

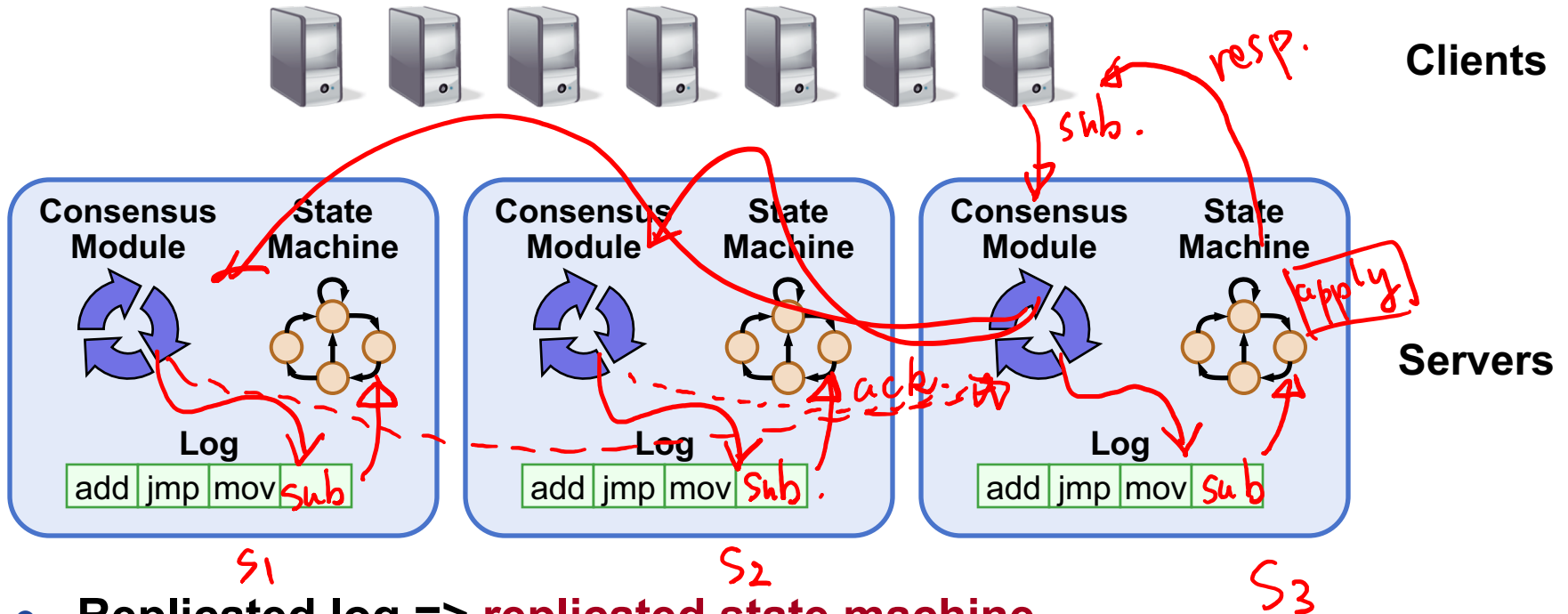
Implementing Replicated Logs with Paxos

John Ousterhout and Diego Ongaro
Stanford University



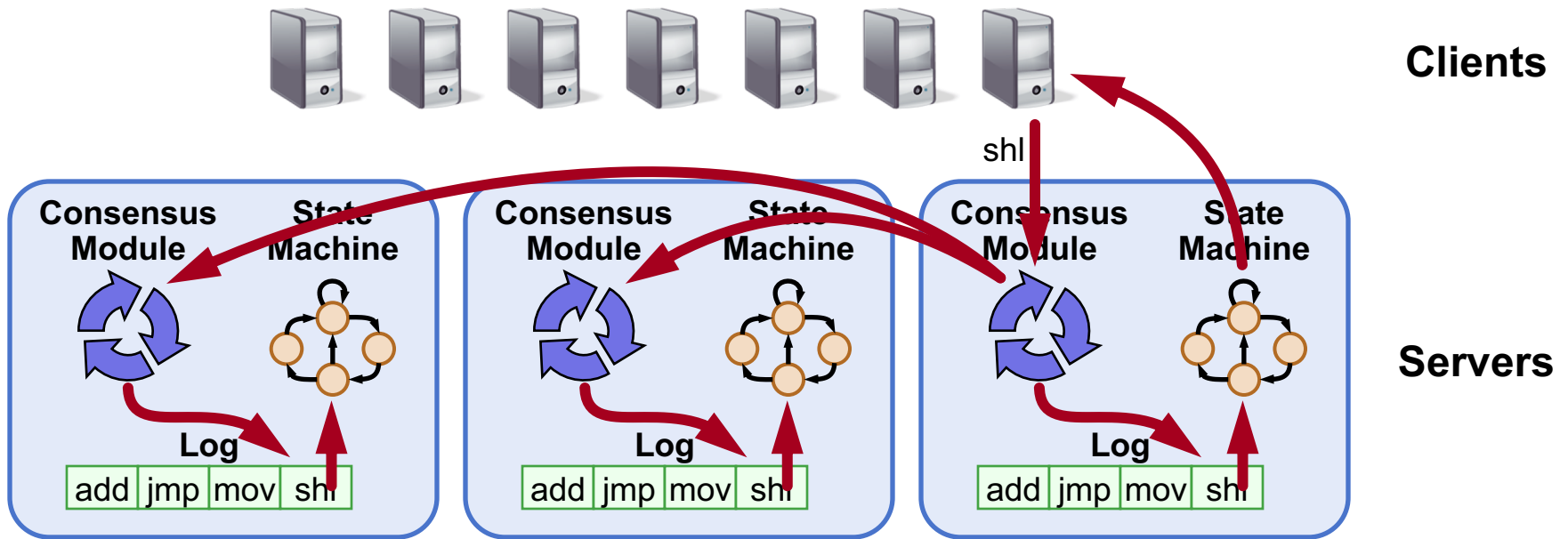
Note: this material borrows heavily from slides by Lorenzo Alvisi, Ali Ghodsi, and David Mazières

Goal: Replicated Log



- **Replicated log => replicated state machine**
 - All servers execute same commands in same order
- **Consensus module ensures proper log replication**
- **System makes progress as long as any majority of servers are up**
- **Failure model: fail-stop (not Byzantine), delayed/lost messages**

Goal: Replicated Log



- **Replicated log => replicated state machine**
 - All servers execute same commands in same order
- **Consensus module ensures proper log replication**
- **System makes progress as long as any majority of servers are up**
- **Failure model: fail-stop (not Byzantine), delayed/lost messages**

The Paxos Approach

Decompose the problem:

- **Basic Paxos (“single decree”):**
 - One or more servers propose values
 - System must agree on a **single value** as **chosen**
 - Only one value is ever chosen
- **Multi-Paxos:**
 - Combine several instances of Basic Paxos to agree on a series of values forming the log

Requirements for Basic Paxos

- **Safety:**

- Only a single value may be chosen
- A server never learns that a value has been chosen unless it really has been

agreement

validity

- **Liveness (as long as majority of servers up and communicating with reasonable timeliness):**

- Some proposed value is eventually chosen
- If a value is chosen, servers eventually learn about it

termination

The term “consensus problem” typically refers to this single-value formulation

Paxos Components

- **Proposers:**

- Active: put forth particular values to be chosen
- Handle client requests

- **Acceptors:**

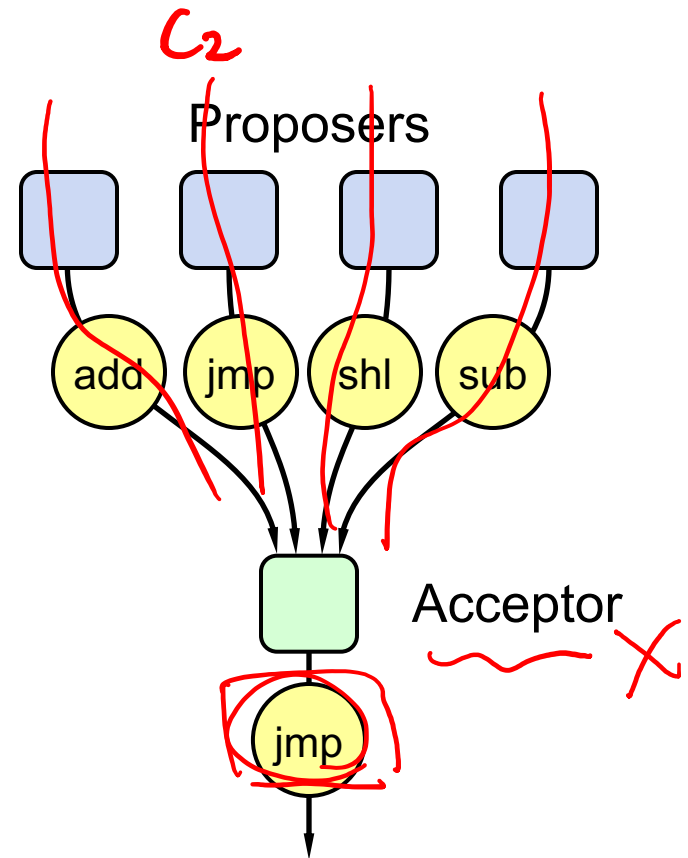
- Passive: respond to messages from proposers
- Responses represent votes that form consensus
- Store chosen value, state of the decision process
- Want to know which value was chosen

