BLOCKCHAIN

Unit3: Class 6

6. Consensus Algorithm – PAXOS

The failures in distributed systems can be broadly classified into three categories:

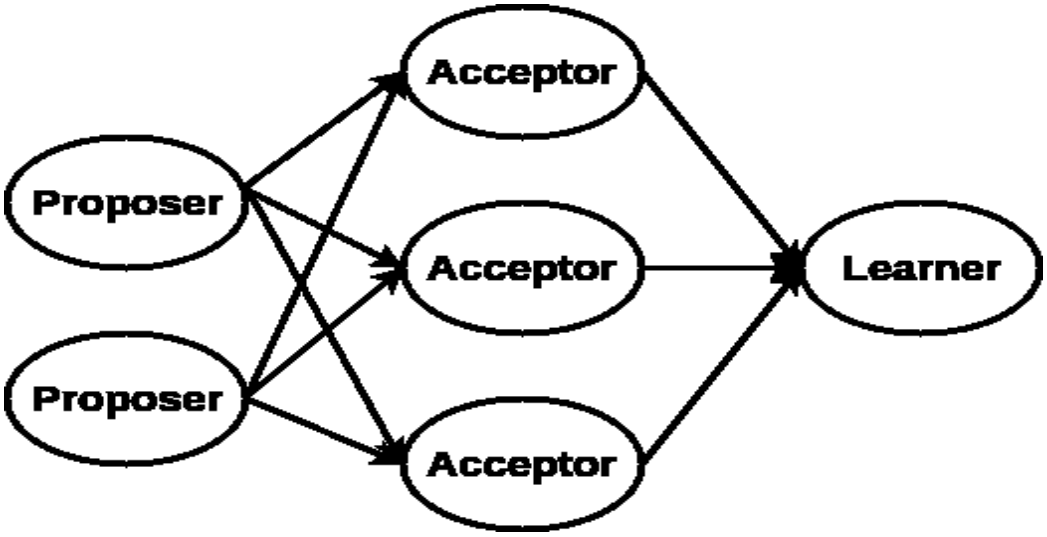
- Crash-fail: The component stops working without warning (e.g., the computer crashes).
- Omission: The component sends a message but it is not received by the other nodes (e.g., the message was dropped).
- Byzantine: The component behaves arbitrarily. This type of fault is irrelevant in controlled environments (e.g., Google or Amazon data centers) where there is presumably no malicious behavior. Instead, these faults occur in what's known as an "adversarial context." Basically, when a decentralized set of independent actors serve as nodes in the network, these actors may choose to act in a "Byzantine" manner. This means they maliciously choose to alter, block, or not send messages at all.
- The first real-world, practical, fault-tolerant consensus algorithm. It's one of the first widely adopted consensus algorithms to be proven correct by Leslie Lamport and has been used by global internet companies like Google and Amazon to build distributed services.
- Crash or Network fault
  - PAXOS
  - RAFT
- Byzantine Fault (including crash and network failures)
  - Byzantine Fault Tolerance (BFT)
  - Practical Byzantine Fault Tolerance (PFBT)
- Paxos is used in permissioned blockchain.
- The network is closed, the nodes know each other, so state replication is possible among the known node.
- Avoid the overhead of mining-do not need to spend anything(power,time, computation) other than message passing.

- The typical use case of Paxos is to replicate state to make it consistently and safely survive crash failures in a private environment.
- Paxos is designed for small clusters, relatively quick operation latency and high throughput.

- First Consensus Algorithm proposed by L. Lamport in 1989
- We have an environment of multiple systems (nodes), connected by a network, where one or more of these nodes may concurrently propose a value (e.g., perform an operation on a server, select a coordinator, add to a log, whatever…). We need a protocol to choose exactly one value in cases where multiple competing values may be proposed.
- We will call the processes that are proposing values **proposers.** We will also have processes called **acceptors** that will accept these values and help figure out which one value will be chosen.

