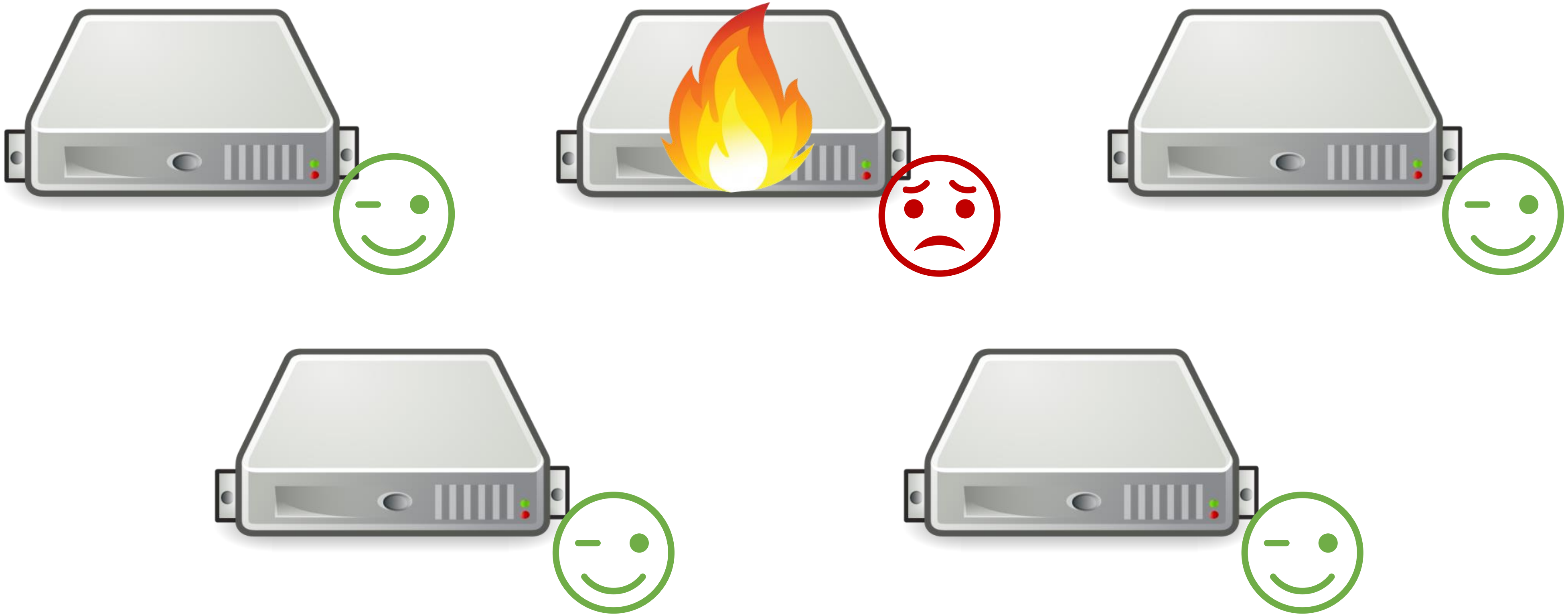
SDPaxos: Building Efficient Semi-Decentralized Geo-replicated State Machines

Hanyu Zhao[*], Quanlu Zhang[†], Zhi Yang[*], Ming Wu[†], Yafei Dai[*]

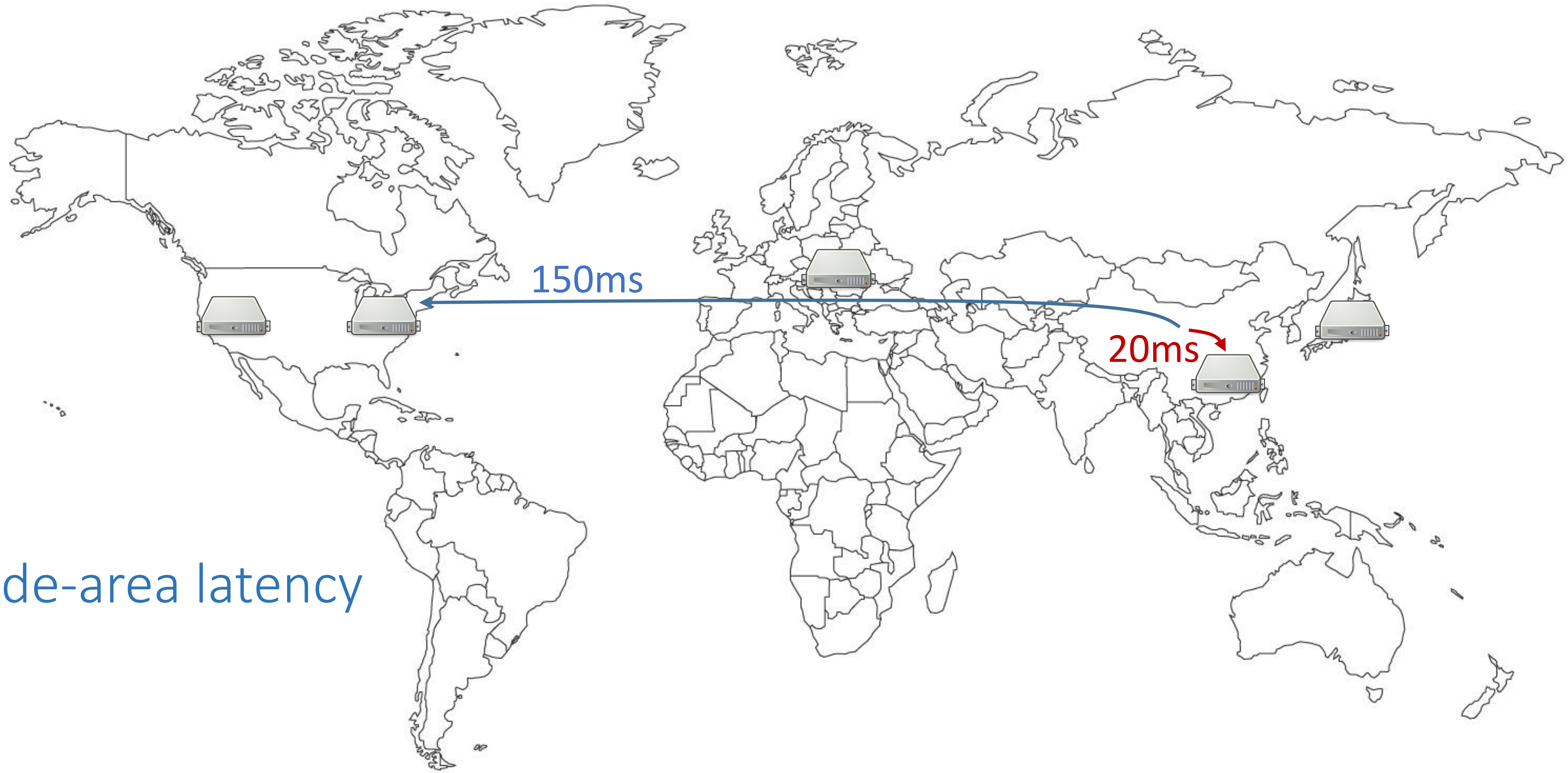
** Peking University*

† Microsoft Research

Replication for Fault Tolerance

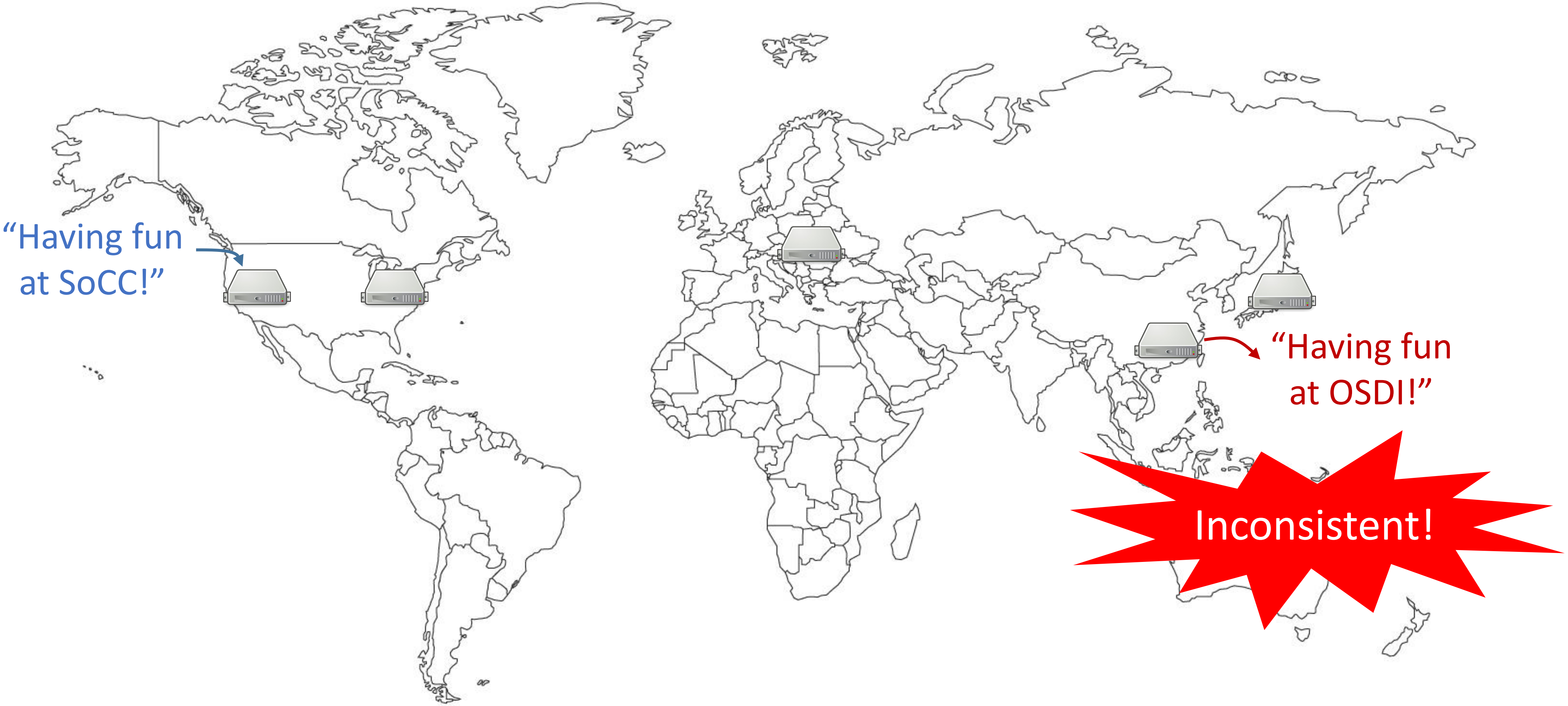


Replication in the Wide Area

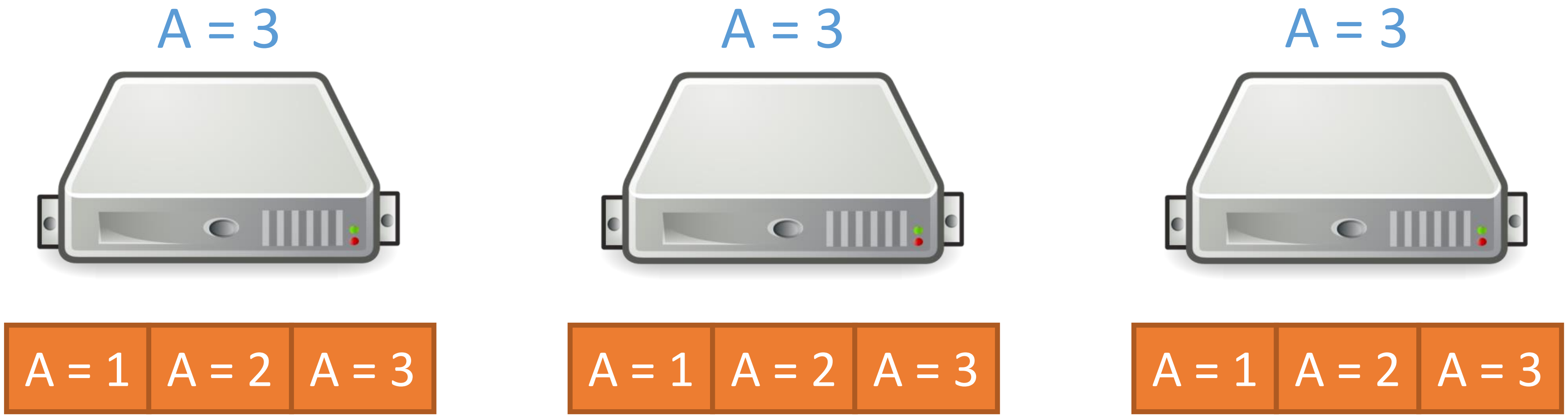


- Reducing wide-area latency for clients

Keeping the Replicated State Consistent



State Machine Replication (SMR)



