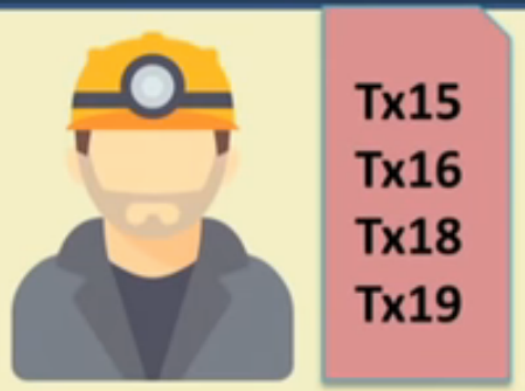
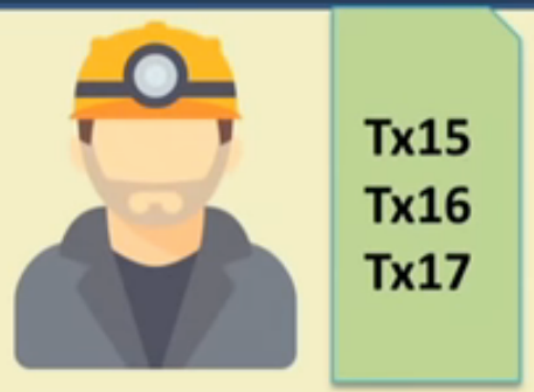
Consensus in Bitcoin

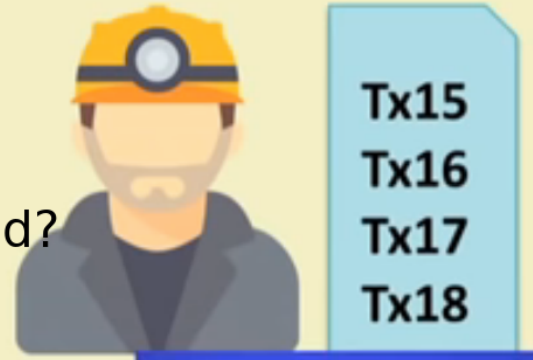




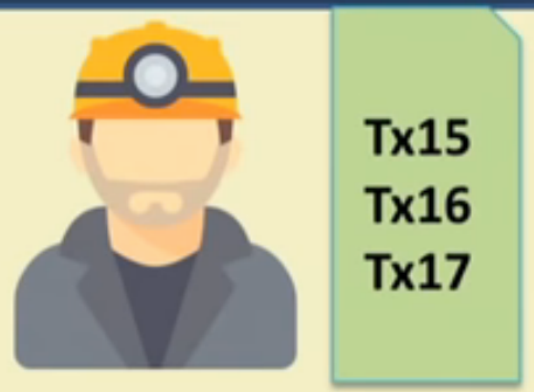
Bitcoin Consensus Objective:

Which block do we add next?

Miners have mined a block but which one of them will get minted?



Consensus in Bitcoin

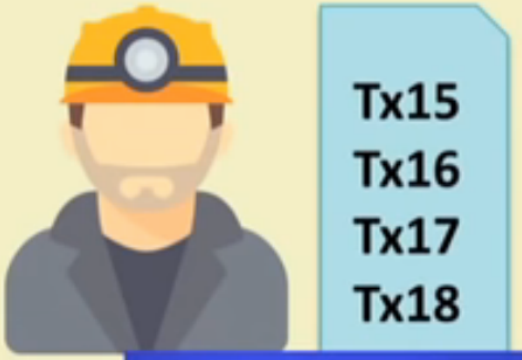


Bitcoin Consensus Objective:

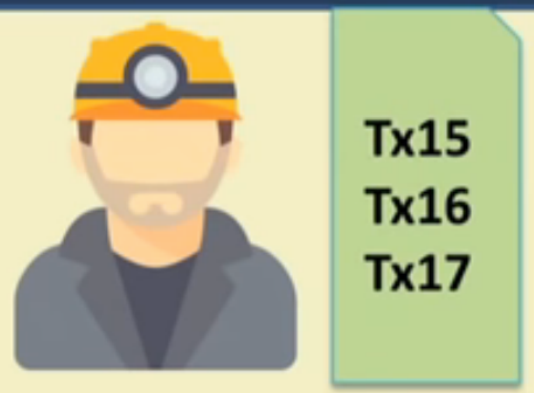
Which block do we add next?

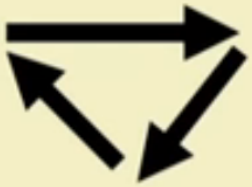
Challenge:

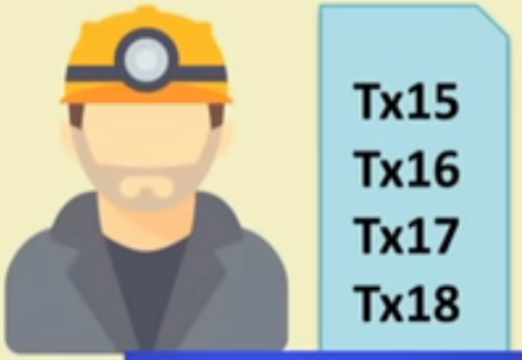
The miners do not know each other



Consensus in Bitcoin



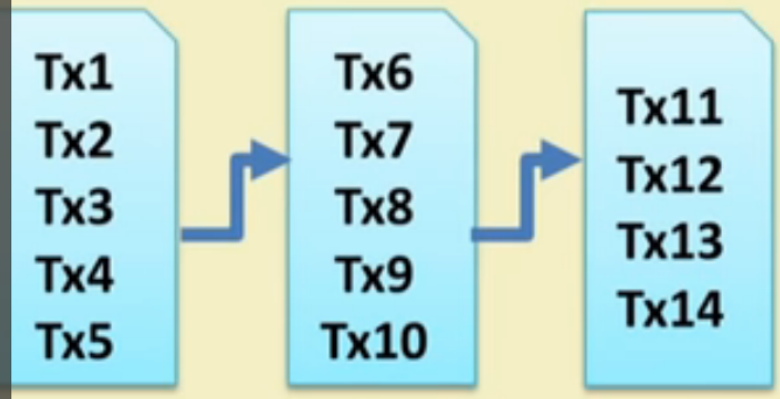


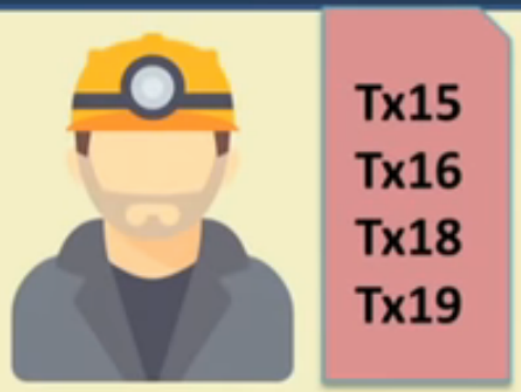


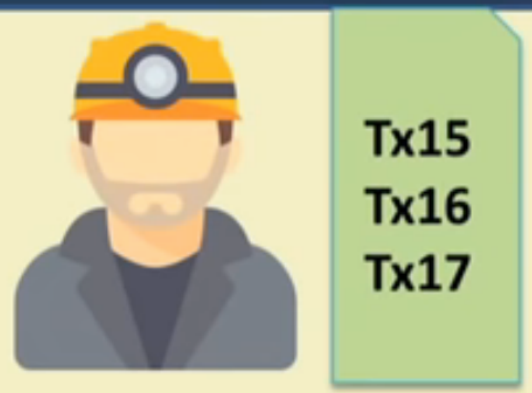
Possible Solution:

Broadcast the information and then apply a choice function – traditional distributed consensus algorithms

Consensus in Bitcoin





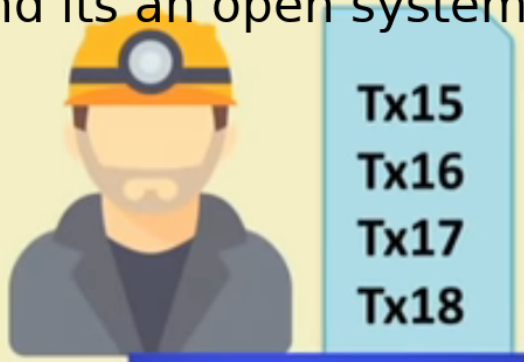


Internet is inherently asynchronous and its an open system

May not be Feasible:

You do not have a global clock! How much time will you wait to hear the transactions

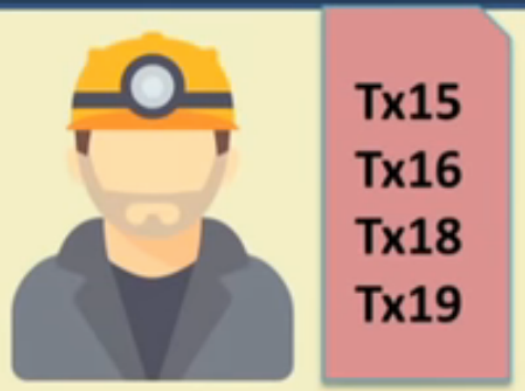
Remember the impossibility result

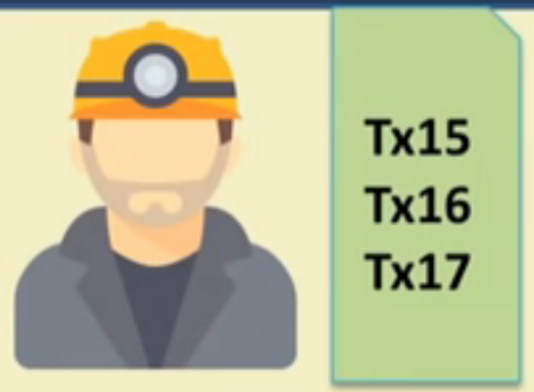


IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

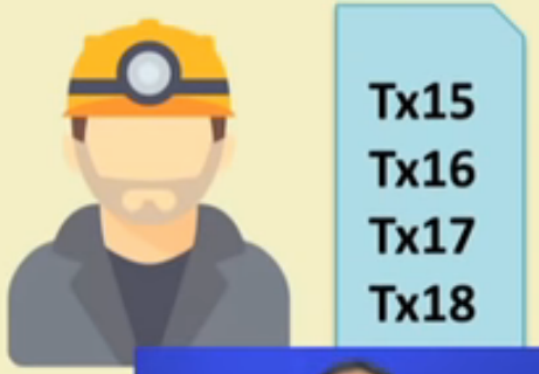
Consensus in Bitcoin



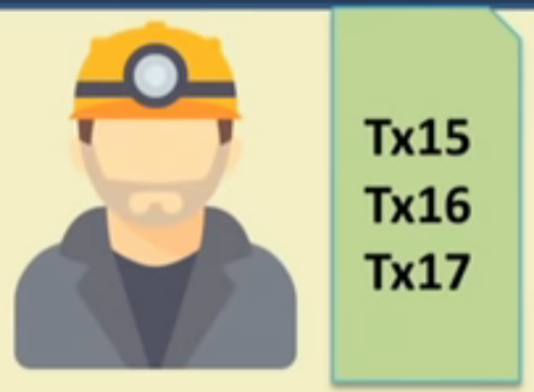


Observation - 1:

- Any valid block (a block with all valid transactions) can be accepted, even if it is proposed by only one miner

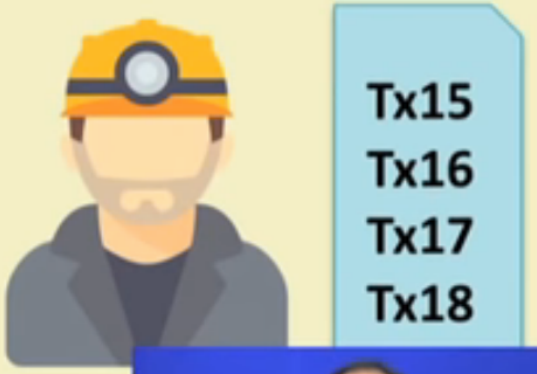


Consensus in Bitcoin

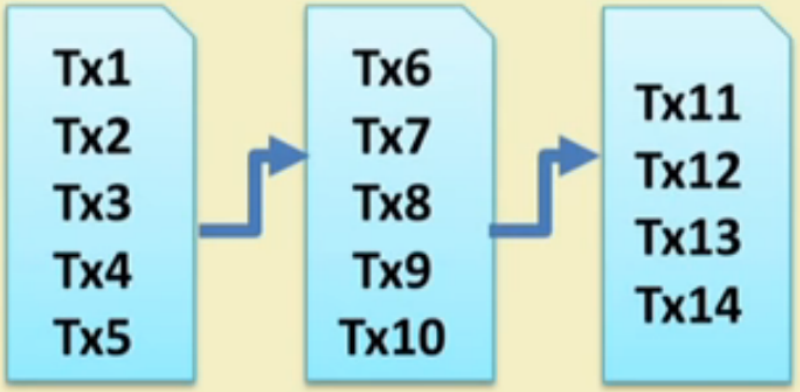


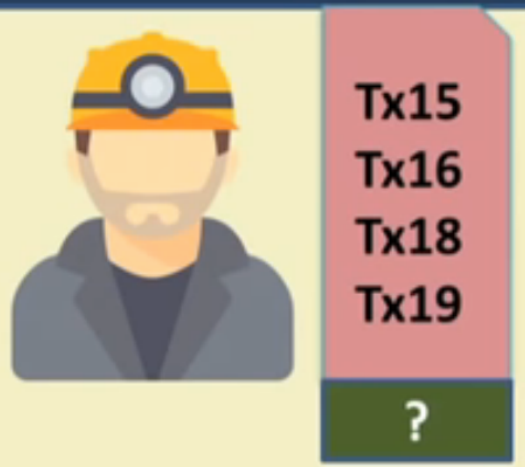
Observation - 2:

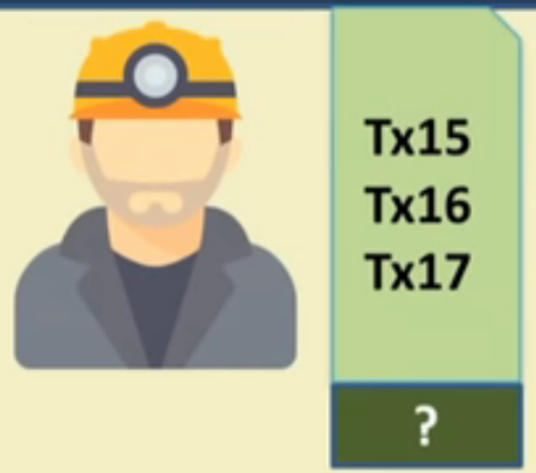
- The protocol can work in rounds
  - Broadcast the accepted block to the peers
  - Collect the next set of transactions



Consensus in Bitcoin

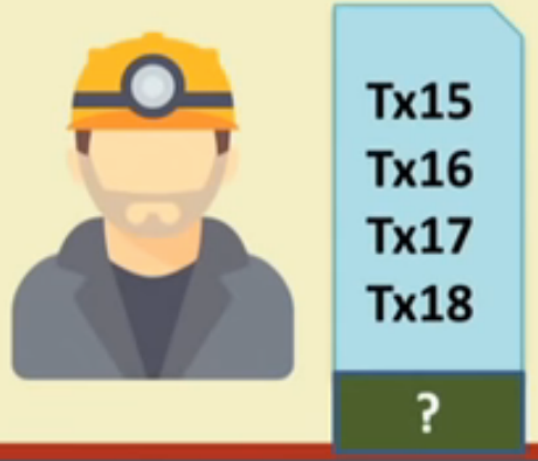




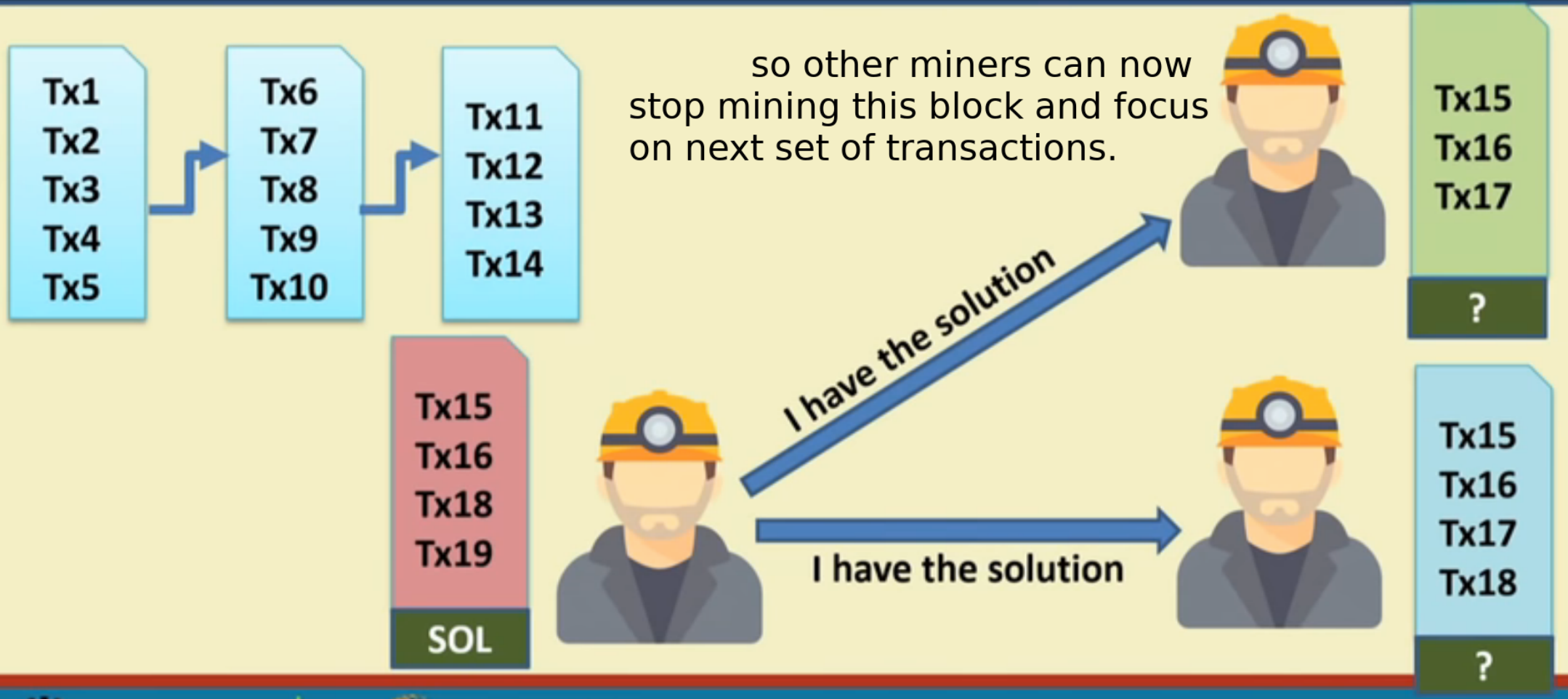


Solution:

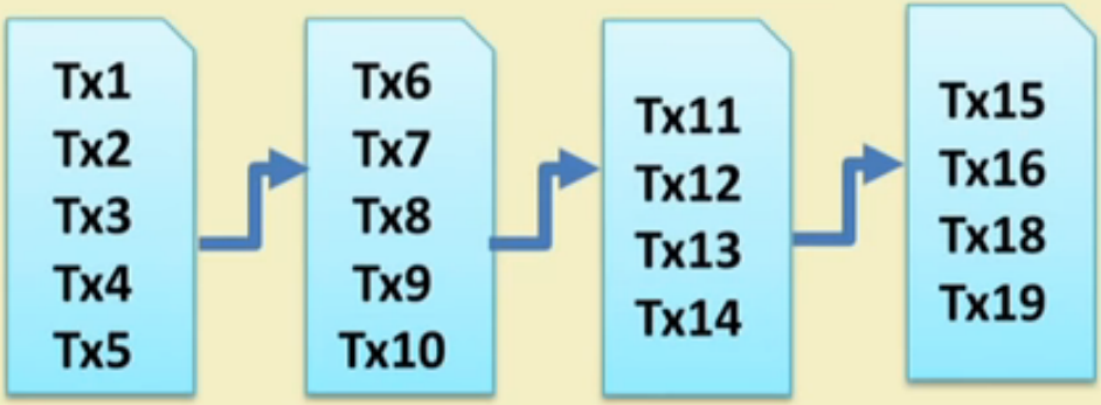
- Every miner independently tries to solve a challenge
- The block is accepted for the miner who can prove first that the challenge has been solved



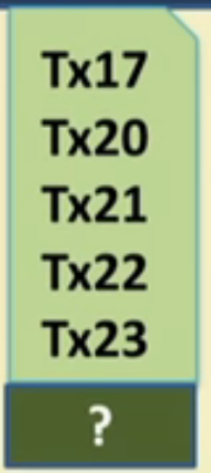
Consensus in Bitcoin

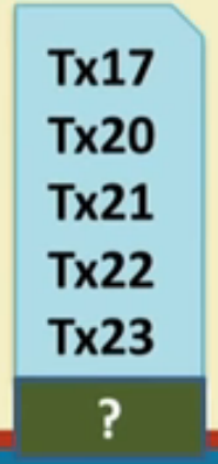


Consensus in Bitcoin



Note: Everyone can see that Tx18 and Tx19 have been committed, but Tx17 has not been committed. Include that in the next round





Proof of Work (Pow)

- An economic measure to deter service abuses by requiring some work from the service requester (usually processing time by a computer)

- The idea came from Dwork and Naor (1992), to combat junk emails
  - You have to do some work to send a valid email
  - The attacker would be discouraged to send junk emails

Dwork, Cynthia; Naor, Moni (1993). "Pricing via Processing, Or, Combatting Junk Mail. Advances in Cryptology". *CRYPTO'92: Lecture Notes in Computer Science No. 740*. Springer

Proof of Work (PoW) Features

- **Asymmetry**
  - The work must be moderately hard, but feasible for the service requester
  - The work must be easy check for the service provider

- Service requesters will get discouraged to forge the work, but service providers can easily check the validity of the work

Sybil Attacks

- Attacker attempts to fill the network with the clients under its control
  - Refuse to relay valid blocks
  - Relay only attacked blocks – can lead to double spending

- Solution:
  - Diversify the connections – Bitcoin allows outbound connection to one IP per /16 (a.b.0.0) IP address

Denial of Service (DoS) Attacks

- Send lot of data to a node – they will not be able to process normal Bitcoin transactions

- Solutions:
  - No forwarding of orphaned blocks
  - No forwarding of double-spend transactions
  - No forwarding of same block or transactions
  - Disconnect a peer that sends *too many* messages
  - Restrict the block size to 1 MB
  - Limit the size of each script up to 10000 bytes
  - …

The Monopoly Problem

- PoW depends on the computing resources available to a miner
  - Miners having more resources have more probability to complete the work

- Monopoly can increase over time (*Tragedy of the Commons*)
  - Miners will get less reward over time
  - Users will get discouraged to join as the miner
  - Few miners with large computing resources may get control over the network

Handling Monopoly and Power Consumption - Proof of Stake (PoS)

- Possibly proposed in 2011 by a Member in Bitcoin Forum - https://bitcointalk.org/index.php?topic=27787.0
  - Make a transition from PoW to PoS when bitcoins are widely distributed

- PoW vs PoS
  - PoW: Probability of mining a block depends on the work done by the miner
  - PoS: Amount of bitcoin that the miner holds – Miner holding 1% of the Bitcoin can mine 1% of the PoS blocks.

Proof of Stake (PoS)

- Provides increased protection https://www.youtube.com/watch?v=M3EFi_POhps
  - Executing an attack is expensive, you need more Bitcoins
  - Reduced incentive for attack – the attacker needs to own a majority of bitcoins – an attack will have more affect on the attacker

- Variants of “stake”
  - Randomization in combination of the stake (*used in Nxt and BlackCoin*)
  - Coin-age: Number of coins multiplied by the number of days the coins have been held (*used in Peercoin*)

Proof of Burn (PoB)

- Miners should show proof that they have *burned* some coins
  i.e. no one will be able to spend that coin
  - Sent them to a verifiably un-spendable address
  - Expensive just like PoW, but no external resources are used other than the burned coins

- PoW vs PoB – Real resource vs virtual/digital resource

- PoB works by burning PoW mined cryptocurrencies

PoW vs PoS vs PoB

PoW

- Do some work to mine a new block
- Consumes physical resources, like CPU power and time
- Power hungry

PoS

- Acquire sufficient stake to mine a new block
- Consumes no external resource, but participate in transactions
- Power efficient

PoB

- Burn some wealth to mine a new block
- Consumes virtual or digital resources, like the coins
- Power efficient

Proof of Elapsed Time (PoET)

- Proposed by Intel, as a part of Hyperledger Sawtooth – a blockchain platform for building distributed ledger applications

- Basic idea:
  - Each participant in the blockchain network waits a random amount of time
  - The first participant to finish becomes the leader for the new block

PoET over Trusted Environments

- How will one verify that the proposer has **really waited** for a **random amount of time**?
  - Utilize special CPU instruction set – *Intel Software Guard Extension* (SGX) – a trusted execution platform
  - The trusted code is private to the rest of the application
  - The specialized hardware provides an attestation that the trusted code has been set up correctly

i.e. complete hardware control

Mining Difficulty

- The difficulty changes for every 2016 blocks
  - Desired rate – one block each 10 minutes
  - Two weeks to generate 2016 blocks

