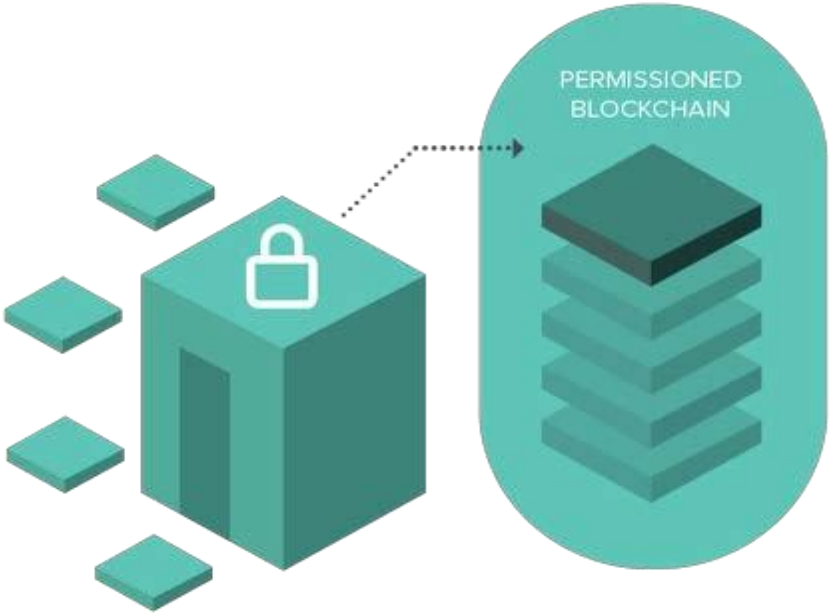
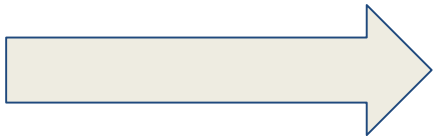


ImageSource:https://nem.io/enterprise/

Permissioned Blockchain - II
Consensus Algorithms

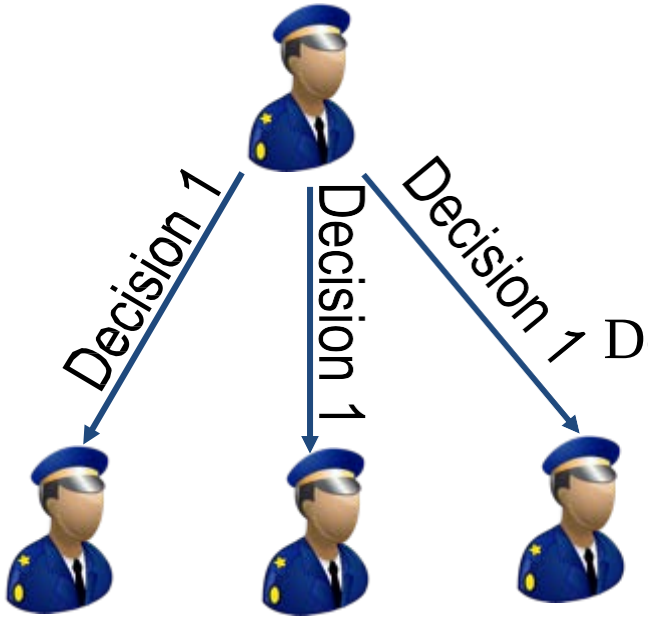
Why Distributed Consensus

One Decision Maker



No Consensus

Why Distributed Consensus



Multiple
Decision Maker

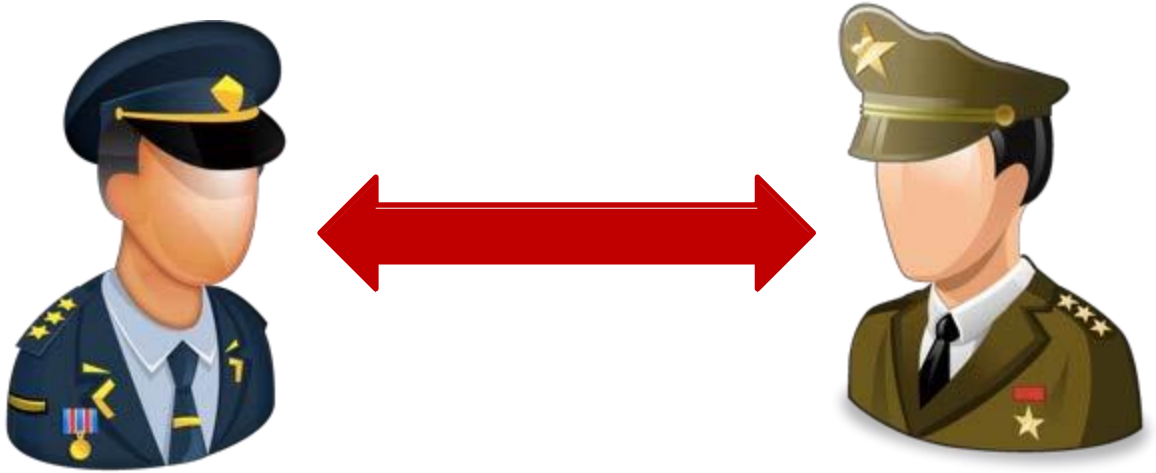
Consensus Required

Why Distributed Consensus

- It **helps in reaching agreement** in case of distributed computing
- In case of replication of state machines, you replicate the **common state** so that all processes have same view
- **Real applications:**
  - **Flight control system:** e.g. Boeing 777 and 787 where we have multiple flights and they want to coordinate their positions among themselves.
  - **Fund transferring system:** Bitcoin and cryptocurrencies
  - **Leader election/Mutual Exclusion** where all the nodes collectively select their leader in the system. Then, we require distributed consensus mechanisms.

