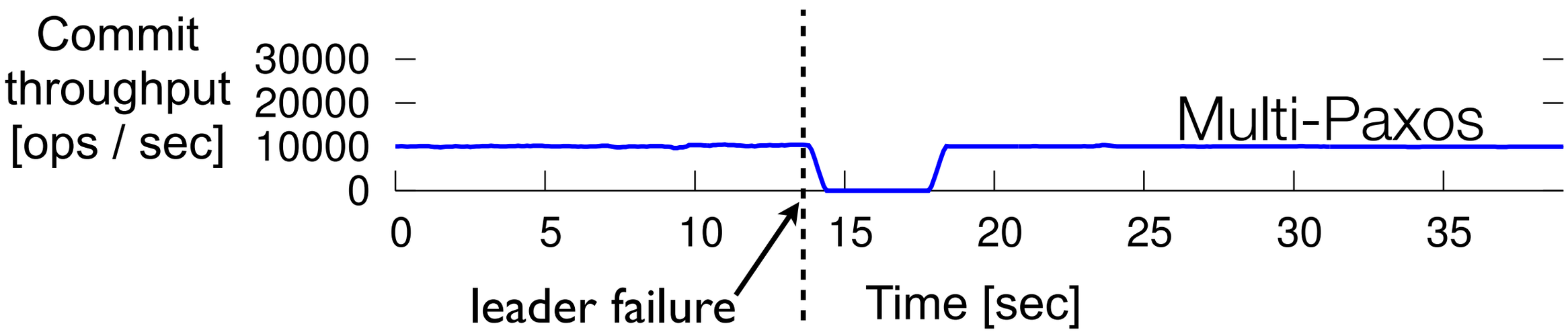
EPaxos: higher throughput w/ batching



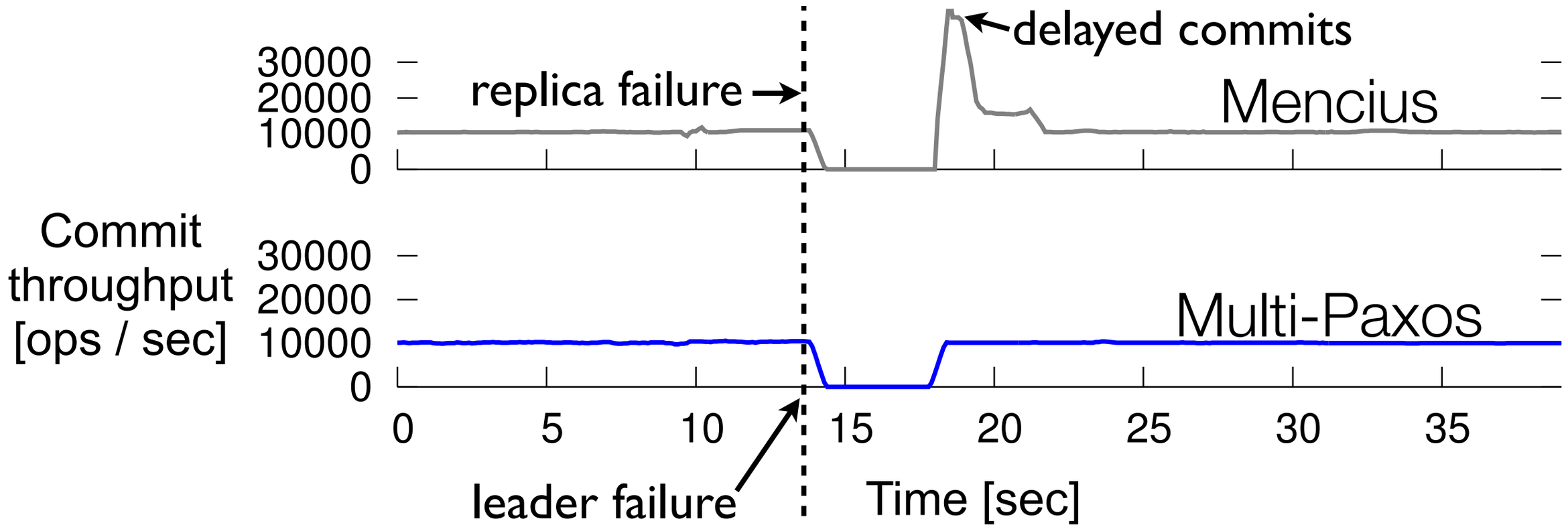
