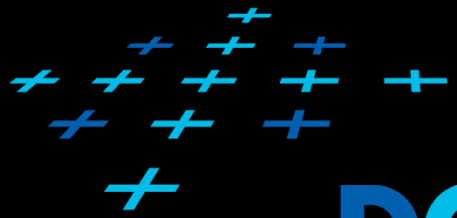


DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

WA 2

Paxos and distributed programming

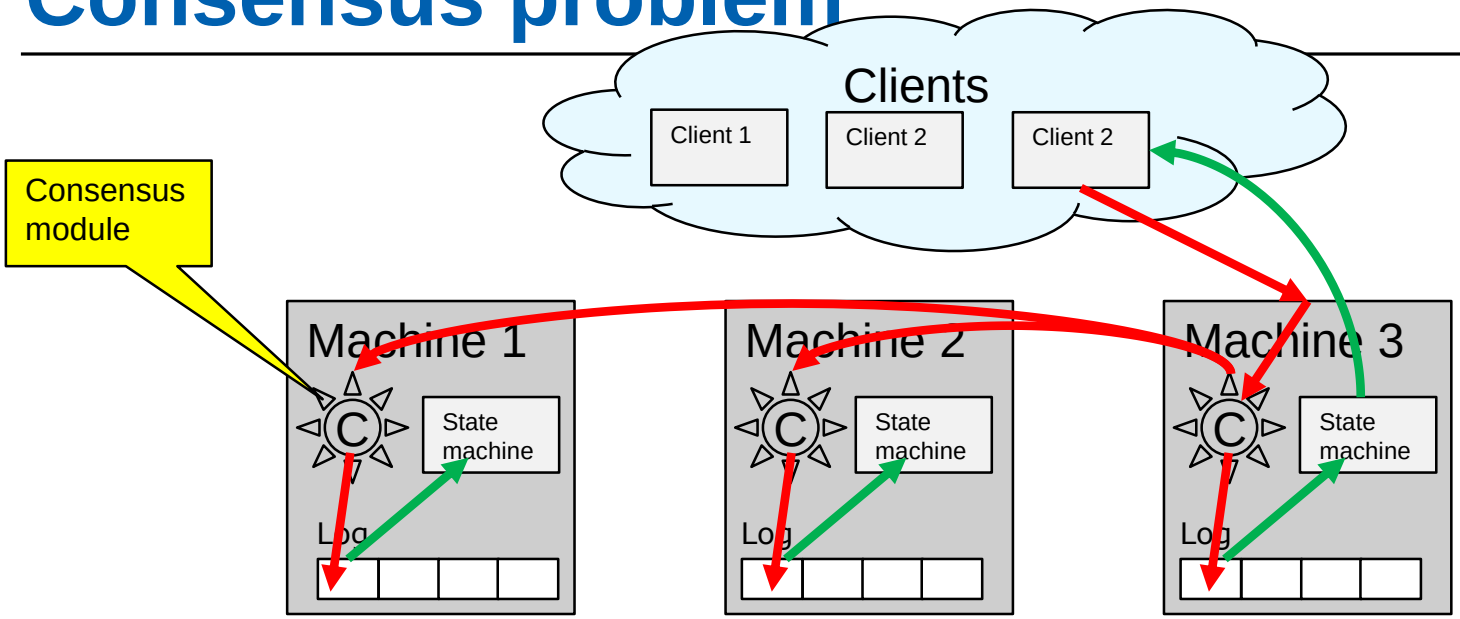
Martin Klíma

Spefics of distributed programming

- Communication by message passing
- Considerable time to pass the network
- Non-reliable network
- Time is not synchronized on all nodes
- Consistency cannot be ensured

- Consensus problem

Consensus problem



- Any of the machines reacts on external inputs
- They get synchronized
- In the case of failure, any of them can carry on functioning
- Log is propagated to all machines <<<< consensus
- A consensus of the state of the log must be found

Consensus problem

- Replicated databases
- Distributed agents
- …
- …

- We accept when a consensus if found on majority of nodes
  - e.g. not all of them have to be up and running
  - we achieve a failure tolerance

Solution

- Basic Paxos
  - One or more servers propose values
  - System must agree on just one → the chosen one
  - Only one is chosen at a time

- Multi-Paxos
  - Combine several instances of Basic Paxos to agree on series of values from the log