A = 1	A = 2	A = 3

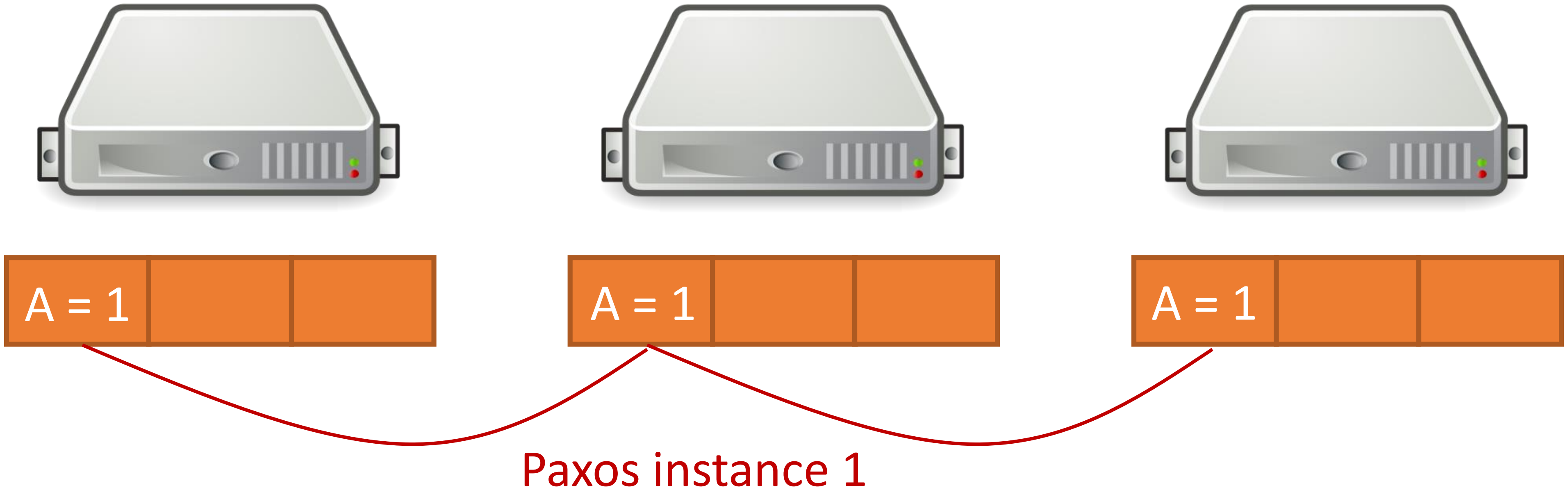
A = 1	A = 2	A = 3

A = 1	A = 2	A = 3

Execute the same sequence of commands in the same order

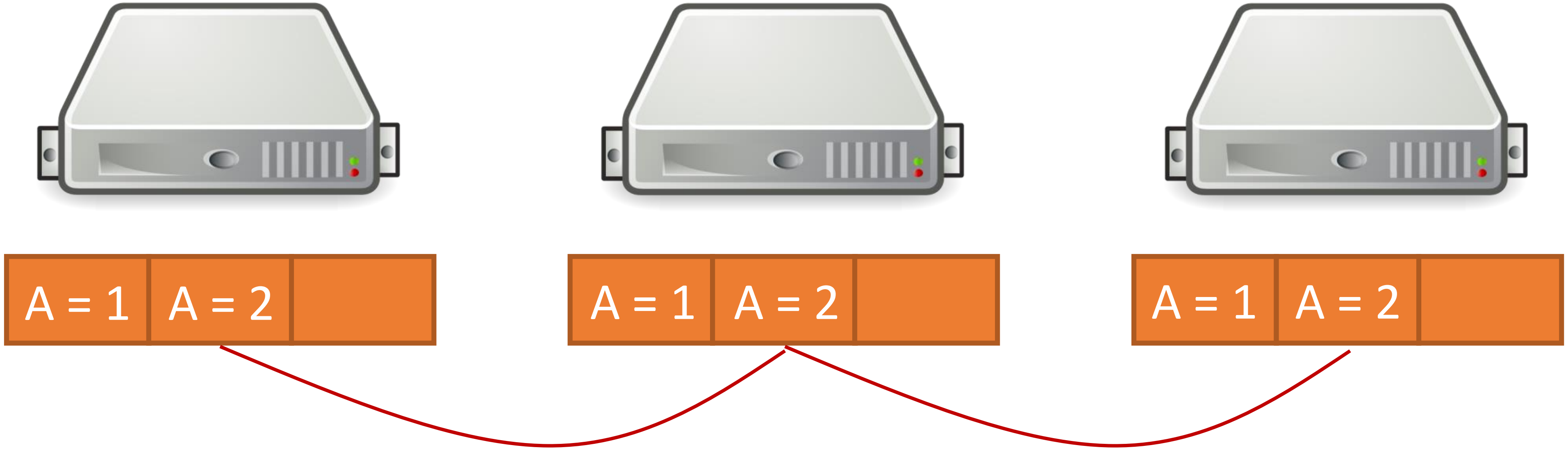
Paxos

- A distributed **agreement** protocol
  - Tolerates F failures given 2F+1 replicas
- Choose a single command for **each command slot** using a **Paxos instance**



Paxos

- A distributed **agreement** protocol
  - Tolerates F failures given 2F+1 replicas
- Choose a single command for **each command slot** using a **Paxos instance**

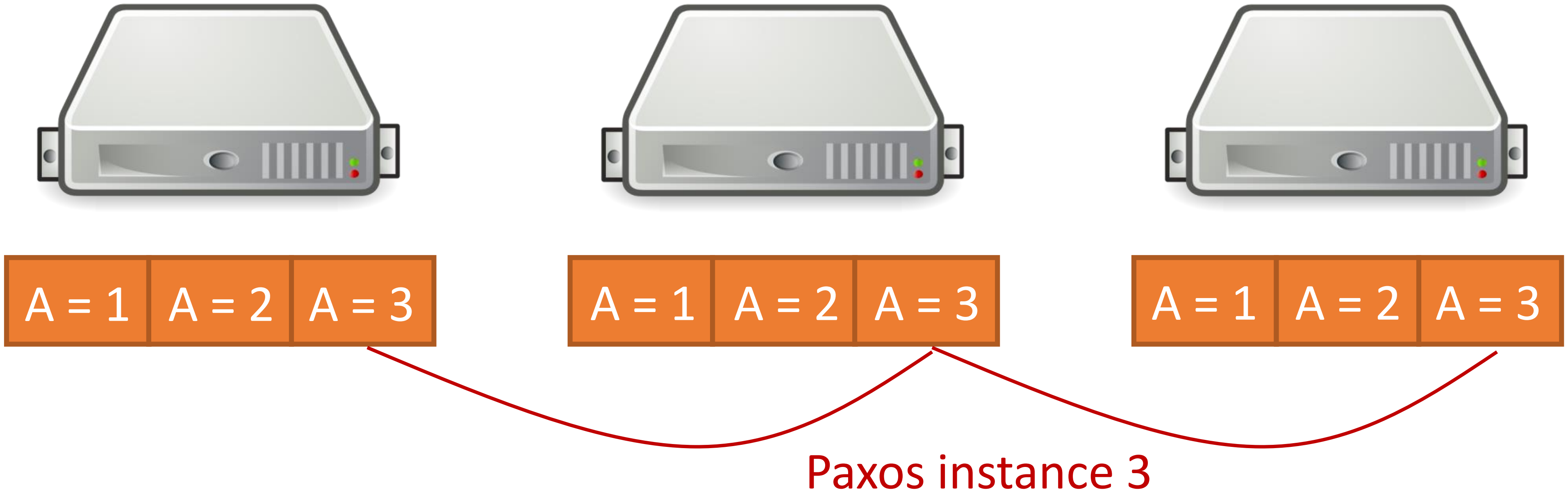


A = 1 | A = 2 |
A = 1 | A = 2 |
A = 1 | A = 2 |

Paxos instance 2

Paxos

- A distributed **agreement** protocol
  - Tolerates F failures given 2F+1 replicas
- Choose a single command for **each command slot** using a **Paxos instance**



