MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Mencius
Building Efficient Replicated State Machines for WANs

Yanhua Mao, Flavio P. Junqueira
Keith Marzullo

Presenter: Yile (Michael) Gu

EECS 591

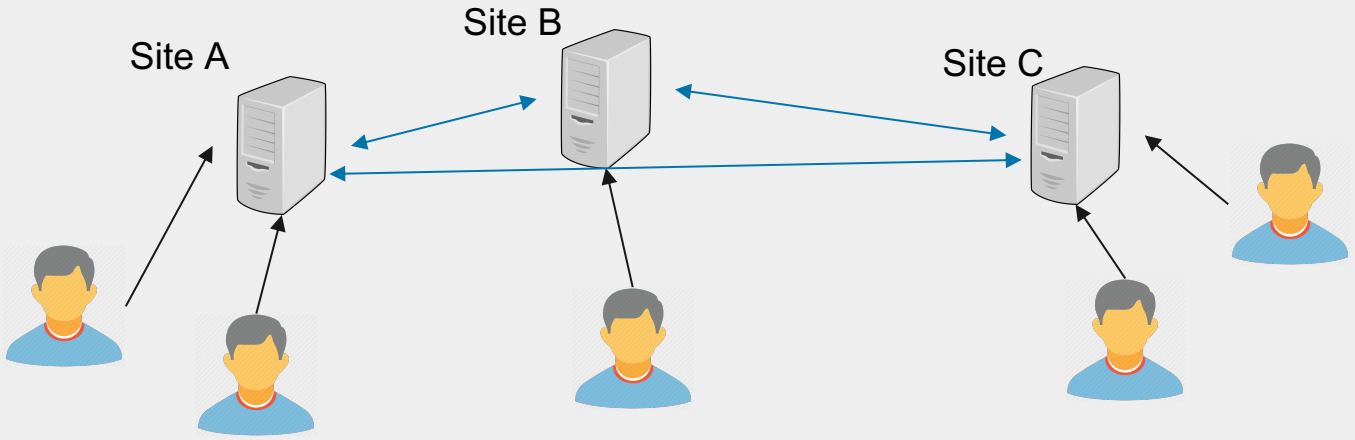
Contents

- Motivation
- Coordinated Paxos
- Deriving Mencius
- Evaluation
- Conclusion

Motivation

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Modern web services are provided in wide-area network

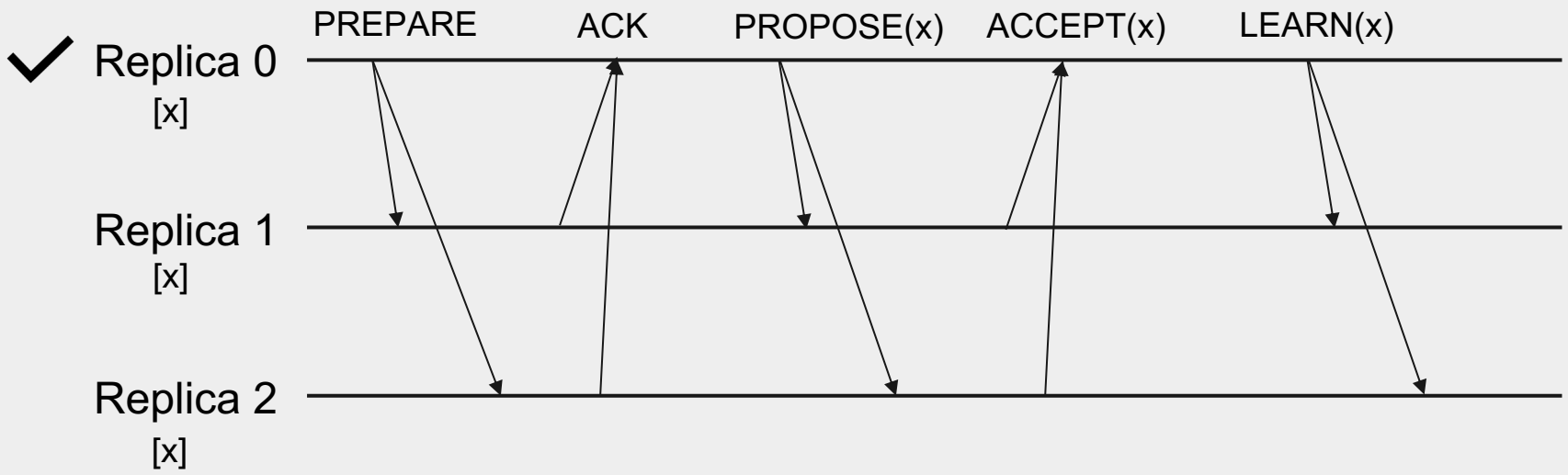


How to provide state machine replication in WAN?

