DISTRIBUTED SYSTEMS

CSCI 4510/6510

Paxos Implementation Details

The Problem

- We need to store a log of events (decrees)

Event 1	Event 2	Event 3	Event 4	Event 5	….

- Want to replicate this log at multiple sites.
  - Need the event to appear in the same order in every log.
  - Need every event to eventually appear in every log.

System Model

- Processes may fail by crashing and may restart.
  - Both crash failures and crash/recovery processes.
  - Processes do not lose state when they crash and recover.

- Messaging is asynchronous.
  - Messages may take arbitrarily long to be delivered.
  - Messages can be duplicated.
  - Messages can be lost.
  - Messages cannot be corrupted.

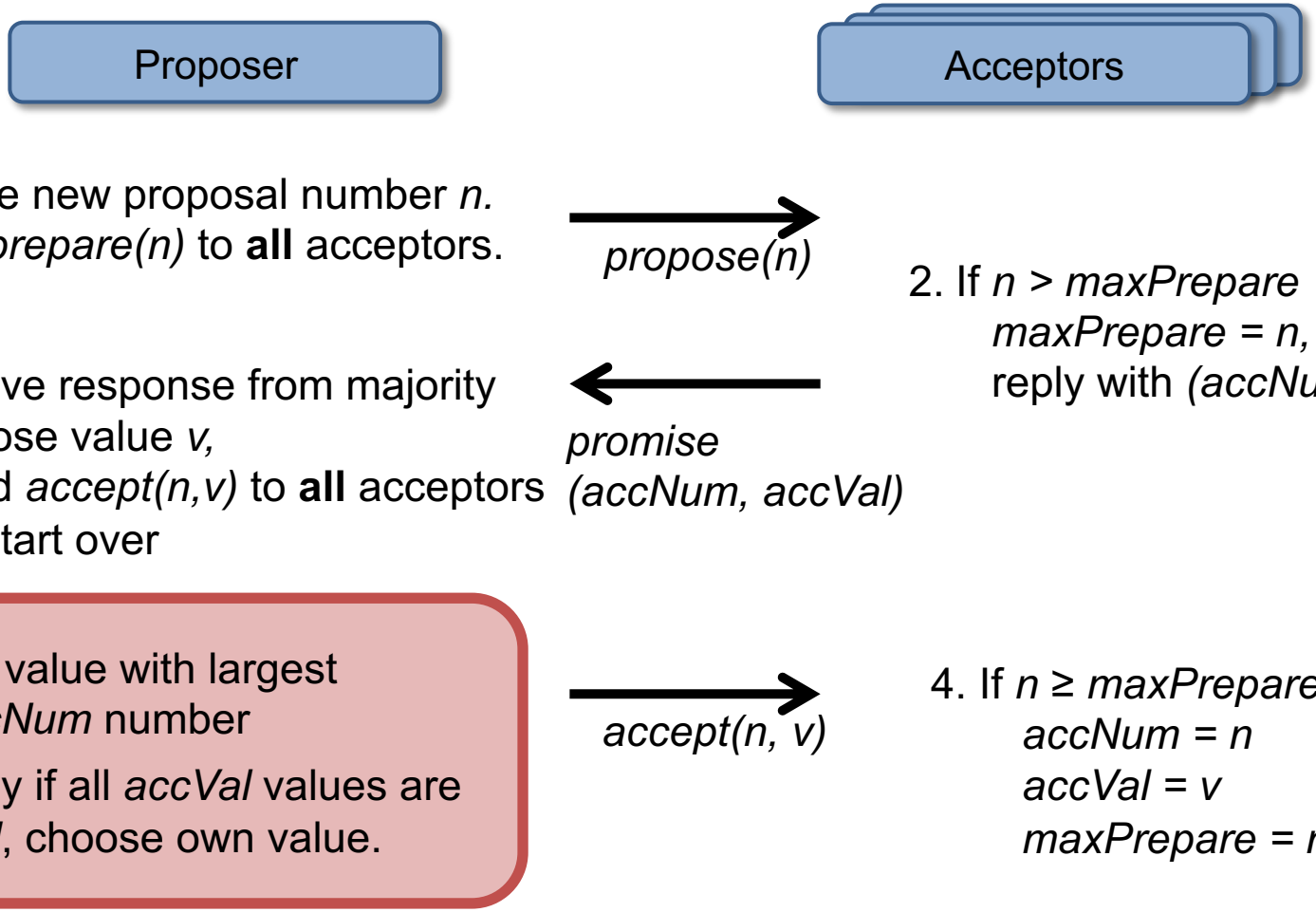
The Paxos Algorithm

- Initial approach: consider the replicated log problem as a sequence of consensus problems, one per log position.