Requirements for Basic Paxos

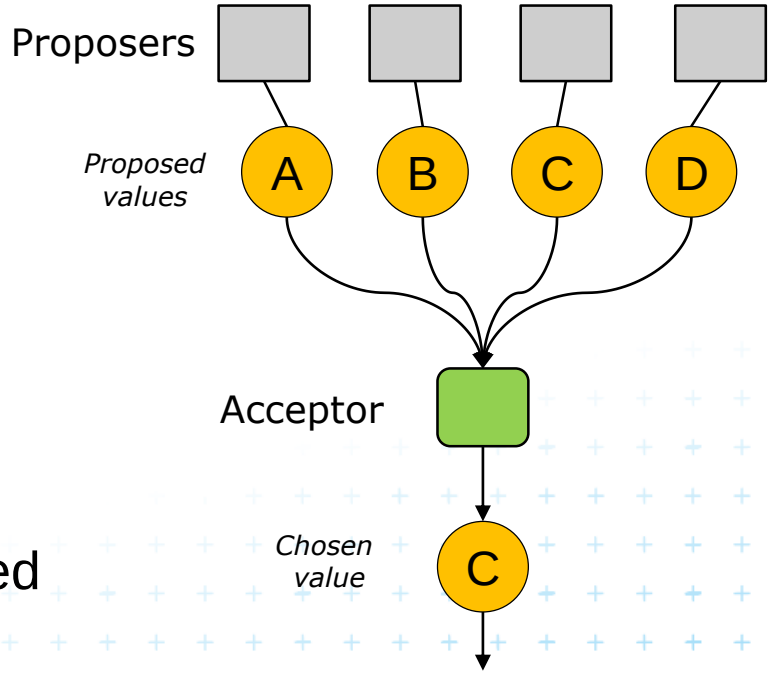
- Only one single value is chosen
- A node never learns that a value has been chosen unless it really is
- Some of the proposed values is finally chosen
- If it is chosen, the nodes will learn about it eventually

Roles of nodes in Paxos

- Proposers
  - Actively propose values into the log
  - Handle client requests
- Acceptors
  - Passively respond to proposes
  - Vote to reach a consensus
  - Store the chosen value (result of voting)
- Listeners
  - Want to know which value was chosen
  - …will be joined with Acceptors

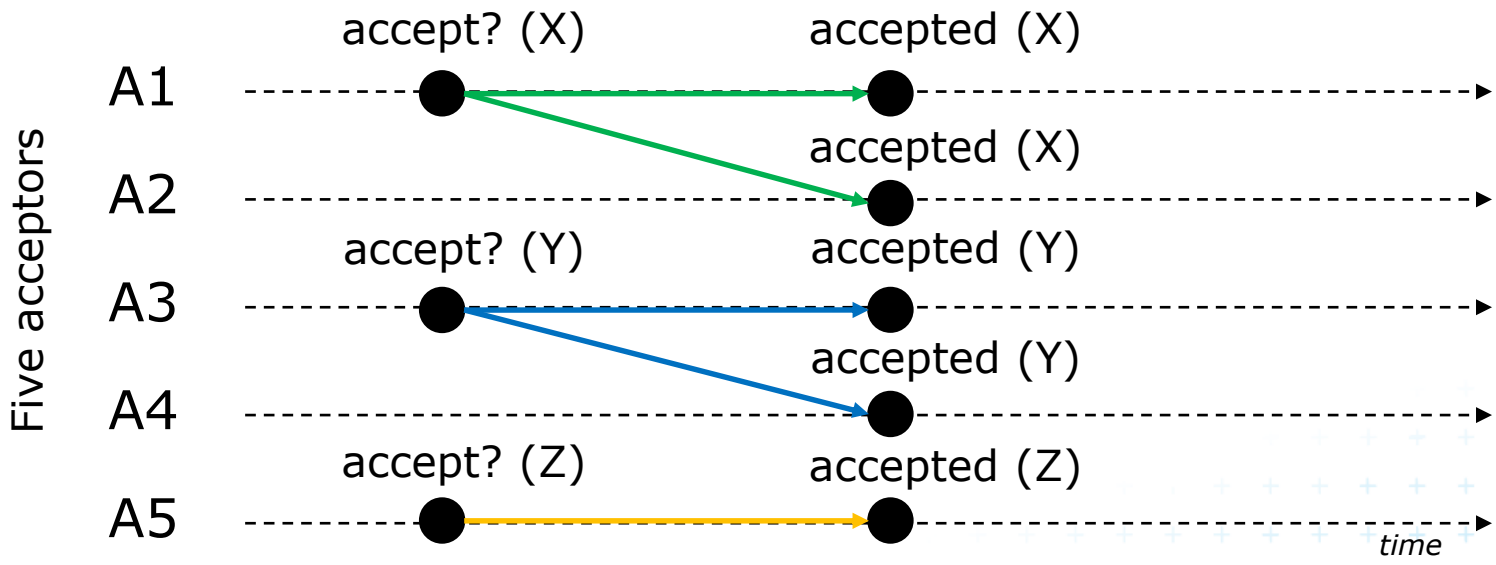
A simple approach (does not work)

- There is a single accepter in the system only
- What if the acceptor crashes after choosing?
- Solution: a quorum
  - Multiple acceptors
  - Value is chosen after being accepted by majority of acceptors
  - If one of them crashes, it still works well



How to reach a quorum?