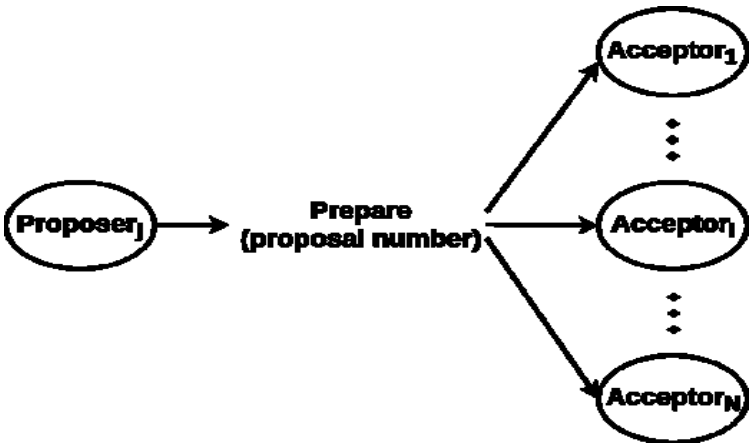
- Paxos simply selects a single value from one or more values that are proposed to it and lets everyone know what that value is. A run of the Paxos protocol results in the selection of single proposed value.
- **Objective:** choosing a single value under crash or network fault.
- System process
  - Making a proposal
  - Accepting a value
  - Handling Failures

PAXOS: Types of Nodes



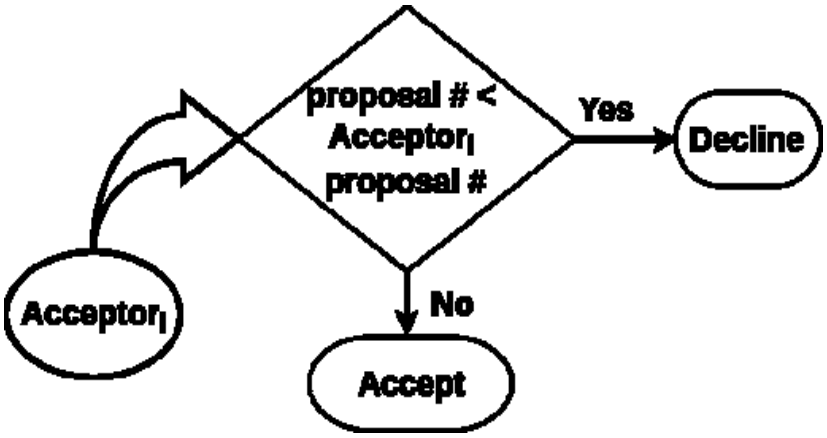
- Proposer: propose values that should be chosen by the consensus
- Acceptor: form the consensus and accept values
- Learner: learn which value was chosen by each acceptor
- When a proposer receives a client request to reach consensus on a value, the proposer must create a proposal number. This number must have two properties:
    - It must be unique. No two proposers can come up with the same number. An easy way of doing this is to use a global process identifier for the least significant bits of the number. For example, instead of an ID=12, node 3 will generate ID=12.3 and node 2 will generate 12.2.
    - It must be bigger than any previously used identifier used in the cluster. A proposer may use an incrementing counter or use a nanosecond-level timestamp to achieve this. If the number is not bigger than one previously used, the proposer will find out by having its proposal rejected and will have to try again.
- A proposer receives a consensus request for a VALUE from a client. It creates a unique proposal number, ID, and sends a PREPARE(ID) message to at least a majority of acceptors.

Making a Proposal: Proposer Process



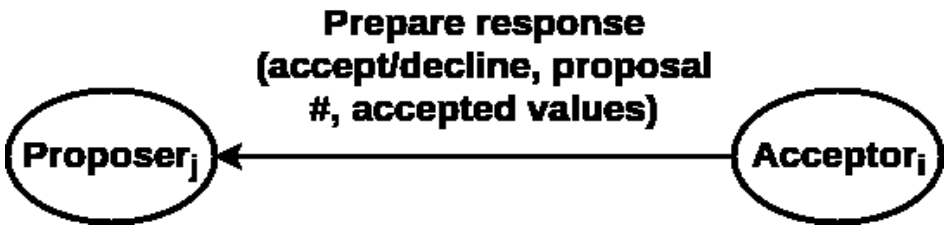
- proposal number: form a timeline, biggest number considered up-to-date

