Unit 3

Index

Consensus

It is a procedure to reach a common agreement on a distributed decentralised environment.

The reason for using a consensus algorithm is as follows:

- **Agreement**- every correct individual agrees on the same value
- **Termination** - when the algorithm ends, all valid individuals decide on a value
- **Validity** - If all individuals propose the same value, all correct individuals decide on that value.
- **Fault tolerant** - the network should work in presence of faults
- **Integrity** - Every correct individual has at most one value.

Proof of Work

This was proposed way before the white paper.

It works on the principle that the solution us hard to find but easy to verify. Each node needs to solve a math problem to find the solution i.e. propose the transaction. The math problem is finding the hash value based on the previous block's hash and the hash value of the input message. The solution will be less than the difficulty of the blockchain i.e. size less than size of difficulty.

The proposed value is called a nonce. This is then verified throughout the chain. The nonce should then be validated by a majority usually 51% but can change based on the chain.

Advantages	Disadvantages
Tested from a long time so is known to be reliable.	Slow and gets more costly as more people join.
More guaranteed. We know there will be no rollback.	51% risk. Here we have to be faster than 51%.
It is trustless. No one can block your transaction from processing.	Can indirectly lead to centralization. The people with the most resources can pool together.

Proof of Stake

A random node is chosen and the block they proposed are checked. They place a stake on it and are rewarded if they are correct. Small win and Big loss.

Pros	Cons
Energy Efficient	Rewards are better the longer you stay
Scales better	Can centralise through wealth
Harder to attack	51% risk. We need to control 51% of the network.

Delegated proof of stake

Here we vote for a bunch of people who in turn vote on the block. This increases centralisation a lot.

Proof of Authority

This is where a selected set of nodes will authorise. This is best for organisations using a private blockchain. This can also be used as a test environment before launching into the public blockchain.

In this anonymity is not a thing. We use real identities and must either invest or follow some standard method to be approved to become a validator. Only validators can make nodes.

We can use this in a supply chain type situation.

Pros	Cons
Energy efficient and fast	Centralised
High risk tolerance to 51% as it is reputation and identity based.	The identities are known. This can lead to IRL manipulation.
Predictable rate of new block generation.	

Proof of Elapsed Time

Everyone is given a random amount of time to wait. After this time they are allowed to add a block to the ledger. This is a permissioned blockchain so verification is all other parts. This is Software Guard Extension by Intel (SGX). The user gets a special key pair from SGX. They use this to get into the blockchain. After this they will start waiting.

Pros	Cons
Low upkeep	Need specialized hardware
Scalable	Identities required.

Proof of Space

This is also called proof of capacity. Her we generate all possible hashes beforehand and store on a hard drive. This uses Shabal cryptographic algorithm i.e. SHA-256. Ex: BurstCoin.

They produce nonces which have 8192 hashes. Each of these hashes are grouped into pairs. Each of these pairs are called scoops. Deadline is the minimum amount of time after the last block was added. After deadline only we can forge the next block.

Steps:

1. Compute scoop number
2. Use scoop to calculate deadline values
3. From all the deadlines choose the lowest deadline value
4. From all the nodes the lowest deadline will be the opportunity to add their block.

Pros	Cons
30 times more efficient and 2 times as fast as PoW.	We need a large storage space.
Can use regular everyday hardware.	Hard to detect malicious hard drive usage.
Drives can be used in other things	Not adopted commonly.

Proof of Burn

Proof of Burn is used by Slimcoin, TGCoin and Factom. We promote periodic burning of coins to promote equality between old and now nodes.

We spend a set of coins to an unspendable escrow account. The person who sends the most wins and can add their block. This promotes constant transactions.

Pros	Cons
Less power needed. Cheap and easy to maintain.	Needs wealthy participants to start.
Motivates more spending.	Not proven to work on a large network
Scalable and used for long term stuff.	Slower

Proof of Concept

lol this is just like prototyping

Fork

This is when there are 2 potential paths the blockchain can go in. This can be because there are some changes in the blockchain. This can be things like adding new features, fixing bugs or a security breach.

There are multiple types of forks:

Codebase Fork

This is like a git fork. This is where we take all the code and change it to our needs. Usually we start the new blockchain with an empty ledger.

Live Blockchain Fork

This is the one where the live one is split into 2 things. This has various types too.

Accidental/ Temporary

This is where 2 miners mine a new block at the same time and some set of nodes accept these blocks. This can happen because of time delays.

Intentional Fork

This is the one where we introduce something new.

Soft Fork

This is where we change the chain in away that is backwards compatible with the old chain. Here the existing rules are tightened.

Hard Fork

This is when it is not backwards compatible. This is when the existing rules are loosened. New rules are put. The nodes running the old software will not consider the new nodes as valid.