Motivation

Paxos Recap

Phase 3: Leader Election

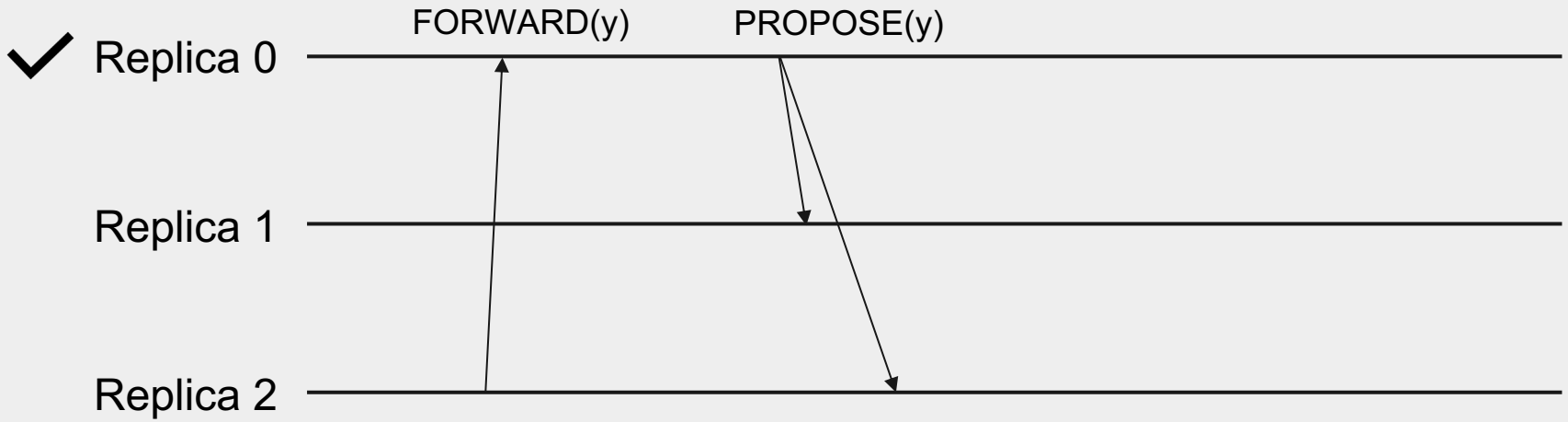


Motivation

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Paxos Recap

Replicas forward client requests to the leader

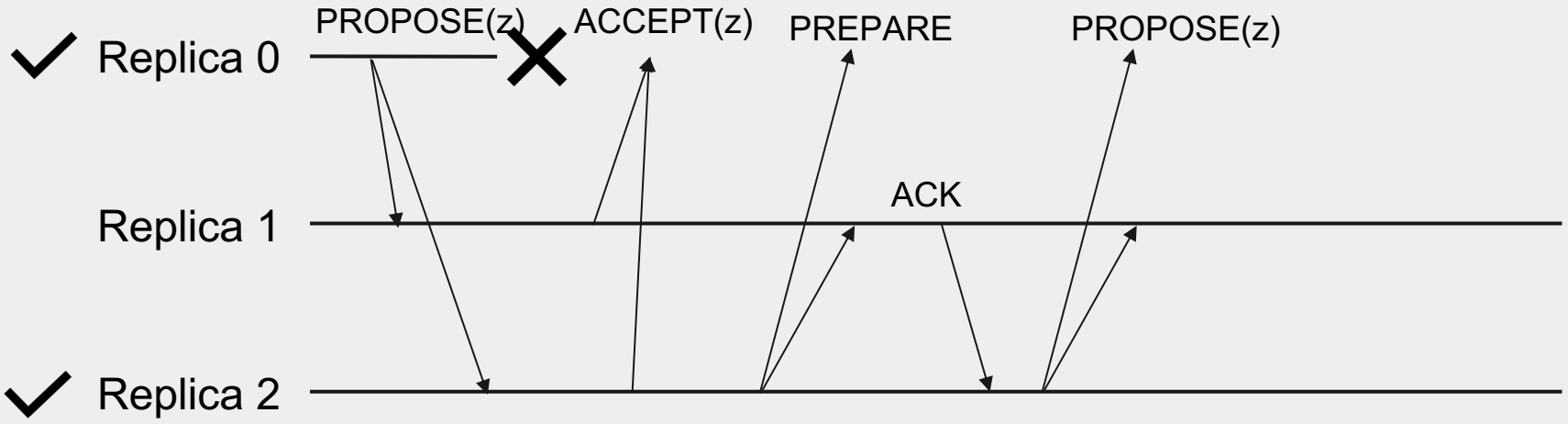


Motivation

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Paxos Recap

Re-elect leader when old leader is suspected



Motivation

Weaknesses of Paxos

Bottleneck at the leader

- Network bandwidth of the leader is saturated first, while channels between others are idle
- CPU utilization of the leader grows linearly as number of replicas grows, undermining scalability

Higher learning latency for replicas

- Replicas get delayed due to FWD and LEARN messages

Coordinated Paxos

Assumptions

Crash failure

- Servers could fail by crash and later possibly recover

Unreliable failure detector

- Detect server failure by timeouts, may have false positives

Asynchronous FIFO communication channel

- TCP transport protocol makes optimizations possible

Coordinated Paxos

Simple Consensus w/ Multiple Instances

- For each instance, only one server is designated as **coordinator**
- Coordinator can propose any command in its instances, while others could only propose *no-op* in them
- For an unbounded number of instances, assign by modular, i.e. server 0: {0, 3, 6, … 3n}, server 1: {1, 4, 7, … 3n+1}, server 2: {2, 5, 8, … 3n+2}