A = 1 | A = 2 | A = 3

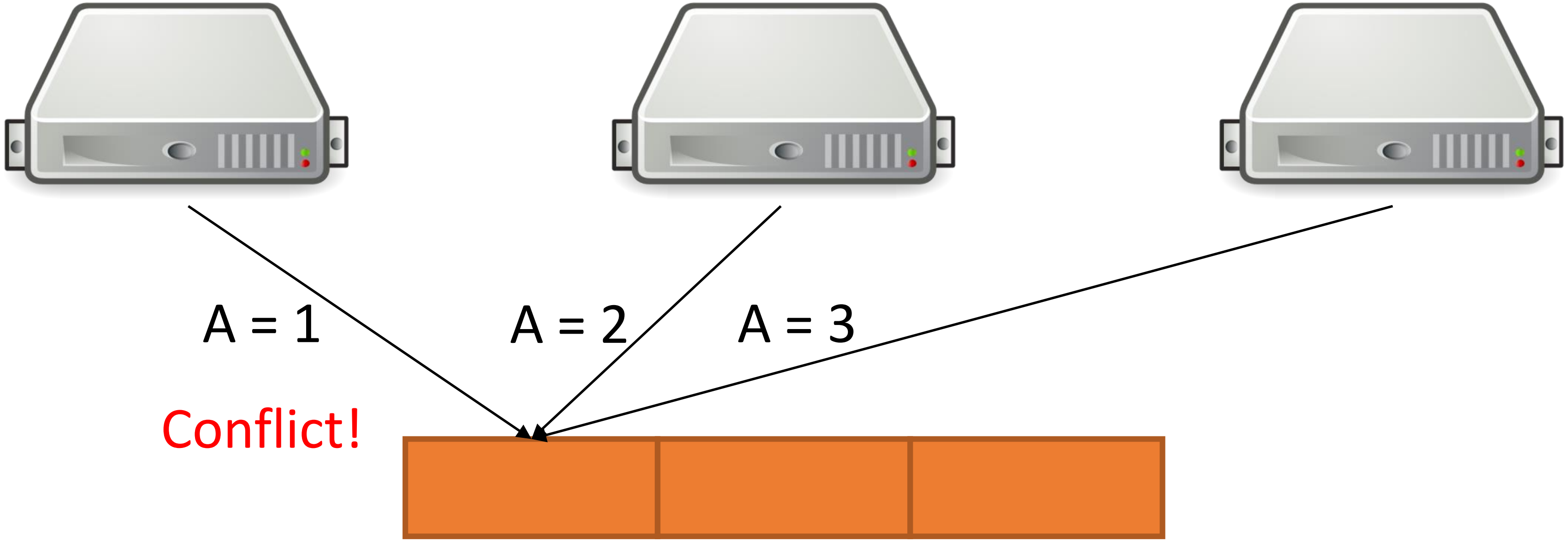
A = 1 | A = 2 | A = 3

A = 1 | A = 2 | A = 3

Paxos instance 3

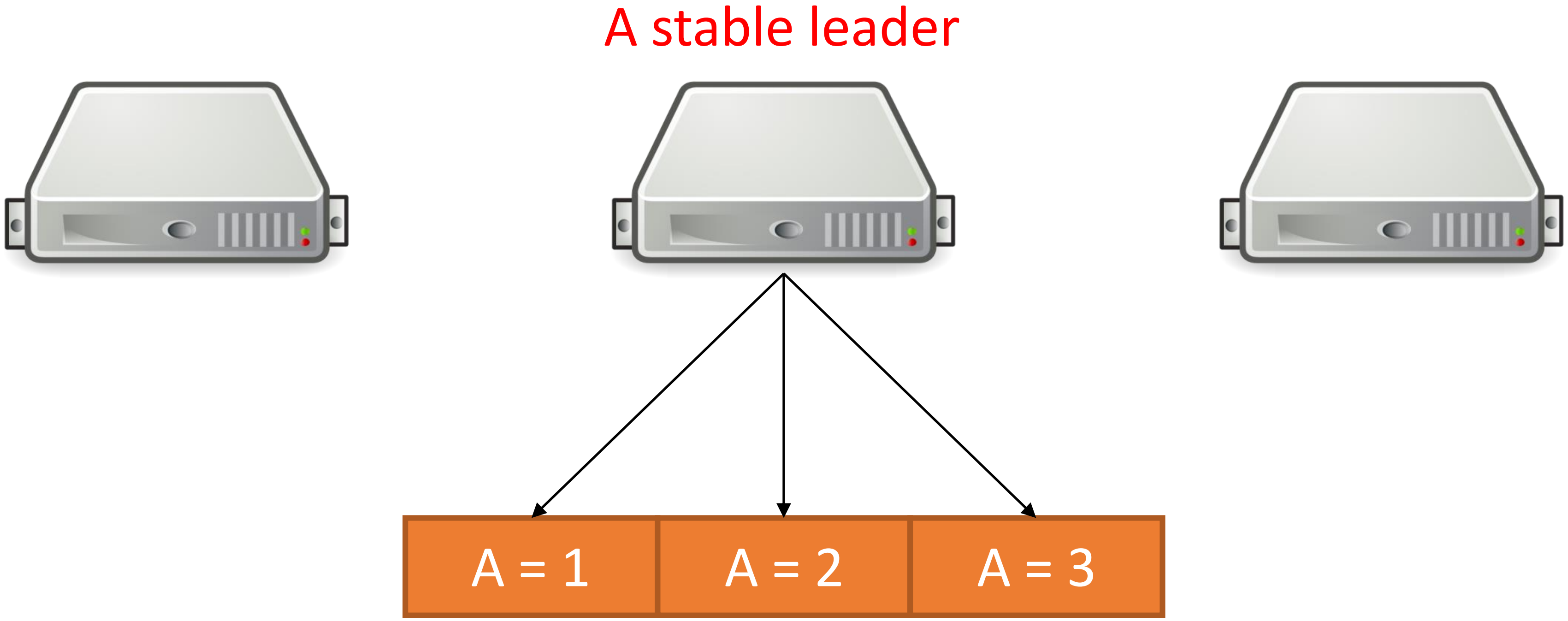
Centralized SMR

- Liveness property of Paxos:
  - There should not be multiple replicas proposing commands in the same instance simultaneously



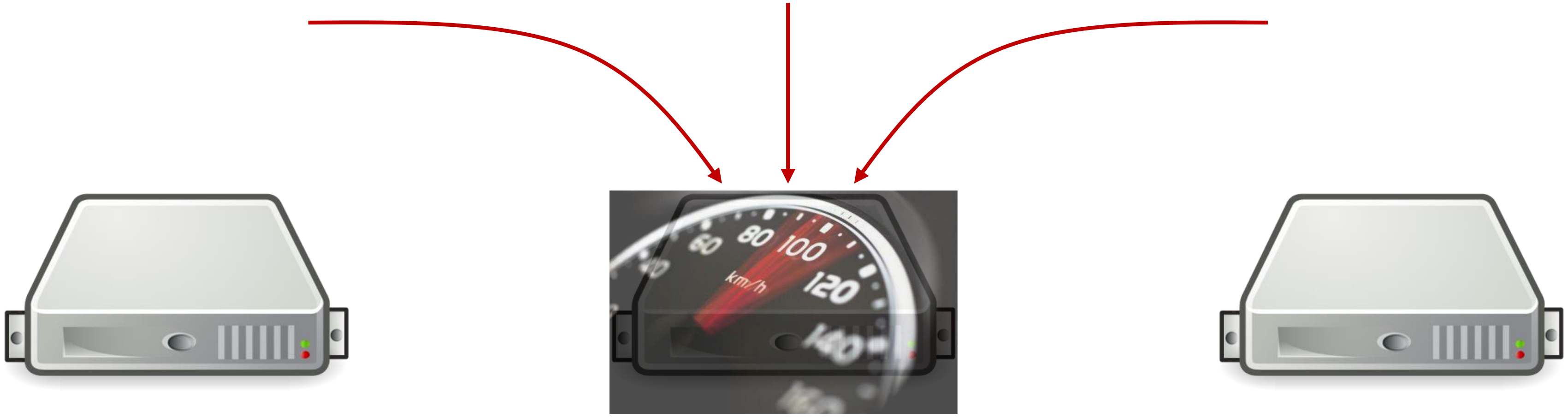
Centralized SMR

- Liveness property of Paxos:
  - There should not be multiple replicas proposing commands in the same instance simultaneously



Drawbacks of Centralized SMR

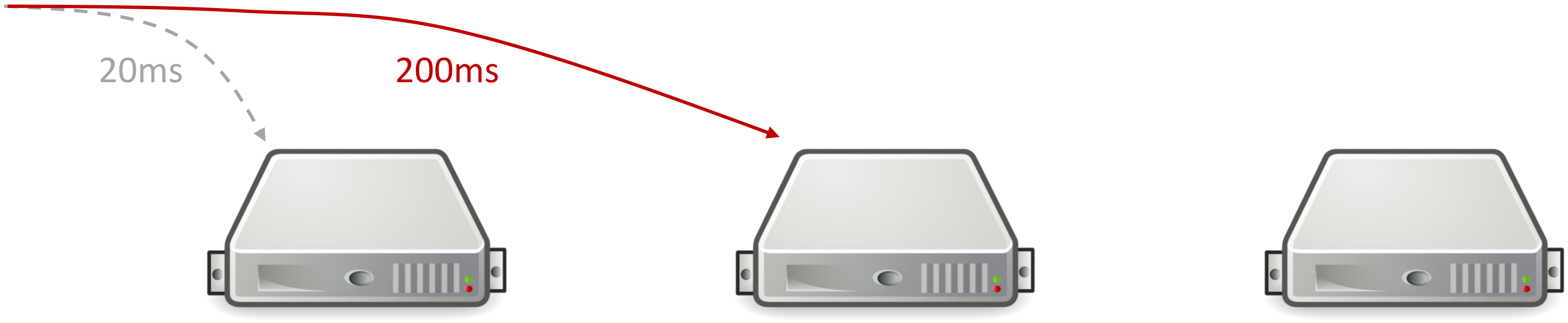
- Potential performance bottleneck
  - Low throughput



Drawbacks of Centralized SMR

- Potential performance bottleneck
  - Low throughput

- High wide-area latency



Drawbacks of Centralized SMR

- Potential performance bottleneck
  - Low throughput

- High wide-area latency

Centralized SMR

Limited performance

Drawbacks of Centralized SMR

- Potential performance bottleneck
    - Low throughput

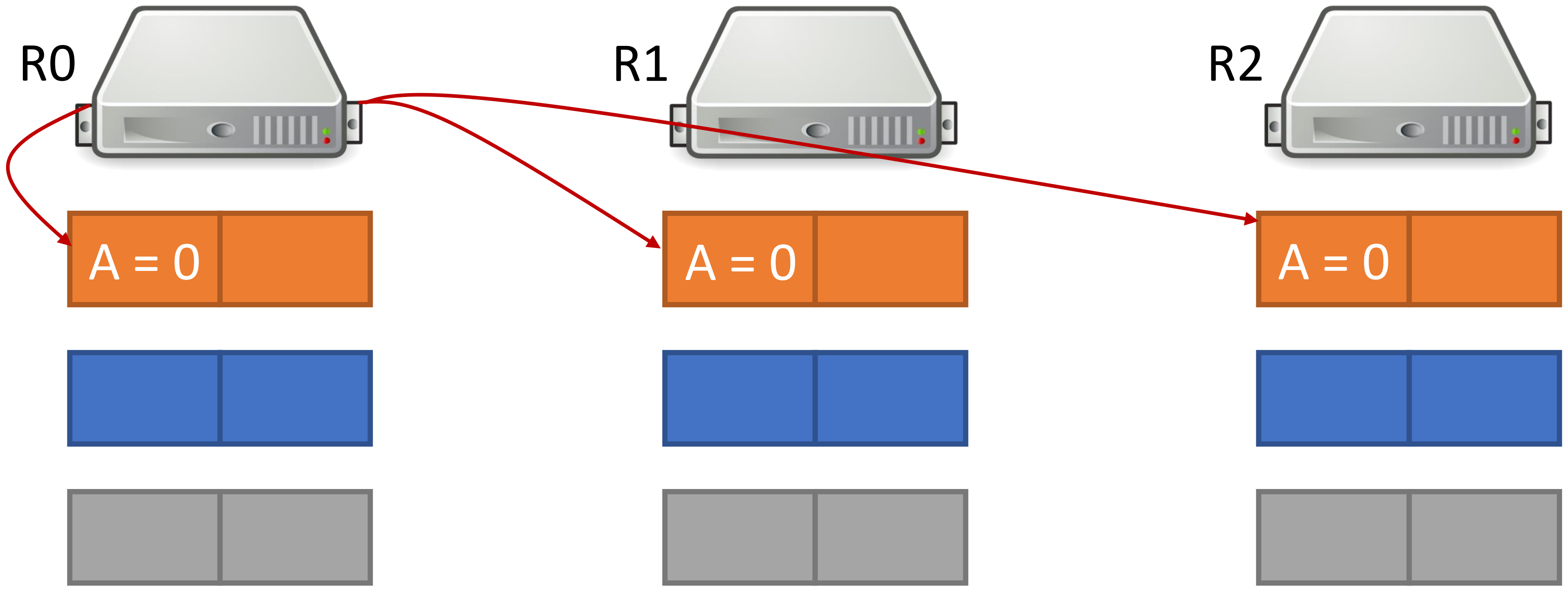
- High wide-area latency

Centralized SMR
Limited performance

Decentralized SMR
High performance?