Why Distributed Consensus

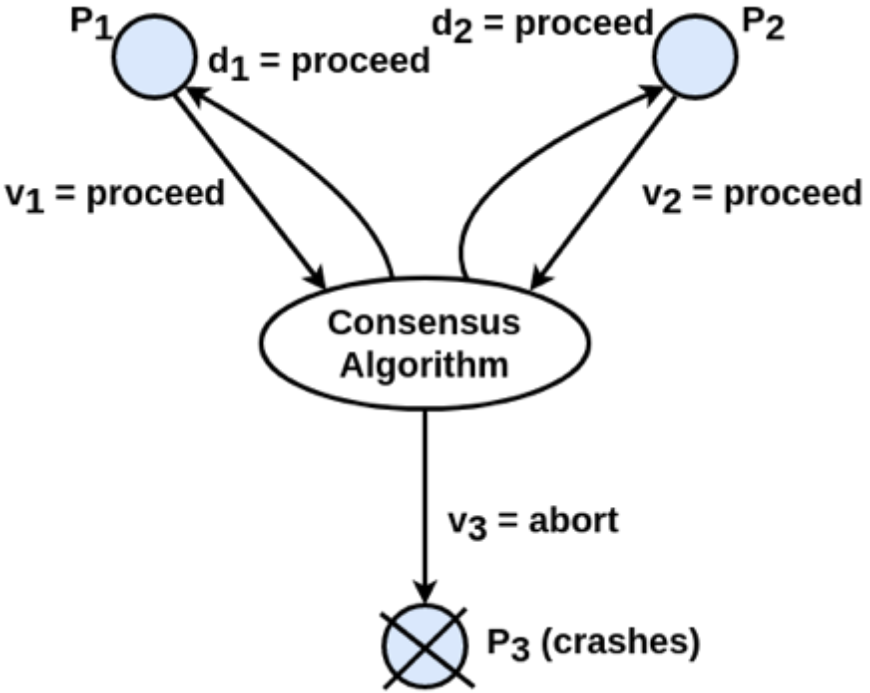
- So, no need of consensus in a single node process.
- **What about when there are two nodes?**
  - Network or partitioned fault/in presence of malicious nodes, consensus cannot be reached
  - **To achieve consensus, you always require more than two nodes**



Faults in Distributed Consensus

- **Crash Fault:** means nodes crashes- no data sent-may recover after some times
- **Network or Partitioned Faults:** N/w gets partitioned and message from one partitioned n/w not gets propagated to other n/w
- **Byzantine Faults:**
  - malicious behaviour in nodes because of
    - hardware fault
    - software error

Consensus for three processes



- Each process P_i (i=1,2,...N):
  - **Undecided state:** proposed certain value v_i from set D. Means every node proposed one state and it is in the undecided state.
  - **Communication state:** exchange values among themselves
  - **Decided state:** set decision variable d_i

Requirements of a Consensus Algorithm
Details we have already looks earlier

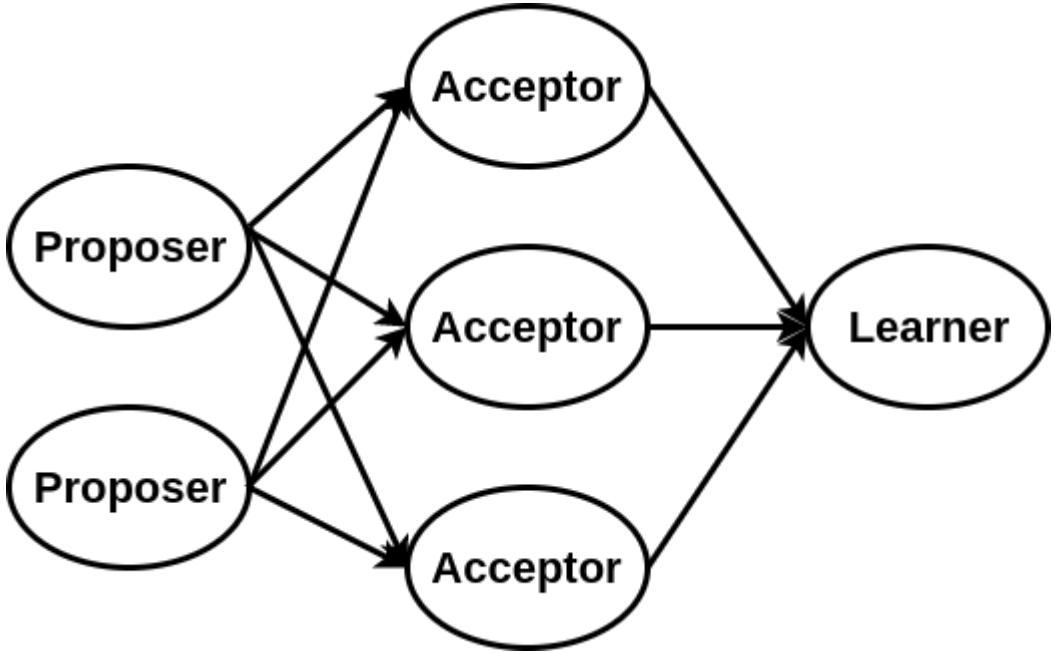
- **Termination:**
  - Eventually each correct process sets its decision variable
- **Agreement:**
  - The decision value of all correct processes is the same. Means every one reached to the common agreement.
- **Integrity:**
  - If the correct processes all proposed the same value, then any correct process in the decided state has chosen that value.

Different Consensus Algorithms for Permissioned environment
They are based on state machine replication principle

- **Both supports Crash or Network Faults but not Byzantine Faults:**
  - PAXOS
  - RAFT

- **Byzantine Faults (including Crash or Network Failures):**
  - Byzantine fault tolerance (BFT)
  - Practical Byzantine Fault Tolerance (PBFT)

PAXOS



- First Consensus Algorithm proposed by L. Lamport in 1989
- **Objective:** choosing a single value under crash or network fault
- **System process**
  - Making a proposal
  - Accepting a value
  - Handling Failures

Source: Lamport, Leslie. "Paxos made simple." ACM Sigact News 32.4 (2001): 18-25.

It took around 13 years to get the paper published because reviewers were not confident that the proposed algorithm is fare enough.