EPaxos: higher throughput w/ batching



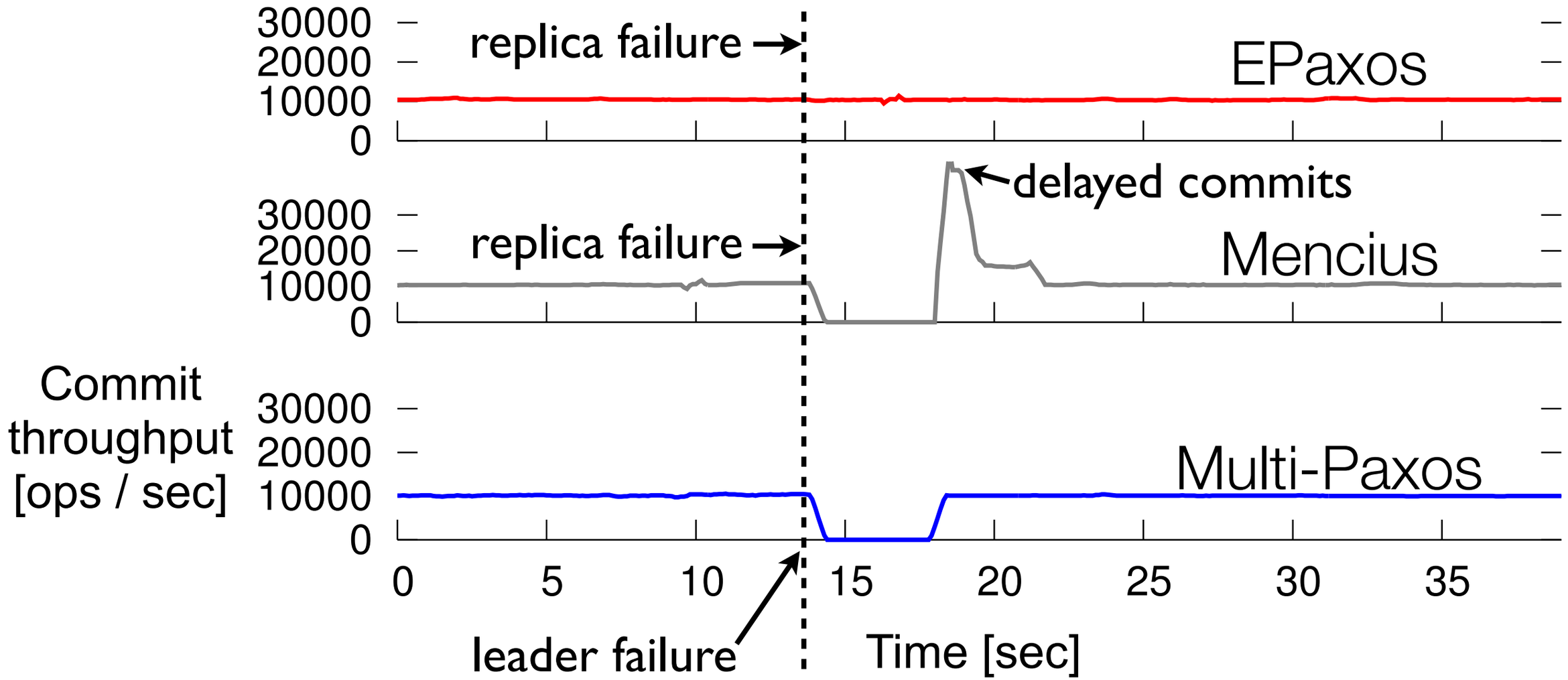
Constant availability