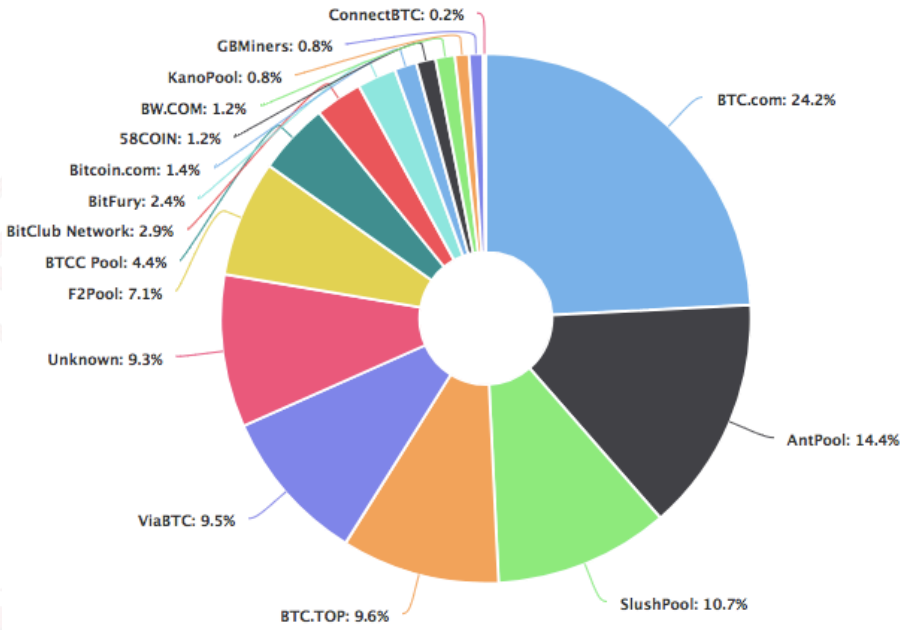
Setting the Difficulty

- Compute the following for every two weeks

```
current_difficulty = previous_difficulty *
      (2 weeks in milliseconds)/(milliseconds to
                                 mine last 2016 blocks)
```

Mining Pool

- Pooling of resources by the miners
  - Share the processing power over a network to mine a new block
  - Split the reward proportionally to the amount of work they contributed



Hash-rate Distribution: blockchain.info

Mining Pool Methods

- Contains hundreds or thousands of miners through special protocols
- B: Block reward minus pool fee
- p: Probability of finding a block in a share attempt ($p = 1/D$), D is the block difficulty

Mining Pool Methods

- Pay per Share (PPS)
  - Instant guaranteed payout to a miner
  - Miners are paid from pool's existing balance, share of a miner is $R = B \times p$
  - Miners get almost equal payment, risk is at the pool operator

Mining Pool Methods

- Proportional
  - Miners earn share until the pool finds a block (end of mining round)
  - $R = B \times \frac{n}{N}$, where n is amount of his own share, and N is amount of all shares in the round
  - Payments are made once a pool finds out a block

Mining Pools – Pros and Cons

- Pros
  - Small miners can participate
  - Predictable mining
- Cons
  - Leads to centralization
  - Discourages miners for running complete mining procedure

Permissioned Model

- A blockchain architecture where users are authenticated apriory
- Users know each other
- However, users may not trust each other – Security and consensus are still required.
- Run blockchain among known and identified participants

Design Limitations

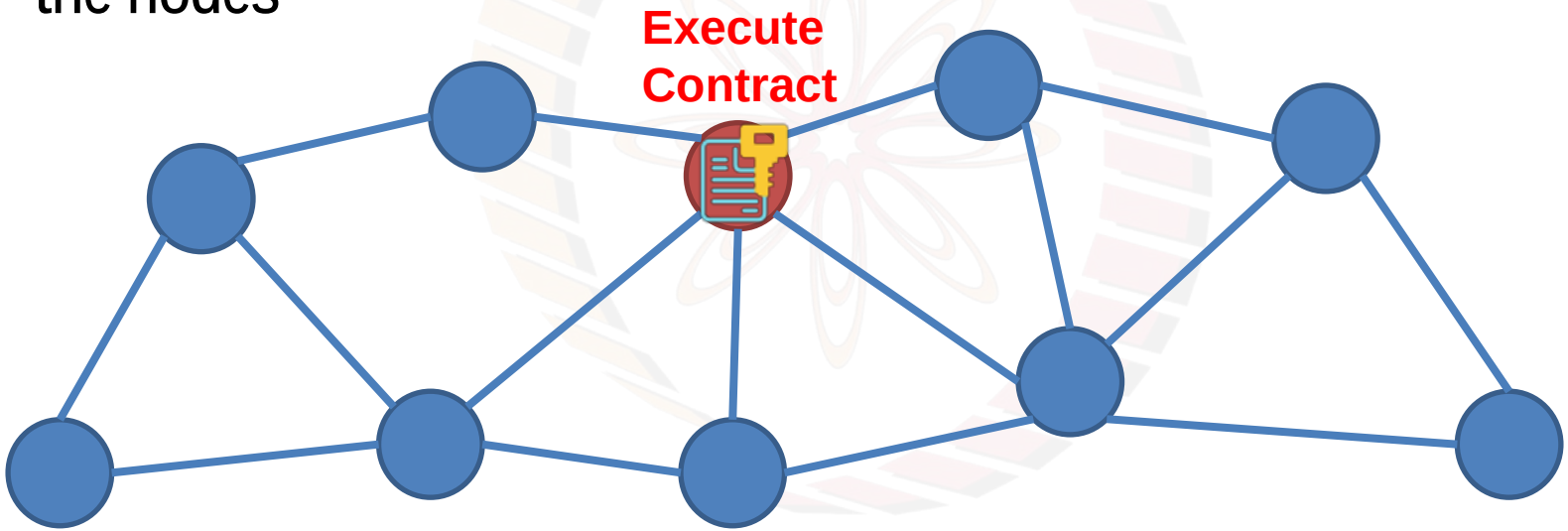
- Sequential Execution
  - Execute transactions sequentially based on consensus
  - Requests to the application (smart contract) are ordered by the consensus, and executed in the same order
  - This give a bound on the effective throughput – throughput is inversely proportional
  - Can be a possible attack on the smart contract platform – introduce contract which will take long time to execute

Design Limitations

- Non-deterministic Execution
  - Smart-contract execution should always needs to be deterministic; otherwise the system may lead to inconsistent states (many fork in the blockchain)
  - Solution: Domain specific language (DSL) for smart contract

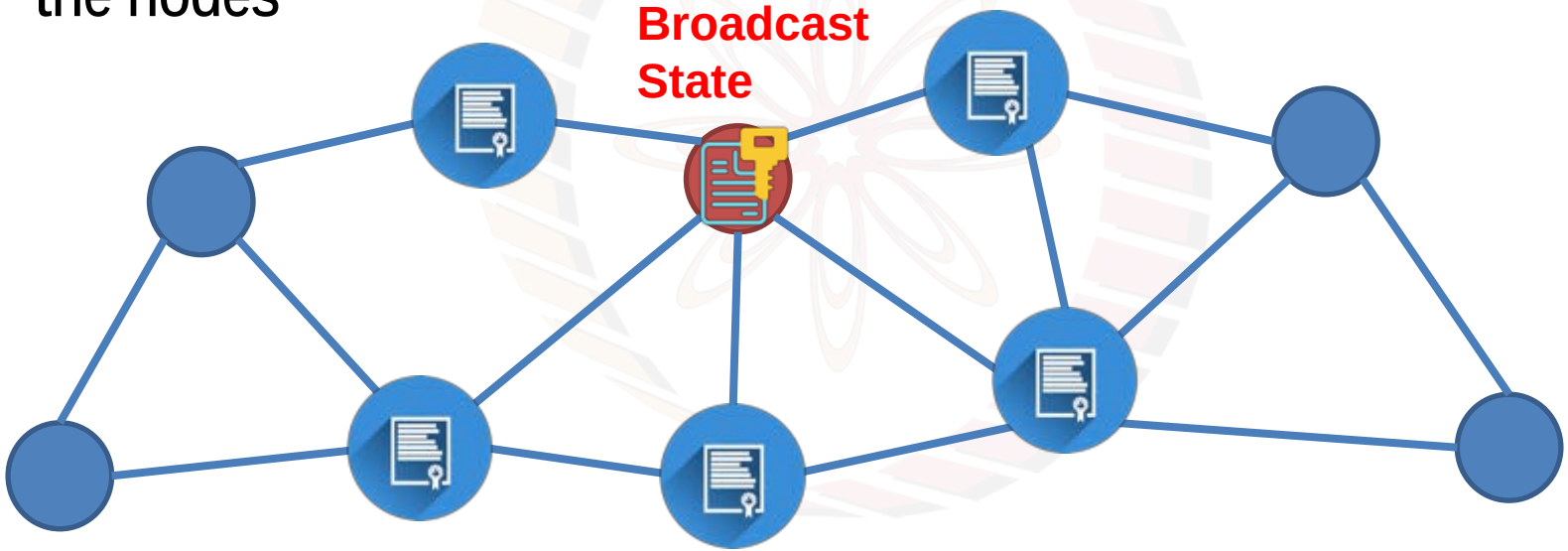
Do We Really Need to Execute Contracts at Each Node

- Not necessary always, we just need state synchronization across all the nodes



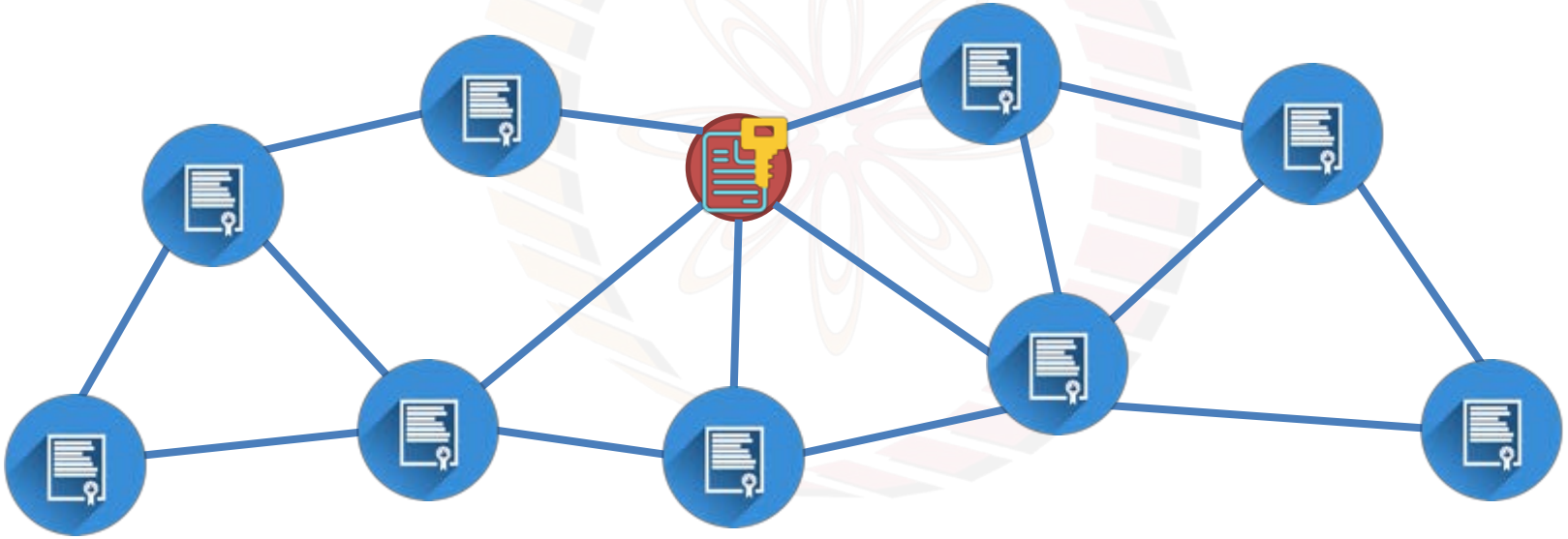
Do We Really Need to Execute Contracts at Each Node