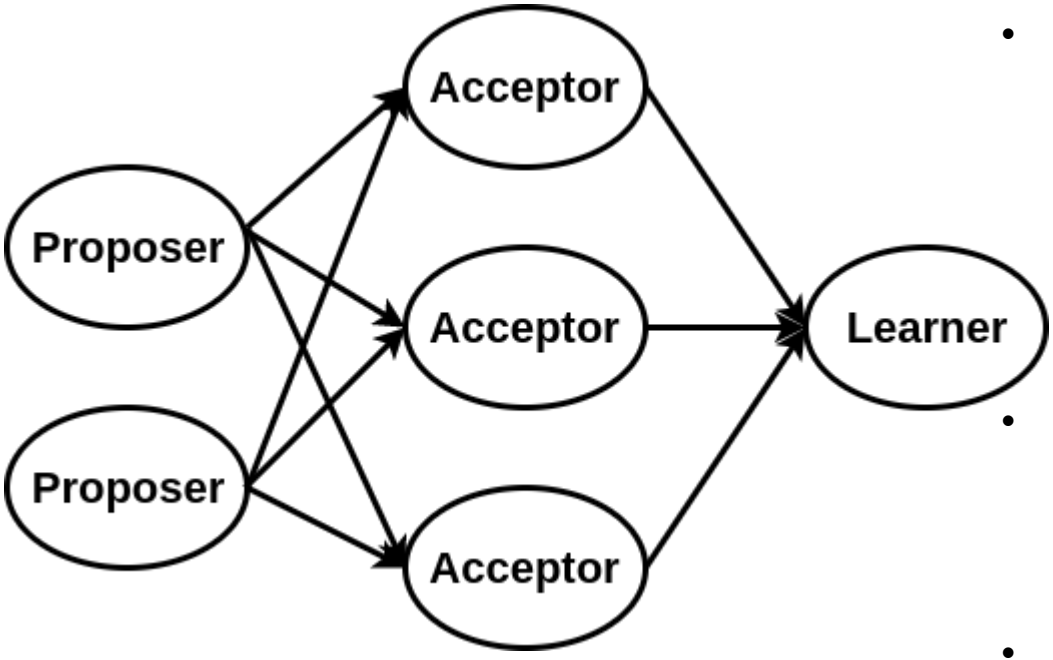
PAXOS Working

- In NU, students have two options after the class either to go K-canteen or NIM-Canteen
- Now students can decides either to go to K-canteen or NIM-Canteen
- **How they select either K-canteen or NIM-Canteen?**
- **Constraints:** All wants to go together, otherwise the party is not fantastic. There is no central leader and every one propose there own view points (means certain values) and through these values consensus can be framed out.
- If I am the member of team of students- I just wait for some amount of time to see someone else from the group is proposing some value or not-If not then I will propose my view point (to go to K-canteen) and see how other students react/accept my value.

PAXOS Working

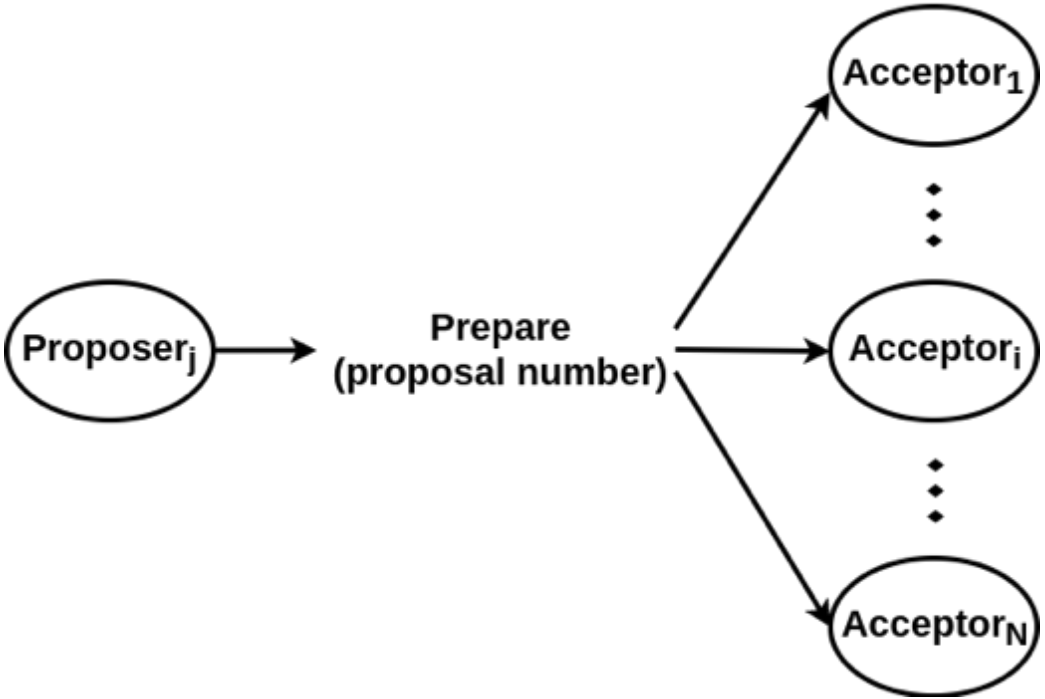
- In this way, the information gets propagated in the n/w and it helps to make the consensus based on the majority decision.
- If majority of the nodes proposes/agreed that we want to go to K-canteen then all goes to K-canteen
- Otherwise, if majority of the nodes proposes/agreed that we want to go to NIM-canteen then all goes to NIM-canteen.
- That is the broad idea behind PAXOS algorithm
- Now we will see how algorithmically it works