For this presentation:

- Each Paxos server contains both components

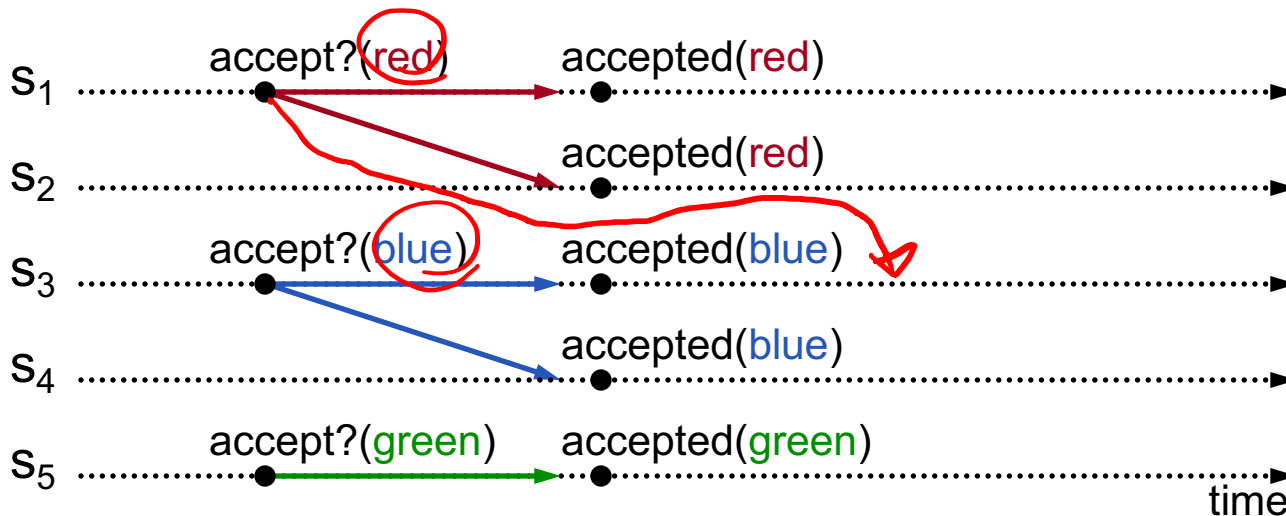
Strawman: Single Acceptor

- **Simple (incorrect) approach:** a single acceptor chooses value
- **What if acceptor crashes after choosing?**
- **Solution: quorum**
 - Multiple acceptors (3, 5, ...)
 - Value v is **chosen** if accepted by **majority** of acceptors
 - If one acceptor crashes, chosen value still available



Problem: Split Votes

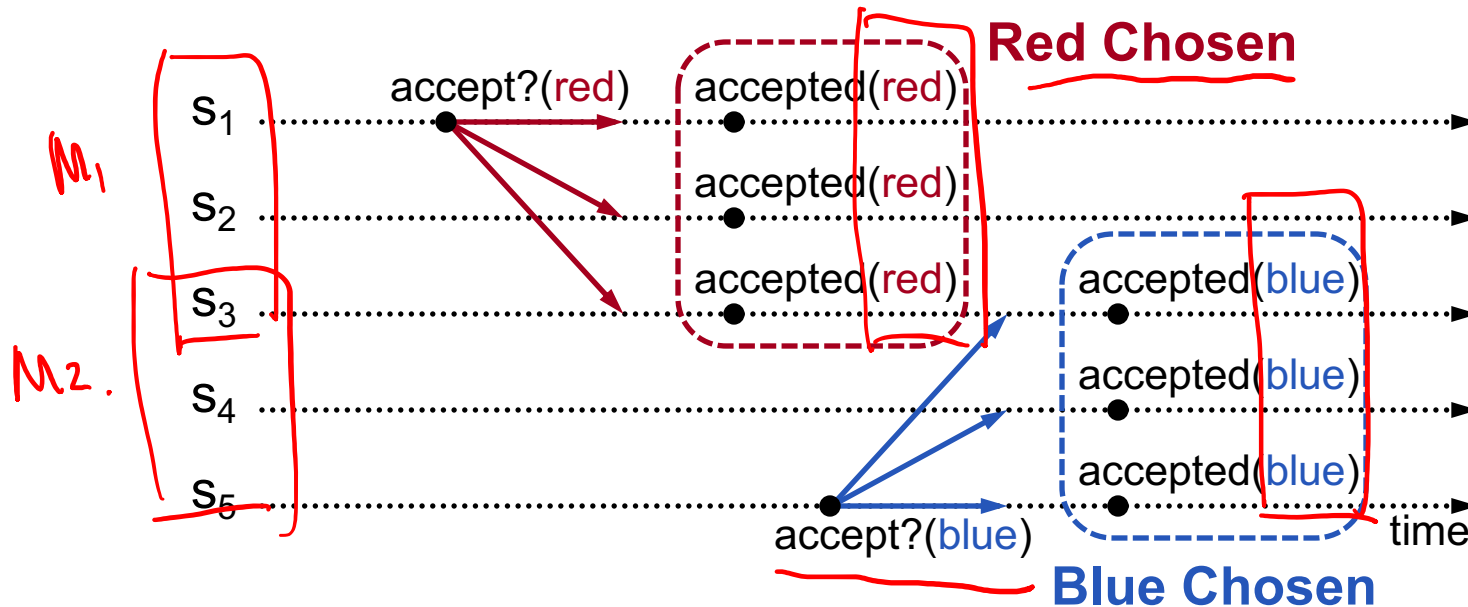
- **Acceptor accepts only first value it receives?**
- **If simultaneous proposals, no value might be chosen**



Acceptors must sometimes accept multiple (different) values

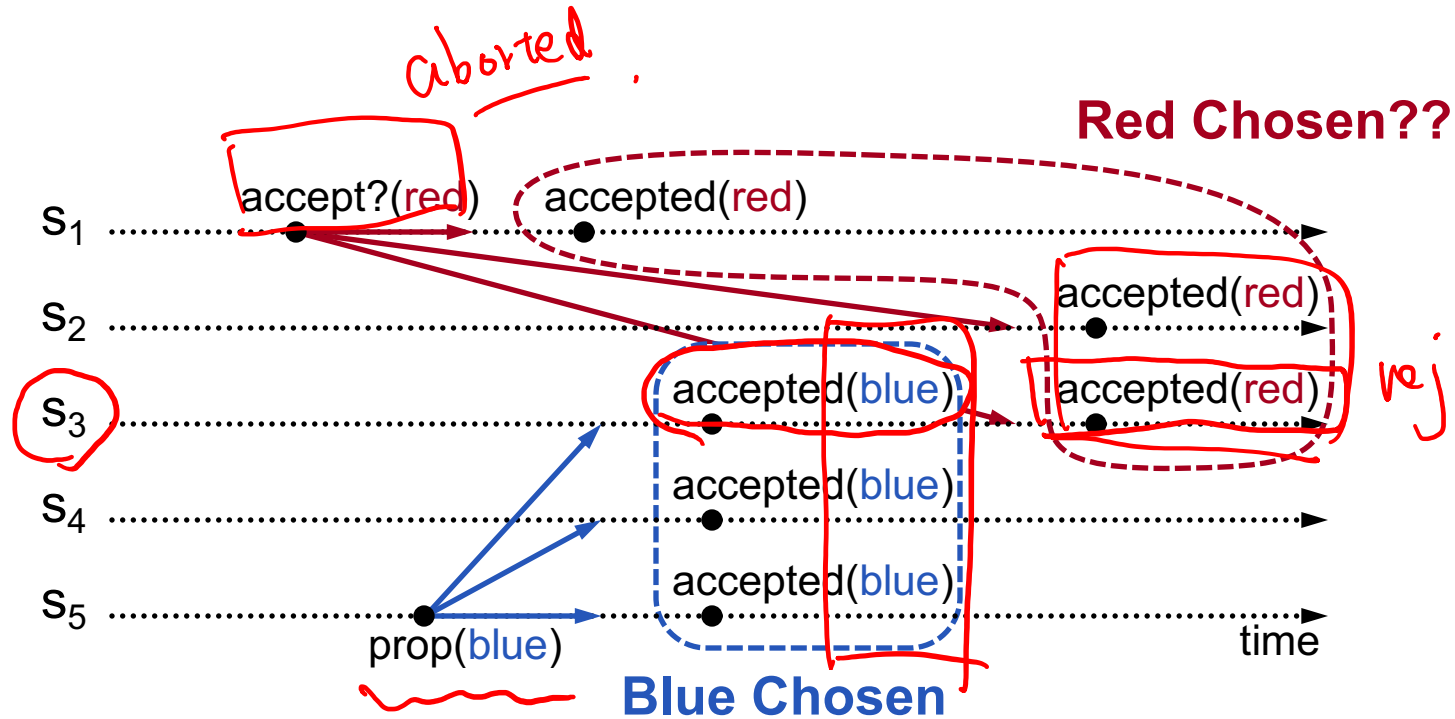
Problem: Conflicting Choices

- Acceptor accepts **every** value it receives?
- Could choose multiple values



Once a value has been chosen, future proposals must propose/choose that same value (**2-phase protocol**)