Decentralizing SMR

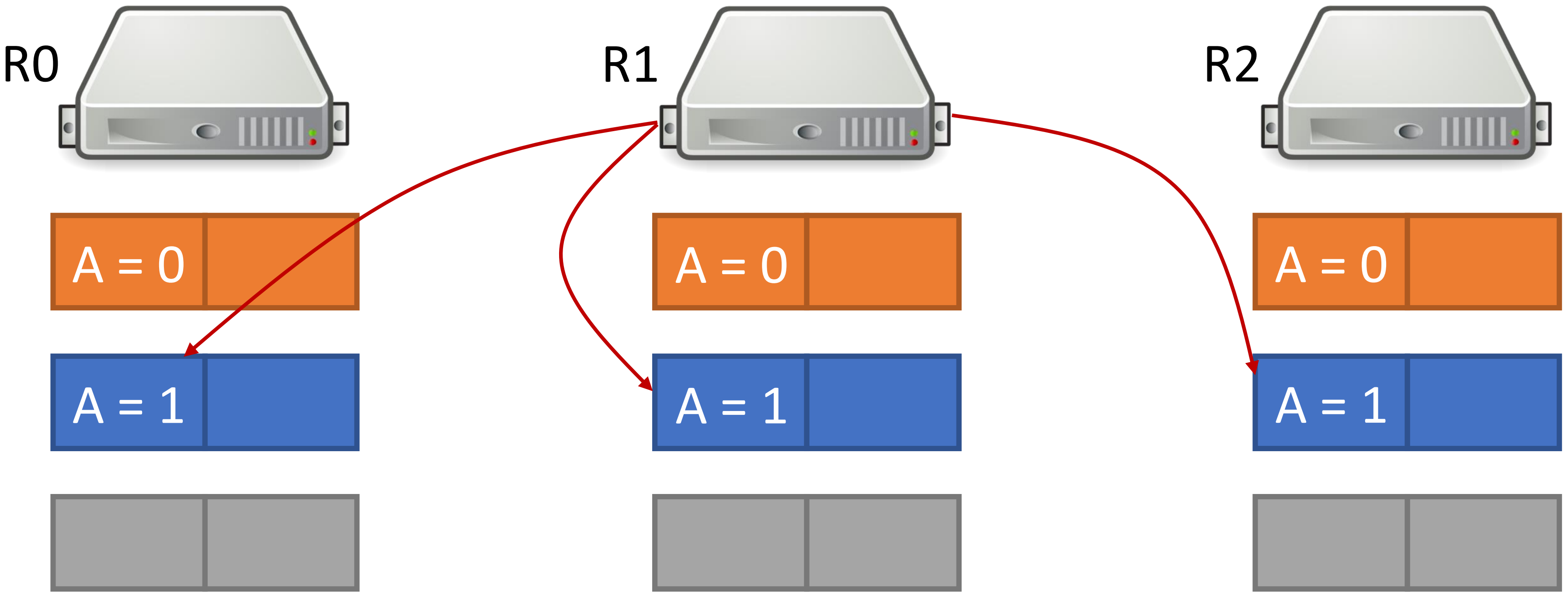
Replicas should propose commands in different command slots



How to *order* them?

Decentralizing SMR

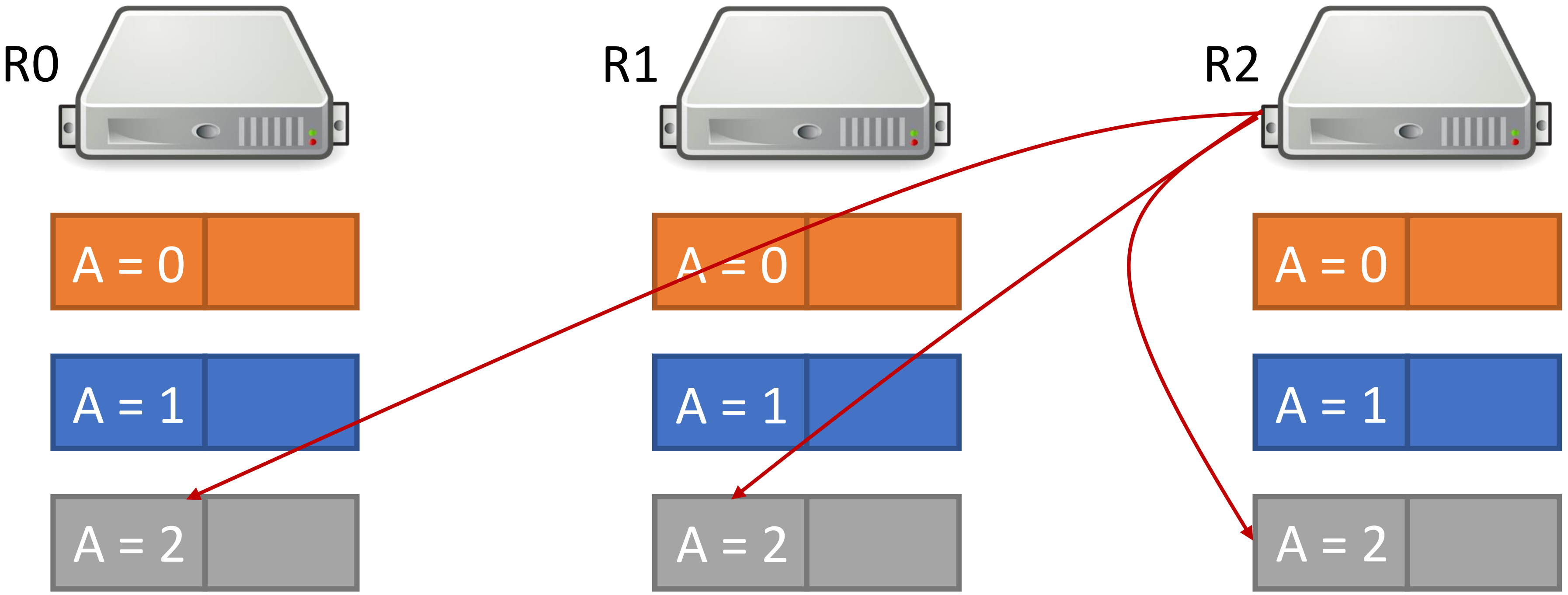
Replicas should propose commands in different command slots



How to *order* them?

Decentralizing SMR

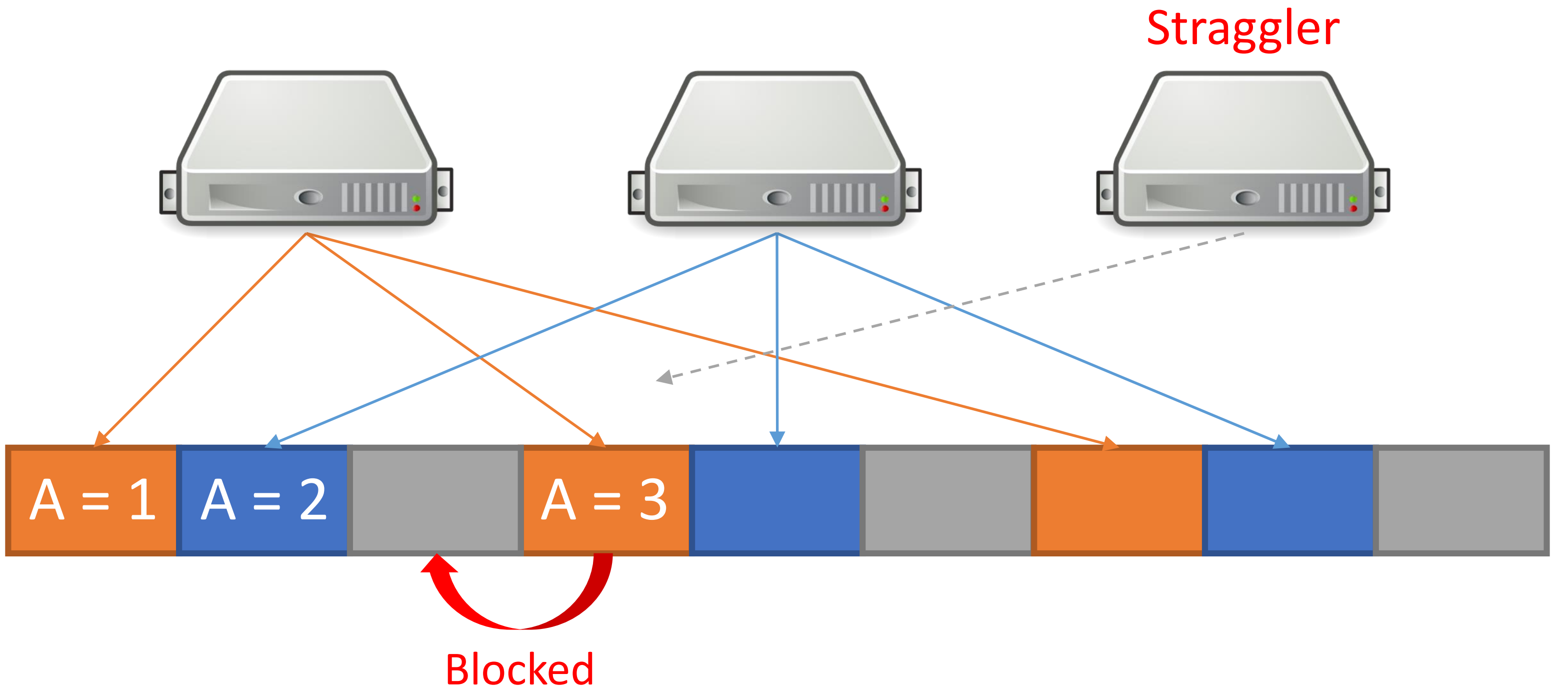
Replicas should propose commands in different command slots



How to *order* them?