PAXOS: Types of Nodes



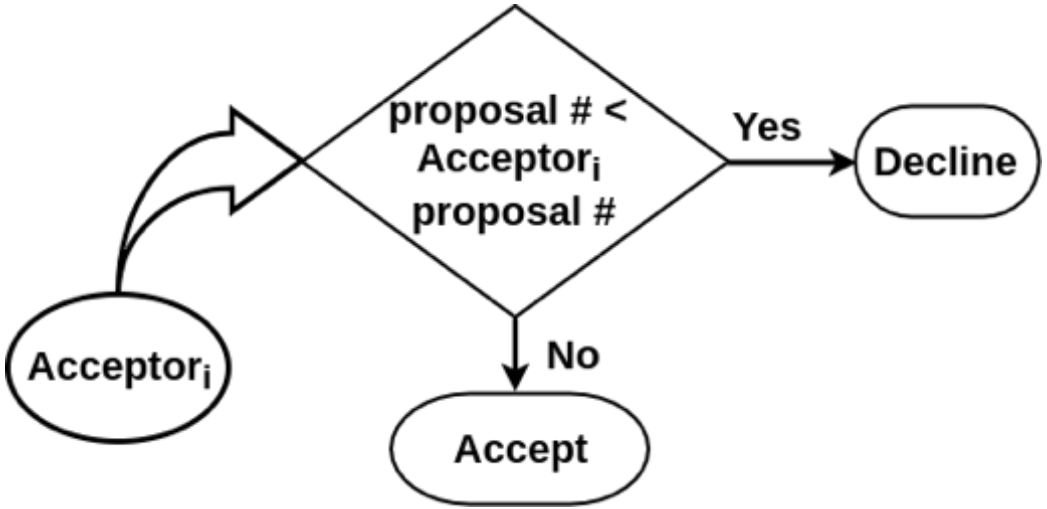
- **Proposer:** they propose values that should be chosen by the consensus algo (like with to go NIM or K canteen)
- **Acceptor:** they form the consensus and accept the values (I am reviewing a proposal to go to NIM-canteen but another proposal will come to me to go K-canteen so I straightaway reject the first one)
- **Learner:** learn which value was chosen by each acceptor then they will collectively accepts that particular value.
- **Every one is the learner in the n/w**, who learns what is the majority decision in the n/w

Process of PAXOS? Making a Proposal: Proposer Process



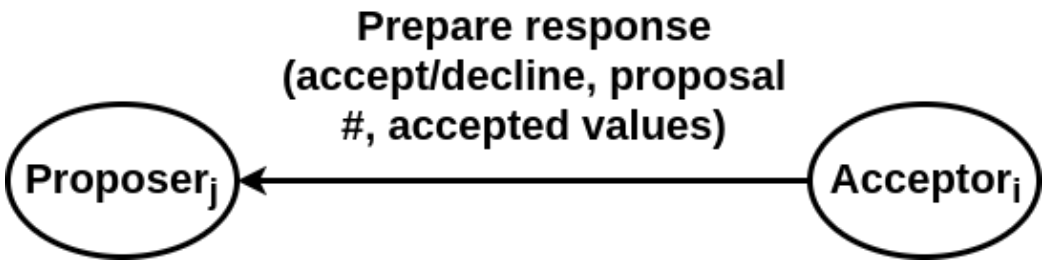
- **Proposal number:** It form a timeline, biggest number considered up-to-date
- **For example,** if two proposals are coming from; P1 -100, P4- 102, then we will accept the proposal coming from P4, as it has biggest number

Making a Proposal: **Acceptor's Decision Making**



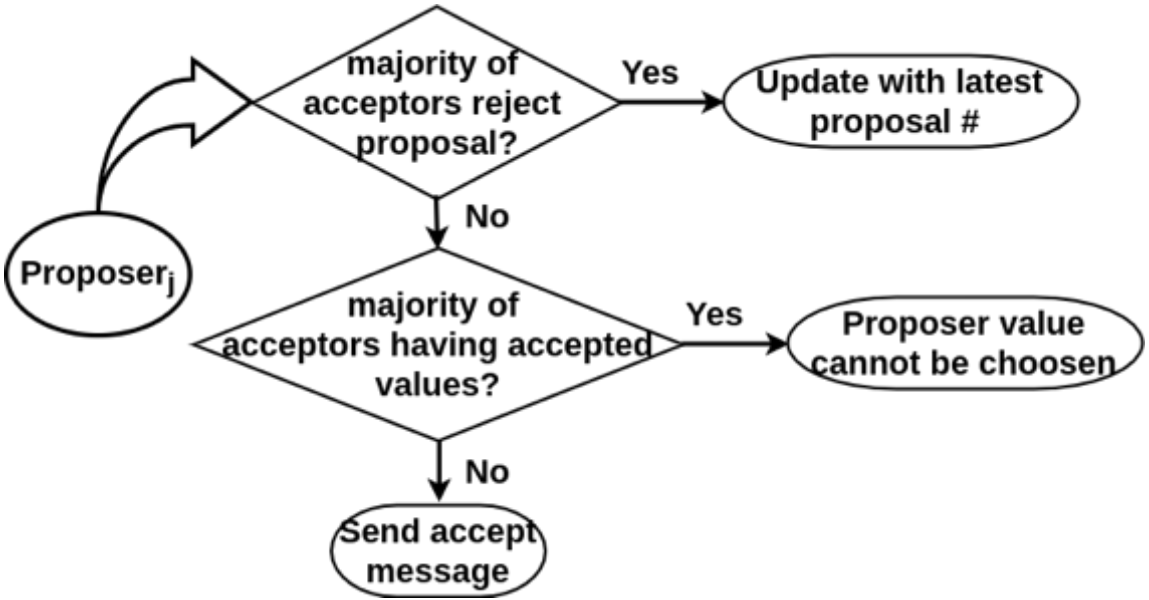
- Each acceptor compares received proposal number with the current known values for all proposer's prepare message
- If the proposal number is less than the acceptor_i proposal number then you accept it
- otherwise reject it