- Not necessary always, we just need state synchronization across all the nodes



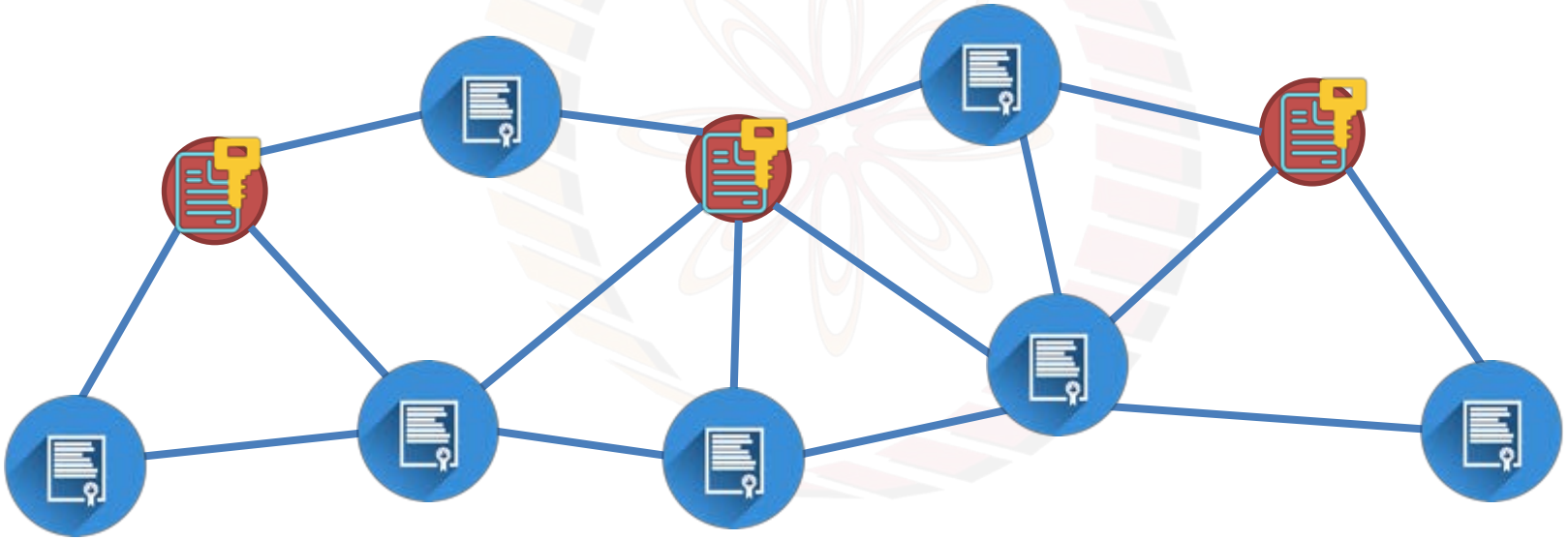
Do We Really Need to Execute Contracts at Each Node

- What if the node that executes the contract is faulty?

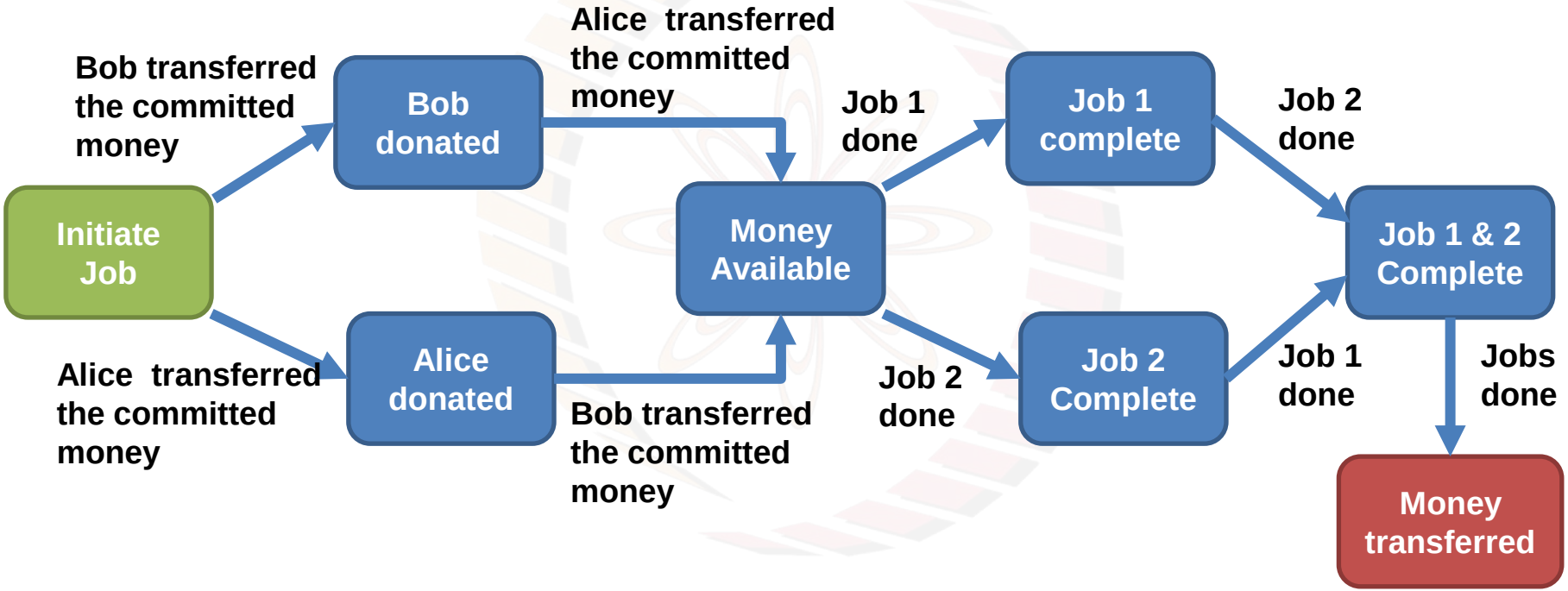


Do We Really Need to Execute Contracts at Each Node

- Use state machine replication – execute contract at a subset of nodes, and ensure that the same state is propagated to all the nodes