Constant availability



Constant availability



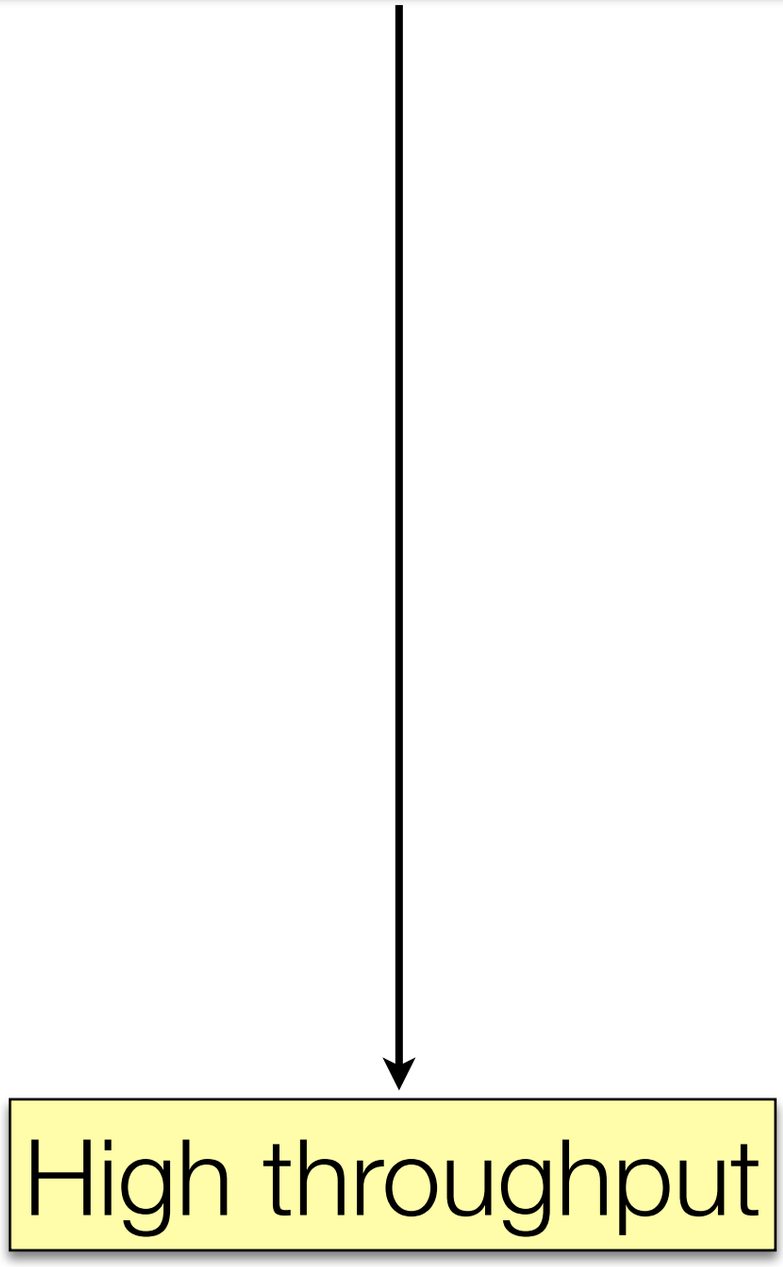
EPaxos insights

EPaxos insights