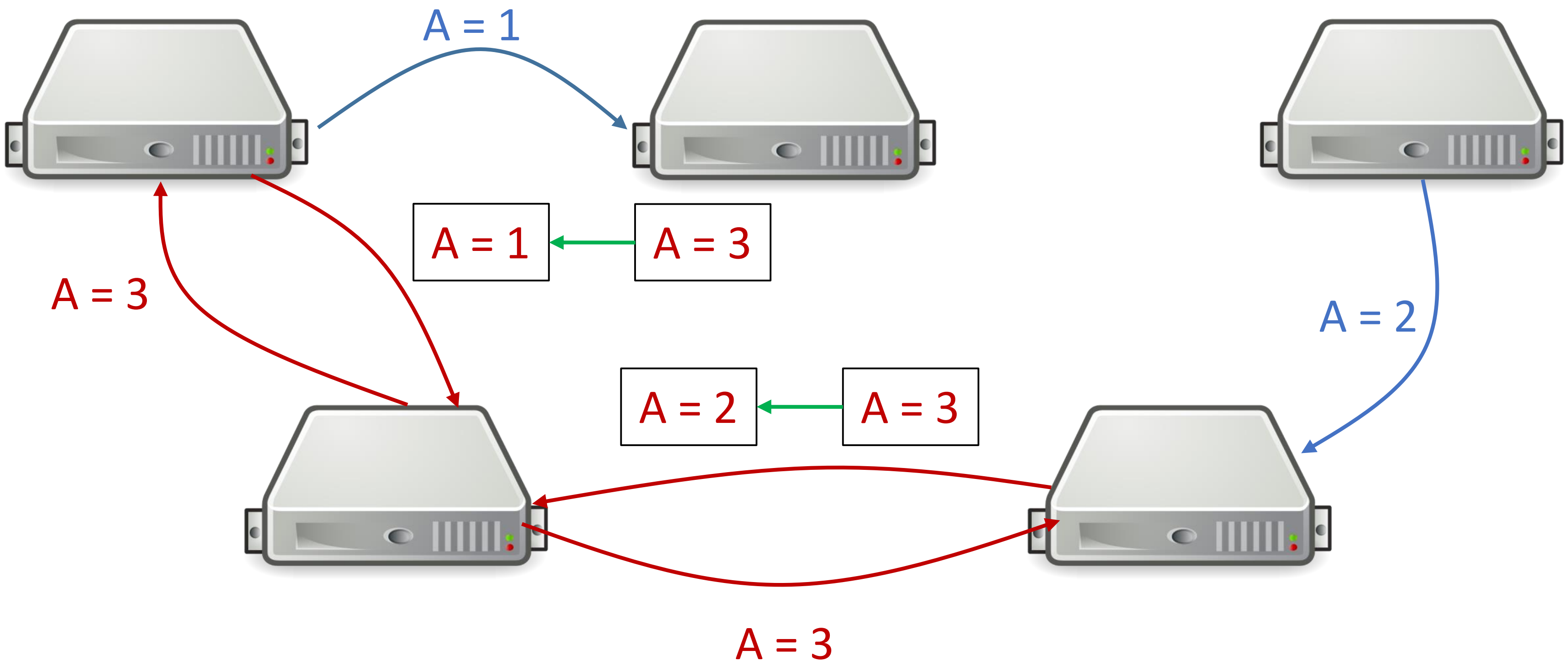
Static Ordering

- The system runs at the speed of the **slowest one**



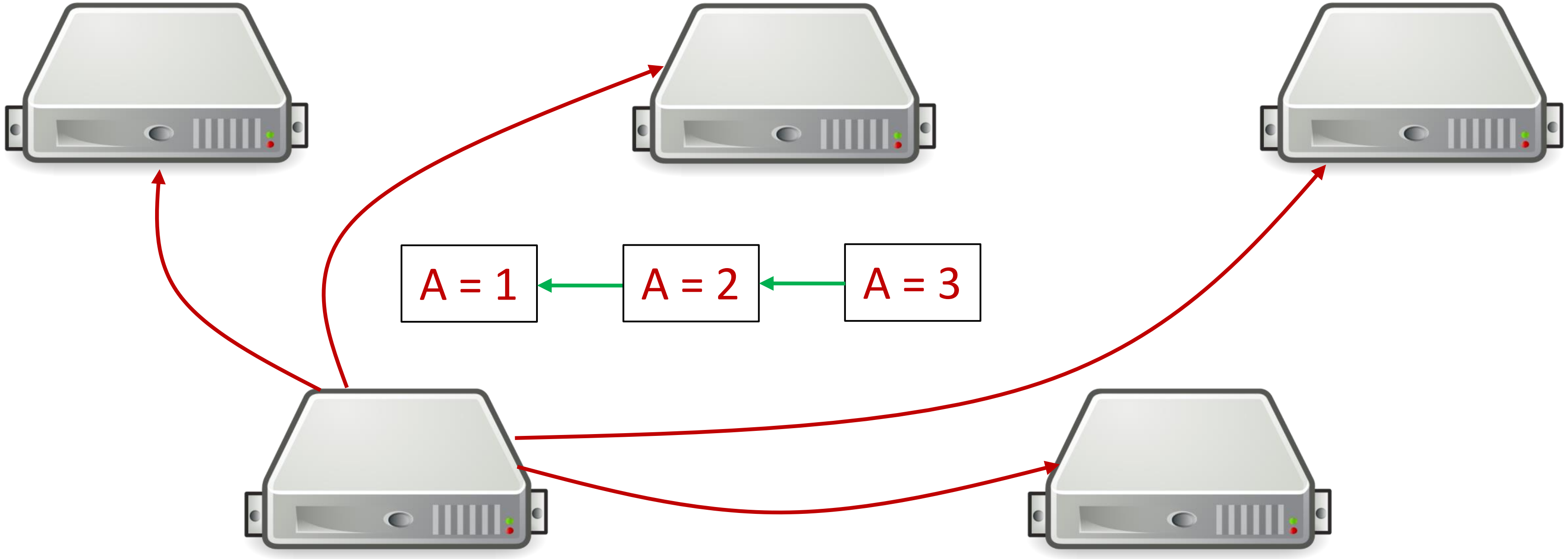
Dependency-based Ordering

- Ordering overhead under contention



Dependency-based Ordering

- Ordering overhead under contention



Drawbacks of Decentralized SMR

- Extra coordination for ordering => performance degradation
  - Lower throughput
  - Higher latency

Centralized SMR
Limited performance

Decentralized SMR
Poor performance stability

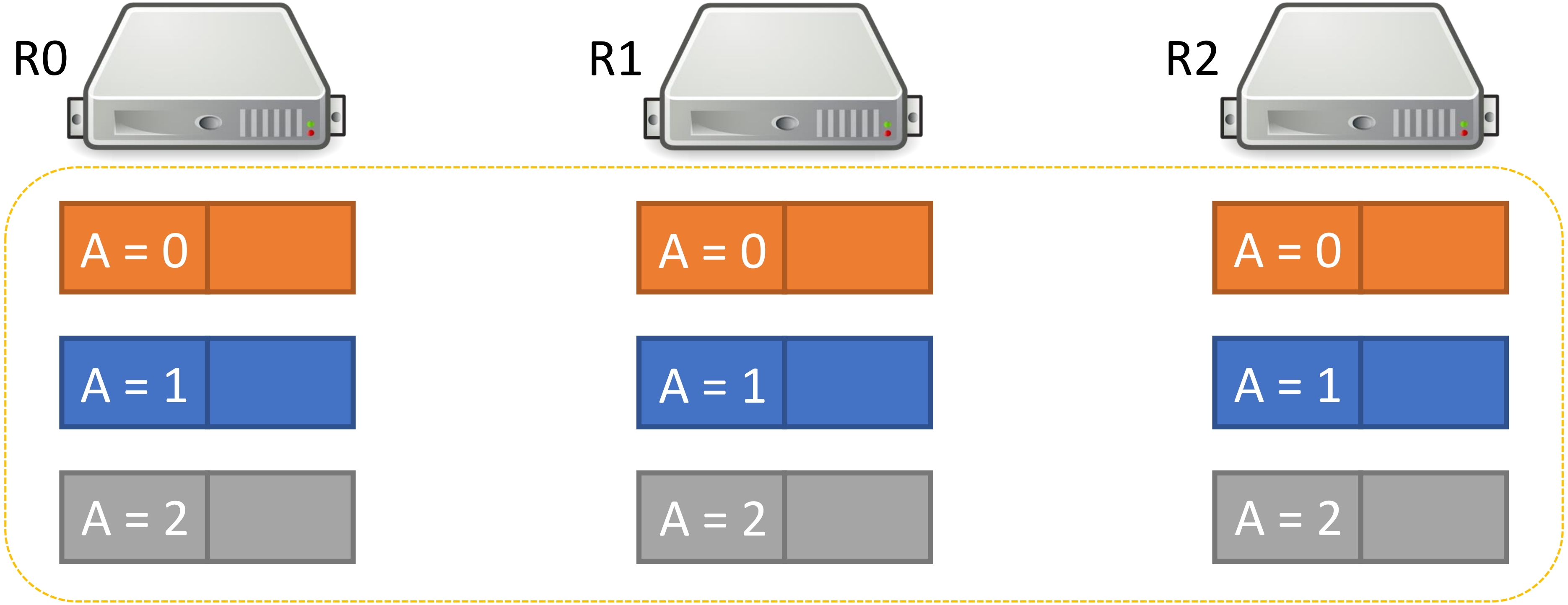
Drawbacks of Decentralized SMR

- Extra coordination for ordering => performance degradation
  - Lower throughput
  - Higher latency