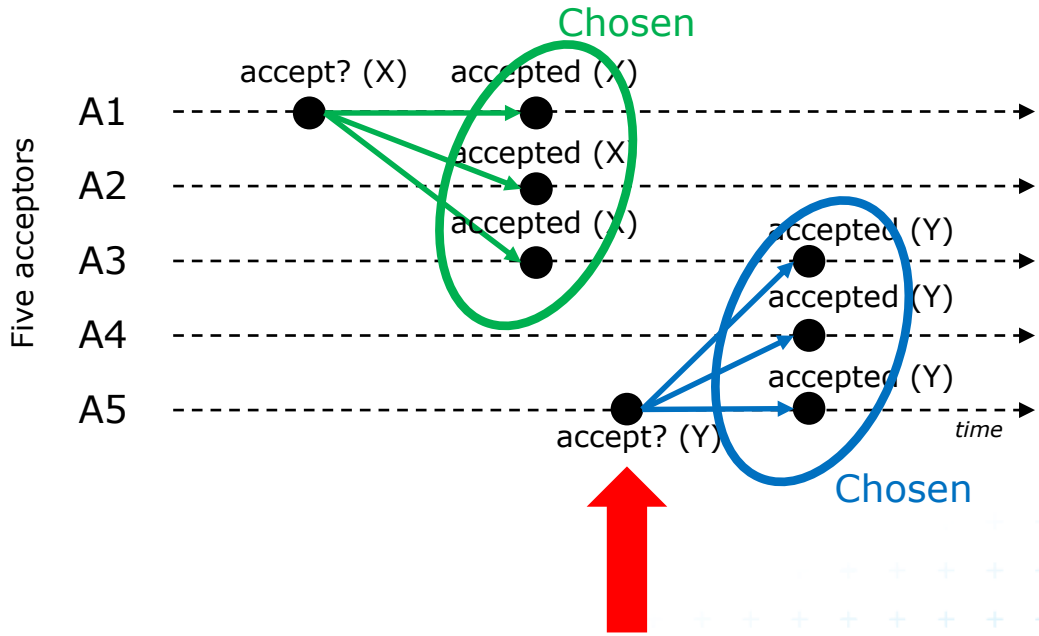
- Simultaneous acceptance may not lead to a consensus



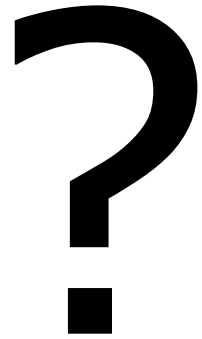
- Acceptors must sometimes change their mind
  - ➔ this means that multiple rounds of voting is necessary!!!

Another approach – 2 phase protocol

- Acceptors accept every value received
- Can choose multiple values



May not accept Y
when X was chosen already

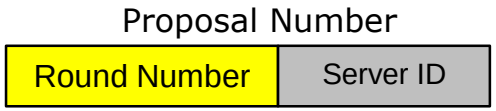


We have to chose
only one value!

Solution – proposal must be ordered

- Unique number for each proposal
  - Higher take priority over lower ones
  - A proposal must be able to choose a proposal number higher than anything used before

- Simple solution



  - Each server remembers maxRound it has seen so far
  - To issue a new proposal – increase maxRound, concatenate server ID
  - maxRound must not be forgotten → store it on disk

