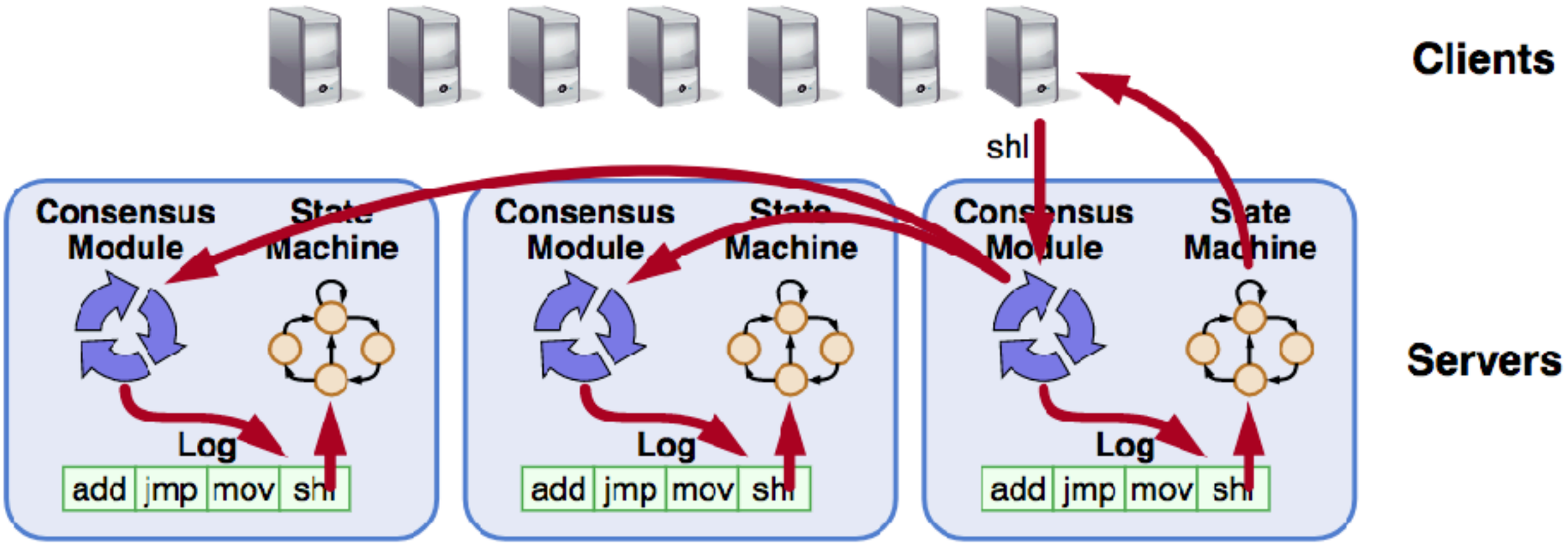
# One step further: Paxos

# Goal of Paxos in practice




- Replicated log —> replicated state machine
  - All servers execute the 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

Picture credit: Ousterhout and Ongaro, Implementing Replicated Logs with Paxos

# Requirements for basic Paxos

- **Safety**
  - Only one value that has been proposed may be chosen
  - If a value is chosen by a process, then the same value must be chosen by any other process that has chosen a value
- **Liveness (as long as majority of servers are up and communicating with reasonable timeliness**
  - Some proposed value is eventually chosen and, if a value has been chosen, then a process can eventually learn the value

> "… it is among the simplest and most obvious of distributed algorithms…" — Leslie Lamport

# The Paxos algorithm

- **Contribution:** Separately consider safety and liveness issues
  - Safety can be guaranteed (**consensus is not violated**)
  - Liveness is ensured during period of synchrony: If things go well sometime in the future (messages, failures, etc.), there is a good chance consensus will be reached (**eventually**)

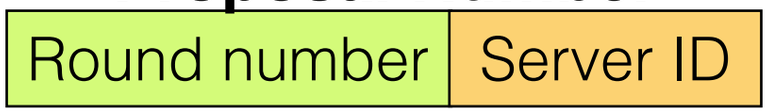
# Paxos components

- **Proposers**
  - Active: put forth particular values to be chosen
  - Handle client requests
- **Acceptors**
  - Passive: response 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
- **Assumption**
  - Each Paxos server contains both components