Conflicting Choices, cont'd



- **s₅** needn't propose **red** (it hasn't been chosen yet)
- **s₁**'s proposal must be aborted (**s₃** must reject it)

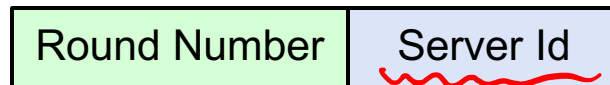
Must order proposals, reject old ones

Proposal Numbers

- **Each proposal has a unique number**
 - Higher numbers take priority over lower numbers
 - It must be possible for a proposer to choose a new proposal number higher than anything it has seen/used before

- **One simple approach:**

Proposal Number



break the tie

- Each server stores maxRound: the largest Round Number it has seen so far
- To generate a new proposal number:
 - Increment maxRound
 - Concatenate with Server Id
- Proposers must persist maxRound on disk: must not reuse proposal numbers after crash/restart

Basic Paxos

Two-phase approach:

- **Phase 1: broadcast Prepare RPCs**
 - Find out about any chosen values
 - Block older proposals that have not yet completed
- **Phase 2: broadcast Accept RPCs**
 - Ask acceptors to accept a specific value

Basic Paxos

Proposers

1) Choose new proposal number n

2) Broadcast Prepare(n) to all servers

4) When responses received from majority:

- If any acceptedValues returned, replace value with acceptedValue for highest acceptedProposal

5) Broadcast Accept(n , value) to all servers

6) When responses received from majority:

- Any rejections (result > n)? goto (1)
- Otherwise, **value is chosen**

Acceptors

3) Respond to Prepare(n):

- If $n > \text{minProposal}$ then minProposal = n
- Return(acceptedProposal, acceptedValue)

6) Respond to Accept(n , value):

- If $n \geq \text{minProposal}$ then acceptedProposal = minProposal = n
acceptedValue = value
- Return(minProposal)

else \bar{j} rej

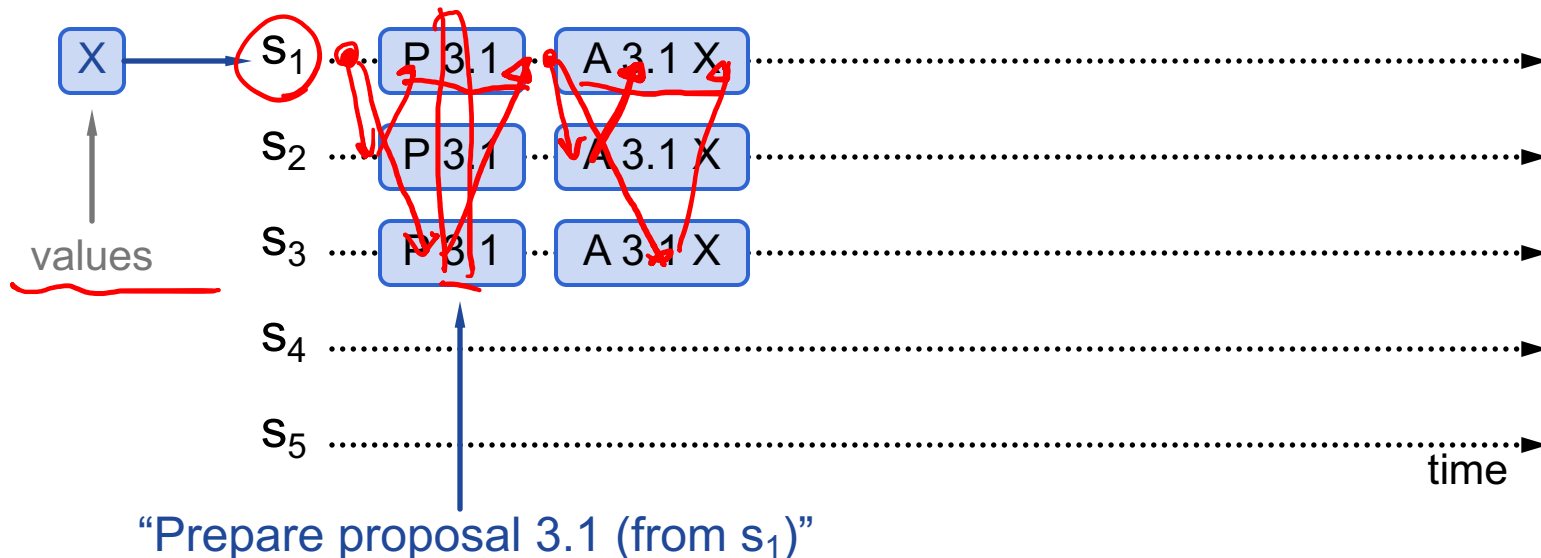
Acceptors must record minProposal, acceptedProposal, and acceptedValue on stable storage (disk)

Basic Paxos Examples

Three possibilities when later proposal prepares:

1. Previous value already chosen:

- New proposer will find it and use it

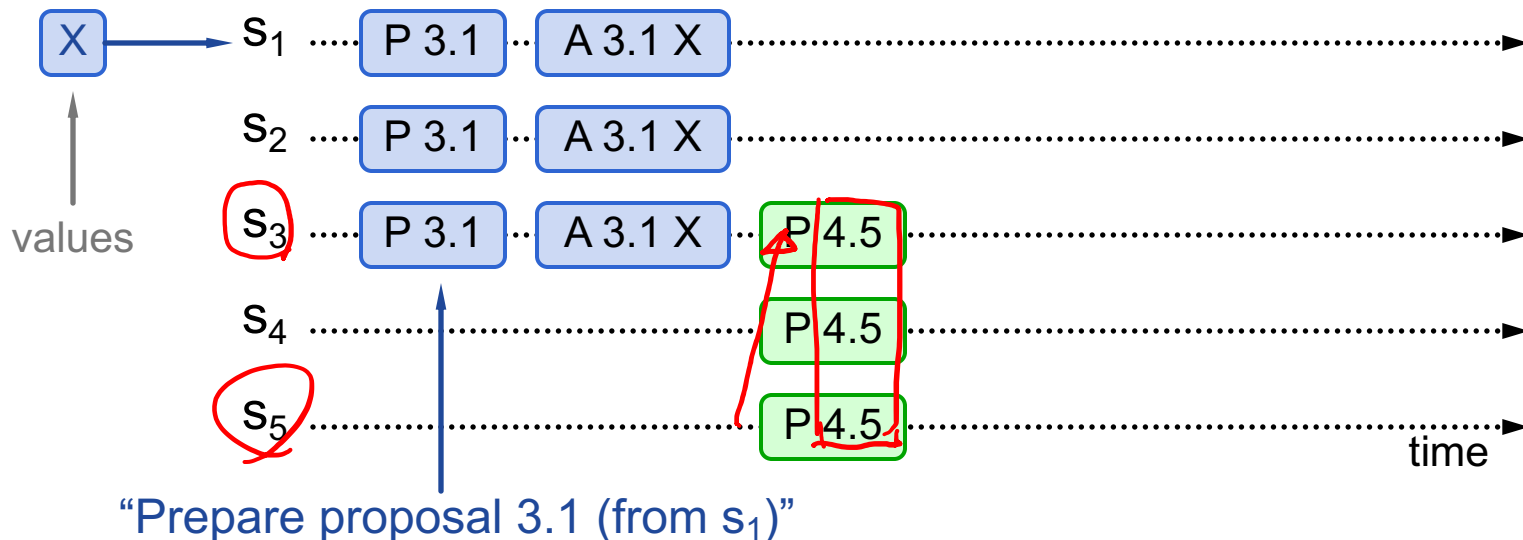


Basic Paxos Examples

Three possibilities when later proposal prepares:

1. Previous value already chosen:

- New proposer will find it and use it

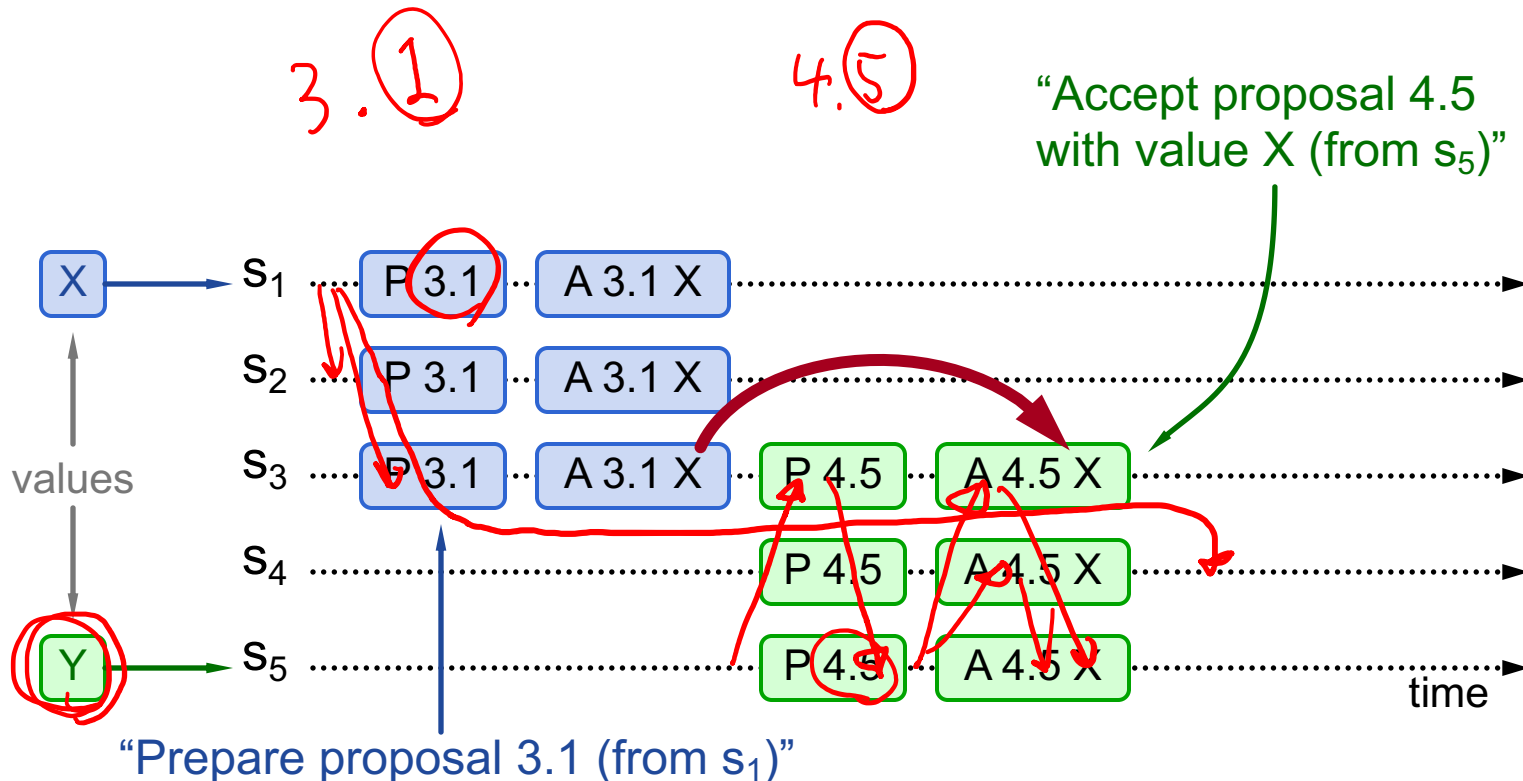


Basic Paxos Examples

Three possibilities when later proposal prepares:

1. Previous value already chosen:

- New proposer will find it and use it

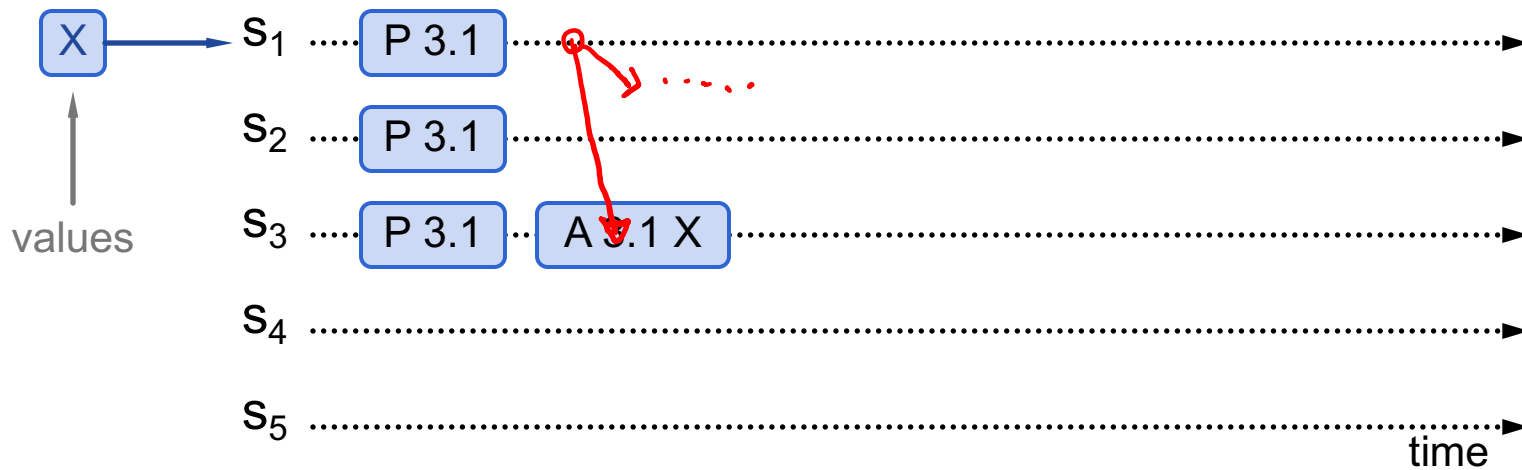


Basic Paxos Examples, cont'd

Three possibilities when later proposal prepares:

2. Previous value not chosen, but new proposer sees it:

- New proposer will use existing value
- Both proposers can succeed

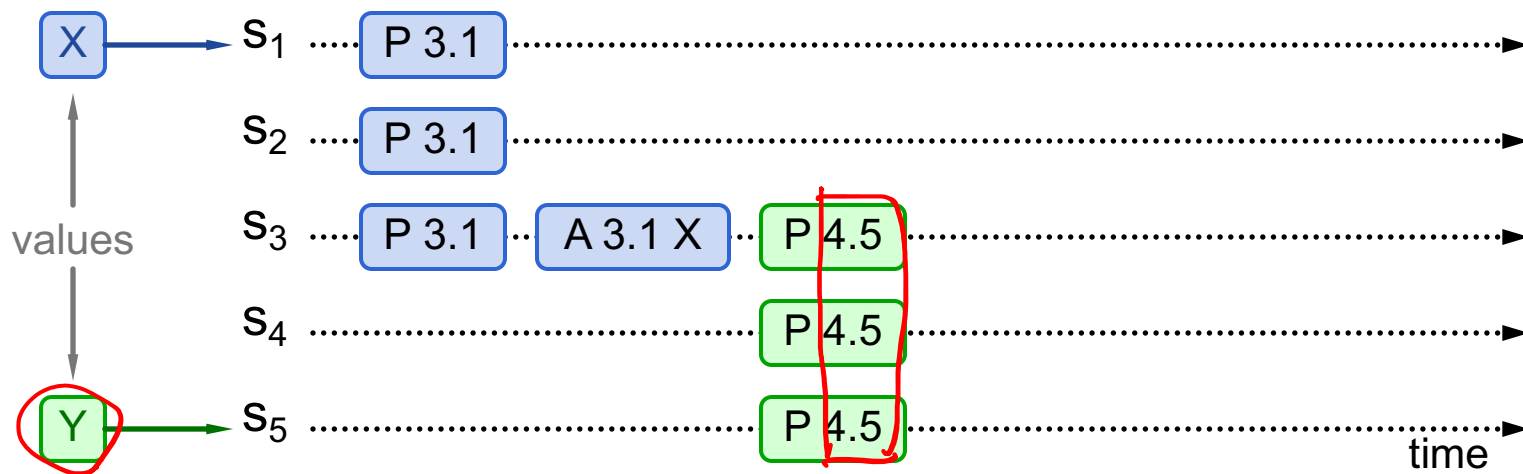


Basic Paxos Examples, cont'd

Three possibilities when later proposal prepares:

2. Previous value not chosen, but new proposer sees it:

- New proposer will use existing value
- Both proposers can succeed

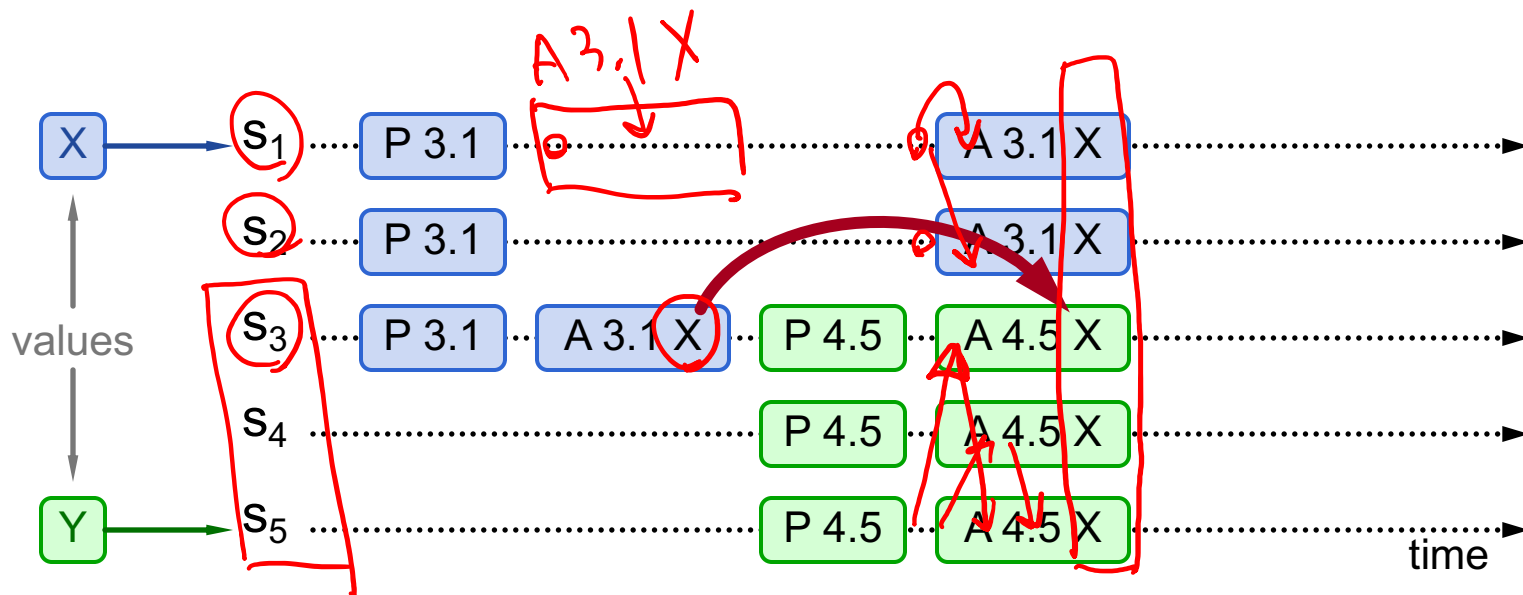


Basic Paxos Examples, cont'd

Three possibilities when later proposal prepares:

2. Previous value not chosen, but new proposer sees it:

- New proposer will use existing value
- Both proposers can succeed

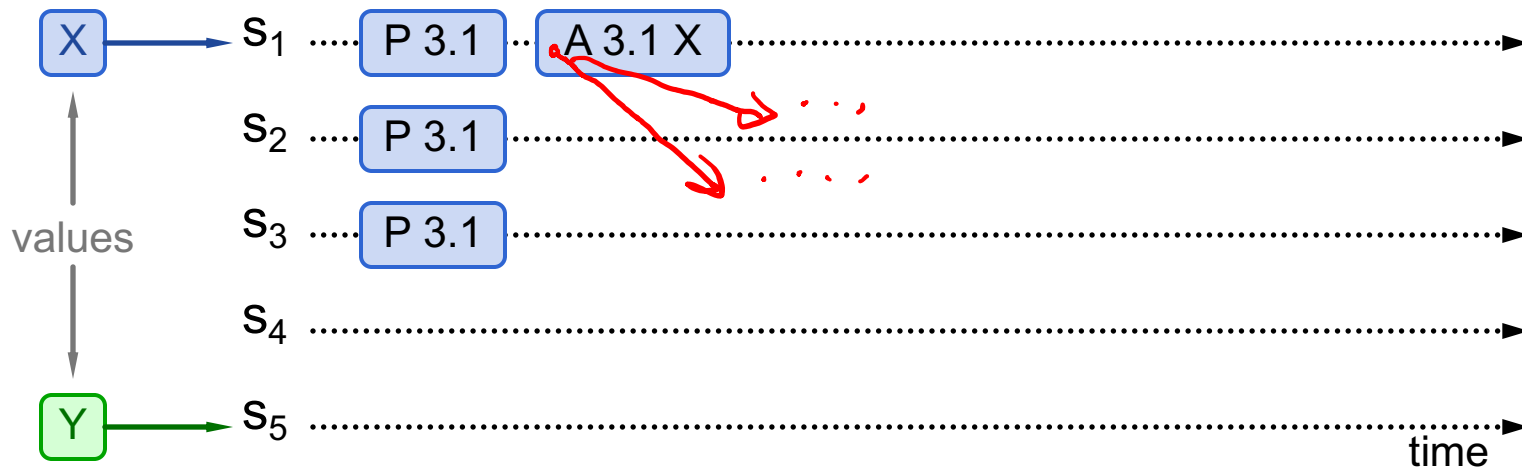


Basic Paxos Examples, cont'd

Three possibilities when later proposal prepares:

3. Previous value not chosen, new proposer doesn't see it:

- New proposer chooses its own value
- Older proposal blocked

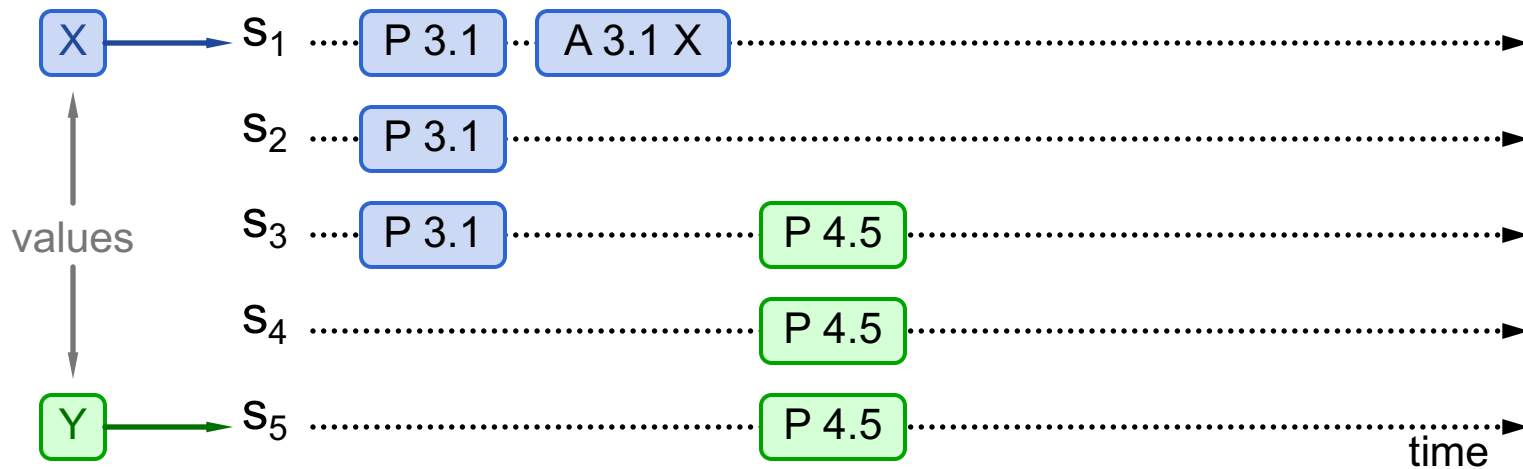


Basic Paxos Examples, cont'd

Three possibilities when later proposal prepares:

3. Previous value not chosen, new proposer doesn't see it:

- New proposer chooses its own value
- Older proposal blocked

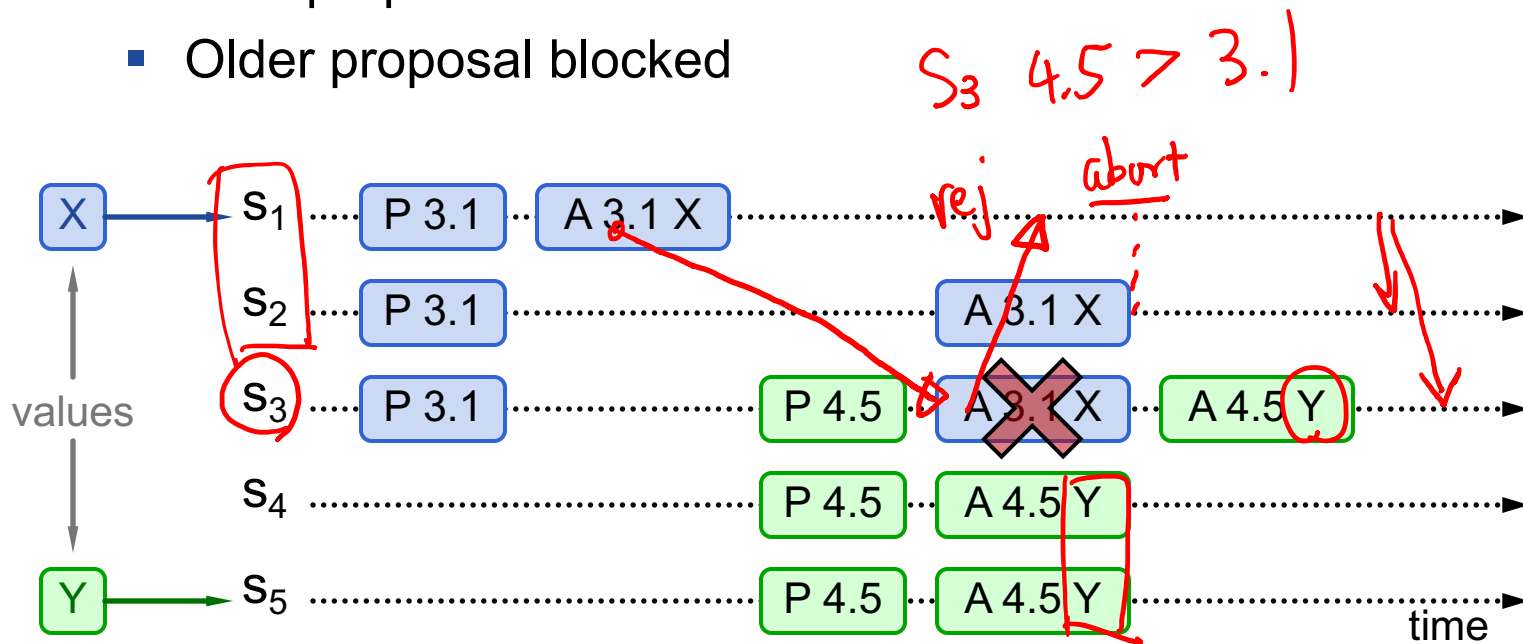


Basic Paxos Examples, cont'd

Three possibilities when later proposal prepares:

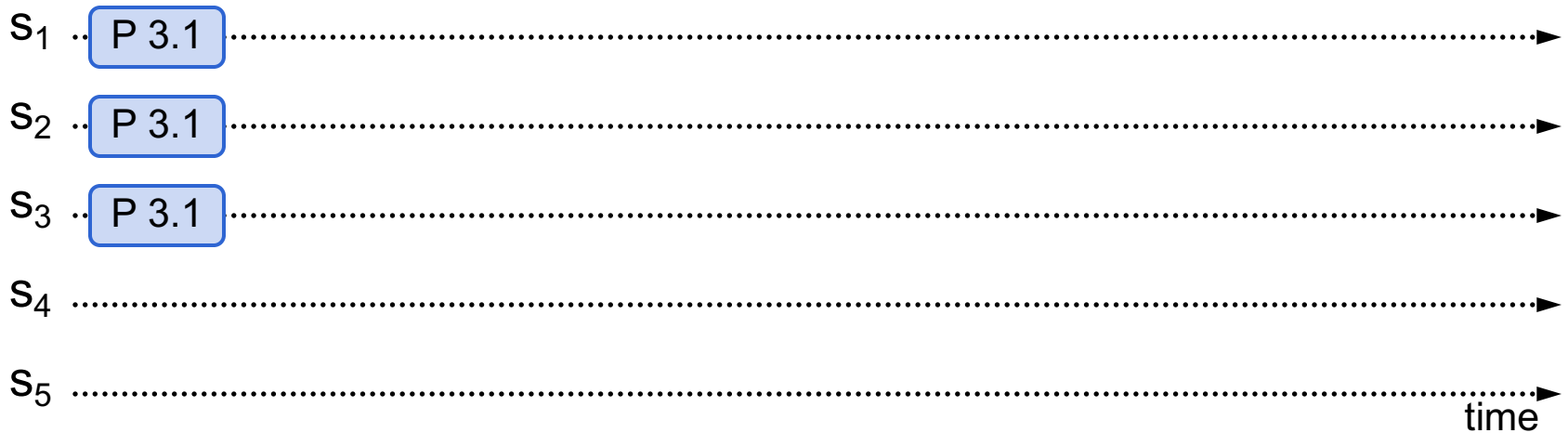
3. Previous value not chosen, new proposer doesn't see it:

- New proposer chooses its own value
- Older proposal blocked



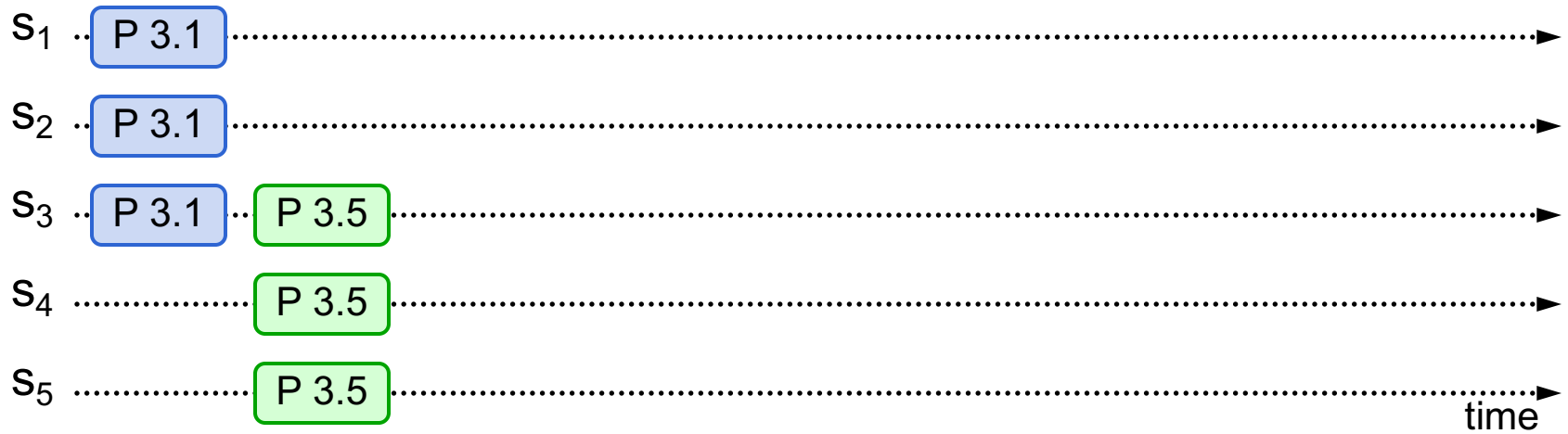
Liveness

- **Competing proposers can livelock:**



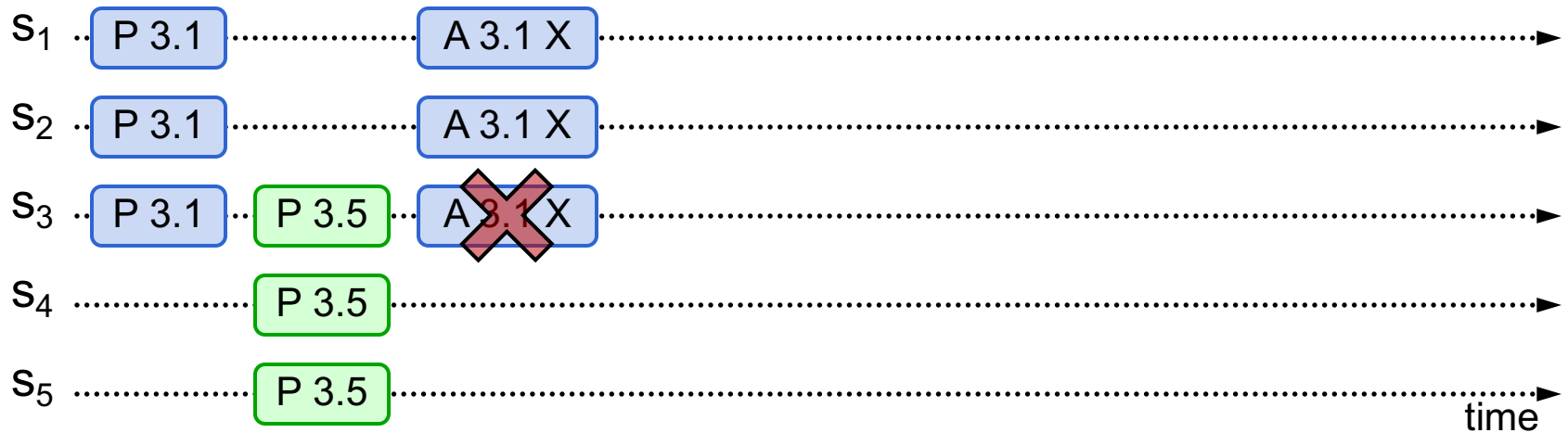
Liveness

- **Competing proposers can livelock:**



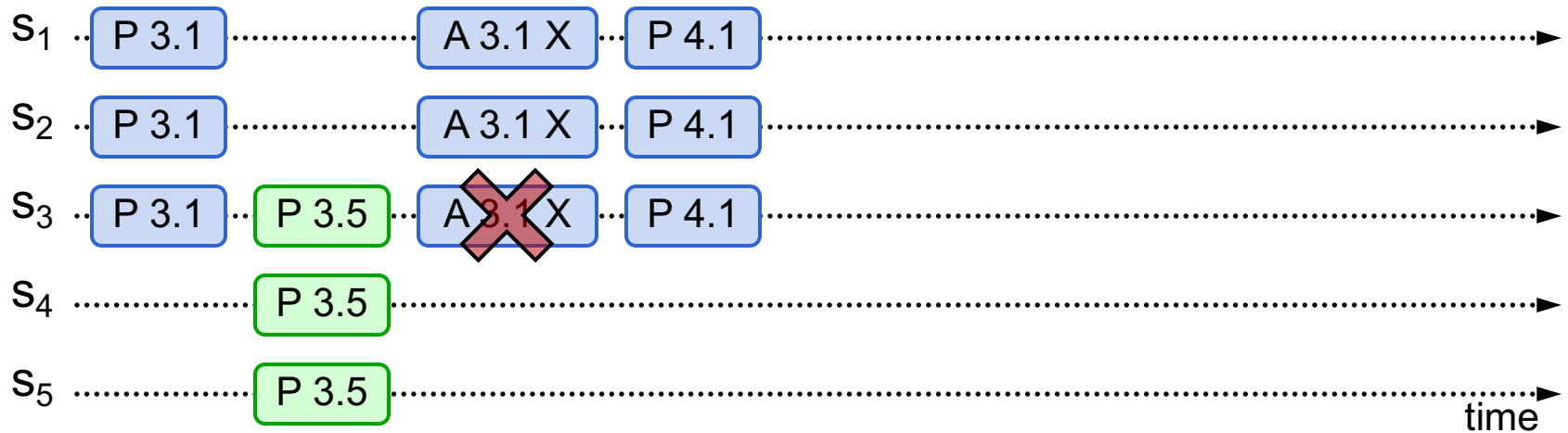
Liveness

- Competing proposers can livelock:



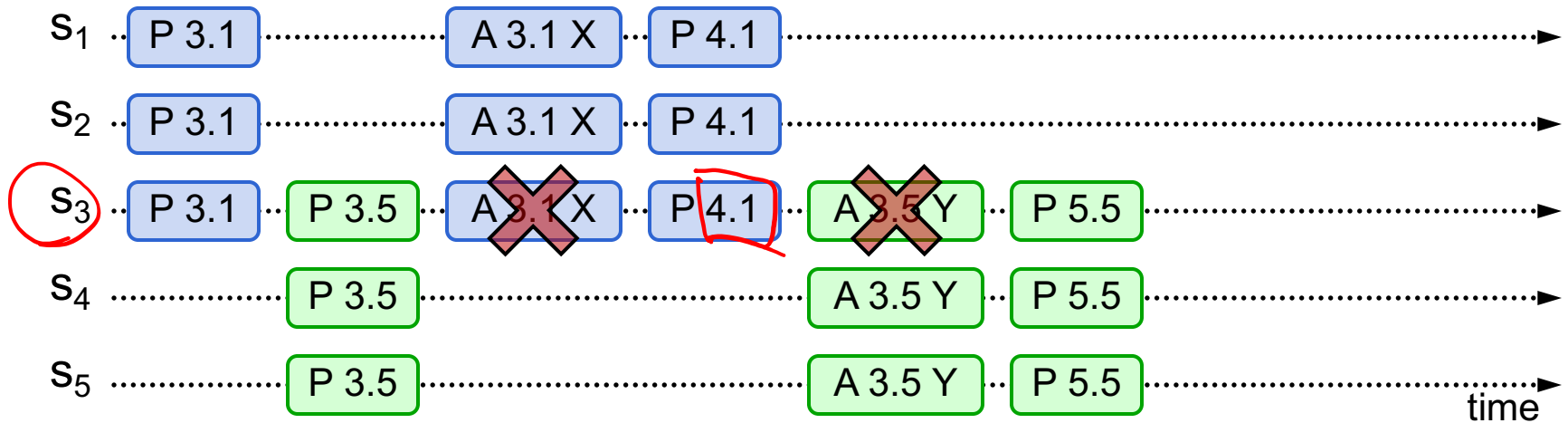
Liveness

- Competing proposers can livelock:



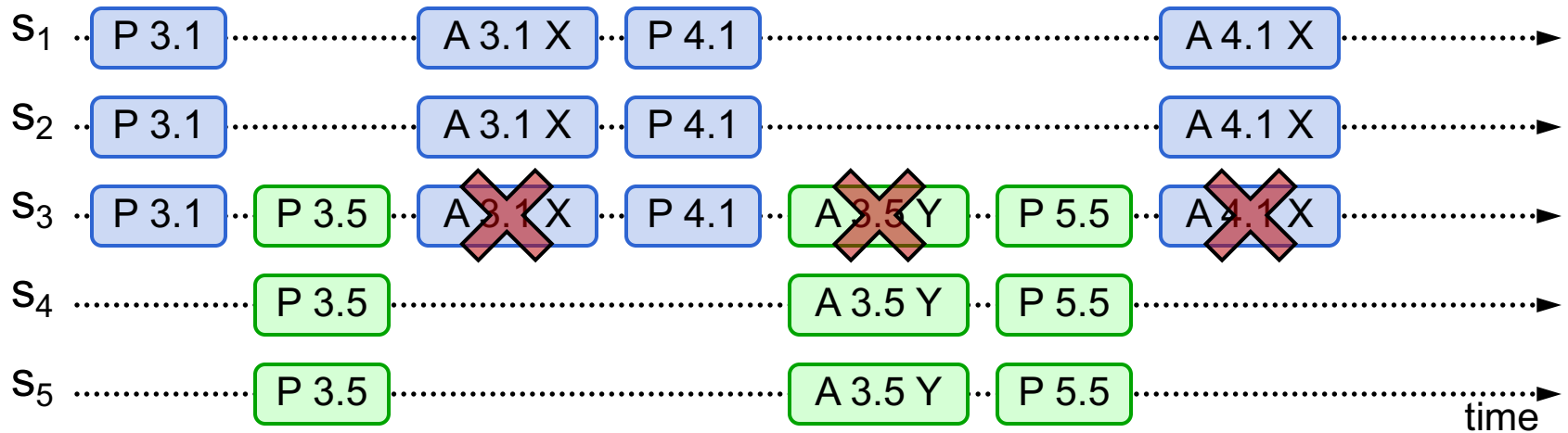
Liveness

- Competing proposers can livelock:



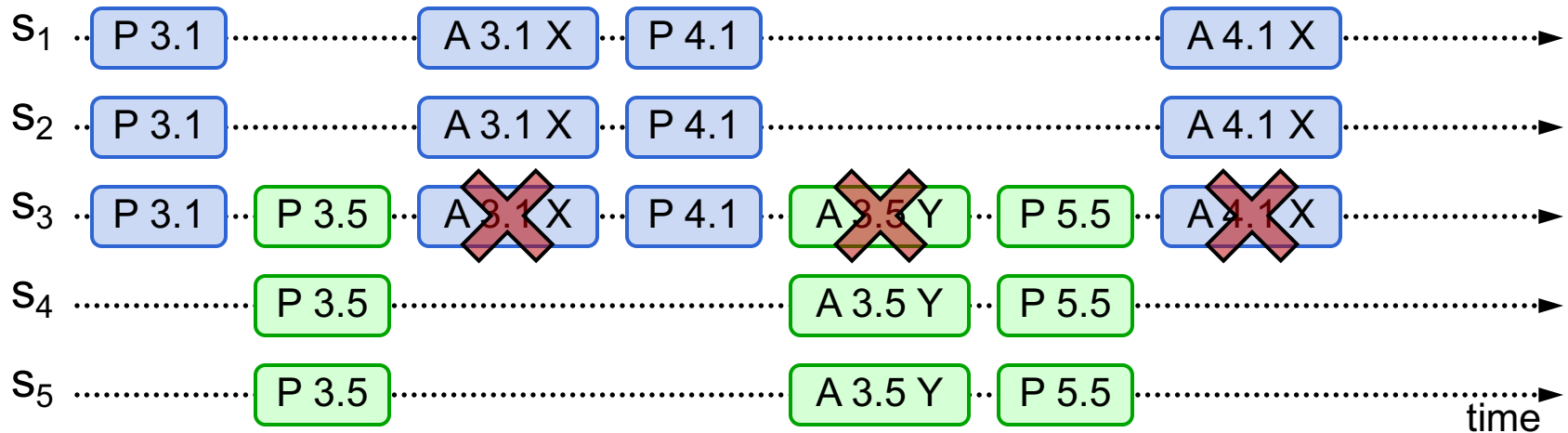
Liveness

- Competing proposers can livelock:



Liveness

- **Competing proposers can livelock:**



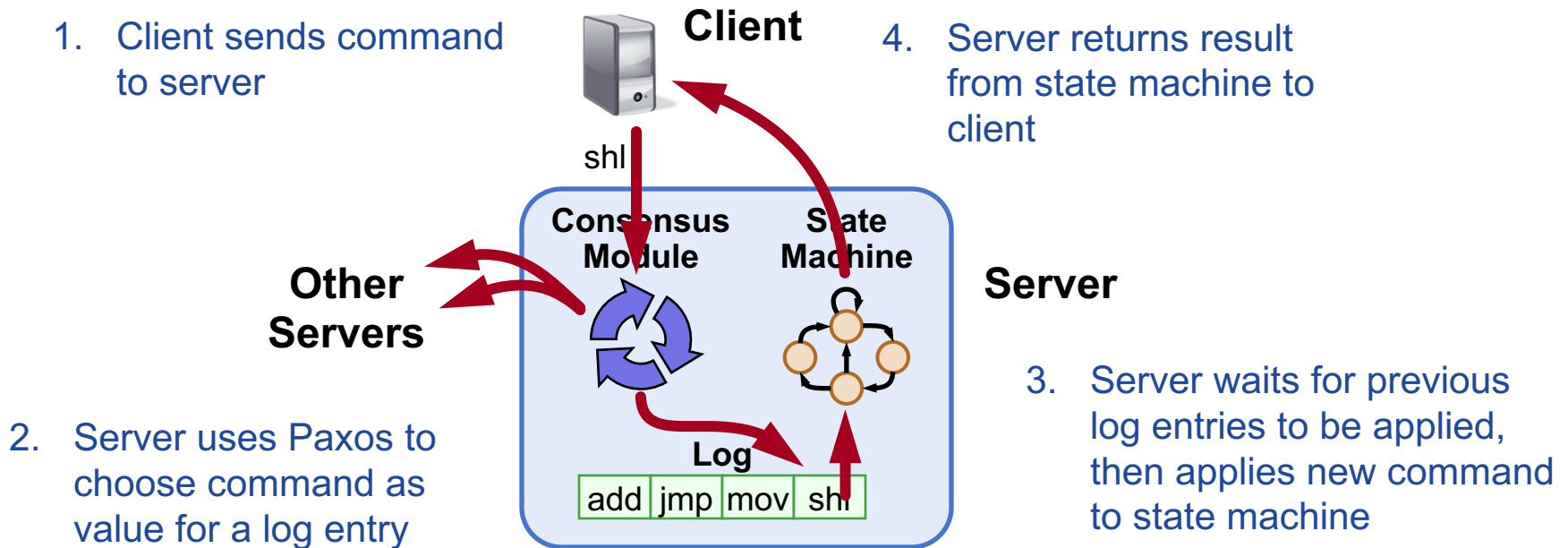
- **One solution: randomized delay before restarting**
 - Give other proposers a chance to finish choosing
- **Multi-Paxos will use leader election instead**

Other Notes

- **Only proposer knows which value has been chosen**
- **If other servers want to know, must execute Paxos with their own proposal**

Multi-Paxos

- **Separate instance of Basic Paxos for each entry in the log:**
 - Add **index** argument to Prepare and Accept (selects entry in log)



Multi-Paxos Issues

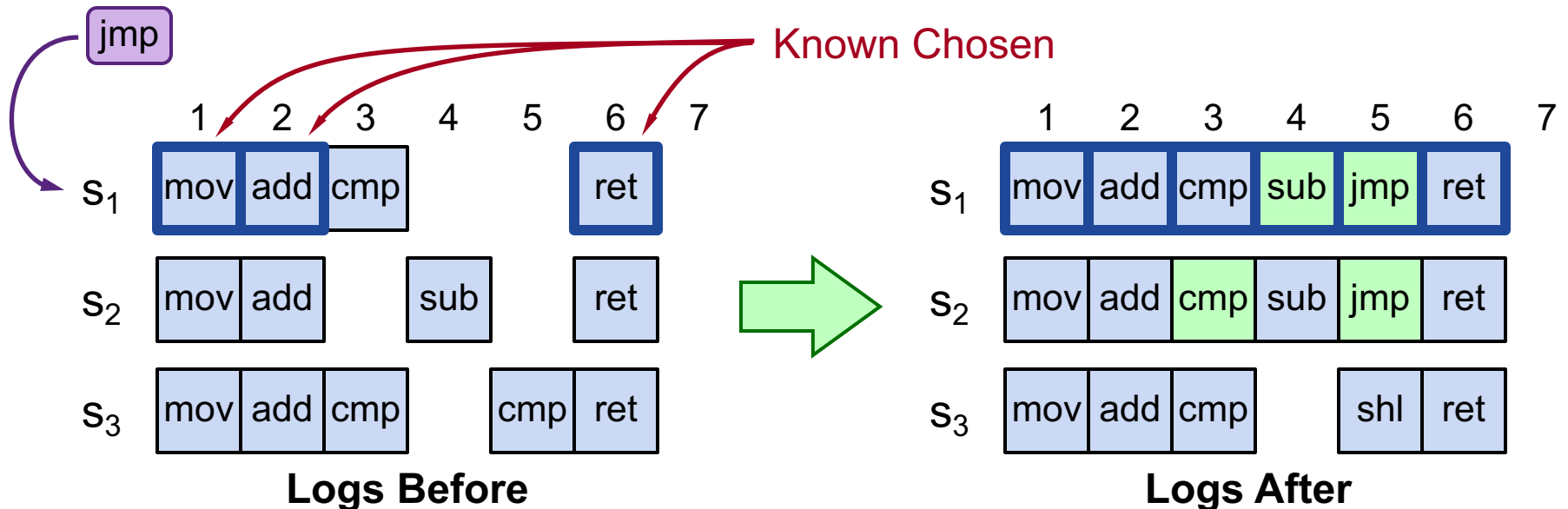
- **Which log entry to use for a given client request?**
- **Performance optimizations:**
 - Use leader to reduce proposer conflicts
 - Eliminate most Prepare requests
- **Ensuring full replication**
- **Client protocol**
- **Configuration changes**

Note: Multi-Paxos not specified precisely in literature

Selecting Log Entries

- **When request arrives from client:**

- Find first log entry not known to be chosen ←
- Run Basic Paxos to propose client's command for this index
- Prepare returns acceptedValue?
 - Yes: finish choosing acceptedValue, start again
 - No: choose client's command



Selecting Log Entries, cont'd

- **Servers can handle multiple client requests concurrently:**
 - Select different log entries for each
- **Must apply commands to state machine in log order**