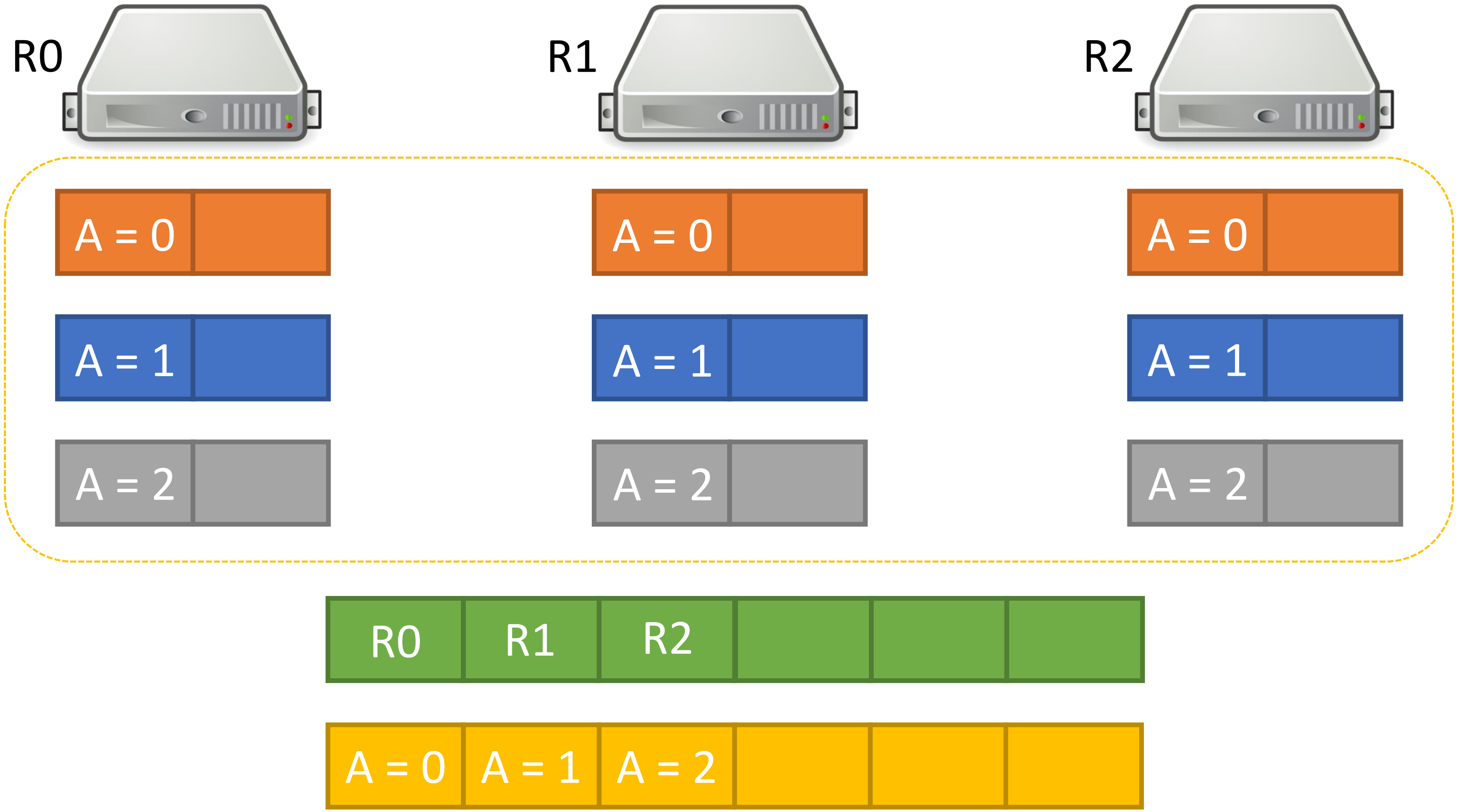
Semi-Decentralized SMR

SDPaxos
- *High performance*
- *Strong performance stability*

SDPaxos Intuition



SDPaxos Intuition



R0

A = 0

A = 1

A = 2

R1

A = 0

A = 1

A = 2

R2

A = 0

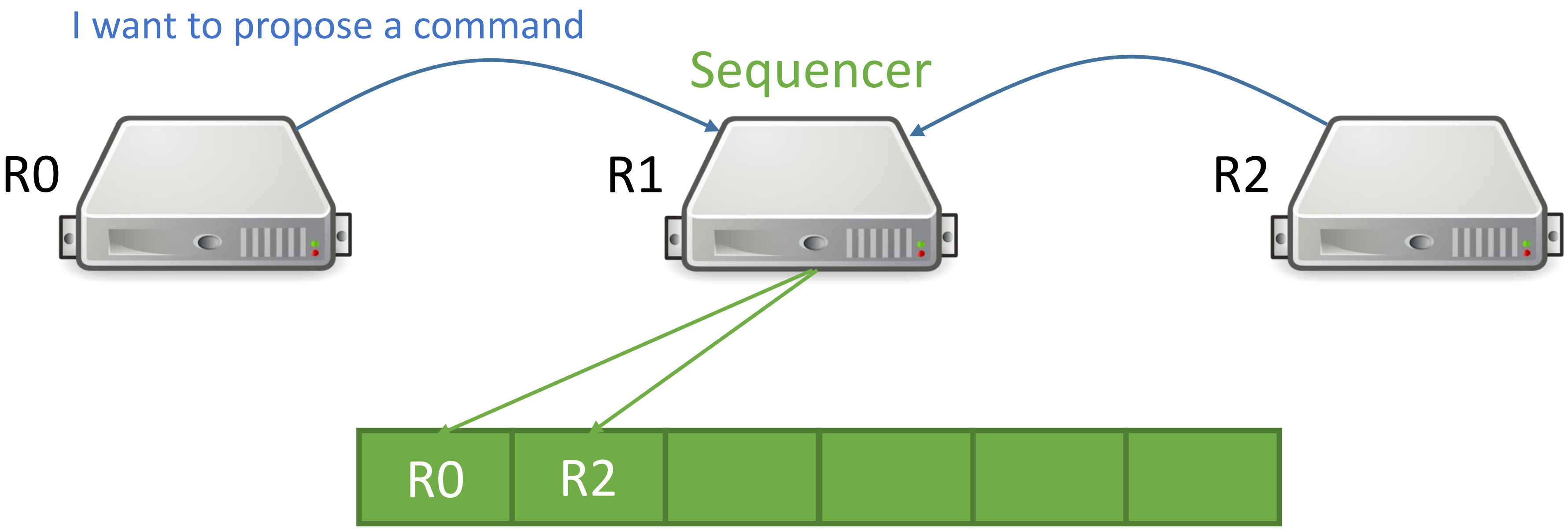
A = 1

A = 2

R0 R1 R2

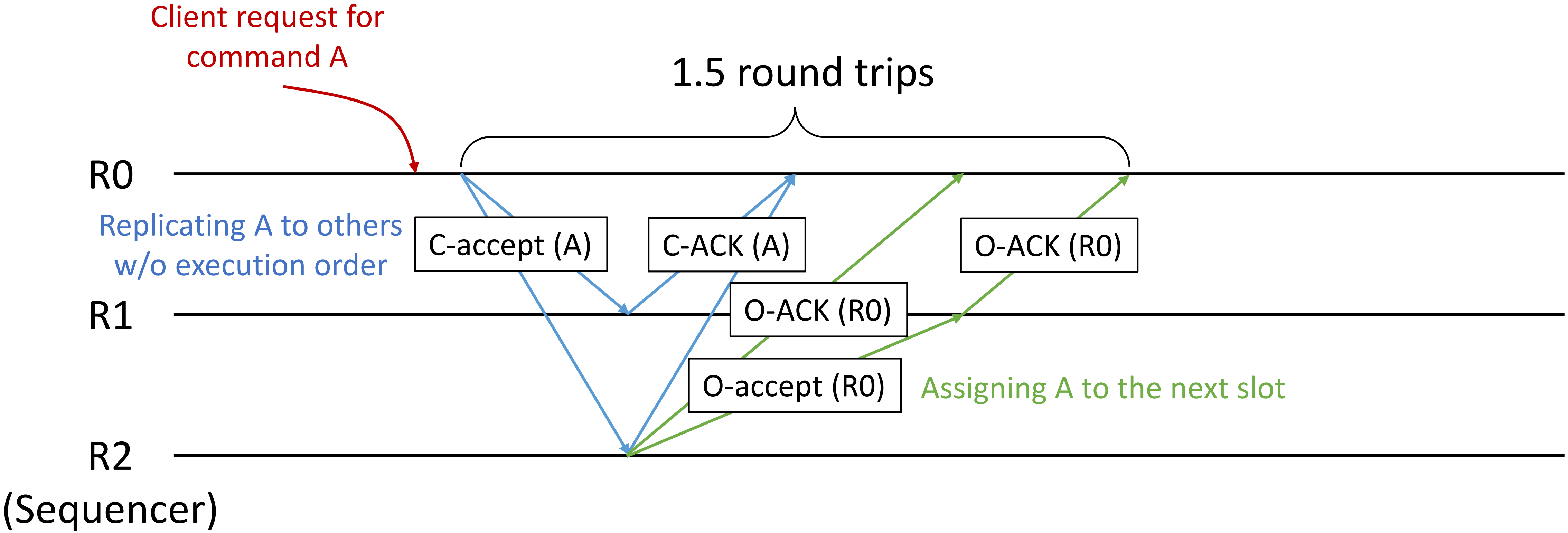
A = 0 A = 1 A = 2

Centralizing Ordering

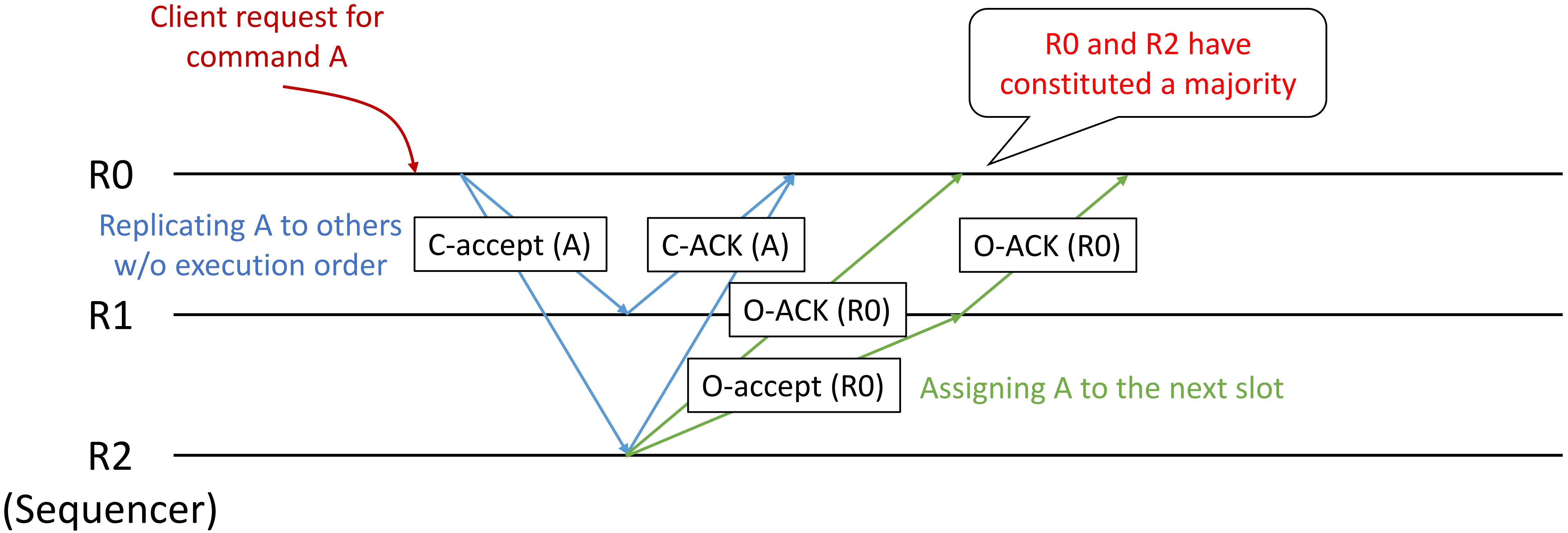


- Dynamical leadership establishment (stragglers won’t block others)
- All commands are serialized (no conflicts)
- Ordering is more lightweight than replicating

SDPaxos: The Basic Protocol


Client request for command A
1.5 round trips
R0
Replicating A to others w/o execution order
C-accept (A)
C-ACK (A)
O-ACK (R0)
O-ACK (R0)
R1
O-accept (R0)
Assigning A to the next slot
R2
(Sequencer)

Reducing Latency for 3 Replicas



Client request for command A

R0 and R2 have constituted a majority

R0

Replicating A to others w/o execution order

C-accept (A)

C-ACK (A)

O-ACK (R0)

R1

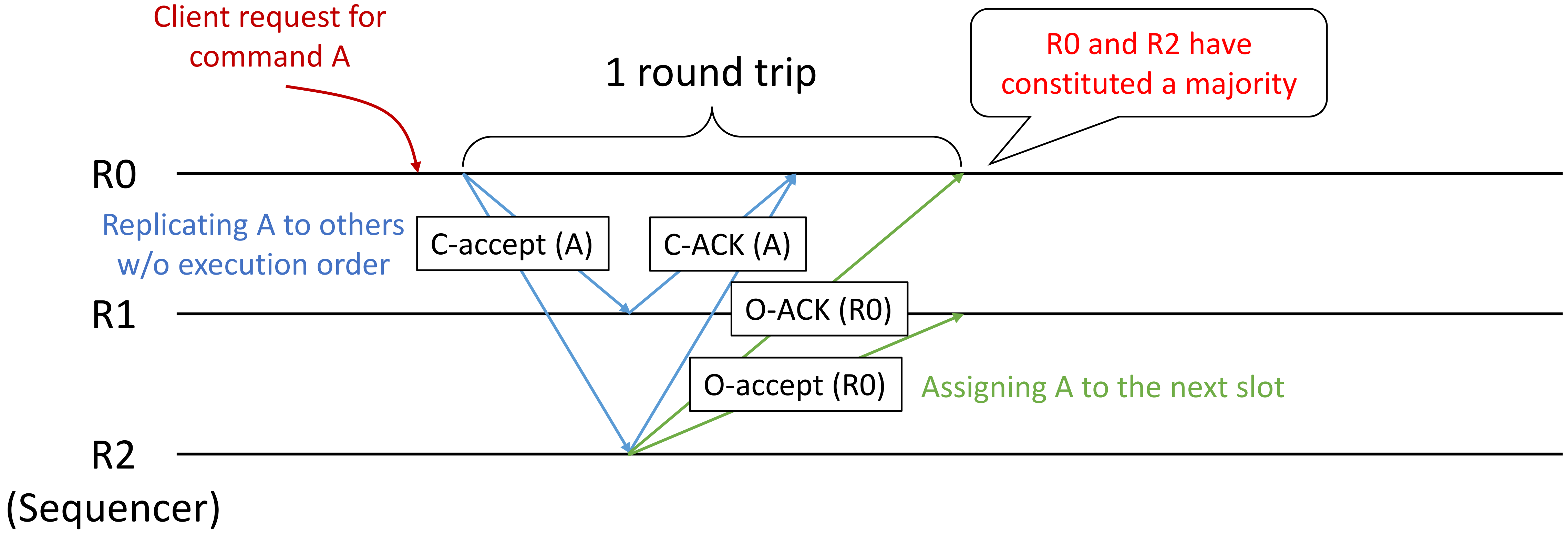
O-ACK (R0)

O-accept (R0)