# 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**




  - 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
    - Reject 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 > minProposal then minProposal = n
   - Return(acceptedProposal, acceptedValue)

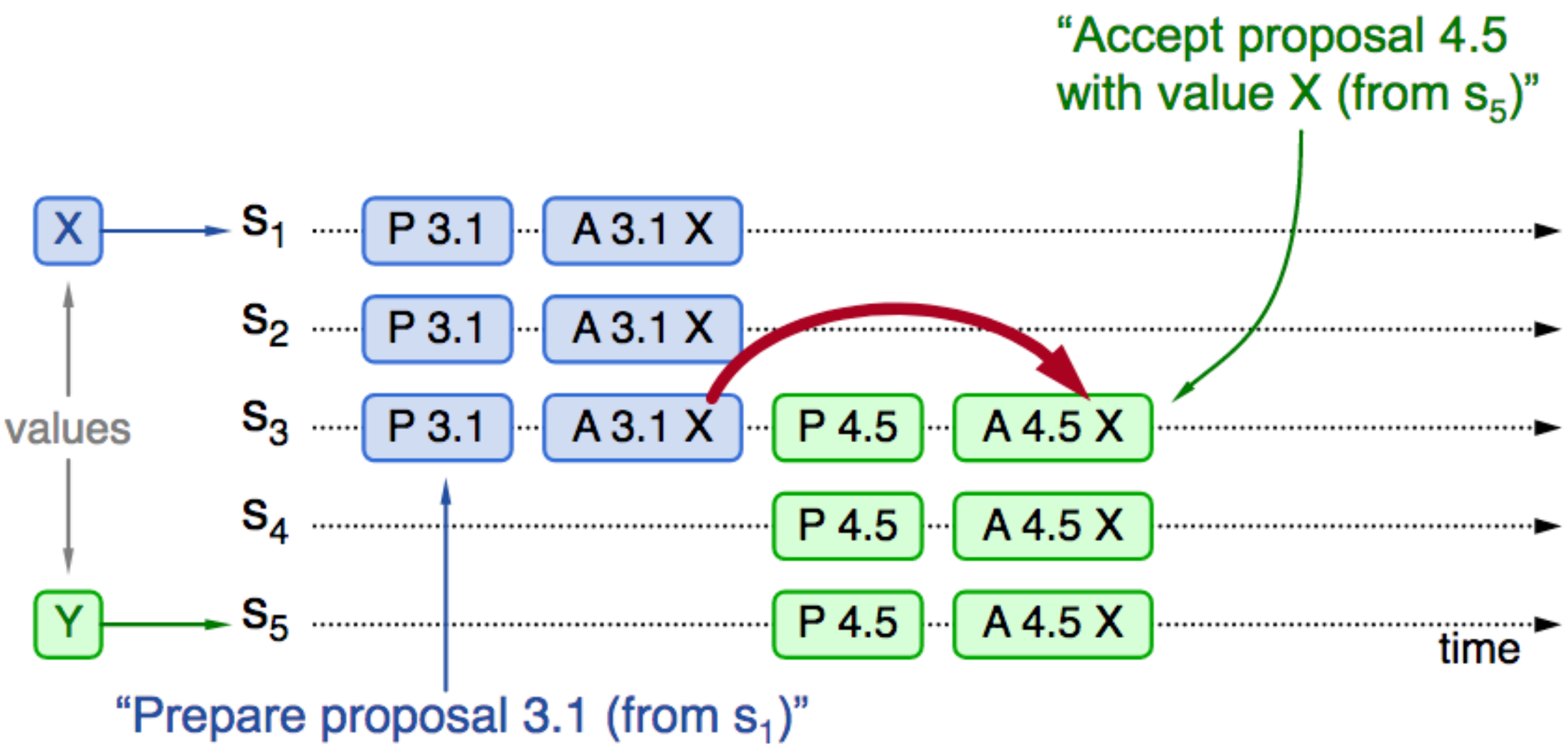
6) Respond to Accept(n, value):
   - If n ≥ minProposal then
     acceptedProposal = minProposal = n
     acceptedValue = value
   - Return(minProposal)