Order commands explicitly

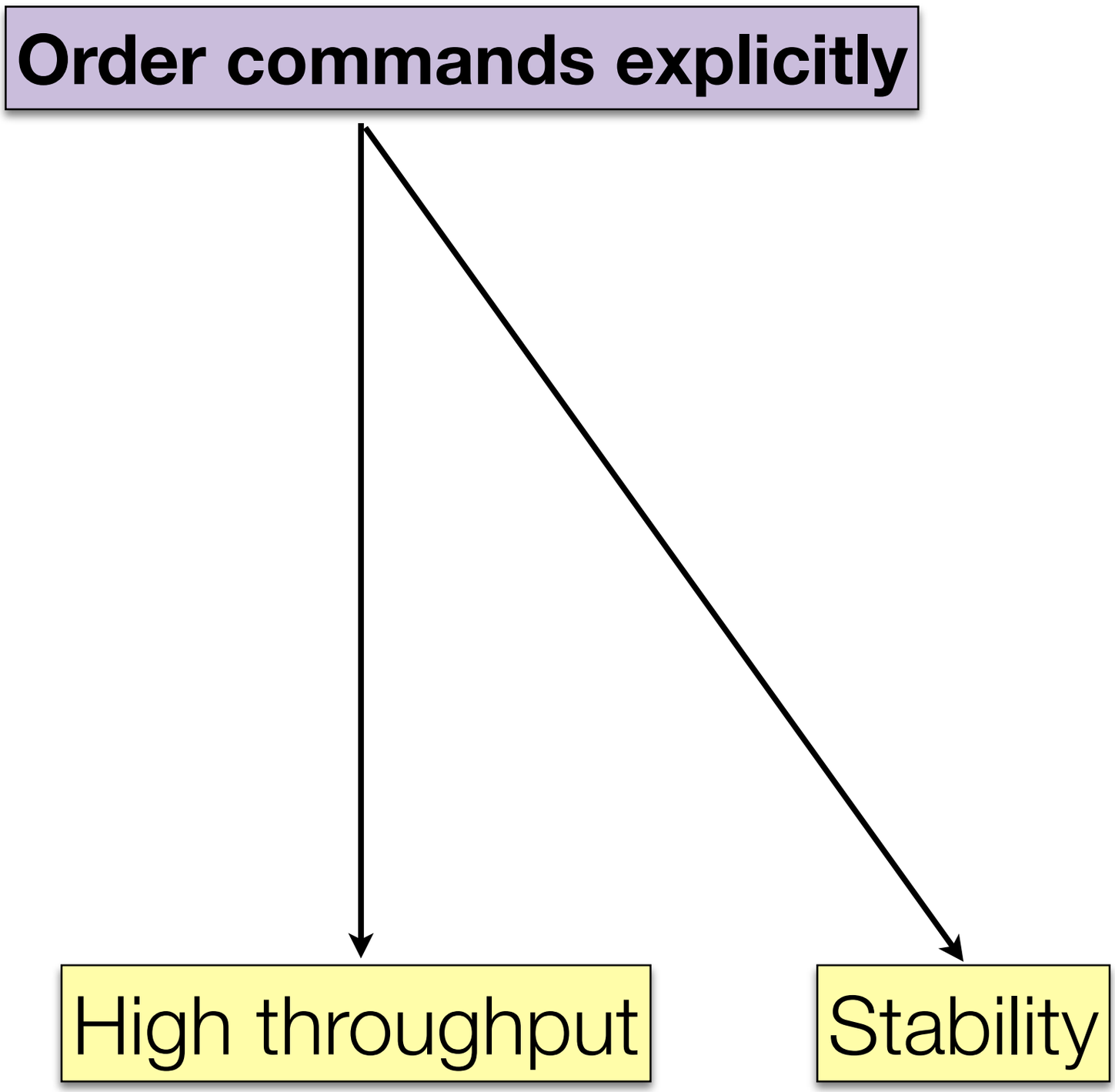
EPaxos insights

Order commands explicitly