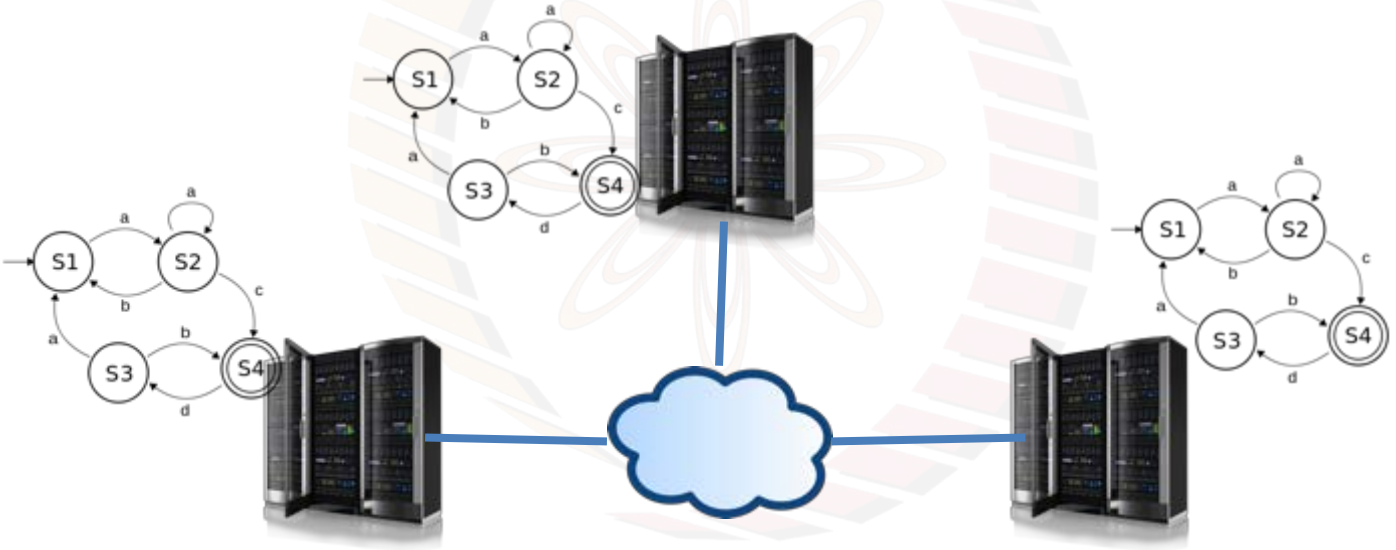


Smart Contract State Machine - Crowd-Funding



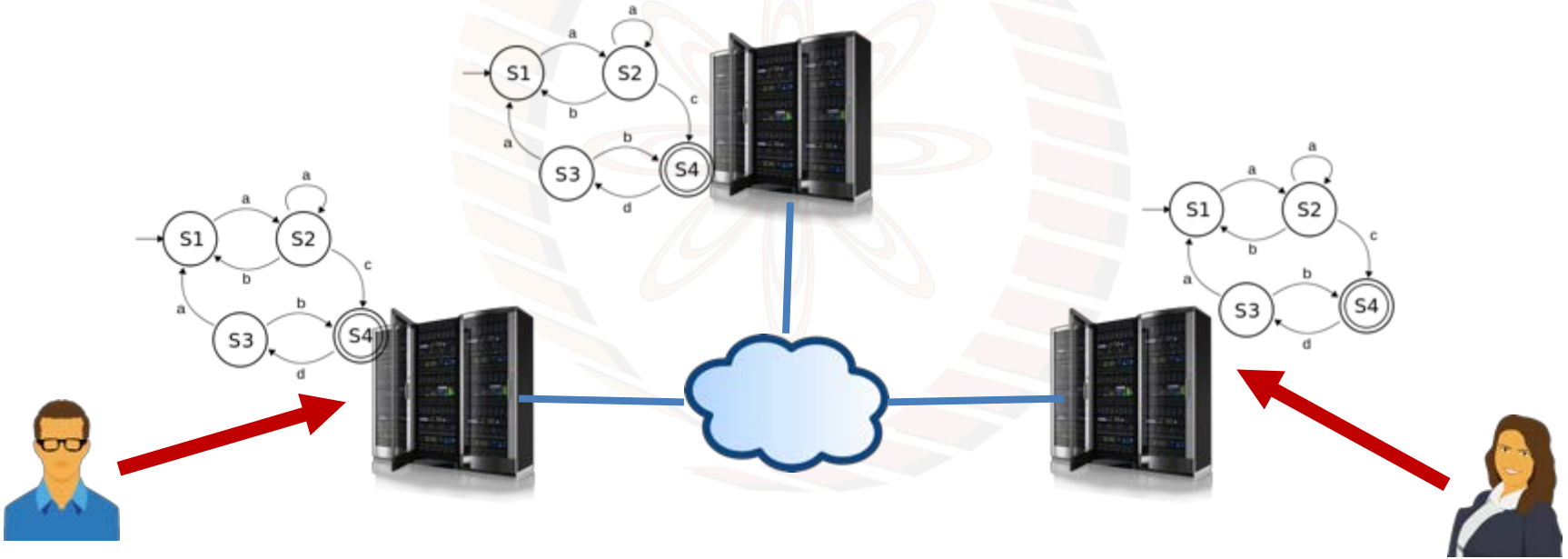
Distributed State Machine Replication

1. Place copies of the state machine on multiple independent servers



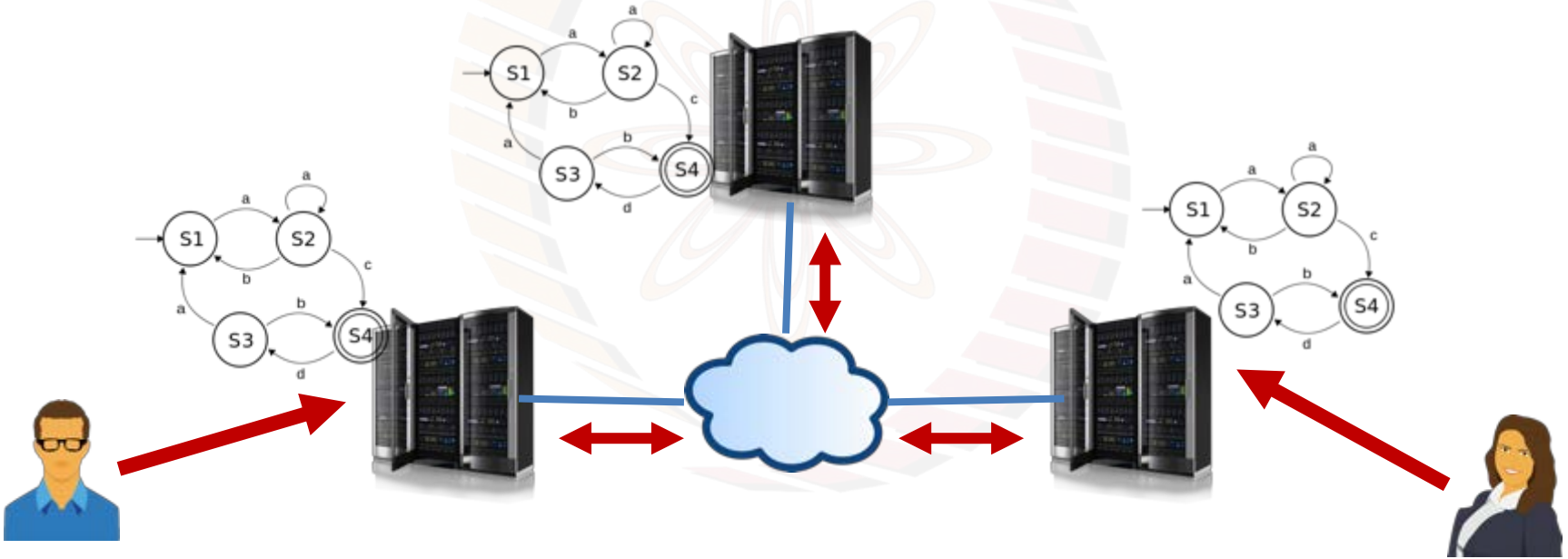
Distributed State Machine Replication

2. Receive client requests, as an input to the state machine



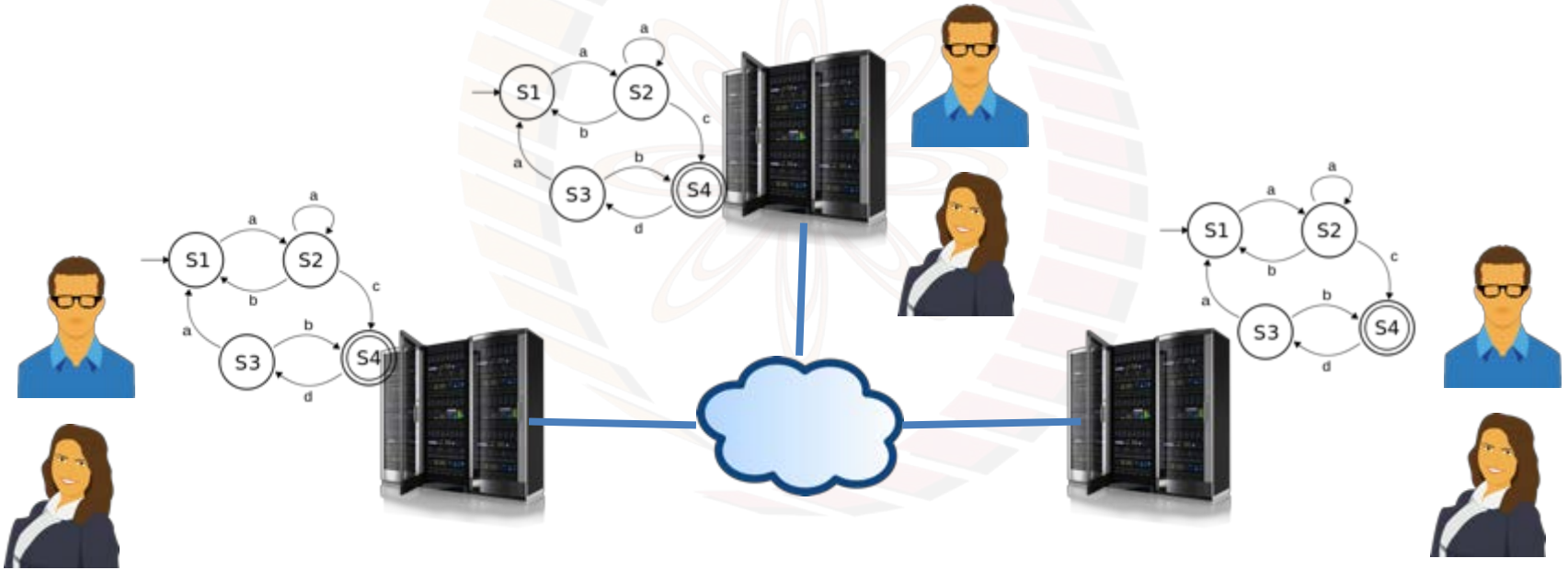
Distributed State Machine Replication

3. Propagate the inputs to all the servers



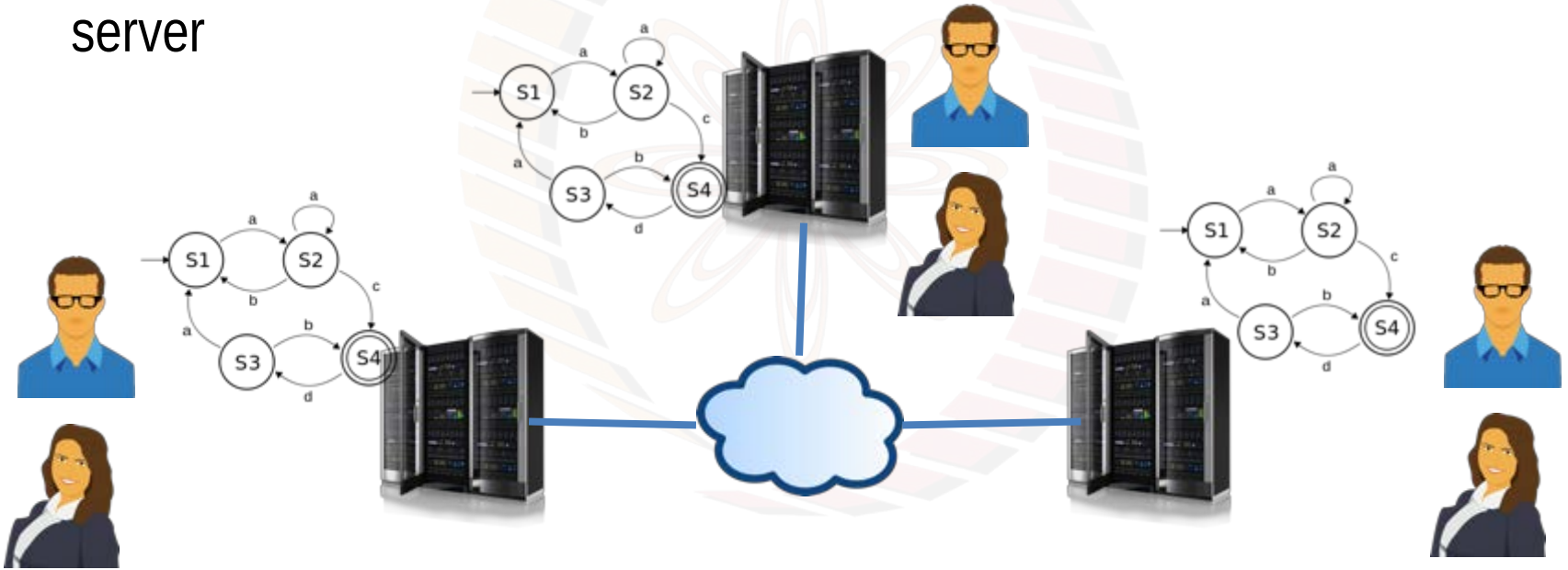
Distributed State Machine Replication

4. Order the inputs based on some ordering algorithm



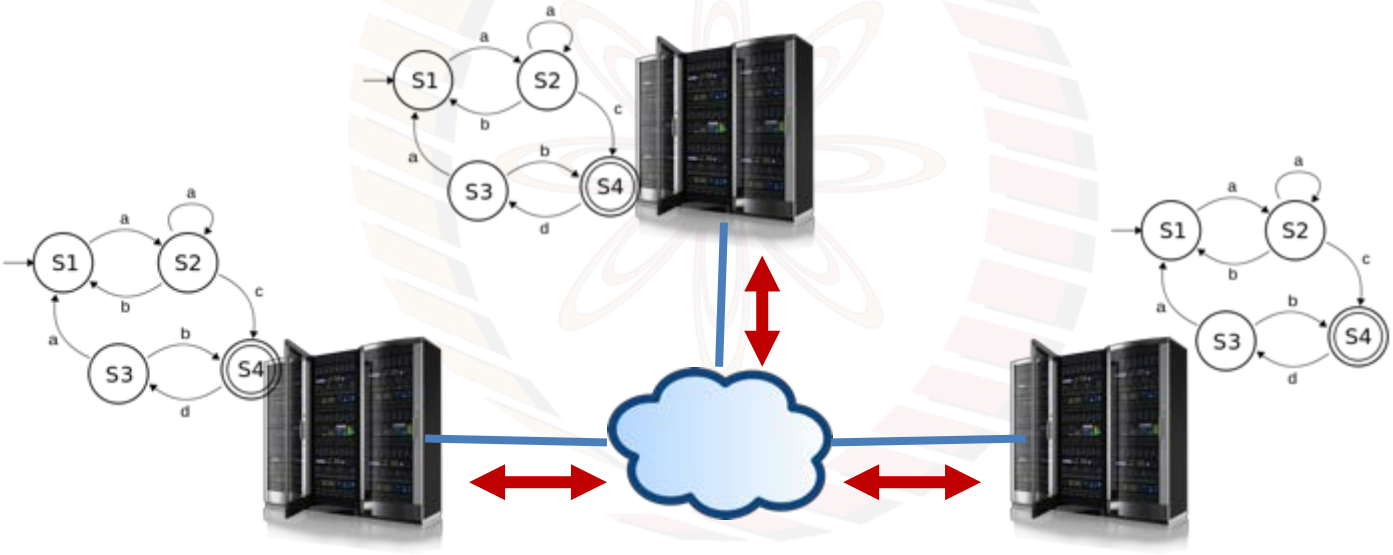
Distributed State Machine Replication

4. Execute the inputs based on the order decided, individually at each server



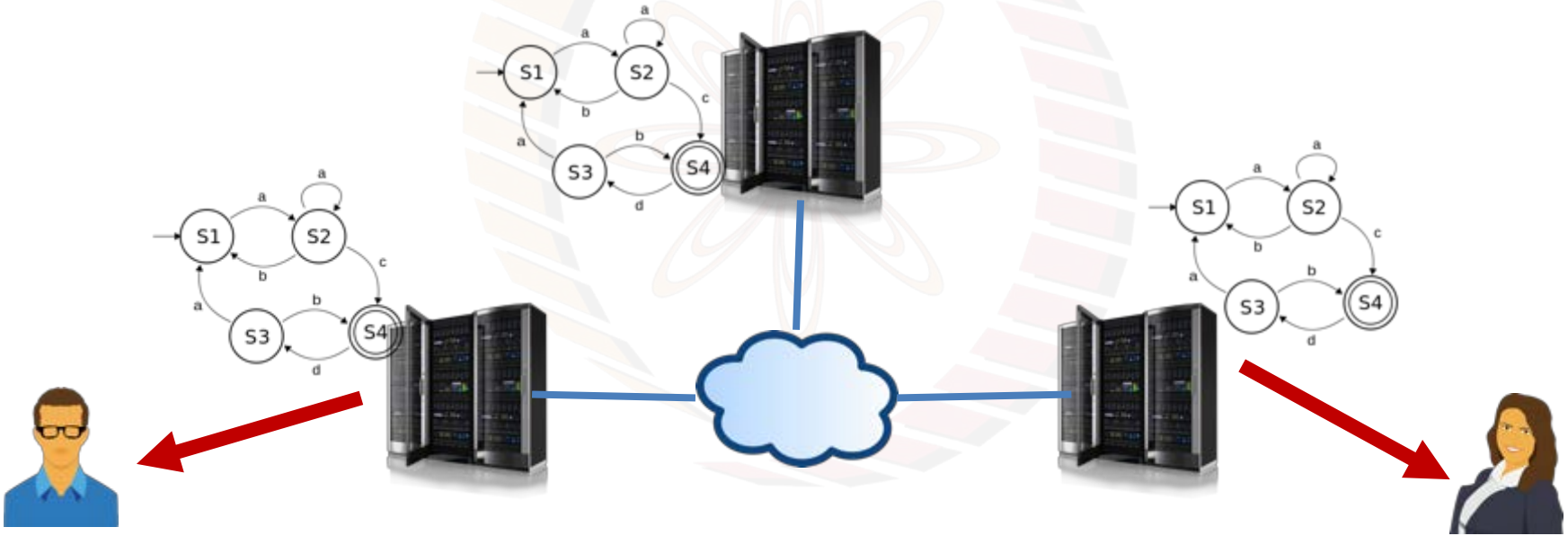
Distributed State Machine Replication

5. Sync the state machines across the servers, to avoid any failure.



Distributed State Machine Replication

6. If output state is produced, inform the clients about the output



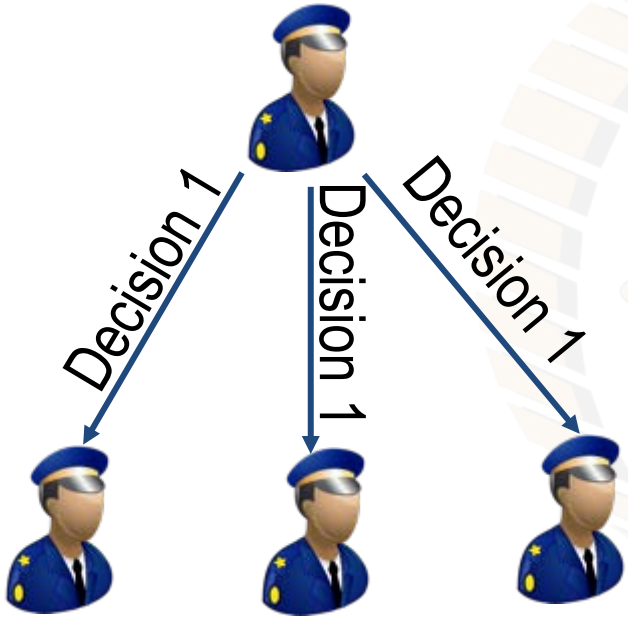
But Consensus is still required.

