DISTRIBUTED SYSTEMS

CSCI 4510/6510

Paxos

Final Requirements

- (P1) An acceptor must accept the first value it receives.
- (P2c) For any v and m, if proposal number m is issued with value v, then there is a majority set S such that either
  - (a) No accepter in S has accepted any value OR
  - (b) v is the value of the highest numbered proposal with number less than m that was accepted by any acceptor in S.
- How to translate these to an algorithm?

Algorithm for Proposer

- To propose a value:
  - Picks a new proposal number m
  - Sends “prepare(m)” message to set of acceptors, requesting:
    - promise not to accept any proposals with lower number than m
    - response with highest-numbered proposal (less than n) that is has accepted

- If proposer receives response from majority of acceptors:
  - Create proposal (m,v)
    - If no responding acceptor has accepted a value, v its own value
    - Otherwise, v is value from with highest-numbered proposal accepted by a responding acceptor
  - Send “accept(m,v)” message to set of acceptors

Algorithm for Acceptor

- Acceptor can receive: “prepare(m)” and “accept(m,v)”

- On receiving “prepare(m)”:
  - Respond only if m is greater than proposal number of any it has already responded to
  - Send (m,v) for highest-numbered proposal it has accepted, if any
  - This response is promise not to accept any proposal numbered less than m.

- On receiving “accept(m,v)”:
  - If not responded to proposal with number greater than m
    - Accept proposal (m,v)
    - Send response to proposer