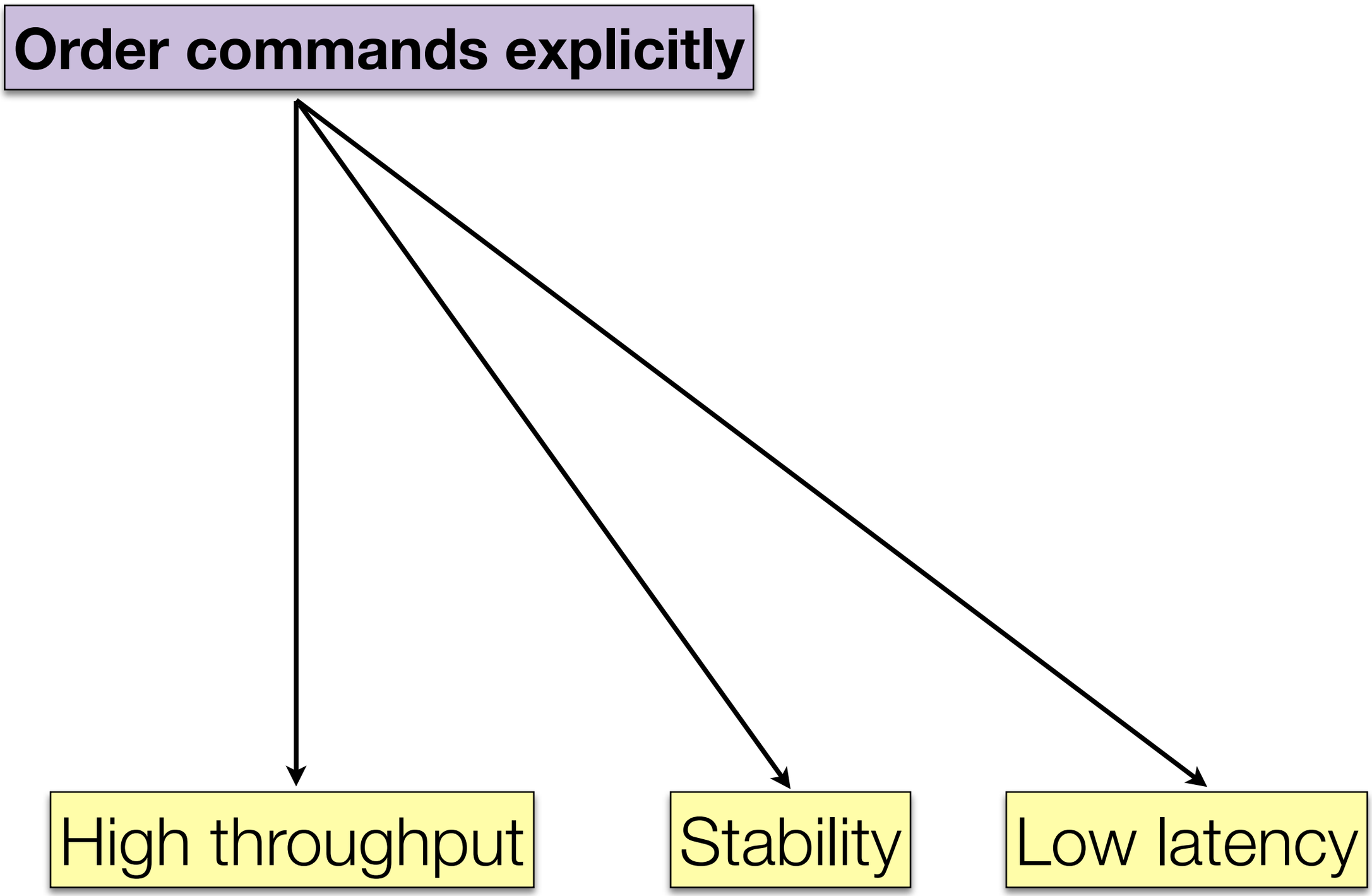


High throughput

EPaxos insights



EPaxos insights



EPaxos insights

Order commands explicitly