Why Distributed Consensus

One Decision Maker

No Consensus

Why Distributed Consensus



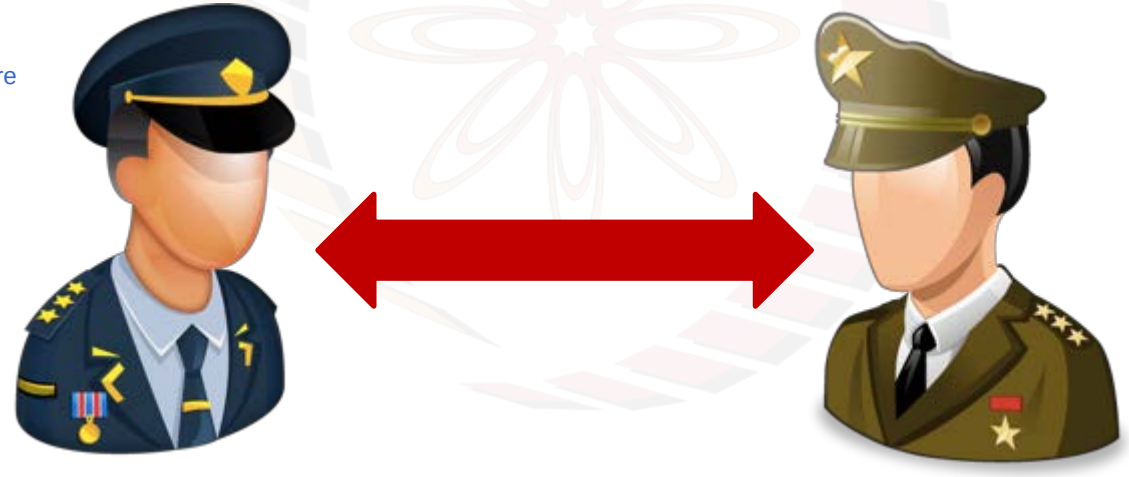
Multiple Decision Maker

Consensus Required

Why Distributed Consensus

- So, no need of consensus in a single node process.
- What about when there are two nodes?
  - Network or partitioned fault, consensus cannot be reached

So we always need more than two nodes

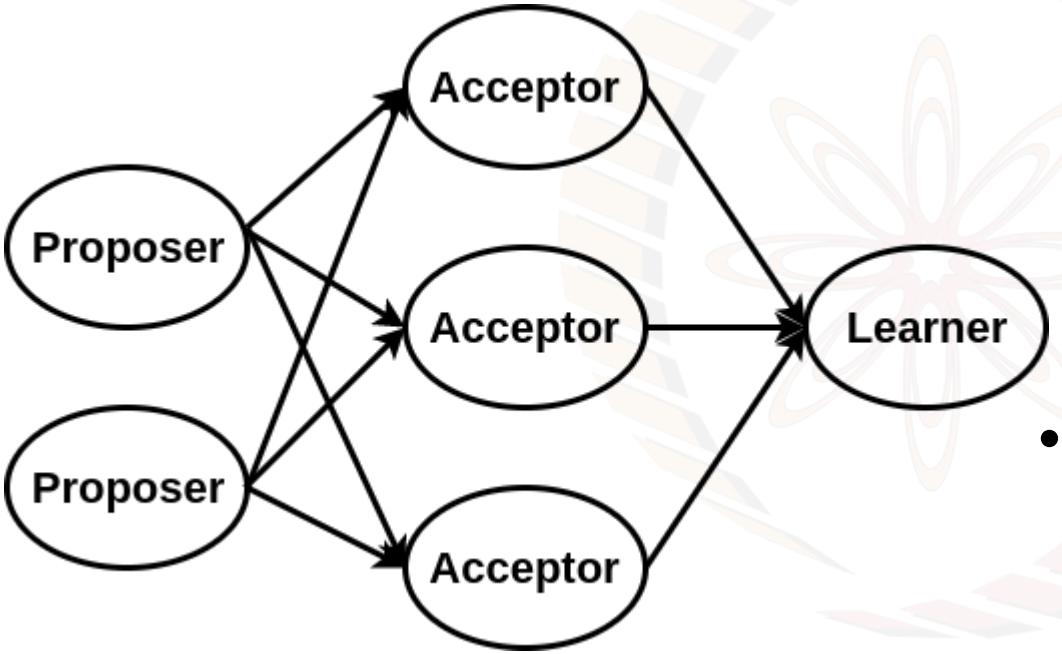


Different Algorithms

- Crash or Network Faults:
  - PAXOS
  - RAFT

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

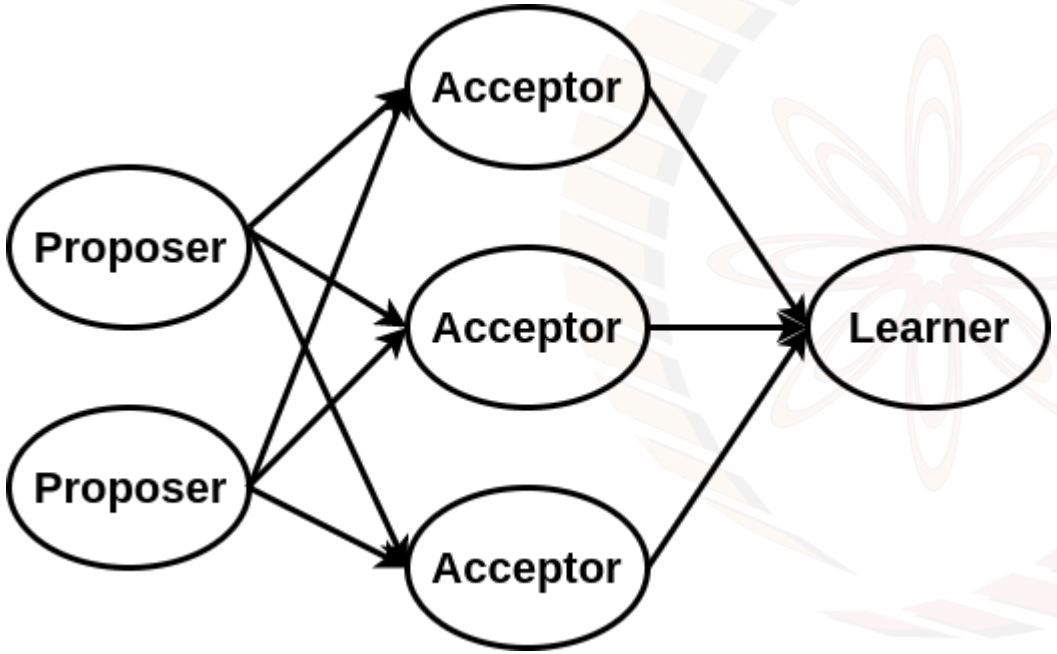
PAXOS



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

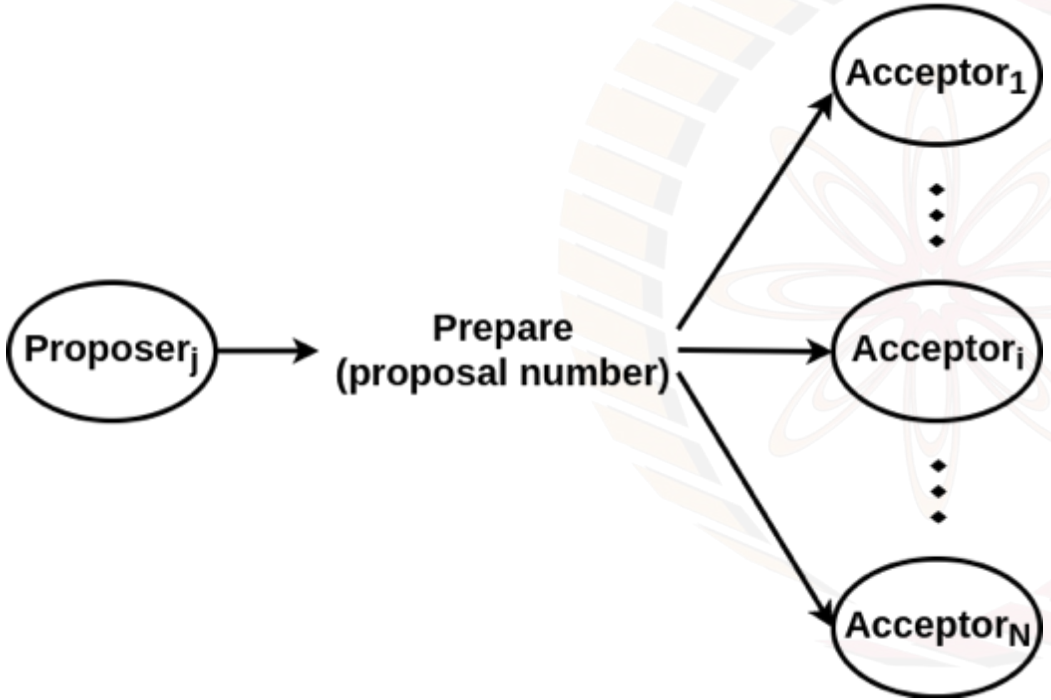
- System process
  - Making a proposal
  - Accepting a value
  - Handling Failures

PAXOS: Types of Nodes



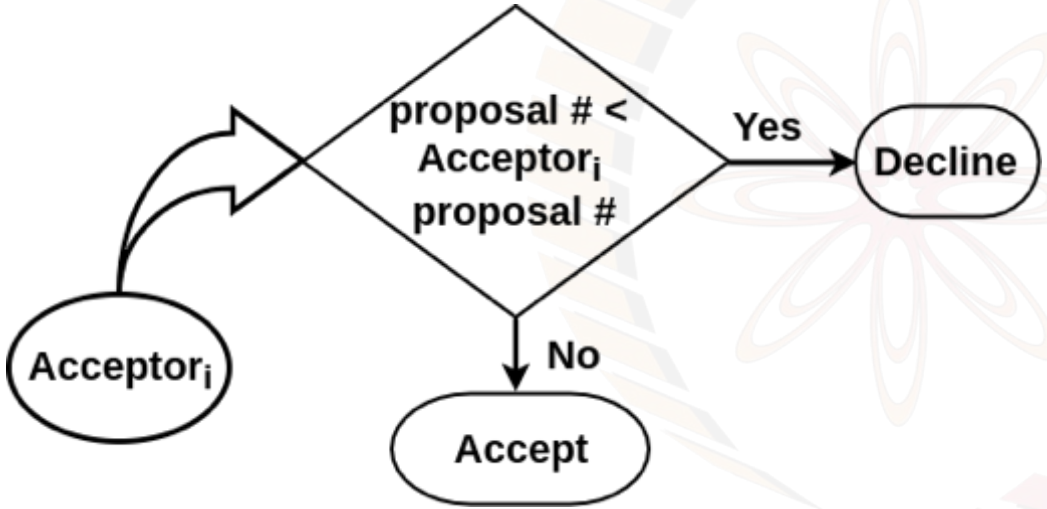
- **Proposer:** propose values that should be chosen by the consensus
- **Acceptor:** form the consensus and accept values
- **Learner:** learn which value was chosen by each acceptor

Making a Proposal: Proposer Process



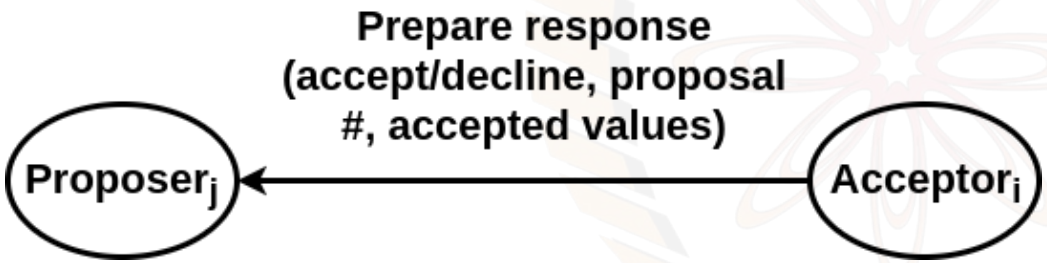
- **proposal number:** form a timeline, biggest number considered up-to-date