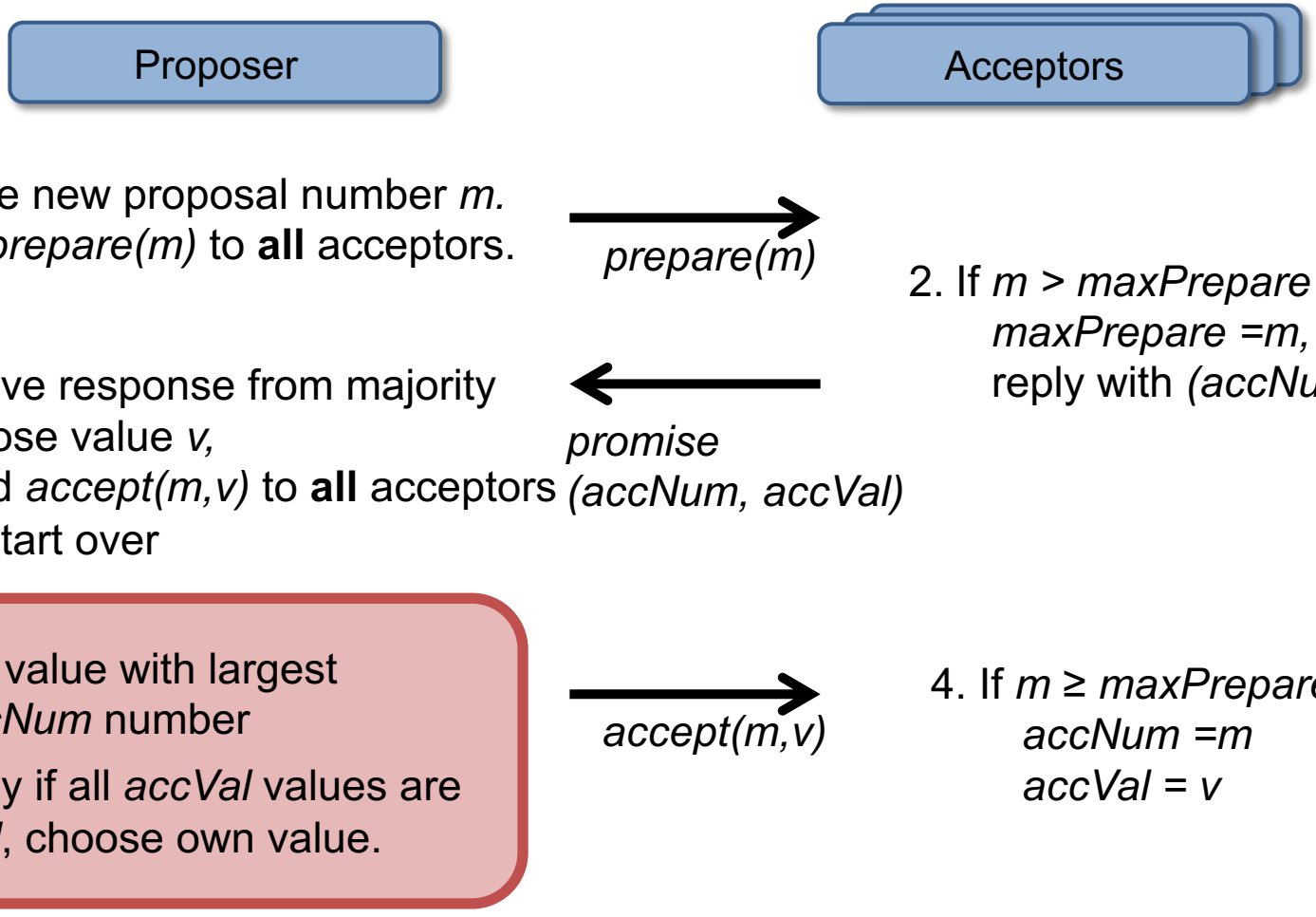
Synod Algorithm Implementation

- Proposers are stateless
  - Safety is not violated if a proposer crashes
  - If proposer recovers, it can start over
  - To ensure liveness, we need a proposer to stay alive “long enough”

- Each acceptor stores
  - maxPrepare: largest proposal number for which it has responded to a prepare message (initially 0)
  - accNum: largest proposal number of proposal it has accepted (initially null)
  - accVal: value of proposal numbered accNum (initially null)

  - The acceptor state must survive crash/recovery

Execution of Synod Algorithm



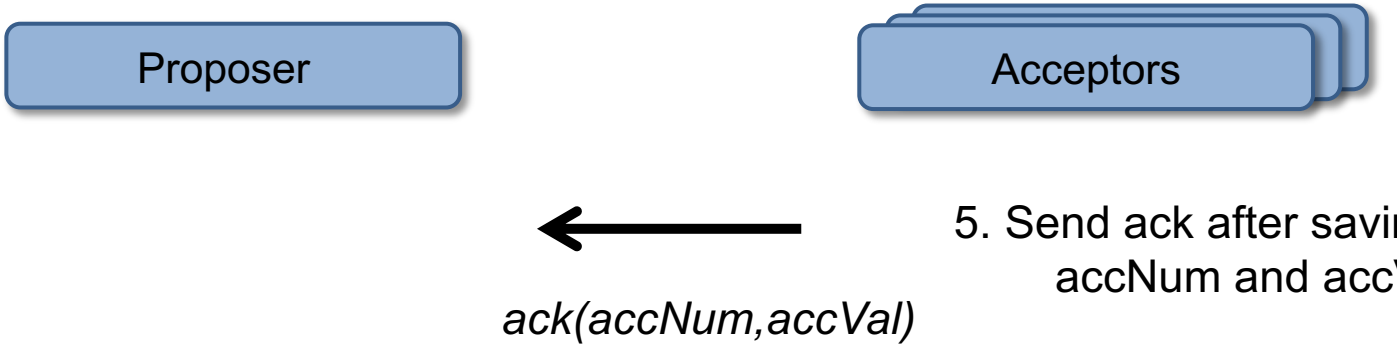
Learning the Accepted Value

- A value is chosen when a majority of acceptors accept it.
- How can the acceptors (learners) determine when this has happened?

- Only once a value has been chosen can it be written in the log.

- With message loss, there is no guarantee that all acceptors will eventually learn the decided value.

Execution of Paxos Algorithm (2)



Correctness of Synod Algorithm

- Safety is guaranteed by algorithm construction.
- What about liveness?
  - Do a majority of acceptors eventually accept a value?
  - Even if acceptors do not fail and messages are not lost?
- Multiple proposers can keep issuing prepare requests with higher numbers.
  - Possible no proposal gets majority of “promises”.

Ensuring Progress

- Can ensure liveness by having a **distinguished proposer**.
  - All proposer funnel proposals through this distinguished proposer.
  - Distinguished proposer issues proposals one at a time.
- Use leader election algorithm to elect this distinguished process.

Full Paxos Algorithm

- The Synod algorithm is used to determined consensus value for a single log entry.

- The Paxos algorithm is a sequence of Synod algorithms.
  - With some optimizations.

Full Paxos Algorithm

- Use leader election algorithm of your choice to elect a leader.
  - Leader acts as **distinguished proposer**.
- When leader elected, it may have missing log entries.
  - It runs Synod algorithm (acts as proposer and acceptor) to learn consensus value for those entries.
- Once holes in log are filled, leader can start processing new requests to create log entries.

Paxos Optimization

- If leader is stable, only need to do “prepare-promise” phase when first elected.
  - Leader does prepare-promise phase until leader’s first value is accepted.
  - * Leader has implicit promise from all acceptors.
- After first value is accepted, leader just uses “accept-ack-commit” phase for subsequent values.