Coordinated Paxos

The coordinator could perform the following actions

Suggest

- This is just like PROPOSE in Paxos

Skip

- A special PROPOSE message containing *no-op*

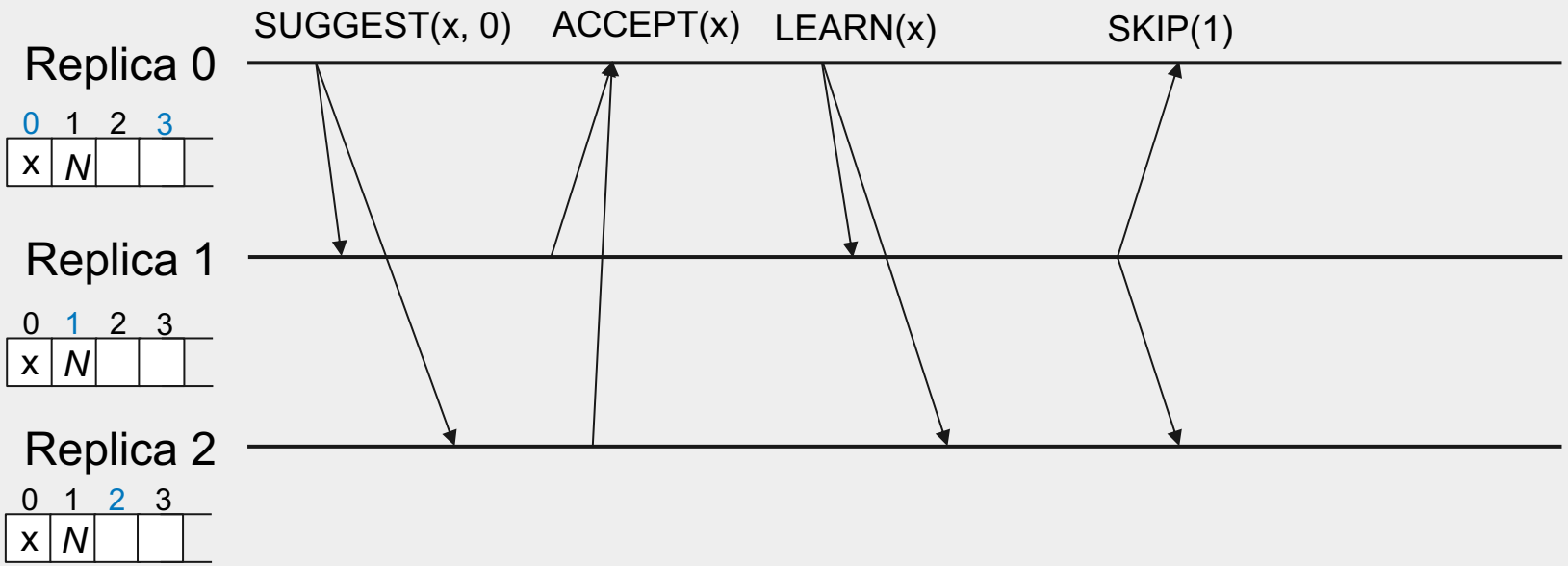
➢ Replicas are safe to learn *no-op* as soon as they receive Skip from the coordinator!

Coordinated Paxos

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Coordinated Paxos

Replica 0 skips first instance 0



Coordinated Paxos

When suspecting that coordinator has failed

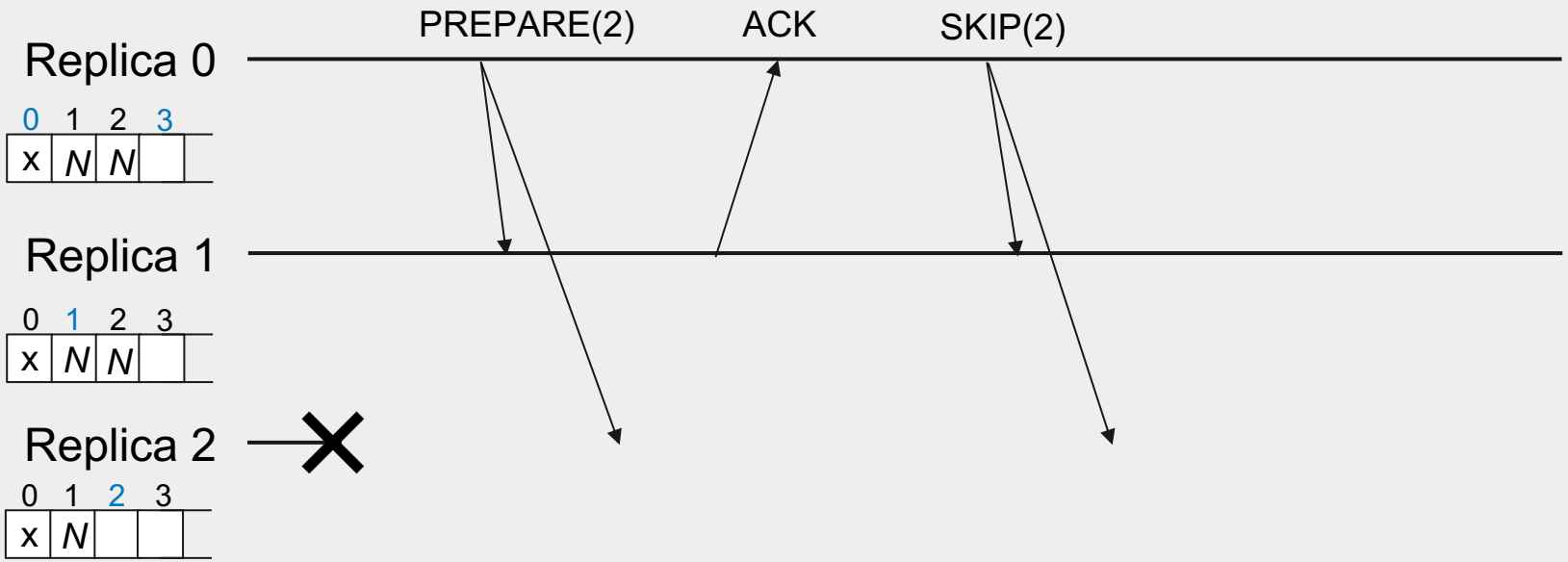
Revoke

- A new leader starts Phase 1 for higher view number
- If the majority acknowledges, start Phase 2
- If no value has been chosen, propose SKIP
- Otherwise, propose SUGGEST on chosen value

Coordinated Paxos

Coordinated Paxos

Replica 0 revokes Replica 2



Coordinated Paxos

Implementing replicated state machines w/ 4 rules

Rule 1

Each server *p* maintains an **index** I_p for the next sequence it should propose, incrementing the index after it suggests upon client request.