Making a Proposal: Acceptor's Decision Making



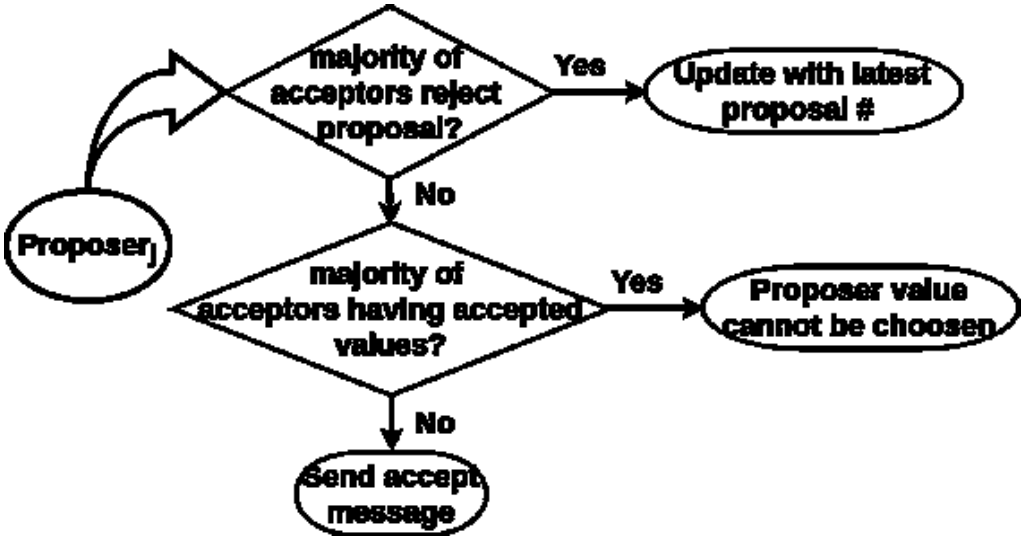
- Each acceptor compares received proposal number with the current known values for all proposer's prepare message.

Making a Proposal: Acceptor's Message



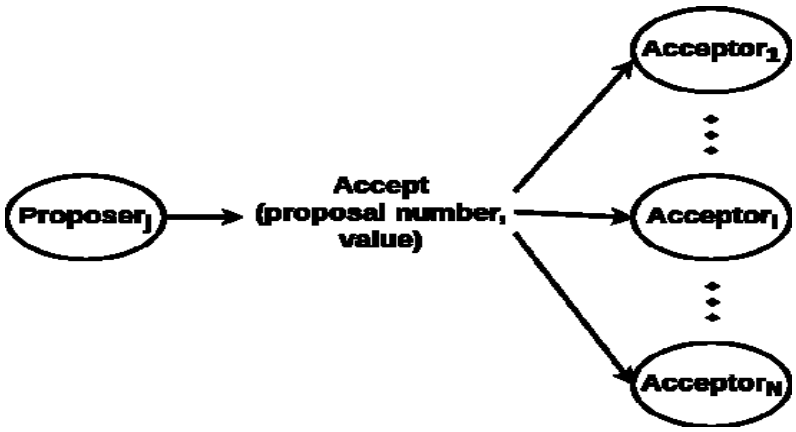
- accept/decline: whether prepare accepted or not
- proposal number: biggest number the acceptor has seen
- accepted values: already accepted values from other proposer

Accepting a Value: Proposer's Decision Making



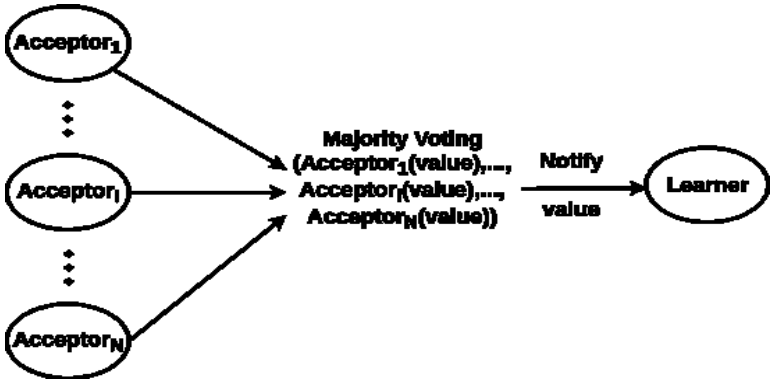
- Proposer receive a response from majority of acceptors before proceeding.