Then Acceptor’s prepare a response Message



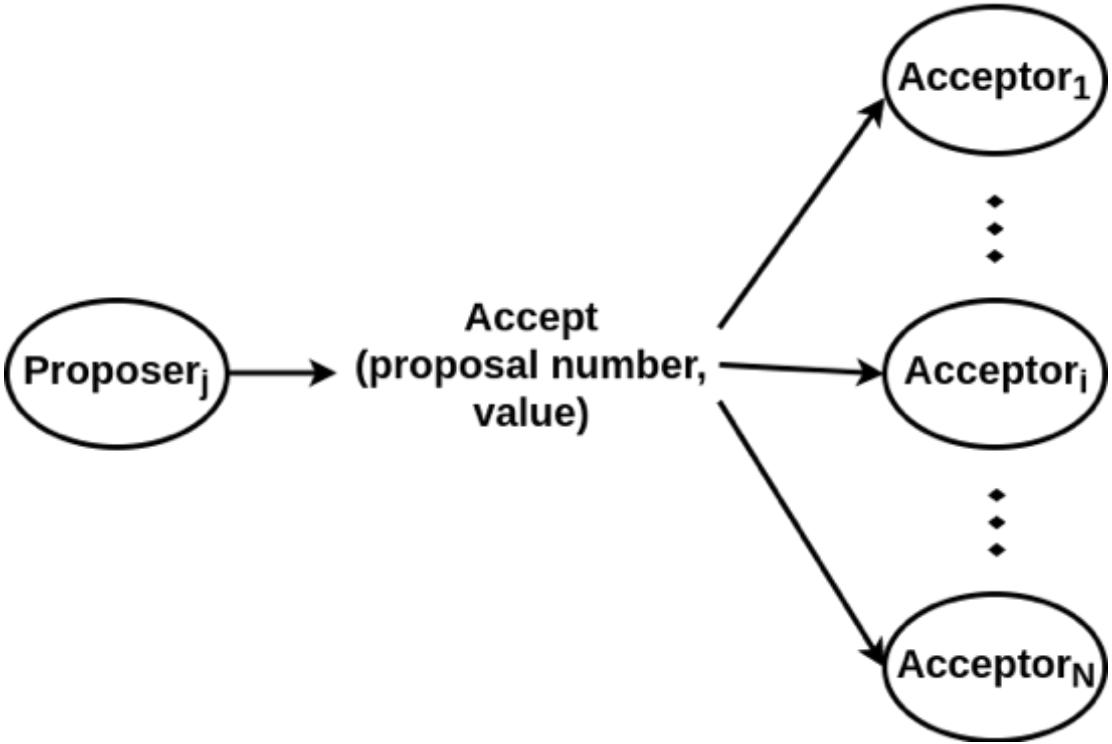
- **accept/decline:** whether prepare accepted or not
- **proposal number:** biggest number the acceptor has seen
- **accepted values:** already accepted values from other proposer and this accepted value is informed to the proposer

Accepting a Value: Proposer's Decision Making



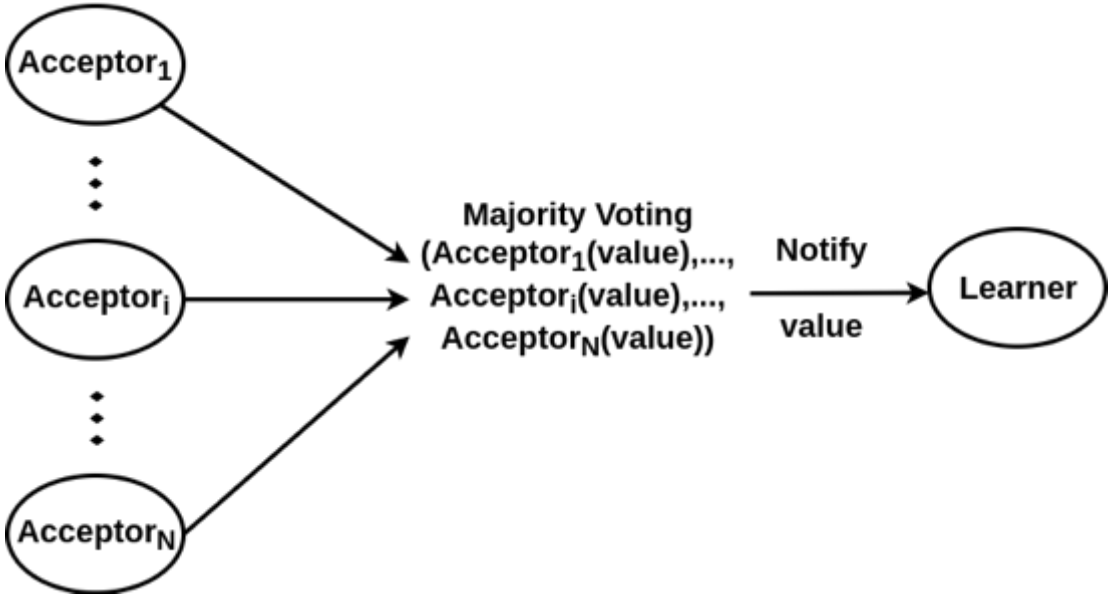
- Proposer receive a response from **majority** of acceptors before proceeding
- Whenever the majority of the acceptors, they are sending some accepted values,
- if they have accepted your value then the value that you have shared is coming to be the consensus

Accepting a Value: **Final stage is “Accept Message"**



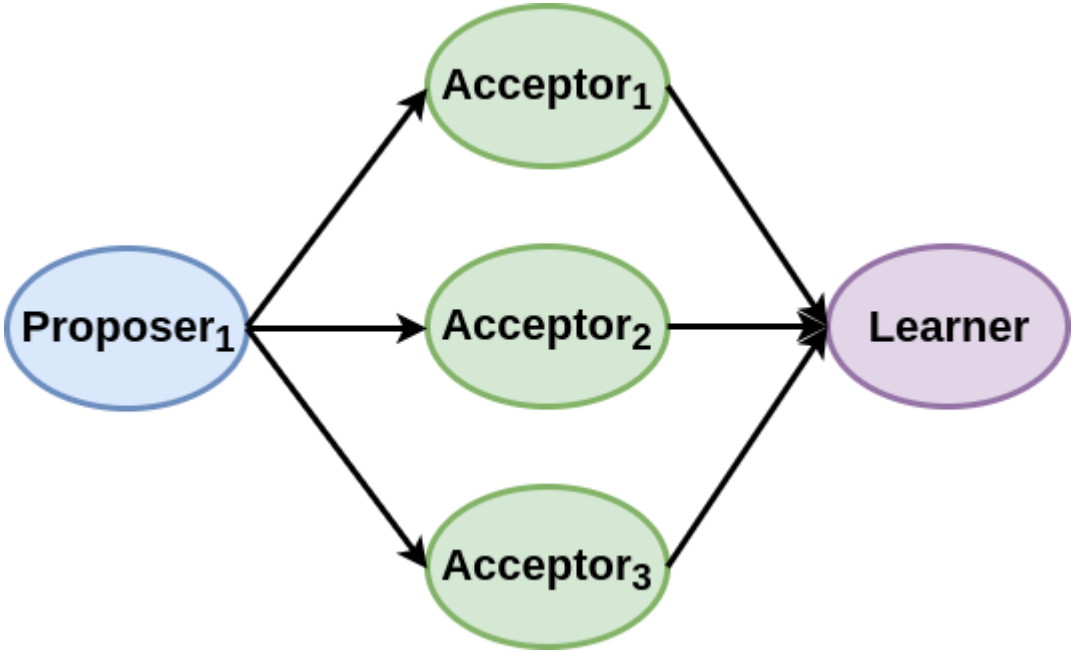
- Proposer sends the accept message to all acceptors, which includes:
- **proposal number:** same as prepare phase value
- **value:** a single value proposed by proposer