Making a Proposal: Acceptor's Decision Making



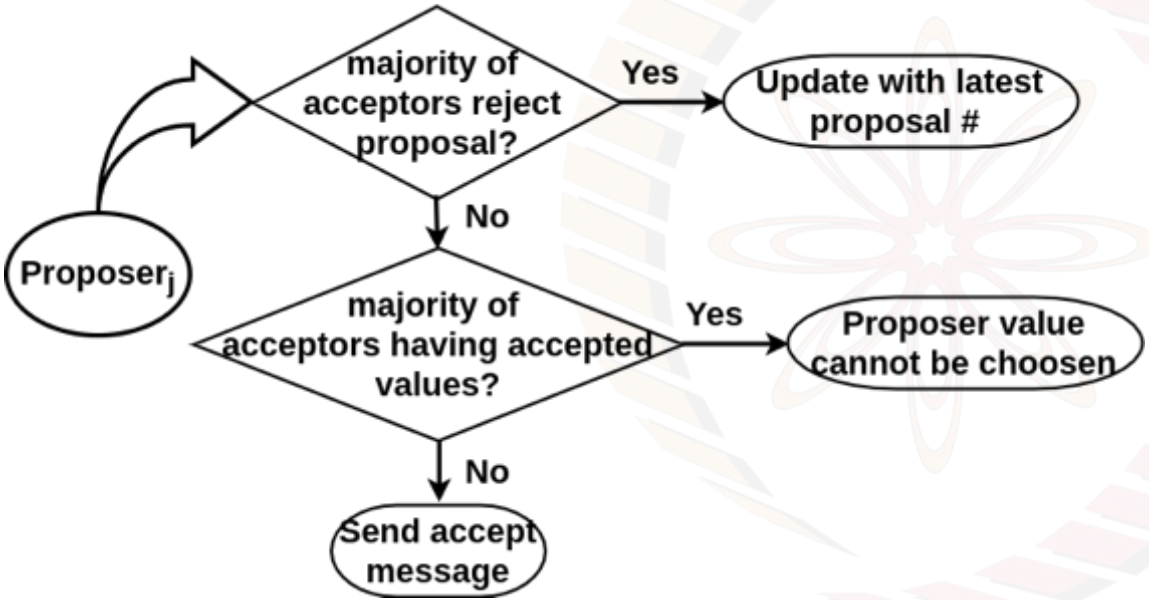
- Each acceptor compares received proposal number with the current known values for all proposer's prepare message

Making a Proposal: Acceptor's Message



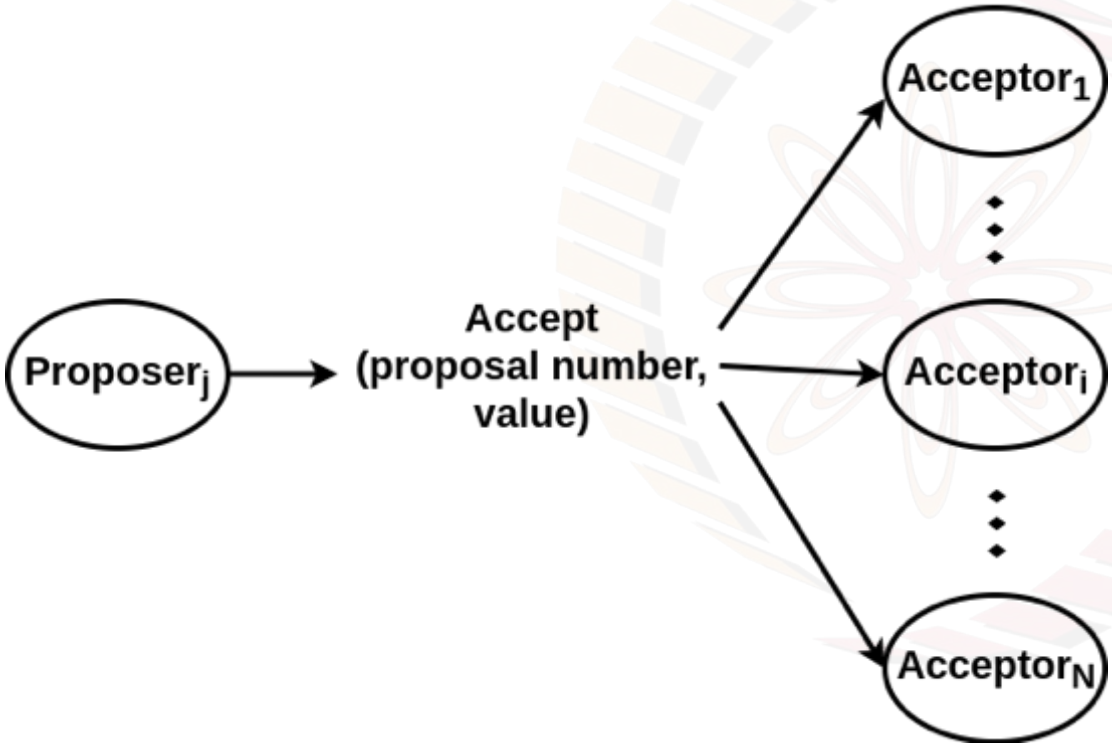
- **accept/decline:** whether prepare accepted or not
- **proposal number:** biggest number the acceptor has seen
- **accepted values:** already accepted values from other proposer

Accepting a Value: Proposer's Decision Making



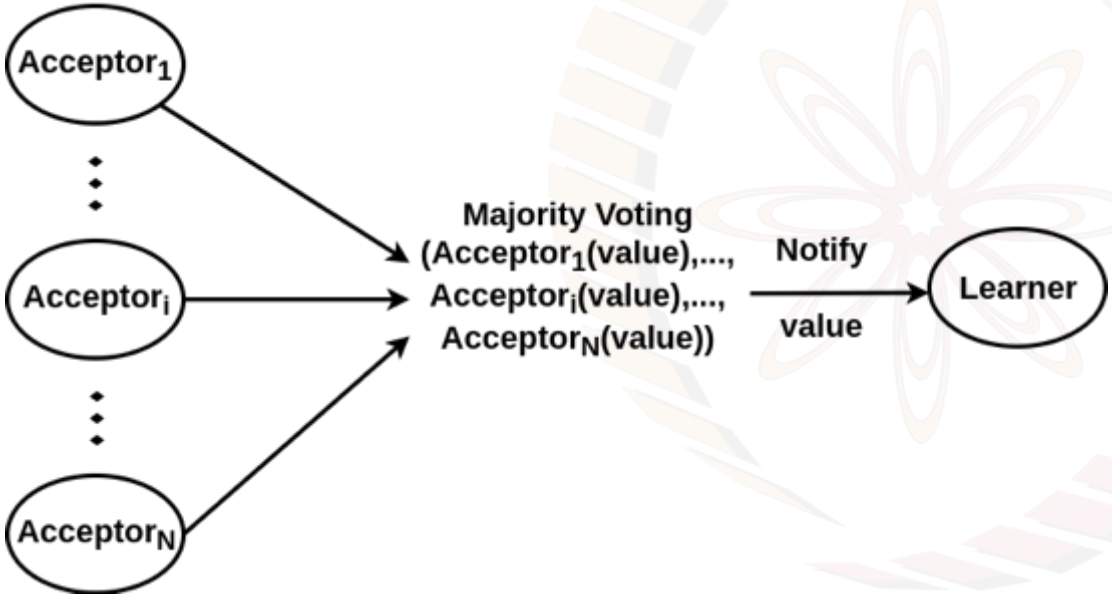
- Proposer receive a response from **majority** of acceptors before proceeding

Accepting a Value: Accept Message



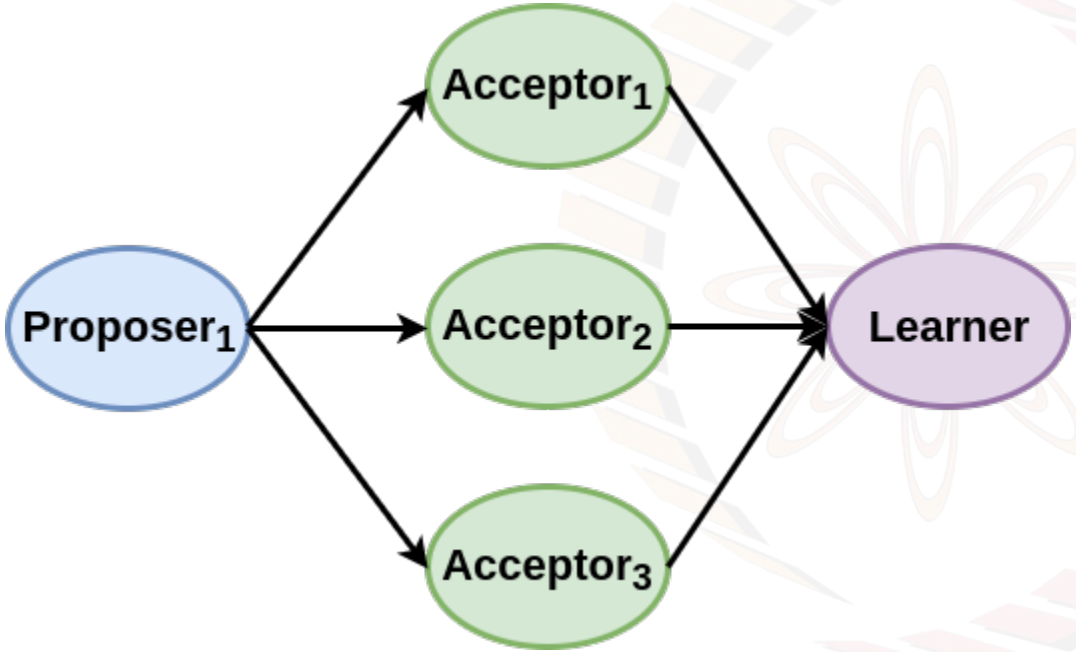
- proposal number: same as prepare phase value
- value: single value proposed by proposer

Accepting a Value: Notifying Learner



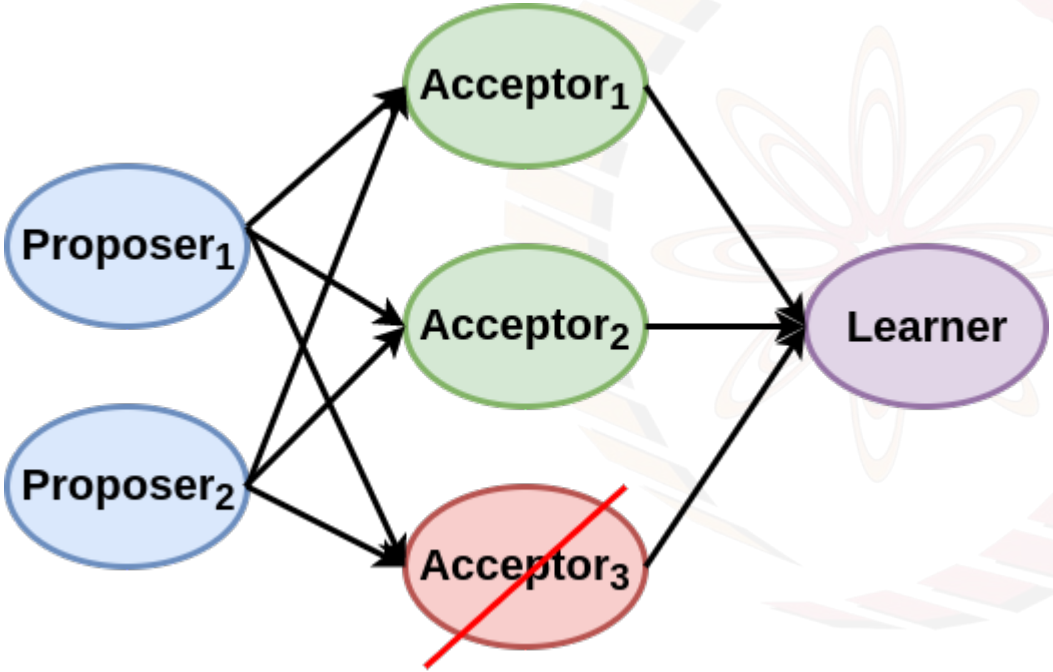
- Each acceptor accept value from any of the proposer
- Notify learner the majority voted value