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

Picture credit: Ousterhout and Ongaro, Implementing Replicated Logs with Paxos

# Basic Paxos examples

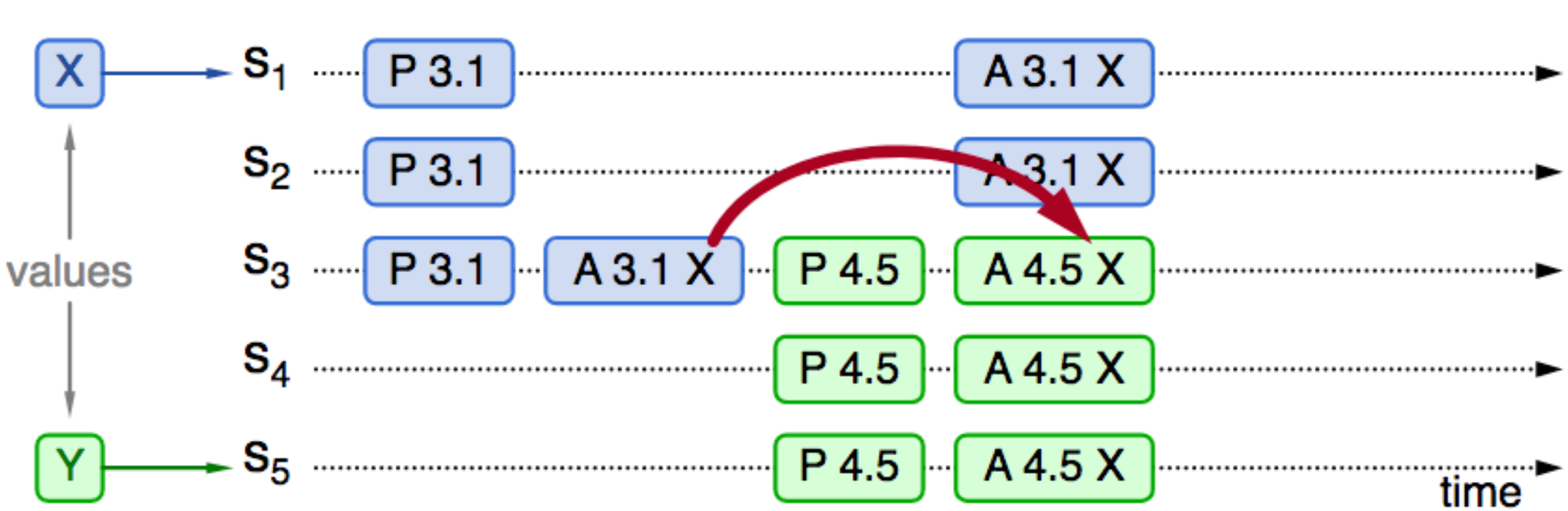
- What if previous value is already chosen
  - New proposer will find it and use it




Picture credit: Ousterhout and Ongaro, Implementing Replicated Logs with Paxos

# Basic Paxos examples

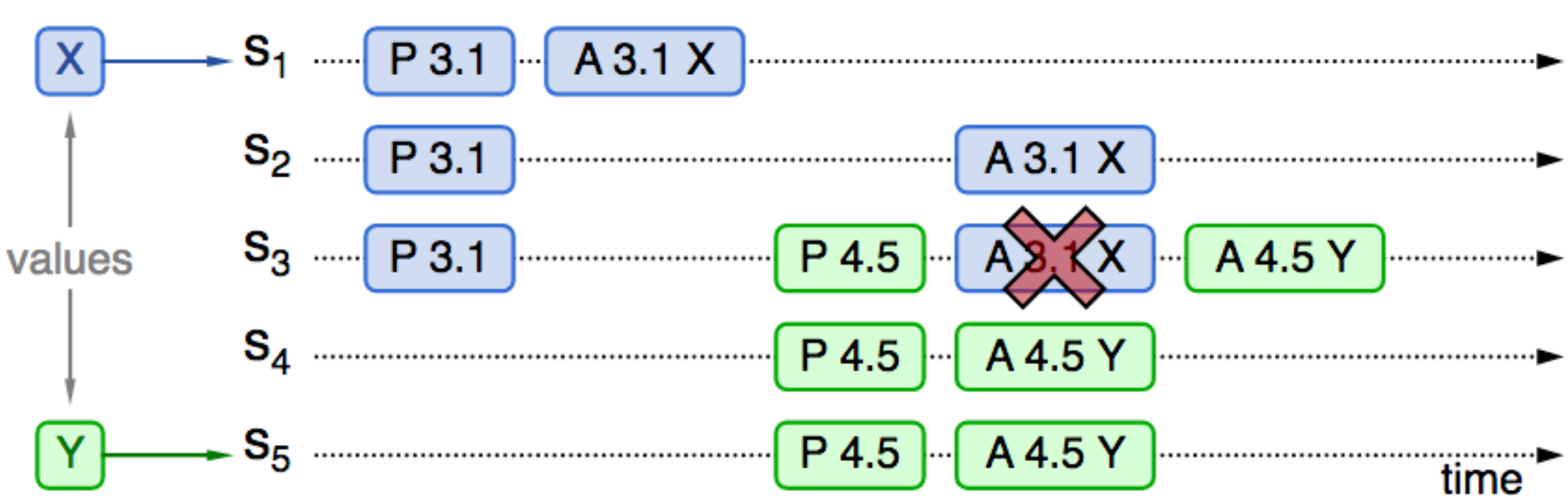
- What if previous value has not been chosen but new proposer sees it
  - New proposer will use existing value
  - Both proposers can succeed