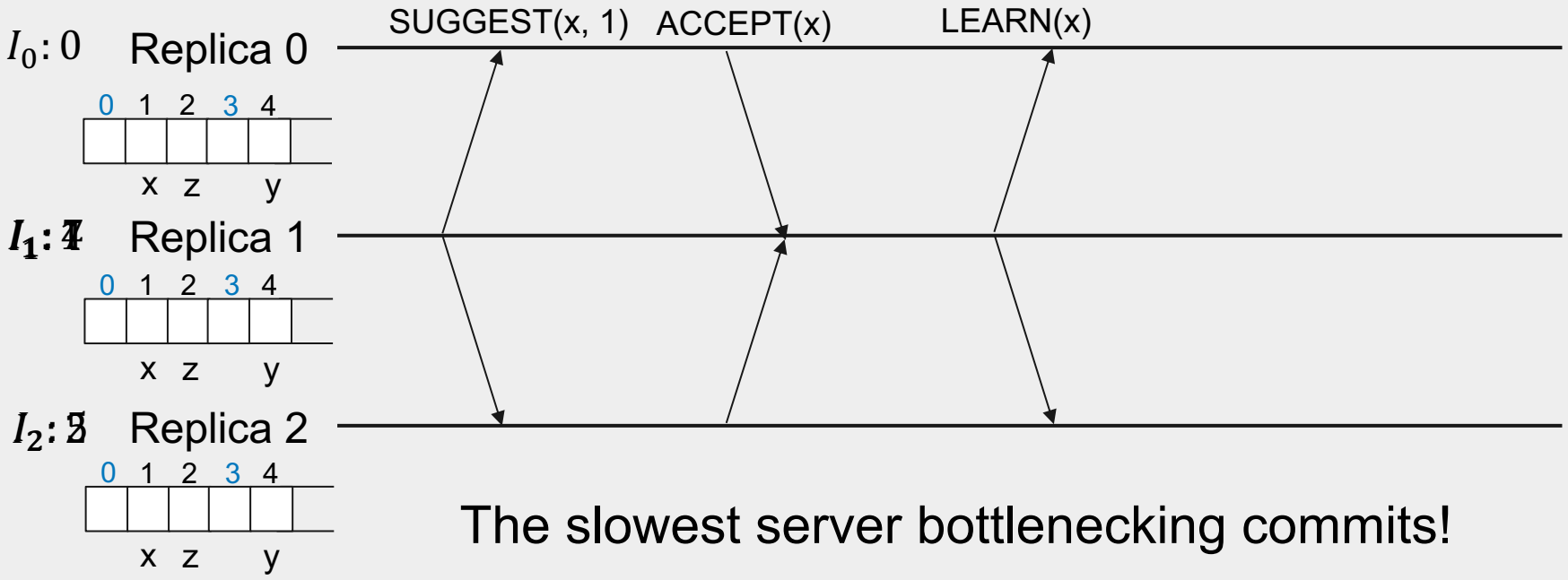
e.g. Replica 1's index is 1, after it suggests on 1, increment to 4

- What if servers generate requests at asymmetric speed?

Coordinated Paxos

Coordinated Paxos

Only Replica 1 & 2 are suggesting requests



The slowest server bottlenecking commits!

Coordinated Paxos

Make slow servers skip their turns

Rule 2

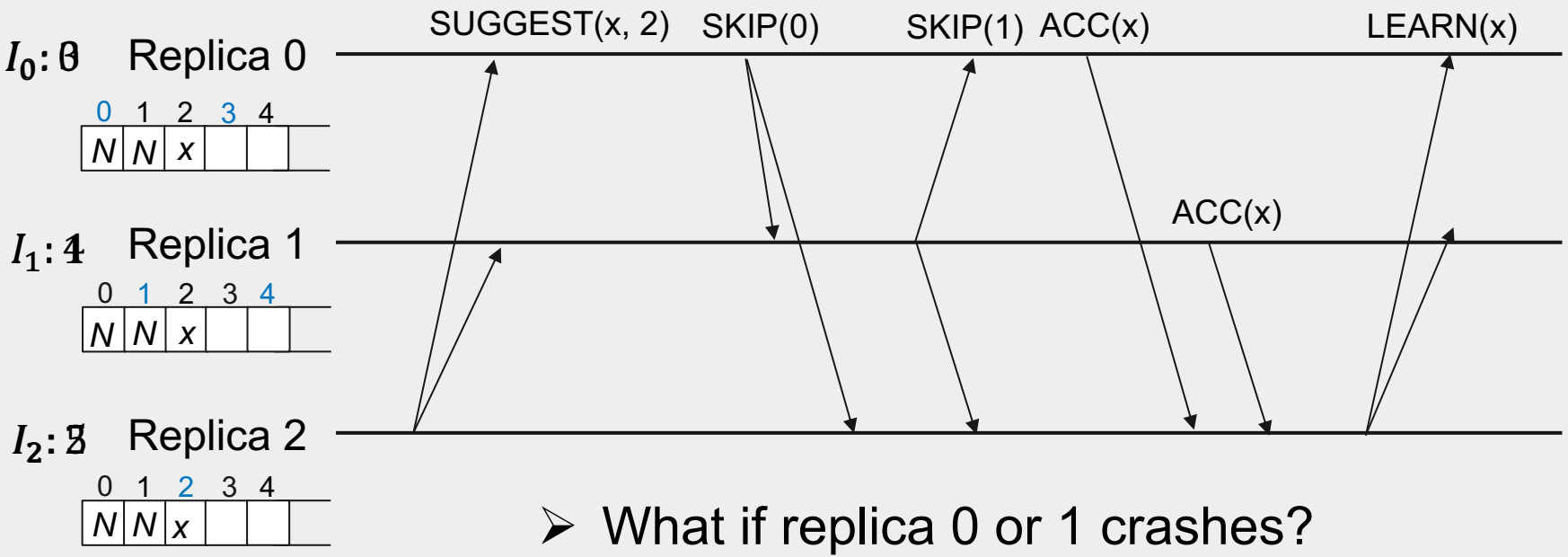
Upon receiving SUGGEST for instance *i*, before sending ACCEPT message, check if $i > I_p$

If so, update index I_p to the smallest instance *p* coordinates, such that $I'_p > i$, and executes SKIP for instance in $[I_p, I'_p]$ that *p* coordinates

Coordinated Paxos

Coordinated Paxos

Only Replica 2 suggests requests



- What if replica 0 or 1 crashes?