Accepting a Value: Notifying Learner



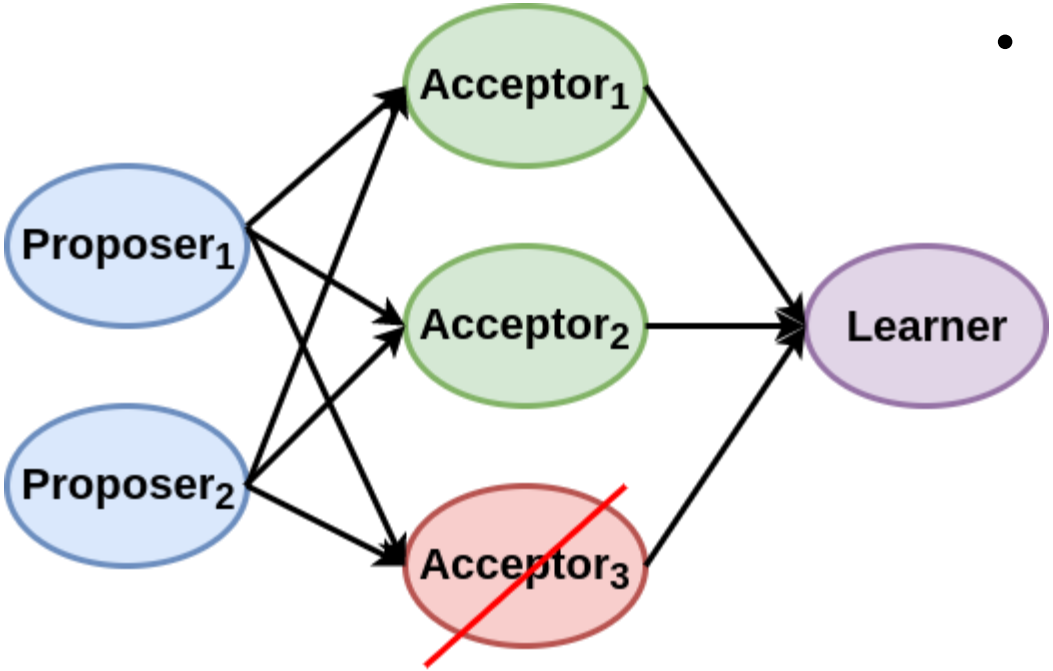
- Each acceptor accept value from any of the proposer then
- Notify learner the majority voted value

If you have a Single Proposer in System: No Rejection



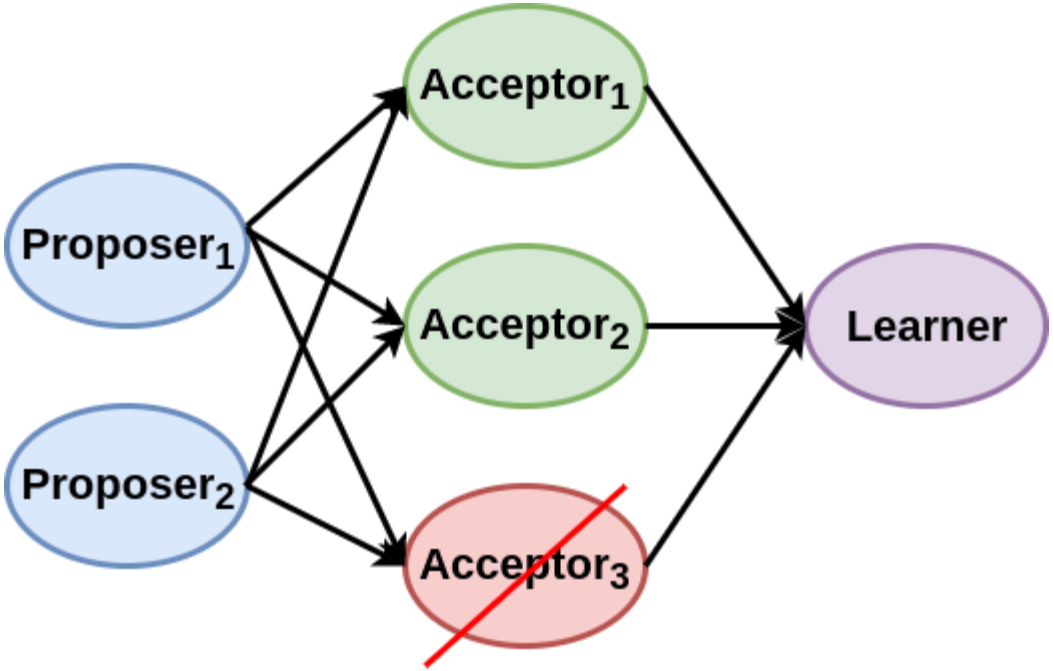
- Proposer always have proposal with biggest number
- Here every acceptors, accept the proposal because it is going to be the biggest
- No proposal rejected
- Only we need to **ensure that no accepter is faulty?? Or majority of the acceptors are non faulty**