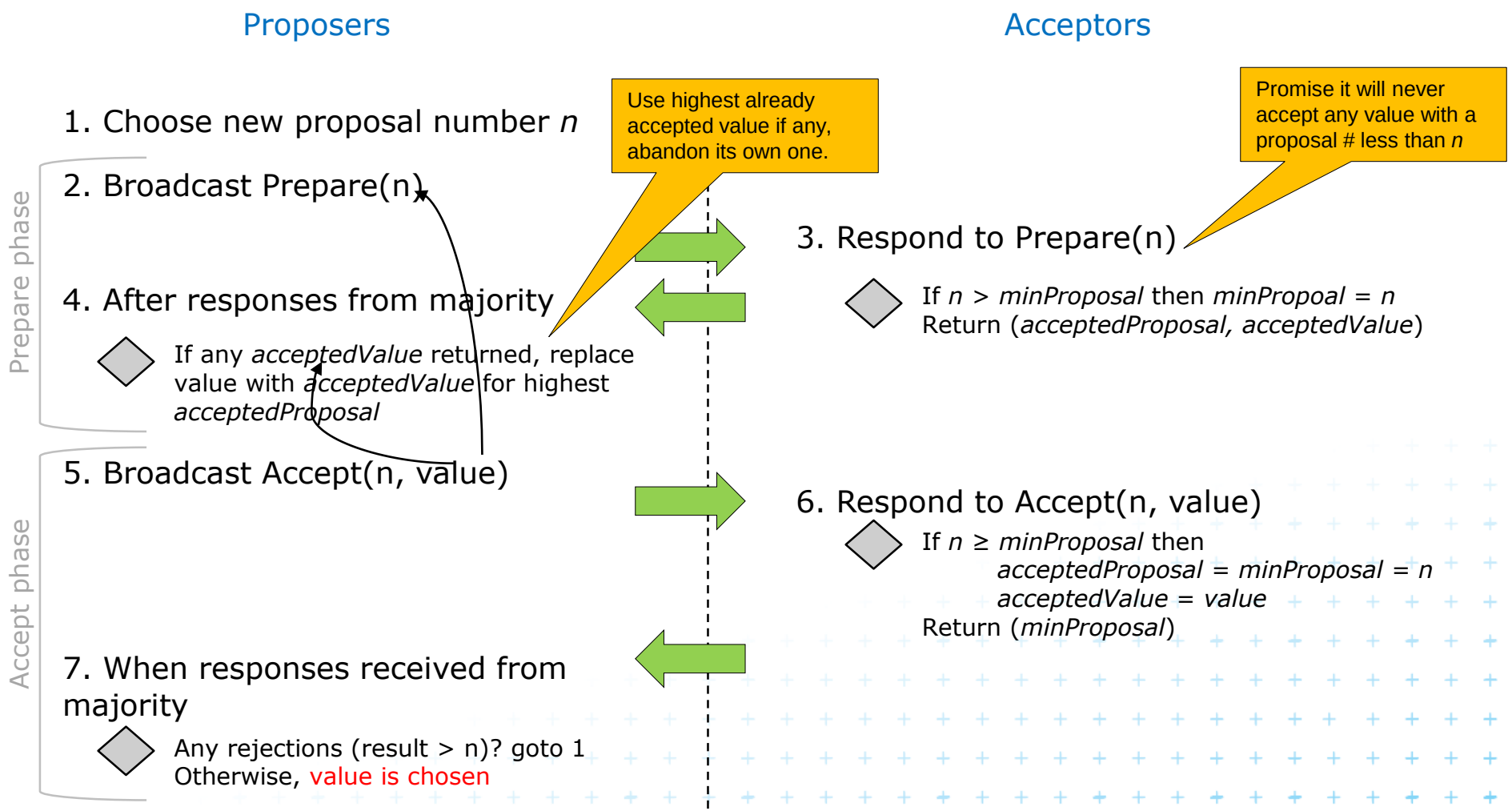
Basic Paxos

Two-phase approach:

- Phase 1: broadcast **prepare** command
- Phase 2: broadcast **accept value** command
  - value is chosen after receiving the majority of positivie accepts