Coordinated Paxos

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Rule 3

If server *p* suspects that server *q* has failed, and suppose C_q is the smallest instance coordinated by *q* but not yet learned by *p*, *p* revokes *q* for instances $[C_q, I_p]$ that *q* coordinates

Rule 4

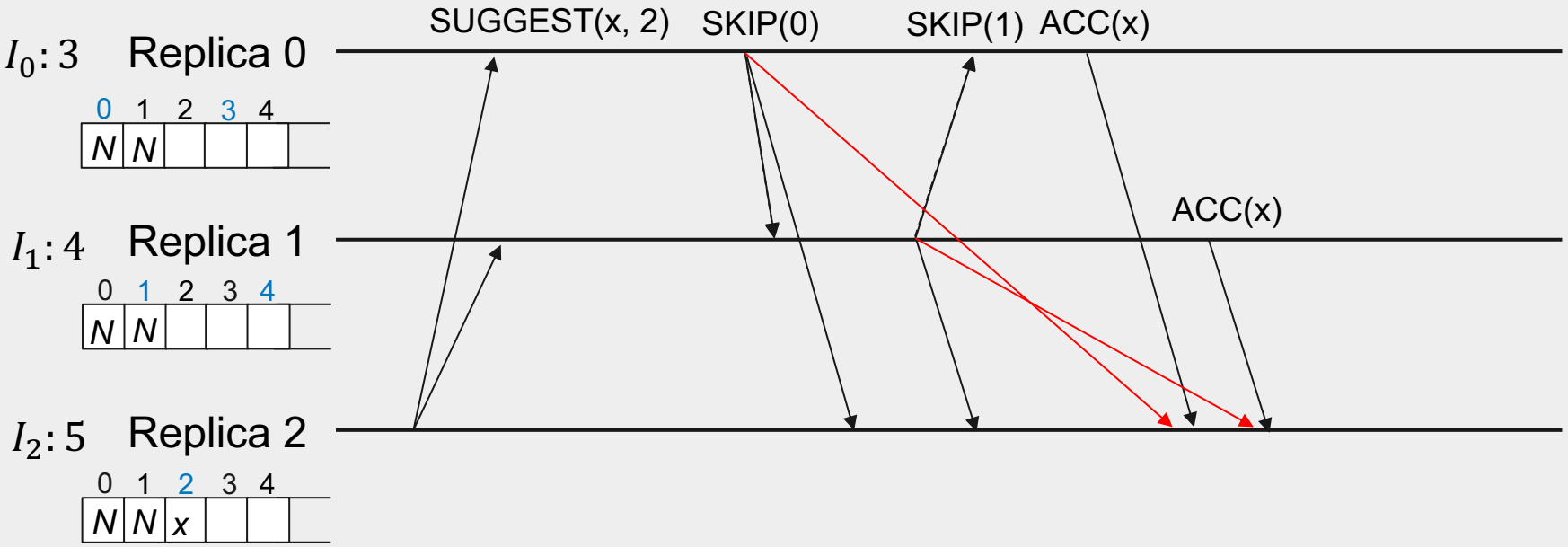
If server *p* suggests a value that is not no-op for instance *i,* but it learns that no-op is chosen for *i*, then *p* suggests the value again

- What if server *q* is falsely suspected, i.e., the network delays the message?

Deriving Mencius

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Could we further reduce message complexity?



Deriving Mencius

Optimization 1

Upon receiving SUGGEST for instance *i* from *q, p* does not send a separate SKIP message to *q* before ACCEPT. *p* uses the ACCEPT message to imply that it would not suggest requests for instance smaller than *i*.

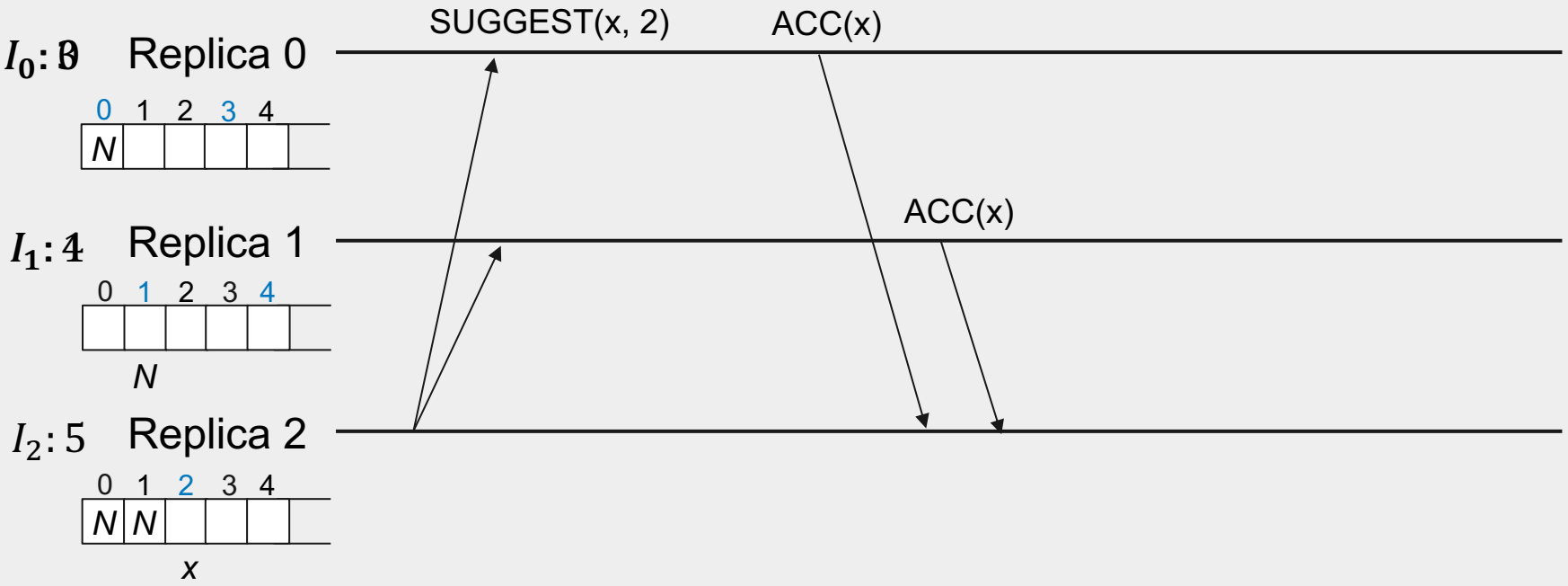
Optimization 2

Also, *p* does not broadcast SKIP to other replicas. *p* waits for SUGGEST from other replicas, and uses ACCEPT message to imply SKIPs similarly