How to Handle Failure: Acceptor Failure



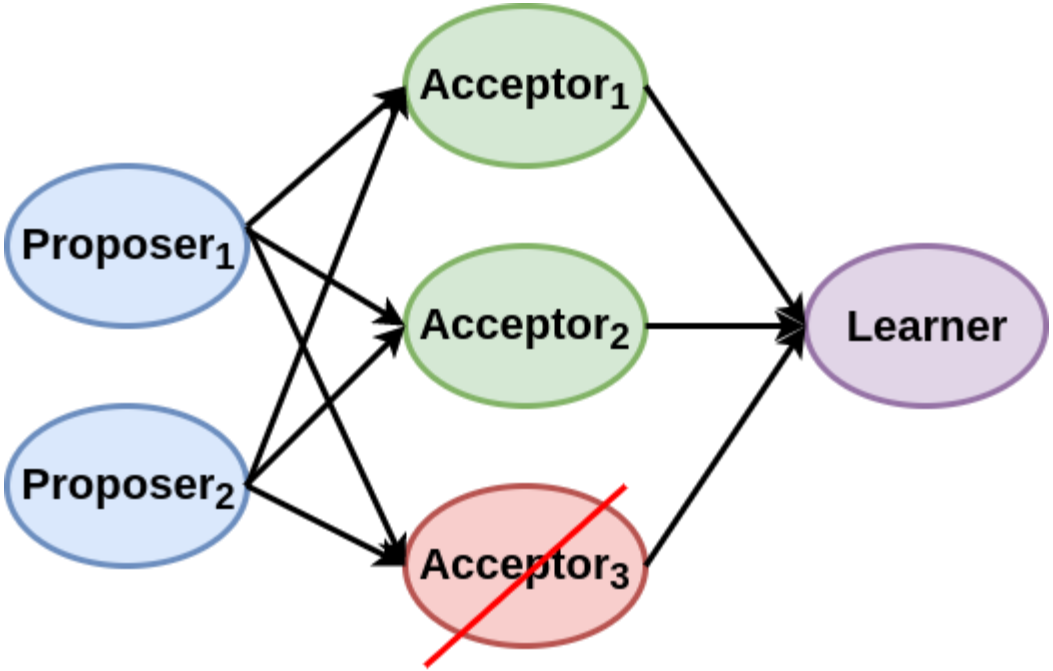
- **Acceptor fails during prepare phase then**
  - No issues, other acceptor can hear the proposal and **cast their vote either in favor or against of the proposal**

Handling Failure: Acceptor Failure



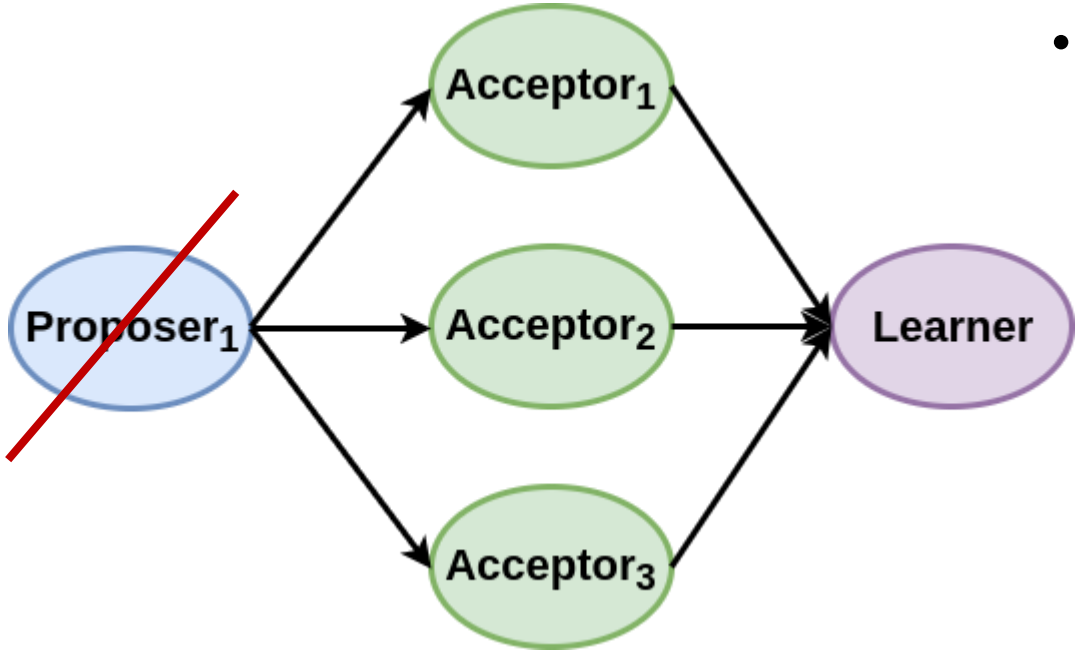
- **Acceptor fails during accept phase then**
  - Again, no issues, other acceptor can vote for the proposal

Handling Failure: Acceptor Failure



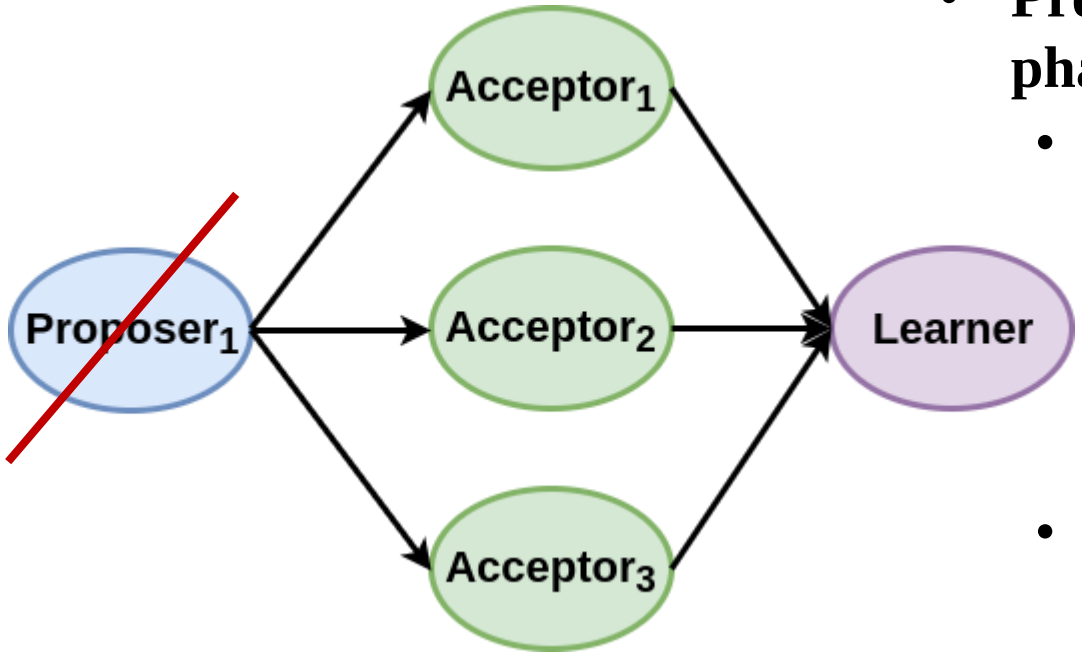
- If more than N/2 - 1 acceptors fail then
  - no proposer get a reply
  - no values can be accepted
  - Means can’t reached to consensus