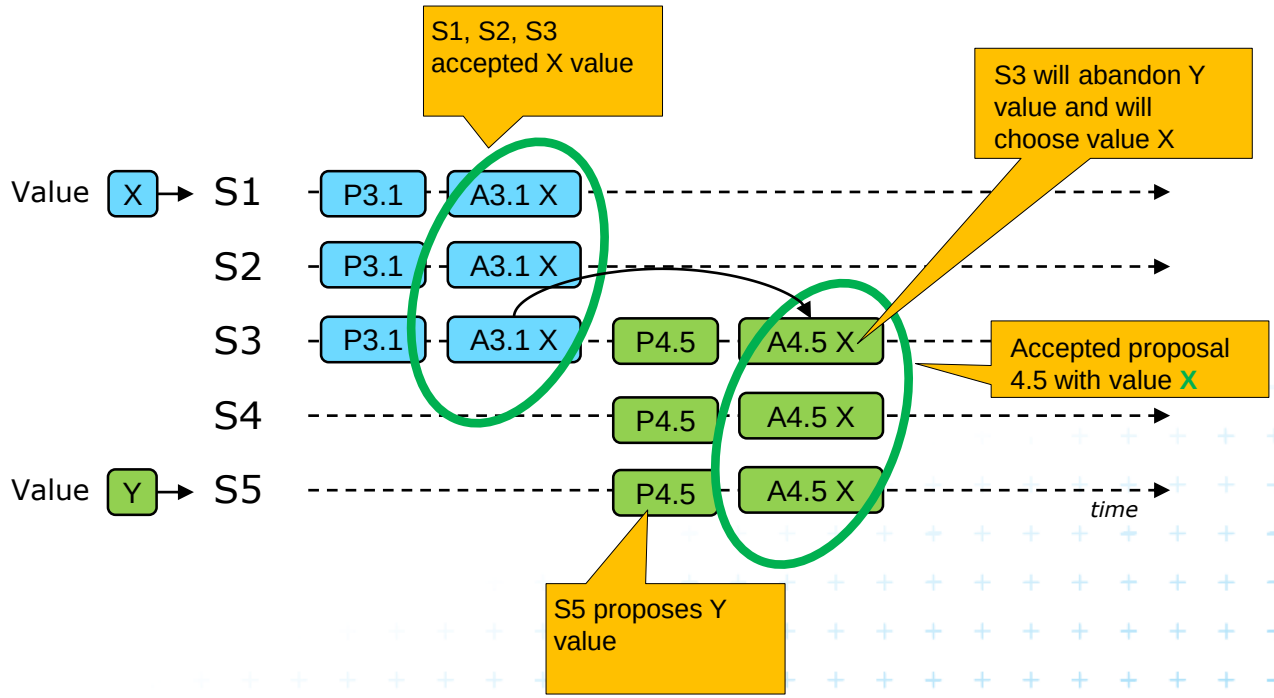
Basic Paxos

Acceptors must remember
minProposal, acceptedProposal, acceptedValue



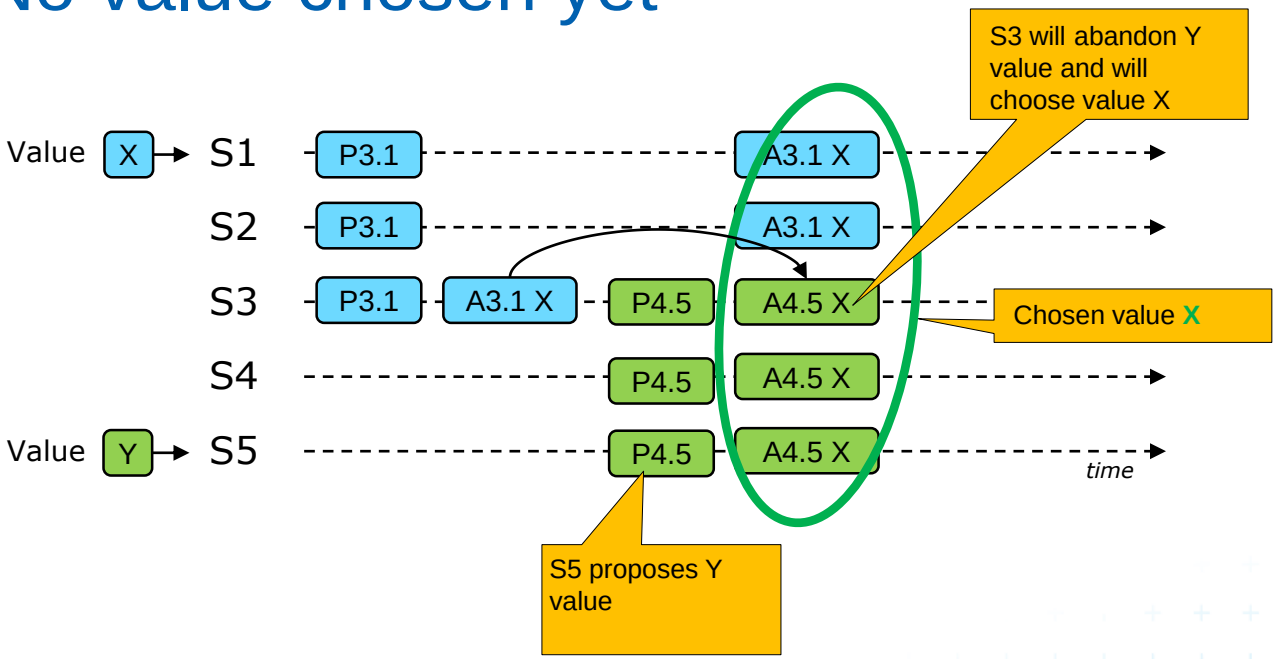
Basic Paxos Example 1

- Situation when a value has been already accepted (blue one)