Deriving Mencius

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

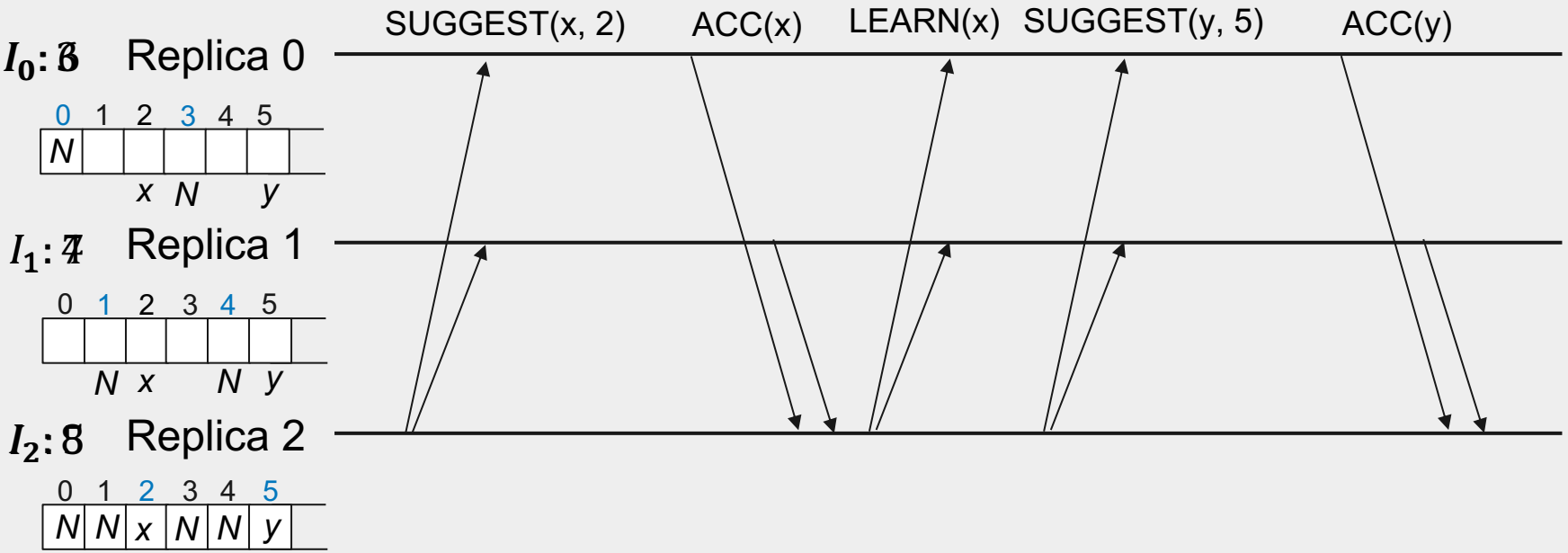
If only replica 2 is suggesting values



Deriving Mencius

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Idle servers take longer time to make progress



Deriving Mencius

Accelerator 1

A server *p* synchronizes its SKIP messages to server *r*, if outstanding SKIPs for *r* is larger than α, or deferred time longer than τ

- Could we reduce message complexity when a server crashes?

Deriving Mencius

Rule 3

If server *p* suspects that server *q* has failed, and suppose C_q is the smallest instance coordinated by *q* but not yet learned by *p*, *p* revokes *q* for instances $[C_q, I_p]$ that *q* coordinates

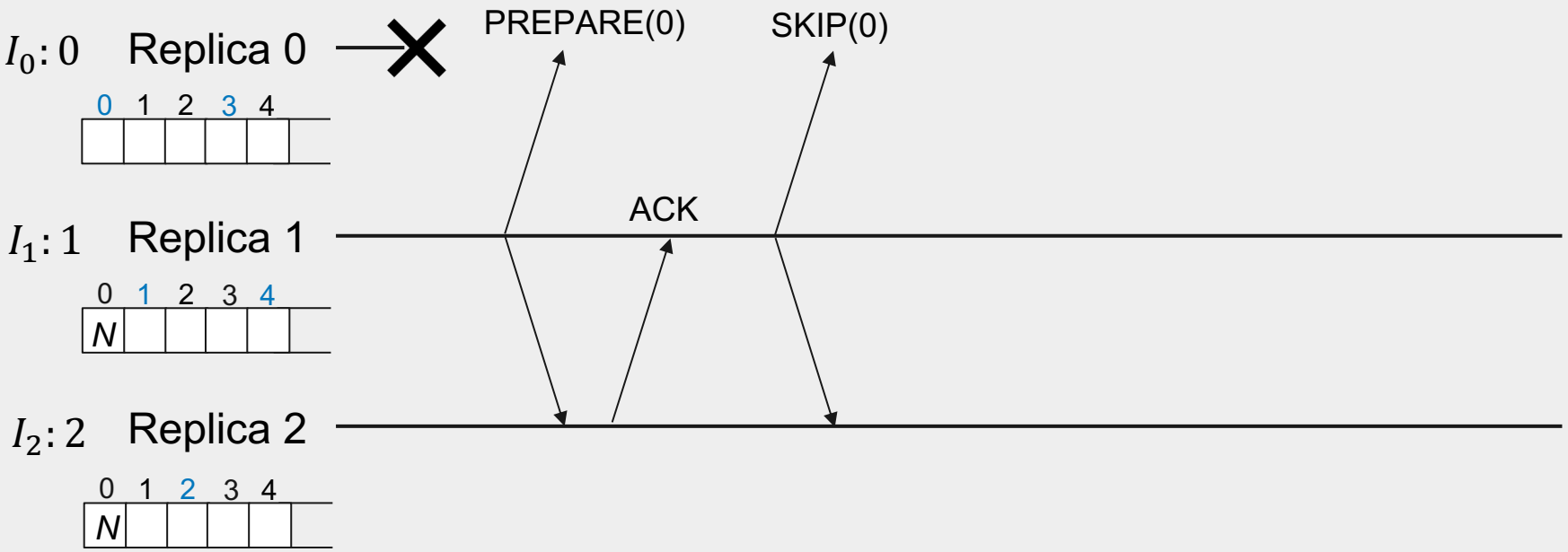
Rule 4

If server *p* suggests a value that is not no-op for instance *i,* but it learns that no-op is chosen for *i*, then *p* suggests the value again