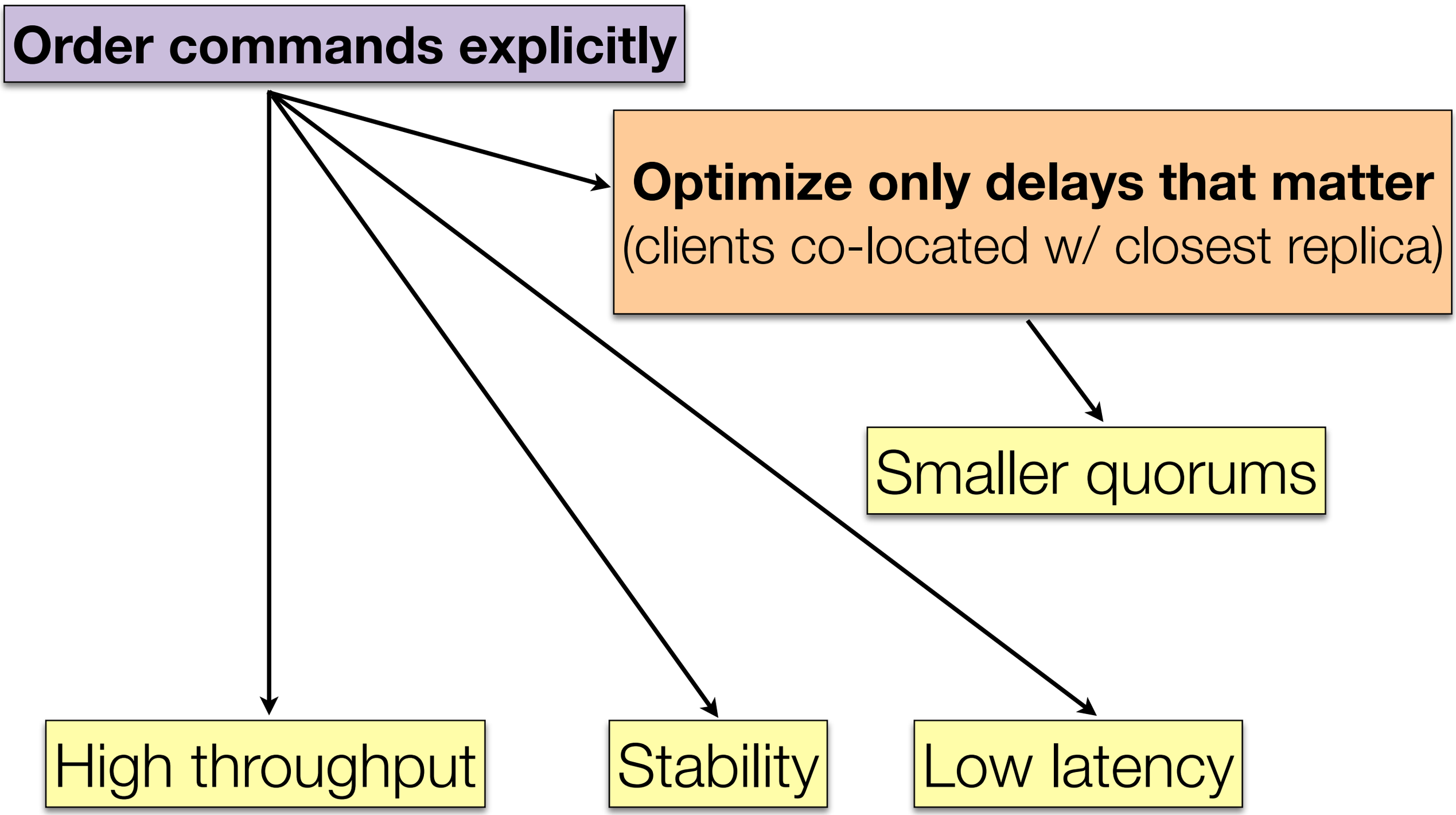
Optimize only delays that matter
(clients co-located w/ closest replica)