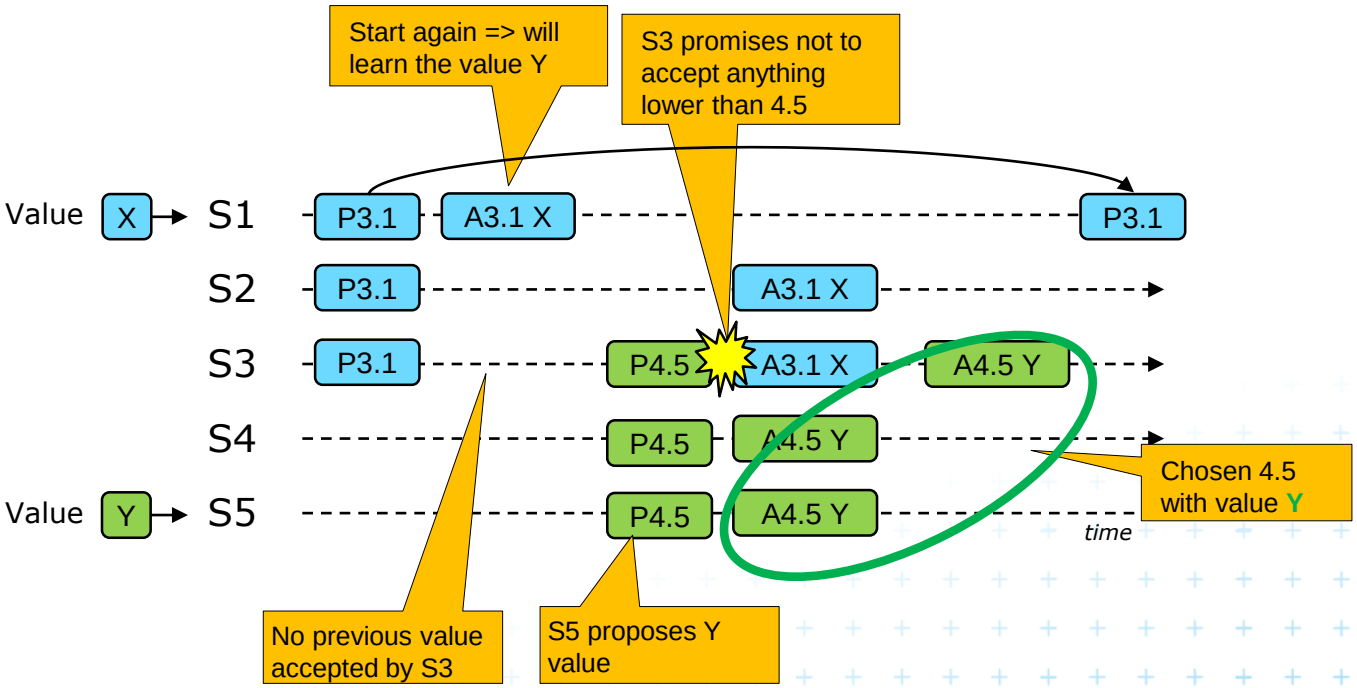
Basic Paxos Example 2

- No value chosen yet



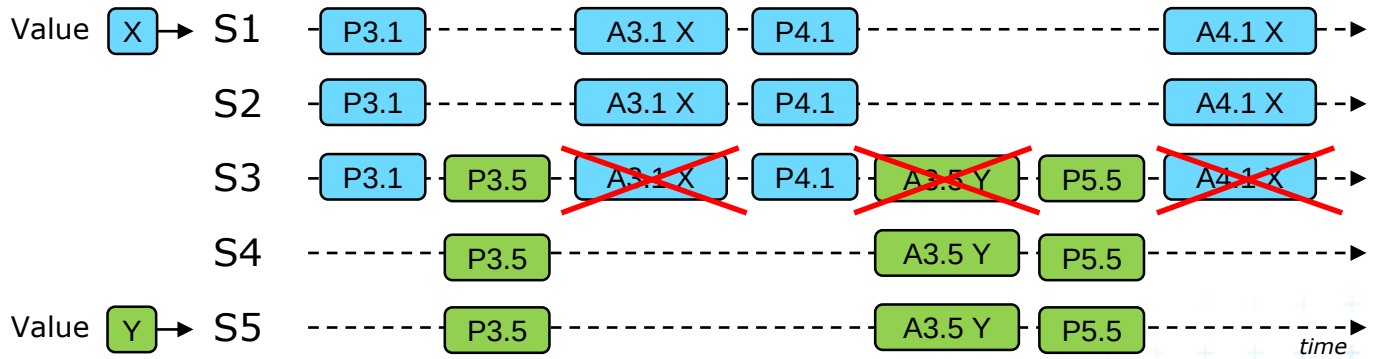
Basic Paxos Example 3

- Later proposal appears



Basic Paxos livelock possible!!!

- Solution
  - randomize restart time
  - use a **Leader role** – leader election



Shortcomings of Basic Paxos

- Communication intensive
  - Leader should reduce conflicts
  - Aim to eliminate Prepare requests
- Ensure full replication
- Changes in topology

⇒ Multi Paxos

Improving efficiency

Leader

- Only Leader will propose new values
  - Accepts requests from clients
  - Acts as proposer and acceptor
- Leader must be Elected
  - Simple algorithm = elect leader with the highest ID
  - Needs to implement a heartbeat

If not leader

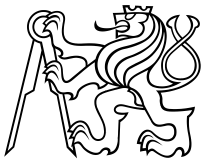
- Reject client requests (redirect to leader)
- Act only as acceptor