Single Proposer: No Rejection



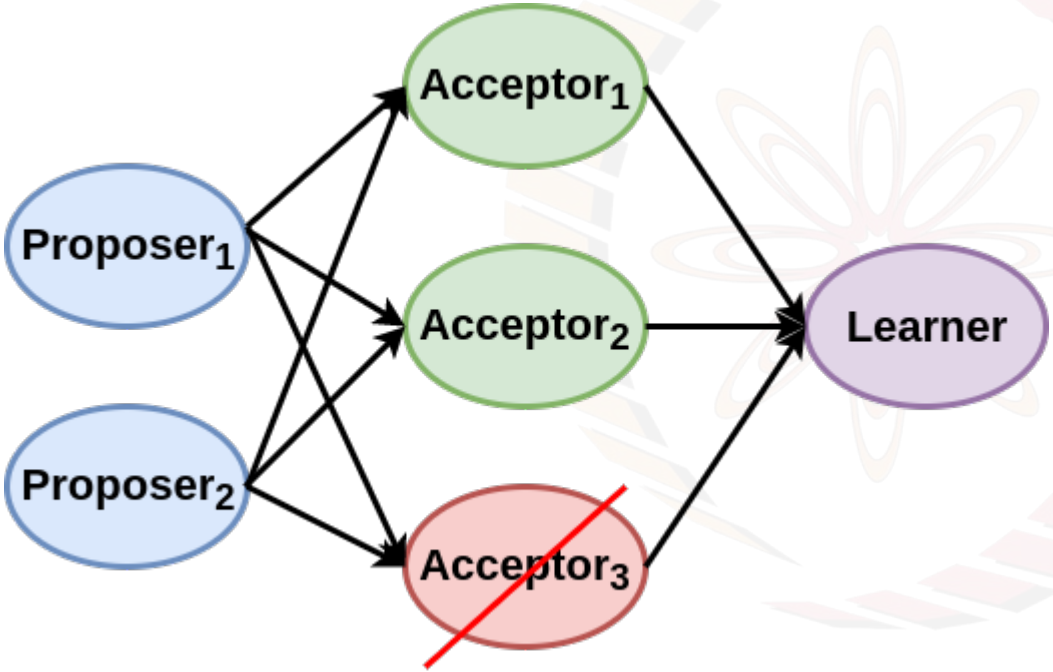
- Proposer always have proposal with biggest number
- No proposal rejected

Handling Failure: Acceptor Failure



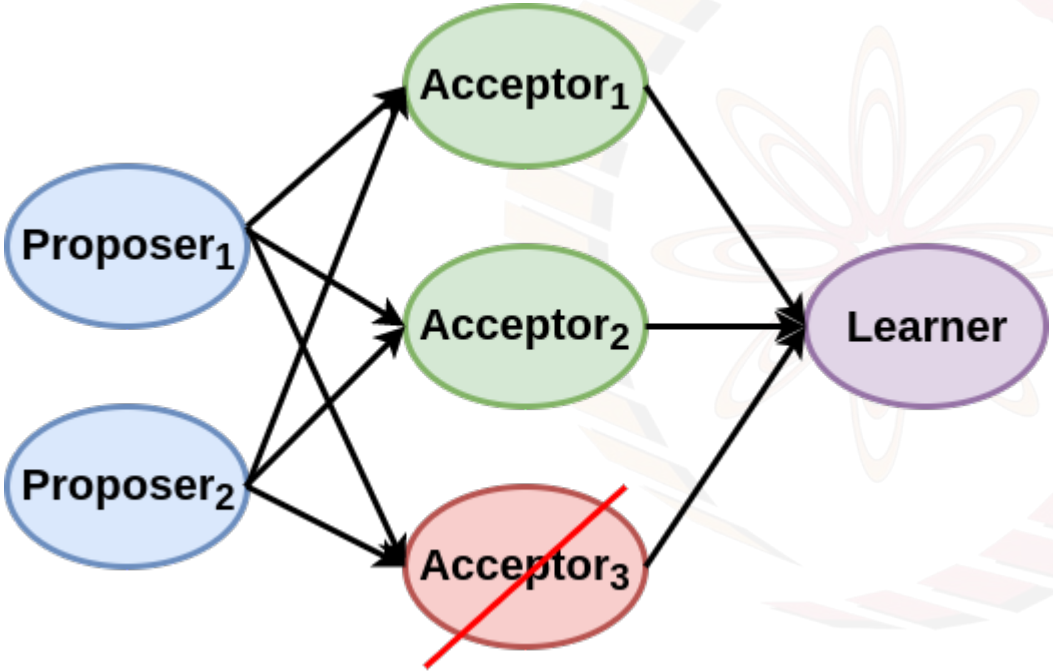
- Acceptor fails during prepare
  - No issues, other acceptor can hear the proposal and vote

Handling Failure: Acceptor Failure



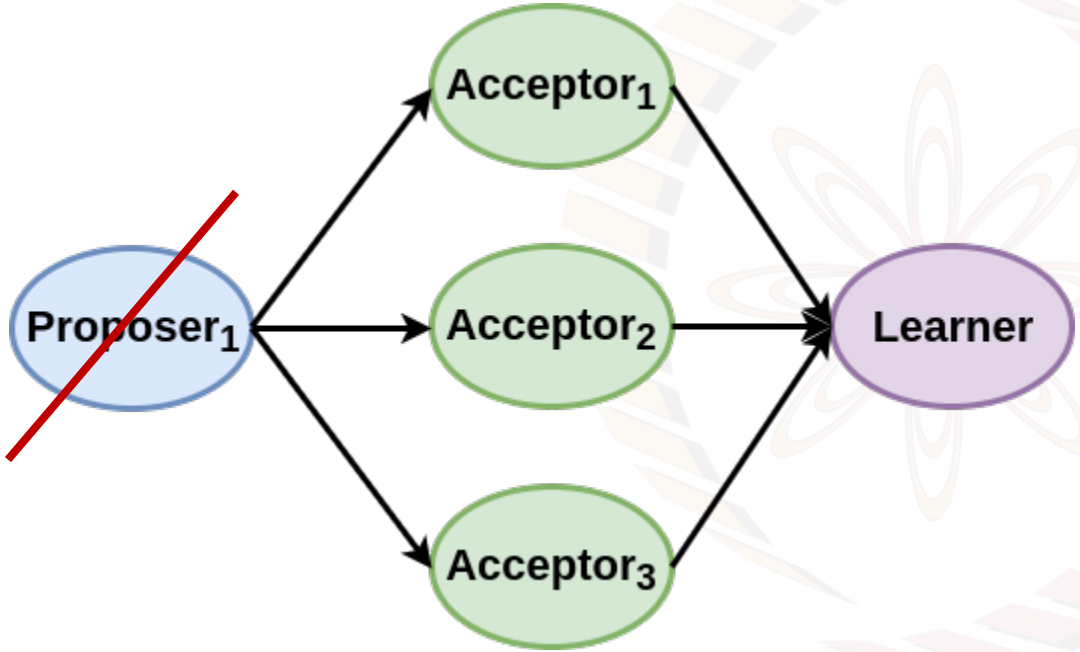
- Acceptor fails during accept
  - Again, no issues, other acceptor can vote for the proposal

Handling Failure: Acceptor Failure



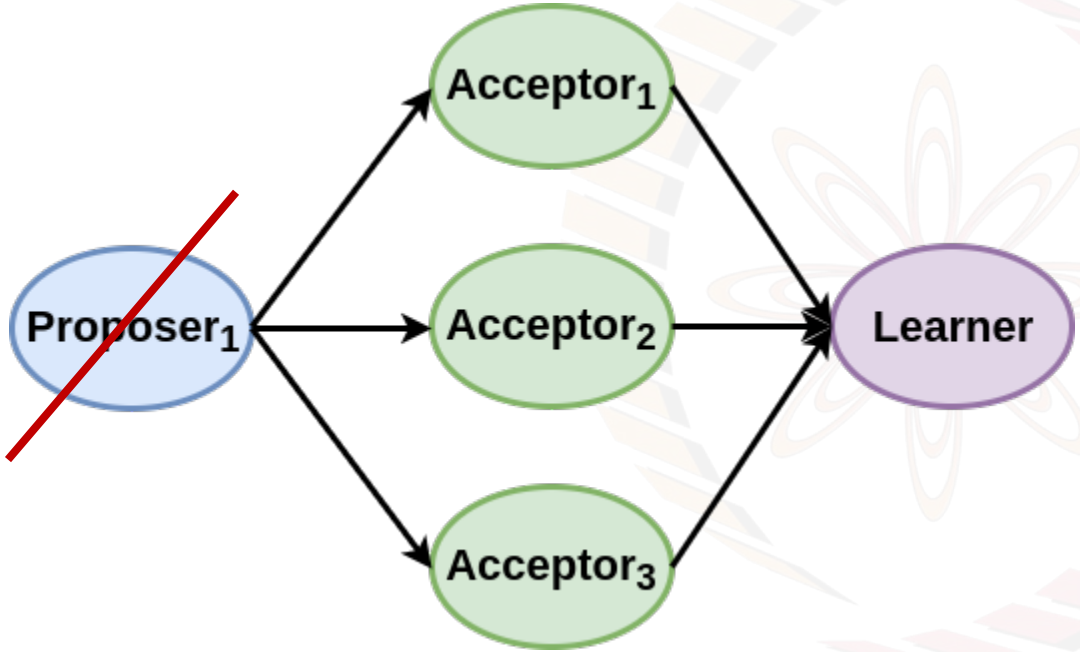
- More than N/2 - 1 acceptors fail
  - no proposer get a reply
  - no values can be accepted

Handling Failure: Proposer Failure



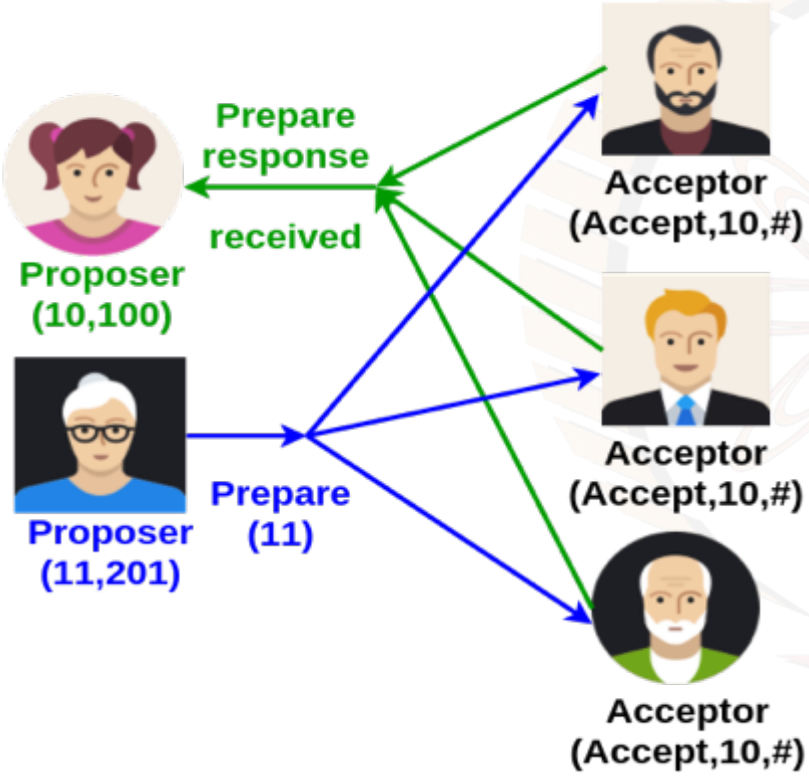
- Proposer fails during prepare phase
  - Acceptors wait, wait, wait, and then someone else become the proposer

Handling Failure: Proposer Failure



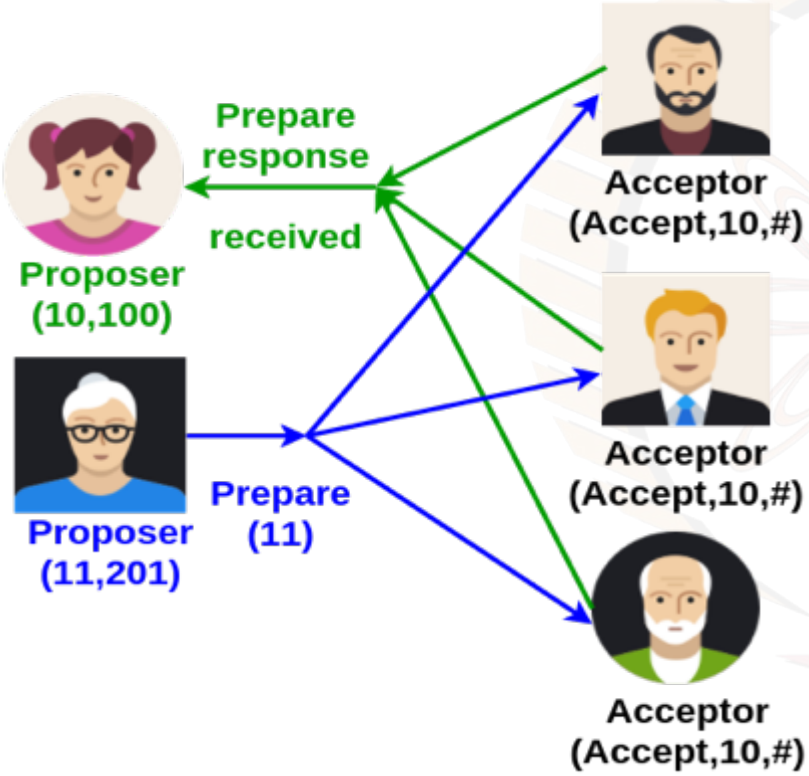
- Proposer fails during accept phase
  - Acceptors have already agreed upon whether to choose or not to choose the proposal

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

Handling Failure: Dueling Proposers



- Use **leader election** - select one of the proposer as leader
- Paxos can be used for leader election !!