Accepting a Value: Accept Message



- proposal number: same as prepare phase value
- value: single value proposed by proposer

Accepting a Value: Notifying Learner



- Each acceptor accept value from any of the proposer
- Notify learner the majority voted value

Procedure

1. The proposer initially prepares a proposal with a proposal number known as prepare message and send it to the acceptors. This proposal number forms a time-line and the biggest number is considered up to date. For

example, between proposal number x and z = x+y, where y ¿= 1, proposal number z will be considered as latest and will be accepted. Prepare message: (proposal number)

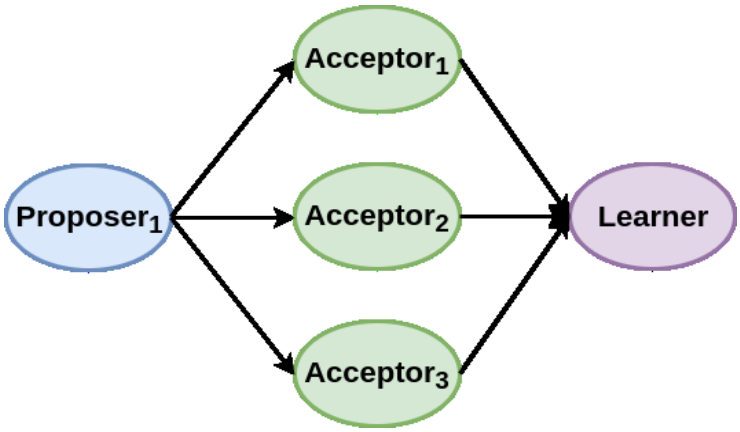
2. Each acceptor compares received proposal number with the current known values for all proposer's proposal message. If it gets a higher number, then it accepts the proposal or else declines it. Then the acceptor prepares the response message of the following form Prepare response: (accept/reject, proposal number, accepted values) where, proposal number is the biggest number the acceptor has seen. and accepted values are the already accepted values from other proposer.

3. Next a vote is being taken based on the majority decision. The proposer checks whether the majority of the acceptors have rejected the proposal. If yes, then the proposer updates it with the latest proposal number. If no, then the proposer further checks whether the majority of the acceptors have already accepted values. If yes then the proposers value can not be selected or else it sends accept message.

4. Finally, the proposer sends the accept message having the following format to all the acceptors. Accept message: (proposal number, value) where proposal number: same as prepare phase value value: single value proposed by proposer.

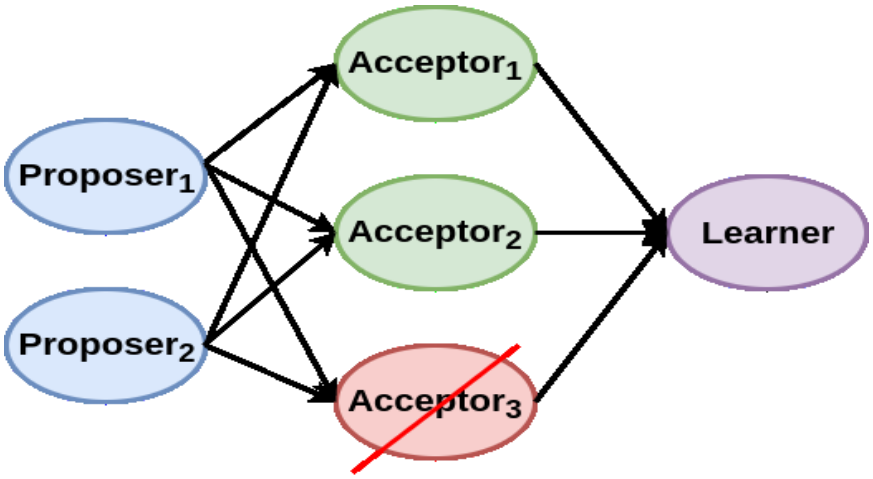
5. whenever the acceptor accepts a value, it informs the learner nodes about it so that everyone will learn about the accepted value.