Improving efficiency - cont

- Reduce # of Prepares
  - Remember why we need prepares?
  - To block old proposals (promise not to accept an old proposal)
  - Find out about possibly chosen values

- Improvement:
  - make the proposal number global
  - acceptor will indicate if noMoreAccepted after the current one
  - if acceptor responds to Prepare with noMoreAccepted, skip future Prepares with that acceptor (until Accept rejected)
  - Once leader receives noMoreAccepted from majority of acceptors, no need for Prepare
  - Only 1 round of calls needed per log entry (Accepts)

Happens when someone else becomes a leader



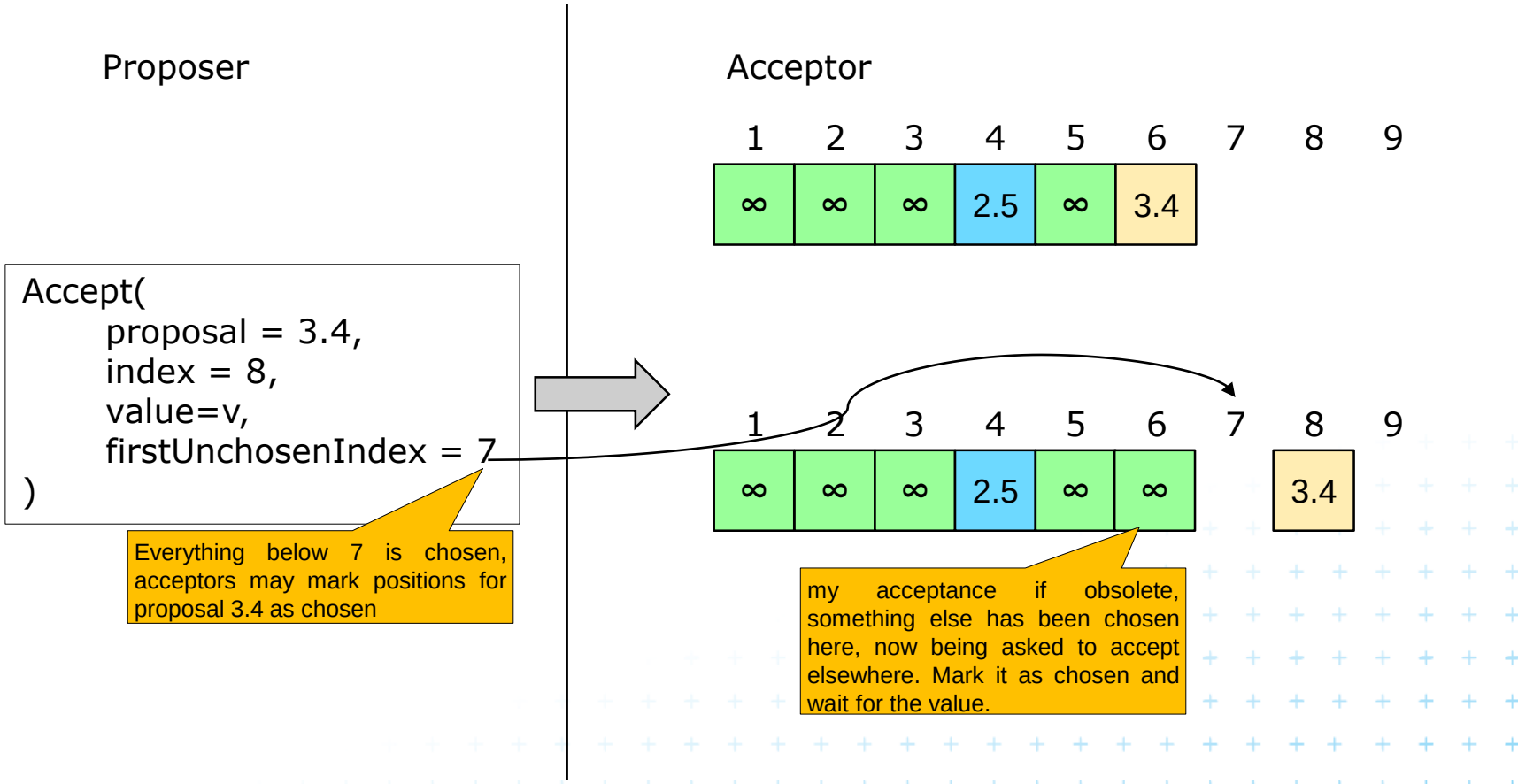
Full Disclosure

- Need to propagate log entries to really all entities, not only to the majority of them
  - full replication
  - so far only proposer knows when an entry is chosen
  - all servers should know!
- Solution
  1. keep retrying Accepts until all acceptors respond
  2. track chosen entries
     - Mark entries that are known to be chosen: acceptedProposal[i] = ∞
     - Each server maintains *firstUnchosenIndex*, first entry not to be marked as chosen yet
  3. Proposer tells acceptors about the chosen entries
     - Include the *firstUnchosenIndex* in Accept calls
     - Acceptor marks all entries *i* as chosen if:

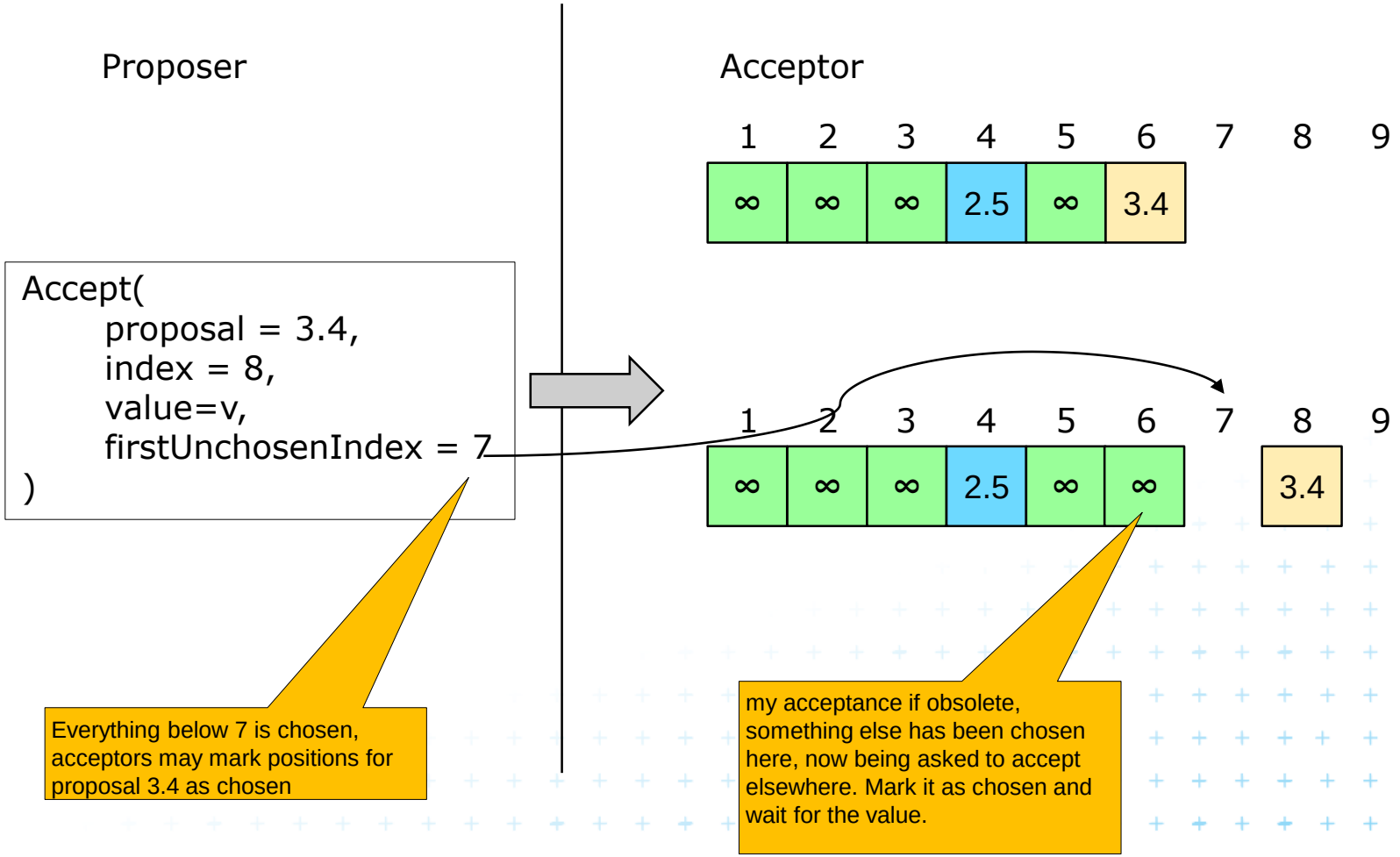
       i < request.firstUnchosenIndex

       acceptedProposal[i] == request.proposal

Result of the improvement so far



Improvement cont.



Full disclosure final

- The Acceptor may still not have the full log replicated.
- Solution: it returns to the acceptor his *firstUnchosenIndex*
- If the Proposer sees, that proposer.firstUnchosenIndex < acceptor.firstUnchosenIndex then
  - proposer sends Success(index, v) message for all indexes missing
  - acceptor remembers
    - acceptedValue[index] = v
    - acceptedProposal[index] = ∞
    - returns firstUnchosenIndex
    - proposer sends additional Success message if needed