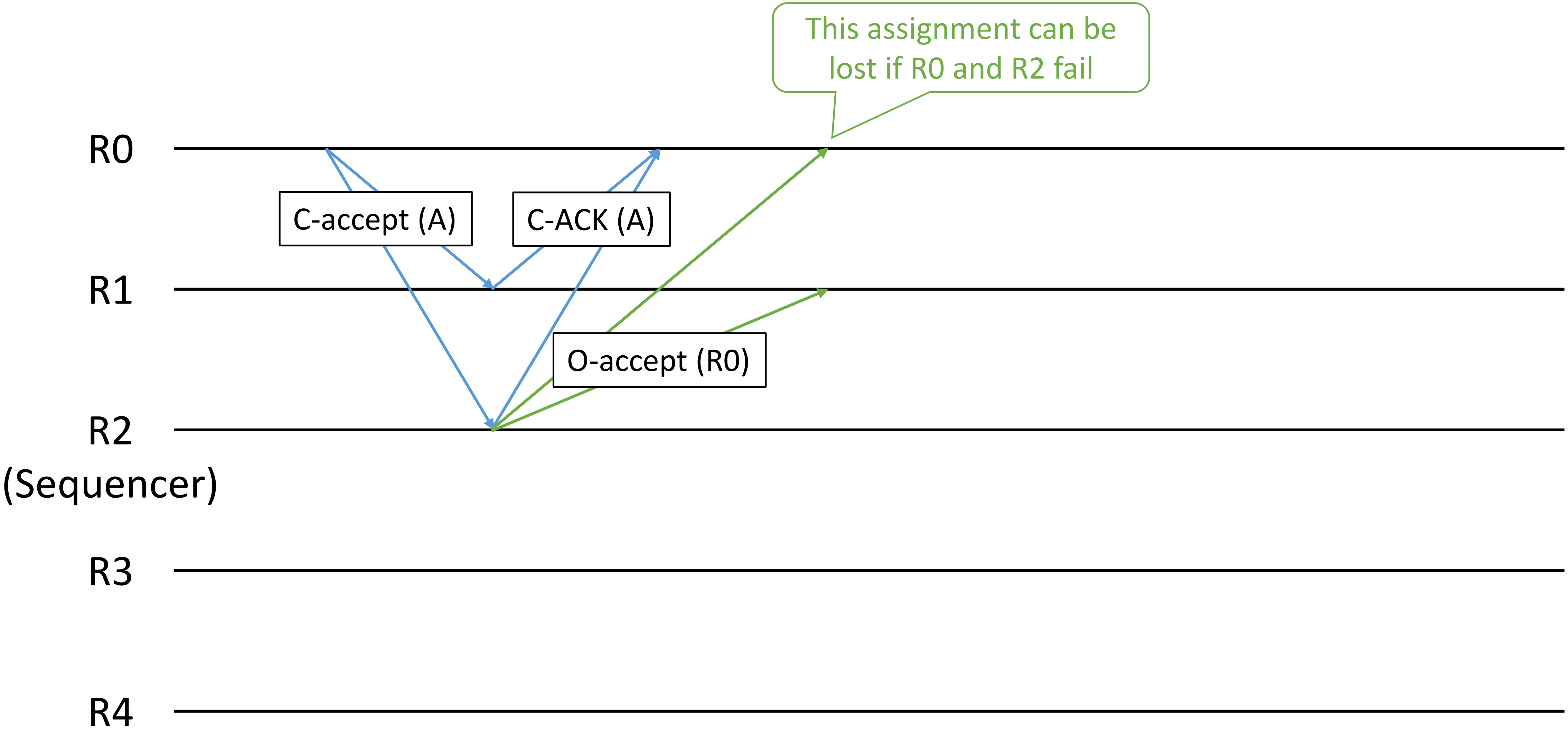
Assigning A to the next slot

R2 (Sequencer)

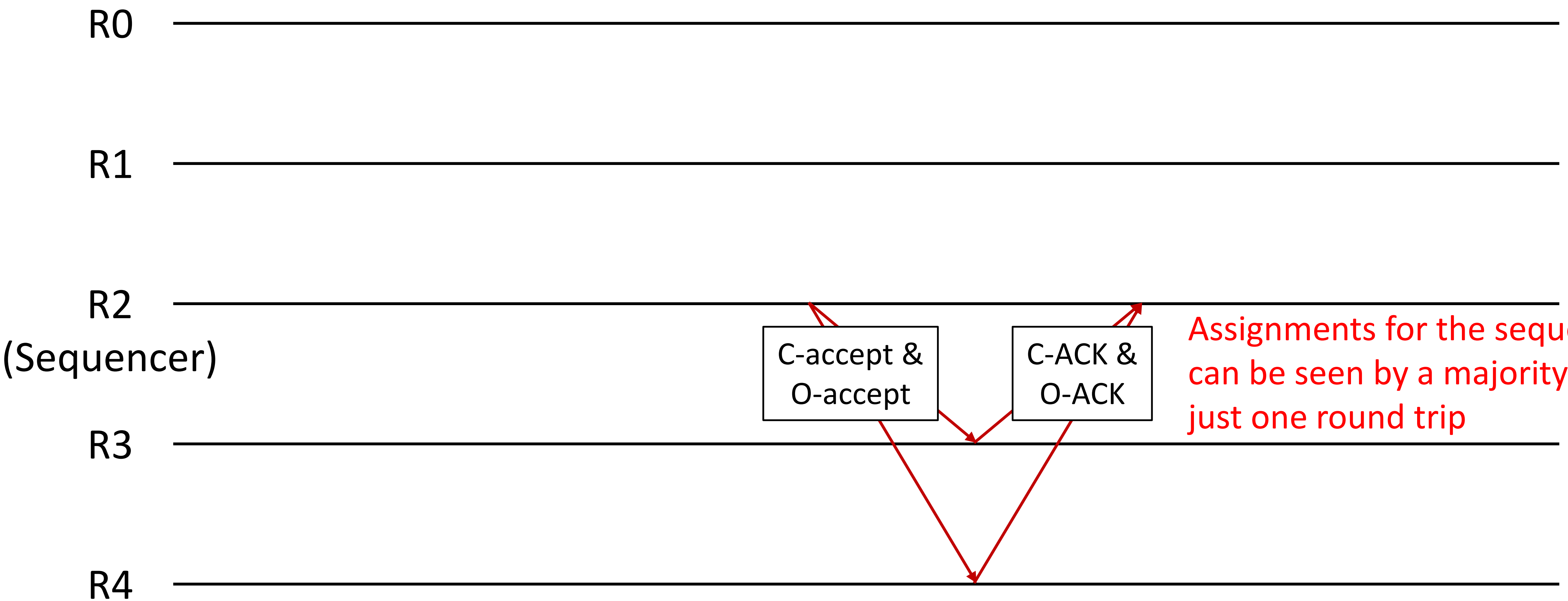
Reducing Latency for 3 Replicas


Client request for command A
1 round trip
R0 and R2 have constituted a majority
R0
Replicating A to others w/o execution order
C-accept (A)
C-ACK (A)
R1
O-ACK (R0)
O-accept (R0)
Assigning A to the next slot
R2
(Sequencer)