Single Proposer: No Rejection

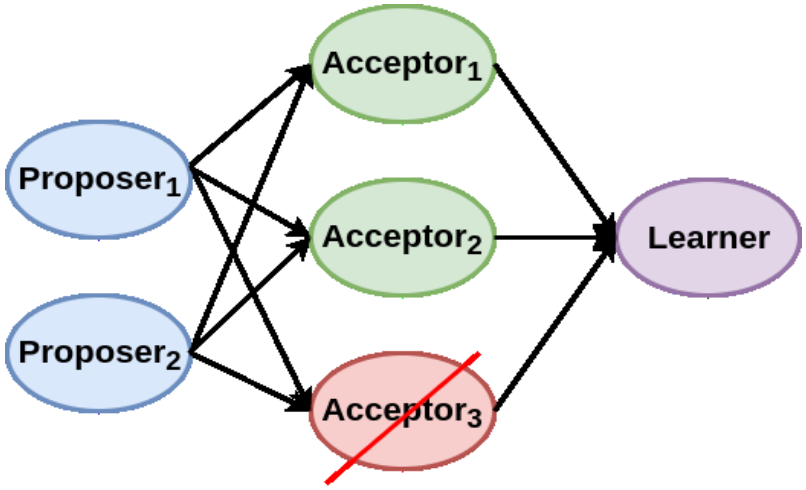


- Proposer always have proposal with biggest number
- No proposal rejected

Handling Failure: Acceptor Failure

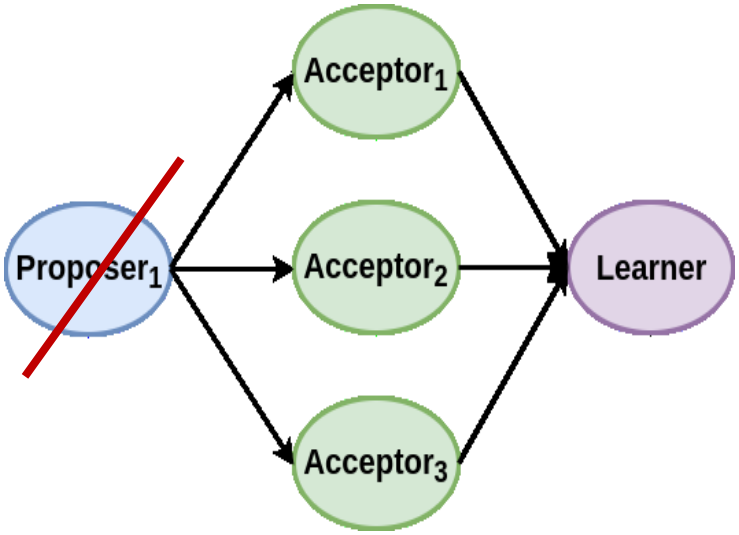


- **Acceptor fails during prepare**
  - No issues, other acceptor can hear the proposal and vote



- More than N/2 - 1 acceptors fail
  - no proposer get a reply
  - no values can be accepted

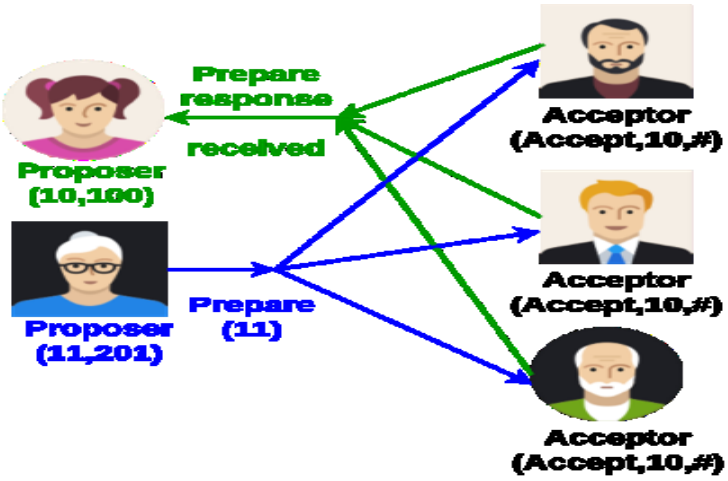
Handling Failure: Proposer Failure



- **Proposer fails during prepare phase**
  - Acceptors wait, wait, wait, and then someone else become the proposer.

- **Proposer fails during accept phase**
  - Acceptors have already agreed upon whether to choose or not to choose the proposal.

Handling Failure: Dueling Proposers



- Proposer received confirmations to her prepare message from majority – yet to send accept messages
- Another proposer sends prepare message with higher proposal number
- Block the first proposer's proposal from being accepted
- Use **leader election** - select one of the proposer as leader
- Paxos can be used for leader election !!