Picture credit: Ousterhout and Ongaro, Implementing Replicated Logs with Paxos

# Basic Paxos examples

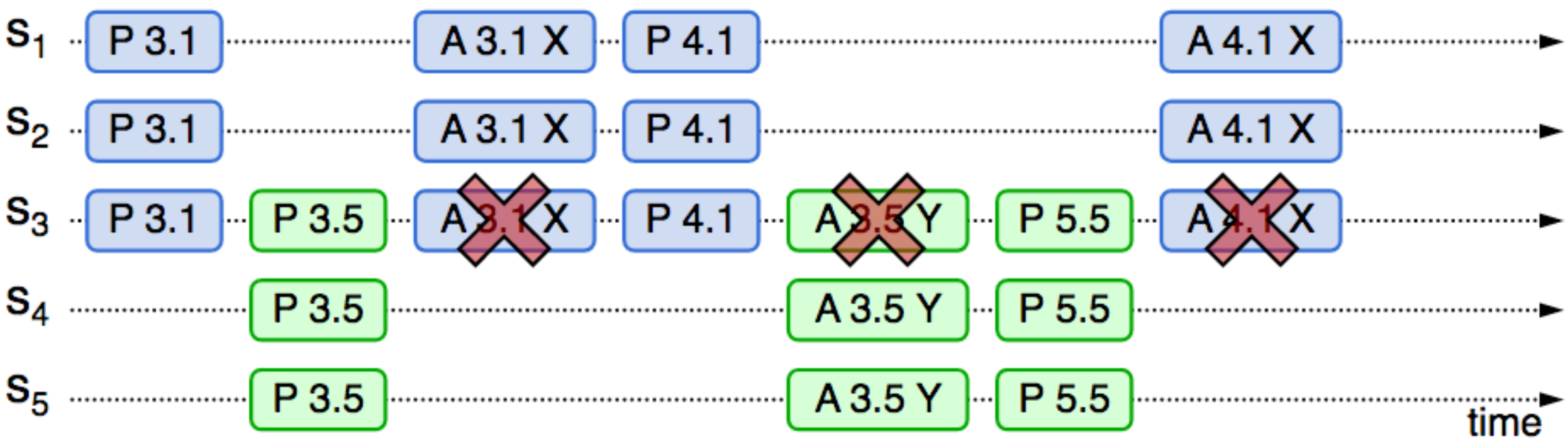
- What if previous value has not been chosen but new proposer doesn't see it
  - New proposer chooses its own value
  - Older proposal rejected




Picture credit: Ousterhout and Ongaro, Implementing Replicated Logs with Paxos

# Liveness

- Competing proposers can **livelock**




- **One solution: randomized delay before restarting**
  - Give other proposers a chance to finish choosing

Picture credit: Ousterhout and Ongaro, Implementing Replicated Logs with Paxos

# Announcements

- Next class: paper presentations and discussions
  - **Raft + Zookeeper**



Apache ZooKeeper™

- Make sure to fill out the paper evaluation form (Google form closes 10 min before class)
- Scribe report on Piazza due by end of next day (Thursday)