Deriving Mencius

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

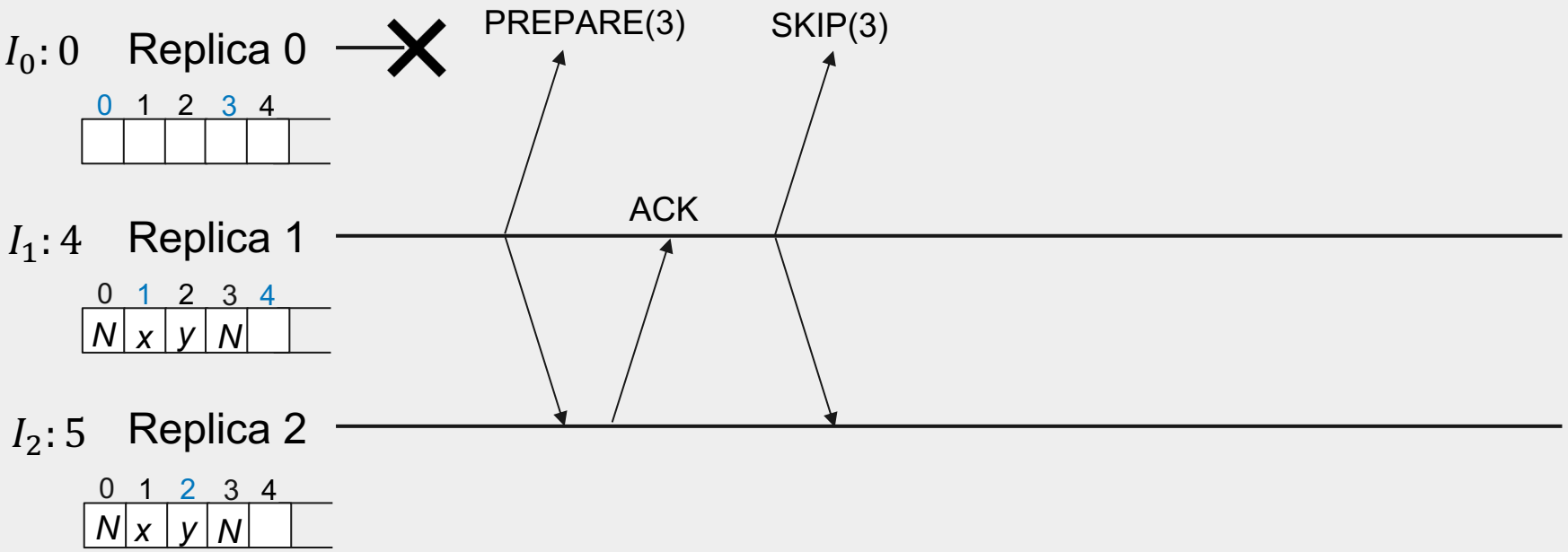
Revoking Replica 0 is repeated



Deriving Mencius

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Revoking Replica 0 is repeated



Deriving Mencius

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Accelerator 1

A server *p* synchronizes its SKIP messages to server *r*, if outstanding SKIPs for *r* is larger than α, or deferred time longer than τ

Optimization 3

If server *p* suspects that server *q* has failed, and suppose C_q is the smallest instance coordinated by *q* but not yet learned by *p*, *p* revokes *q* for instances $[C_q, I_p + 2\beta]$ that *q* coordinates

➢ Could we reduce message complexity when a server crashes?

Evaluation

Experiment Settings

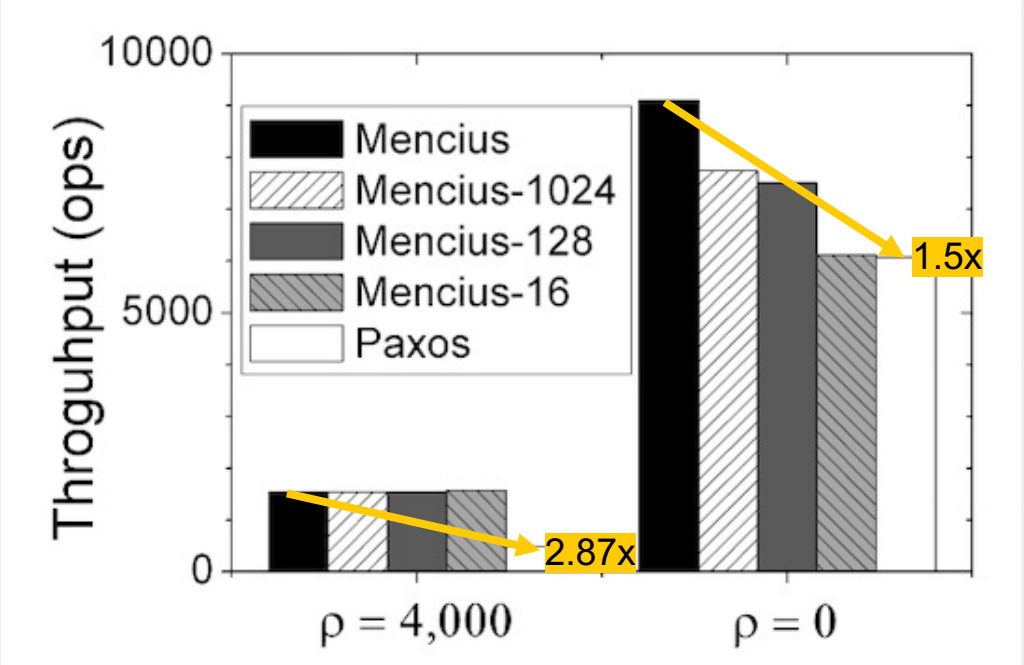
- Use Mencius to implement a read/write register service for K registers
- Every command could either be read or write
- Every command may contain a payload of ρ bytes
- Three sites simulating three datacenters, each site contains one server node
- When ρ = 4000, the system is network-bound, and when ρ = 0, the system is CPU-bound