Event 1	Event 2	Event 3	Event 4	Event 5	….

- The algorithm used to reach consensus on a single log entry is called the **Synod Algorithm**.
- The algorithm to replicate a sequence of log entries is called **Paxos**.
- Some people call the Synod Algorithm “Single Paxos” and call the Paxos Algorithm “Multi Paxos”
  - I don’t like these people.

Execution of Synod Algorithm



Execution of Paxos Algorithm (2)

ack(accNum,accVal)

4 (continued).
Send ack after saving accNum and accVal

5. If receive ack from majority
send commit(v)

commit(v)

6. Record v in log.

Log is stored in stable storage.

With message loss, there is no guarantee that all learners will eventually learn the chosen value.

Synod Implementation Details (I)

- When proposer sends “prepare(n)”, it may not receive “accept” from a majority of acceptors.
  - Possible message loss
  - Or proposal number too small
- Solution:
  - Wait a for responses for some time interval
  - Wait random “backoff” interval
  - Start propose step again, with new, larger proposal number
- Similar problem can happen when proposer is waiting for “ack” messages.
  - Same solution – wait random amount of time and start propose step again with new, larger proposal number.

Synod Implementation Details (II)

- Proposal numbers must be unique.

- One way to do this:
  - Site keeps a counter i, initially 0.
  - For each new proposal, use proposal number n= i + “site_id”
  - E.g., for site 1, proposal numbers are 1, 11, 21, 31, …

Synod Implementation Details (III)

- What if proposer's proposal number is less than acceptor's maxPrepare?
  - Acceptor can ignore propose and accept messages
  - Proposer will have to wait for "timeout"
  - Faster if acceptor sends a "nack" messages to tell proposer
    - Your proposal number is too small
    - This is my accNum
  - Proposer can start new proposal with larger proposal number (after random waiting period)

Synod Implementation Details (IV)

- If proposer crashes in the middle of Synod algorithm
  - Proposal is lost – not stored in stable storage
  - Should store last-used proposal number in stable storage, when proposer recovers, it can start with larger proposal number (for a different proposal).

- For each acceptor, initially:
  - accNum = 0
  - accVal = null

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.
  - Each log entry has it's own Synod algorithm.
  - Can also include some optimizations.
- Can ensure liveness by having a leader site act as the **distinguished proposer**
  - All proposers funnel proposed values through this distinguished proposer
  - Distinguished proposer issues proposals, one for each value at a time
  - Each proposal issued for a new log entry

Paxos Algorithm Details

- Use leader election algorithm of your choice to elect a leader (to act as distinguished proposer).

  .

- When leader elected, it may have missing log entries.
  - Leader runs Synod algorithm (acts as proposer and acceptor) to learn consensus value for those entries.
  - Uses a dummy value for these entries.

- Once holes in log are filled, leader can start processing new requests to create additional log entries.

Paxos Optimization (I)

- Leader must complete entire Synod algorithm one time.
  - For new log entry.
  - Its proposed value must be the value that is accepted.

- After this proposal is accepted, leader can skip “prepare-promise” phase.
  - For any subsequent proposal, start with “accept” message with proposal number 0.

Paxos Optimization (II)

- An acceptor accepts proposal 0, unless it accepts competing proposal with higher number first
  - So it is safe if there are multiple leaders at the same time.
  - Only one can start with accept phase.

- If leader cannot complete accept phase (does not receive enough acks).
  - This means some other site issued a proposal.
  - Should rerun leader election algorithm to elect single leader.
  - Funnel failed proposal through new leader.

Paxos Implementation Details

- The proposal numbers should be re-initialized for each log entry.

- Since sites may be proposing for different log entries at the same time, proposals should contain both a log entry number and a proposal number.