High throughput

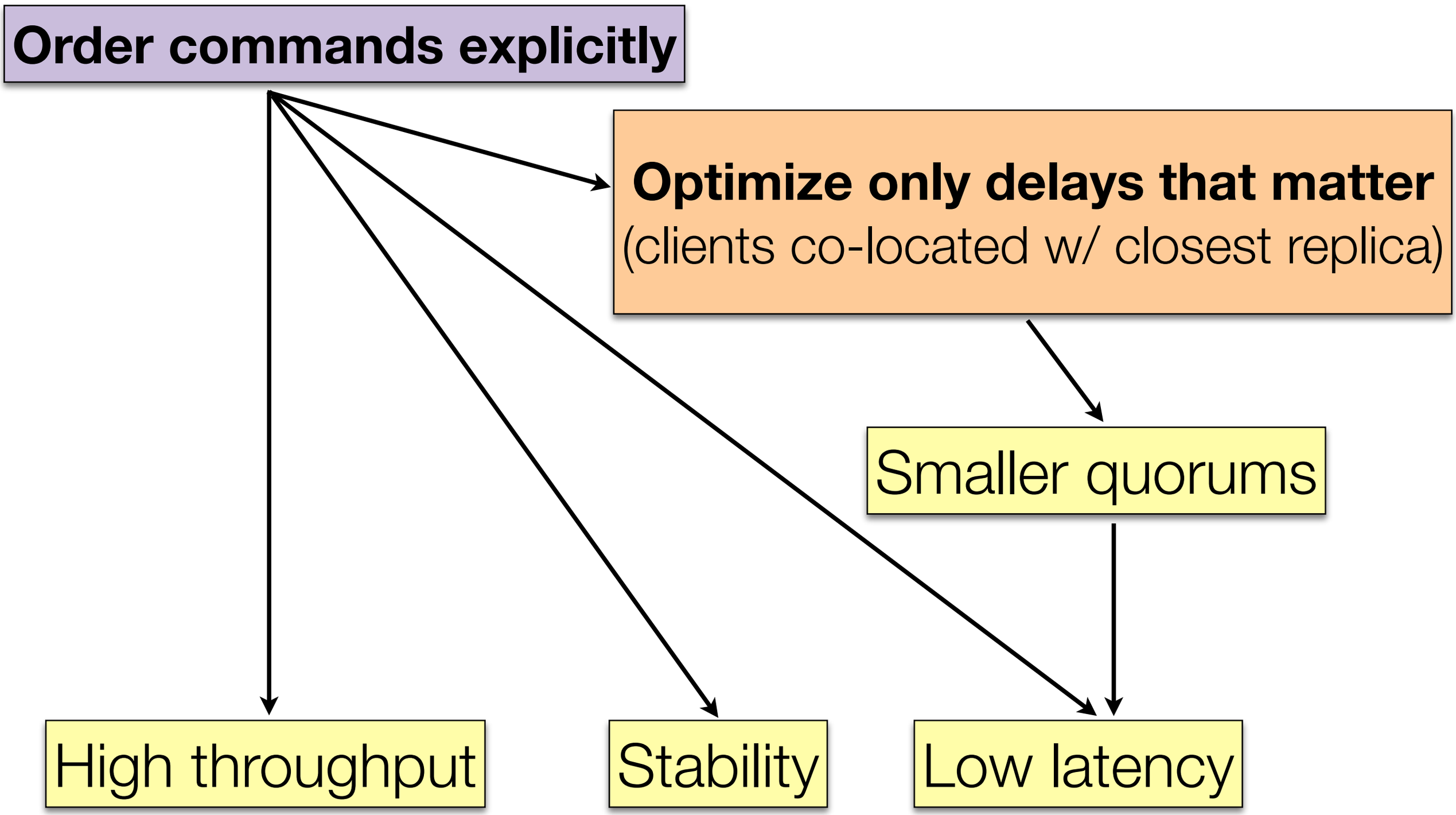
Stability

Low latency

EPaxos insights



EPaxos insights



Formal Proof

TLA+ Spec

http://cs.cmu.edu/~imoraru/epaxos/tr.pdf

Open Source Release

http://github.com/efficient/epaxos