How to handle when Proposer Failure



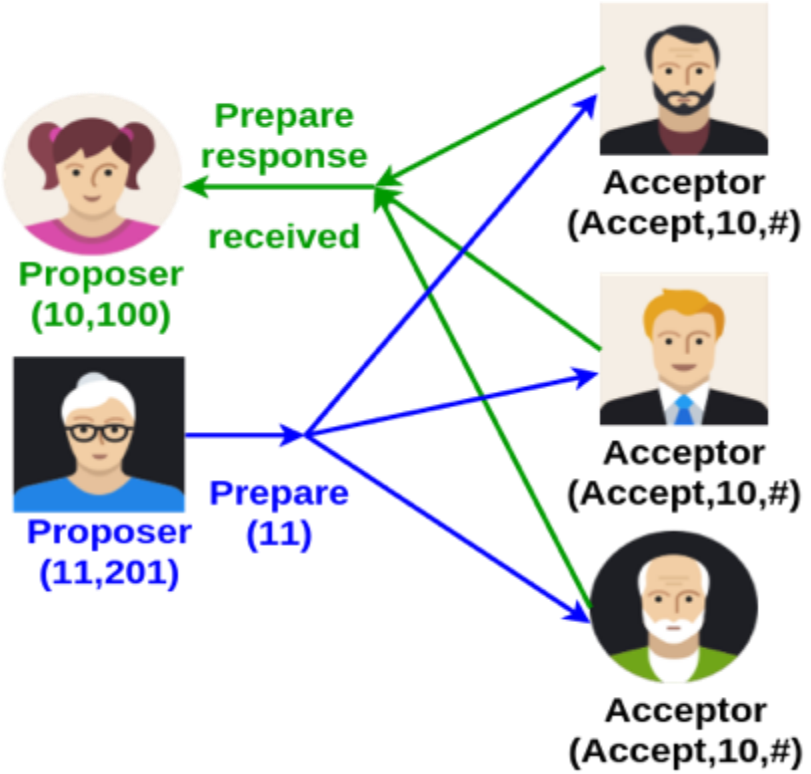
- **Proposer fails during prepare phase (means no one proposing any value)**
  - Means no one proposing any value
- Acceptors wait, wait, wait, and then someone else become the proposer

Handling Failure: When Proposer Failure



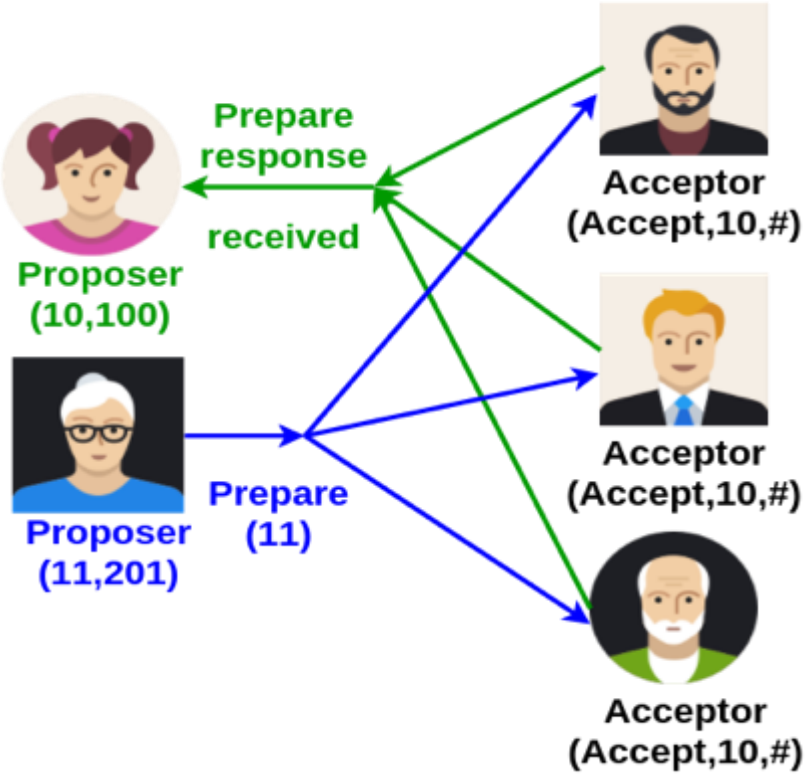
- **Proposer fails during accept phase**
  - Acceptors have already agreed upon whether to choose or not to choose the proposal based on majority vote.
  - They have shared the majority vote among themselves and from it they can find whether the proposal has been accepted or not.

How to handle **“Dueling Proposers Attack”**



- Proposer received confirmations to her prepare message from majority
  – yet to send accept messages
- Another proposer sends prepare message with higher proposal number
- To break the tie between the two proposers
- Block the first proposer’s proposal from being accepted

Which Block you need to Block in Dueling Proposers



- Use "**leader election algorithm" to handle this attack -** select one of the proposer as leader

- **Paxos** can be used for leader election !!