Reducing Latency for 5 Replicas



Reducing Latency for 5 Replicas

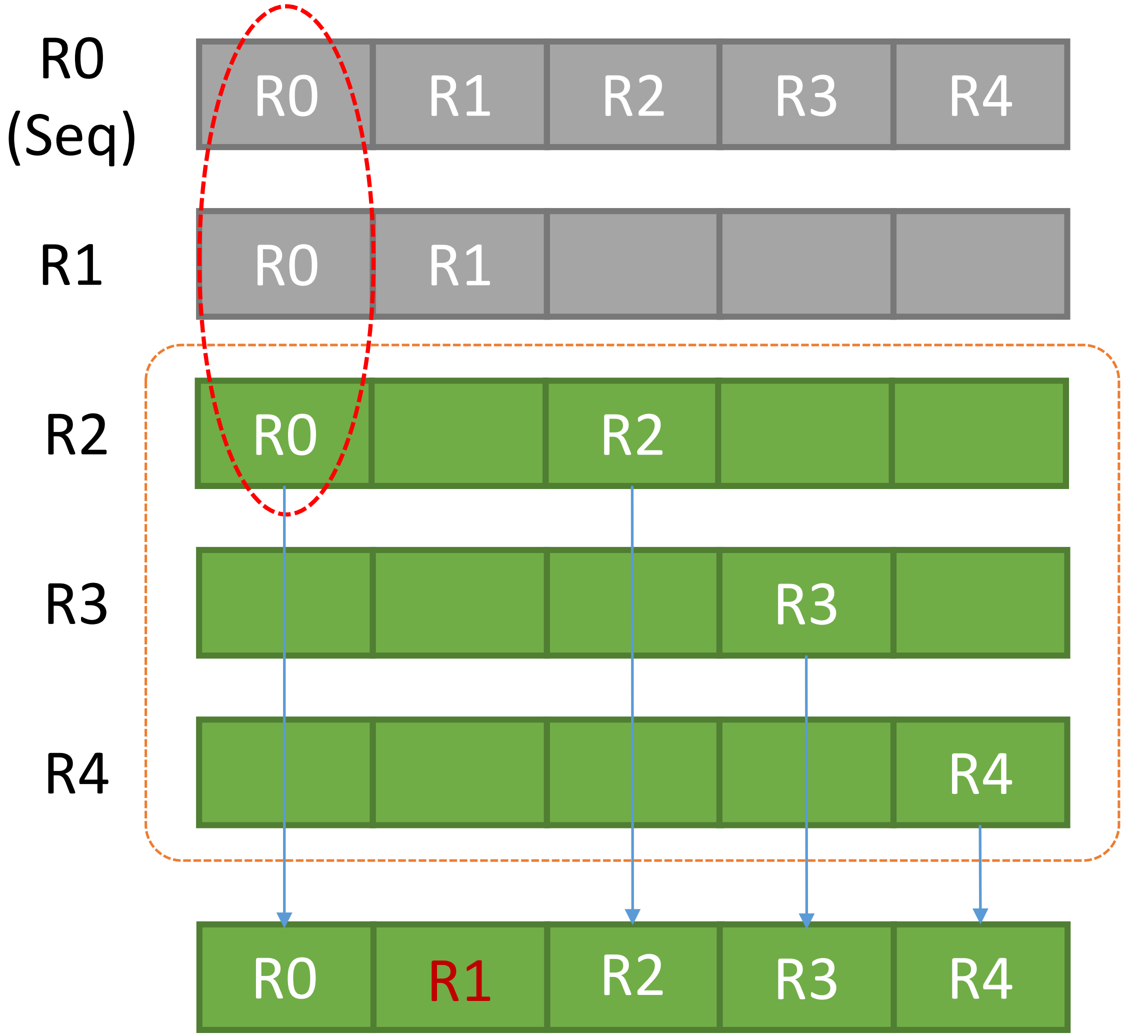


Assignments for the sequencer can be seen by a majority in just one round trip

Handling Failures for 5 Replicas

R0 (Seq)	R0	R1	R2	R3	R4
R1	R0	R1			
R2	R0		R2		
R3				R3	
R4					R4

Handling Failures for 5 Replicas



R0 (Seq)	R0	R1	R2	R3	R4
R1	R0	R1			
R2	R0		R2		
R3				R3	
R4					R4
	R0	R1	R2	R3	R4

More Details in the Paper

- The detailed protocol and fault tolerance approach

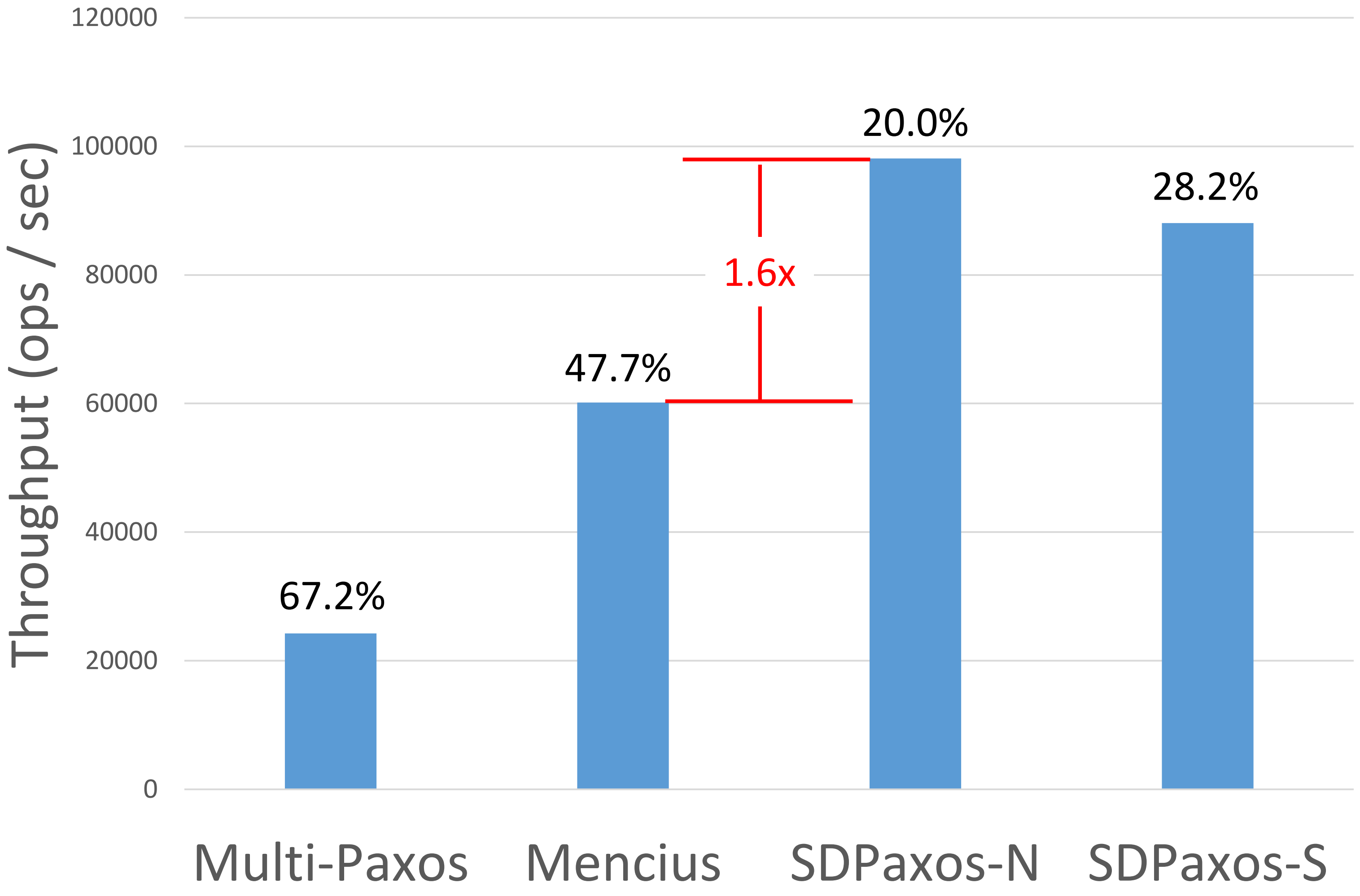
- Reads bypassing Paxos
  - Leveraging the centralized ordering to perform fast and safe reads

- Performance optimizations
  - Lightening the load of ordering
  - Straggler detection
  - …

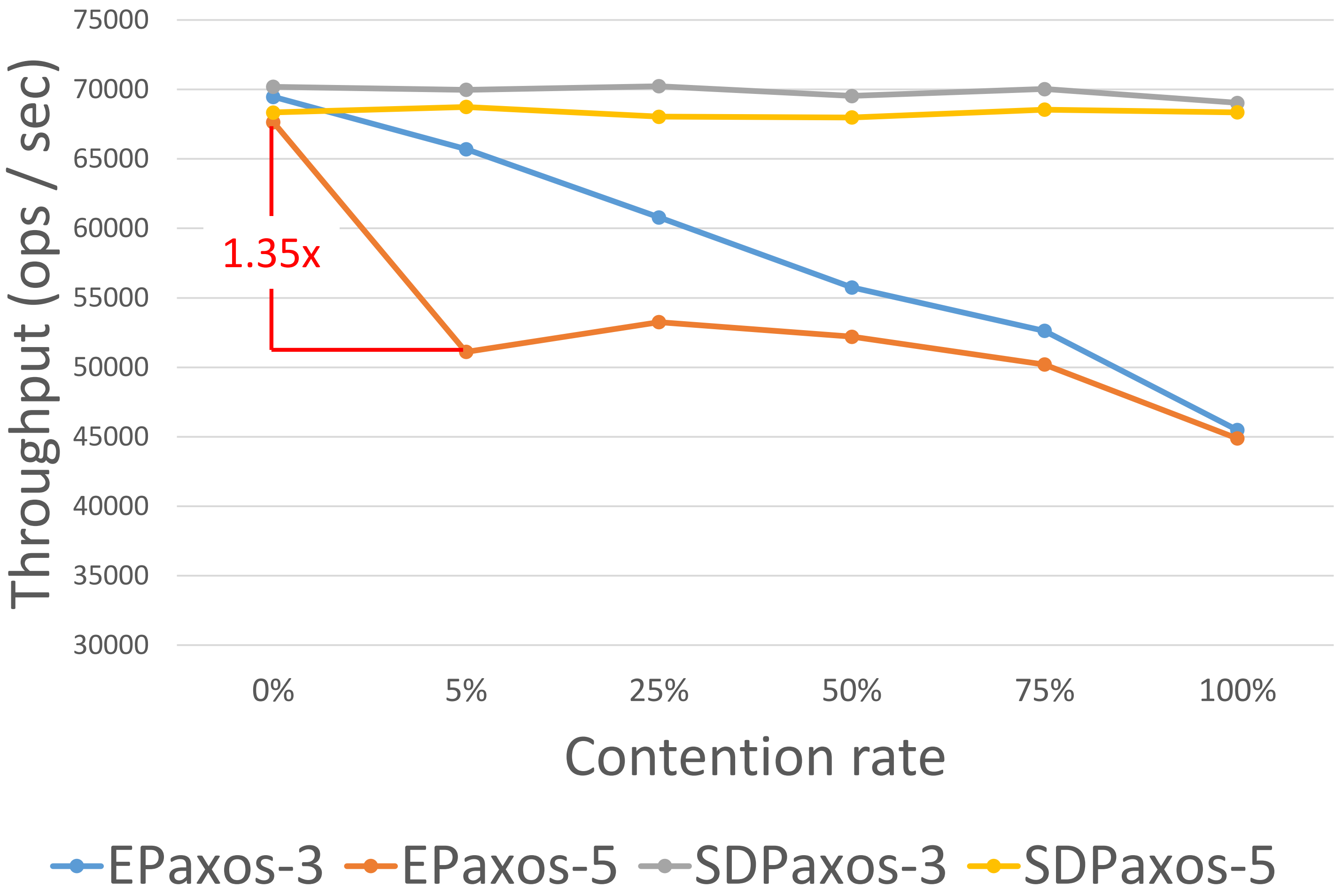
Experimental Setup

- Baselines
  - Multi-Paxos
  - Mencius
  - EPaxos
- Workload: a replicated key-value store
- Testbed: Amazon EC2 m4.large instances
  - Wide-area experiments: CA, OR, OH, IRE, SEL

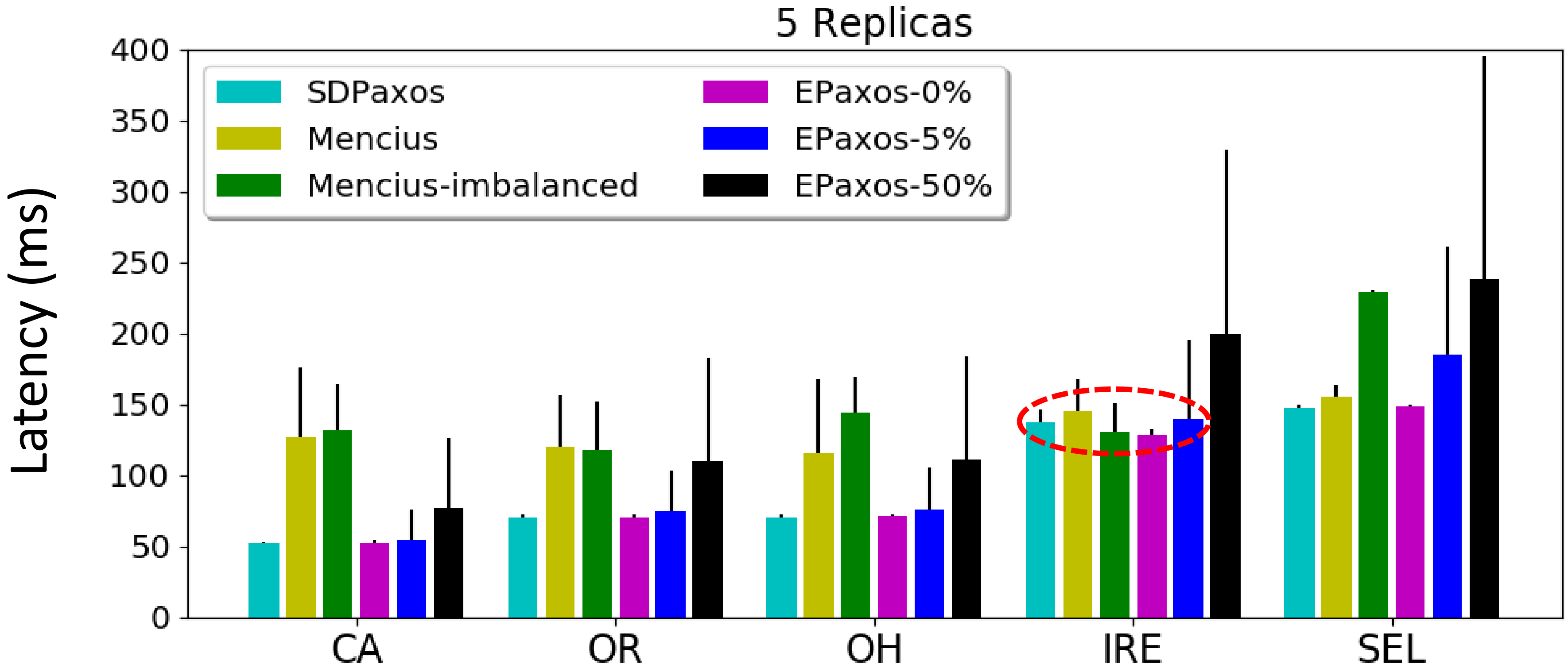
Performance Stability against Stragglers



Performance Stability against Contention



Wide-area Latency



- SDPaxos achieves optimal number of round trips
- SDPaxos's latency is relevant to the distance to the sequencer (IRE)
- SDPaxos's latency is not impacted by stragglers or contention

Conclusion

- The first semi-decentralized SMR protocol
  - High performance
  - Strong performance stability

- One-round-trip under realistic configurations tolerating one or two failures

- High throughput, low latency with stragglers, under contention or in ideal cases

Q & A

Thanks!