Evaluation

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Throughput

- When network-bound, Paxos is limited by the leader's bandwidth
- When CPU-bound, Paxos reaches 100% CPU utilization on leader, but 50% on others. Mencius reaches 100% CPU utilization on all 3 servers

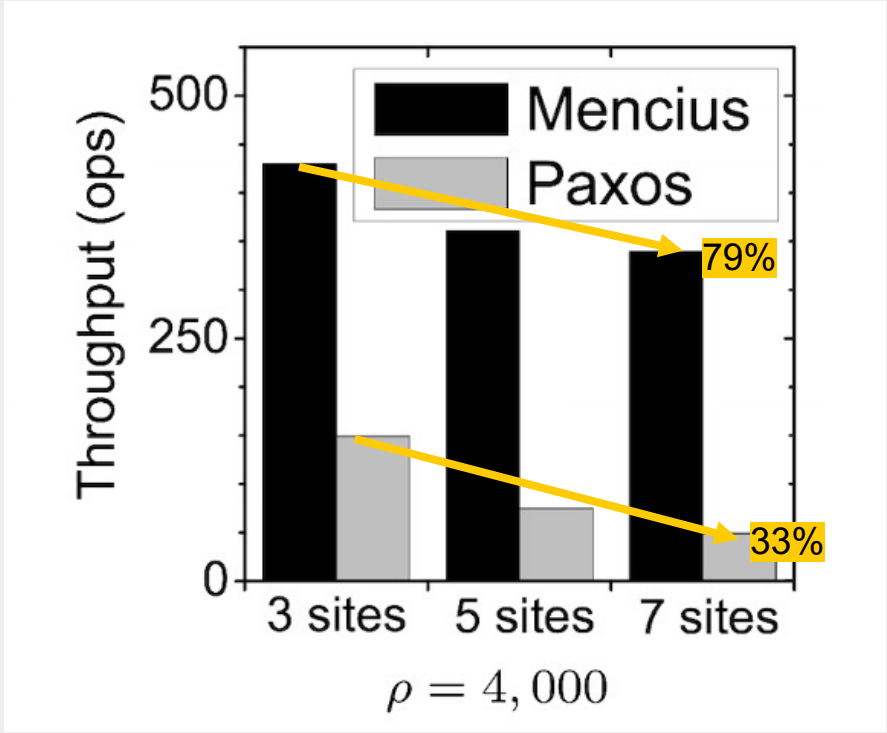


Evaluation

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Scalability

- When CPU-bound, Paxos experiences bottleneck at leader, while Mencius could use extra processing power
- When network-bound, Mencius uses bandwidth provided by new sites, thus experiencing smaller throughput drop

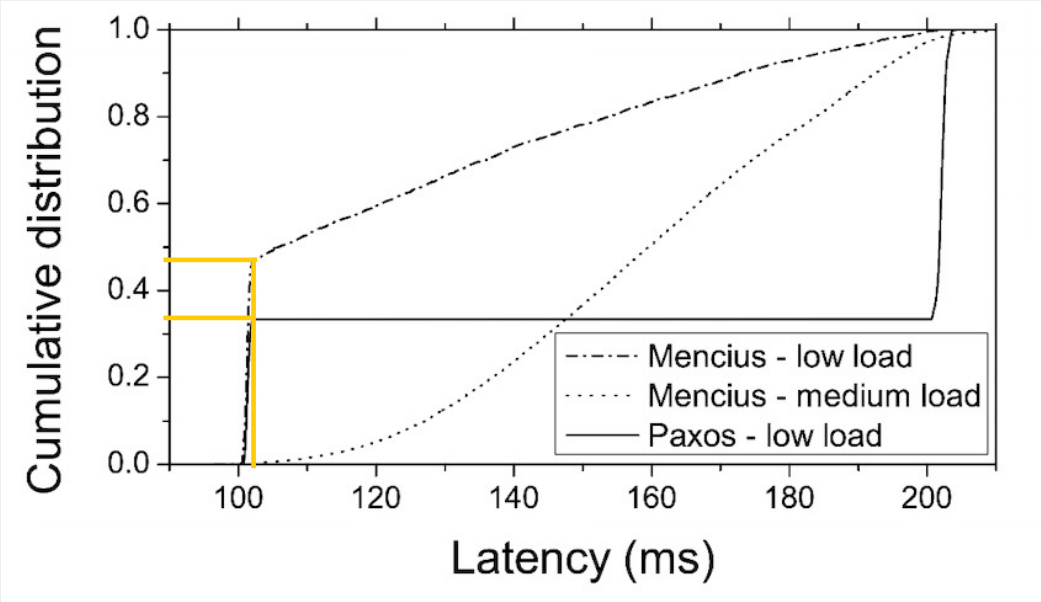


Evaluation

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Latency

- For Paxos, one third of the requests has 100ms commit latency, while 50% of Mencius has so
- Average commit latency is 167ms for Paxos, and 155ms for Mencius



Conclusion

Mencius Summary

- Mencius = Coordinated Paxos + 3 Optimizations + 1 Accelerator
- Adjustable parameters for different network settings
- Similar to Paxos, tolerate *f* failures for totally 2*f* + 1 servers
- Higher throughput than Paxos when either CPU-bound or network bound
- Better scalability and latency than Paxos

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Discussion

- In terms of coordinator and assume no failures, is Mencius always non-blocking upon receiving a quorum of ACCEPT messages?
- Despite improved throughput and latency of Mencius compared to Paxos, it seems that many infrastructures still use Paxos as the commit protocol. What are some of the cases where Mencius may not work?