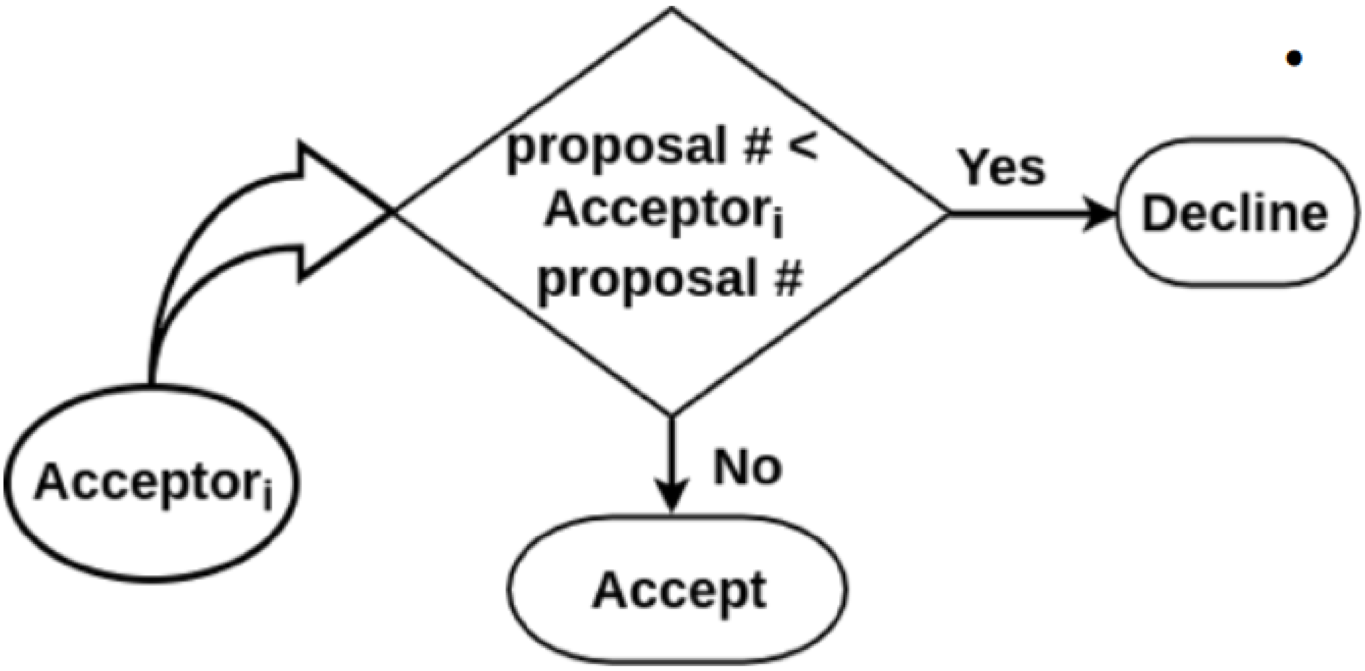
PAXOS

This is the older consensus algorithm.

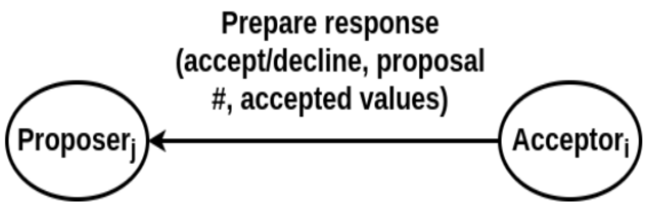
3 nodes in PAXOS:

- Proposer: -_-
- Acceptor: Voter
- Learner: Learns based on acceptors

Here we make a timeline for proposals. The largest number is considered the latest.



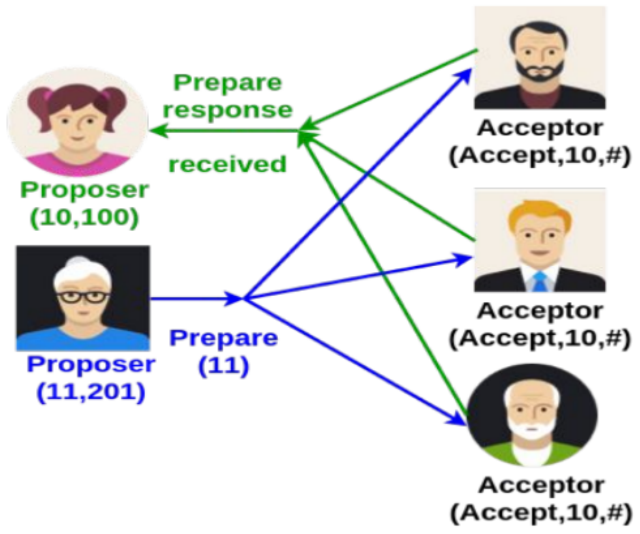
Making a Proposal: Acceptor's Message



Prepare response (accept/decline, proposal #, accepted values)
Proposer$_j$
Acceptor$_i$

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

Handling Failure: Dueling Proposers



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

Here before 11 could be proposed the accept for 10 was already sent. So 10 is chosen.

RAFT

The leader proposes the new blocks and the followers vote.

Steps in the algorithm:

1. Election timeout: Followers wait for leader communication.
2. Candidacy declaration: Timeout triggers a follower to become a candidate.
3. Vote granting: Candidates request votes from other nodes.
4. Majority vote: If a candidate gets a majority of votes, it becomes the leader.
5. Heartbeats and leader confirmation: The leader sends periodic heartbeats to maintain authority.
6. Handling split votes: If no majority is reached, a new election is initiated.
7. Handling leader failure: Nodes transition to candidates if no heartbeat is received.

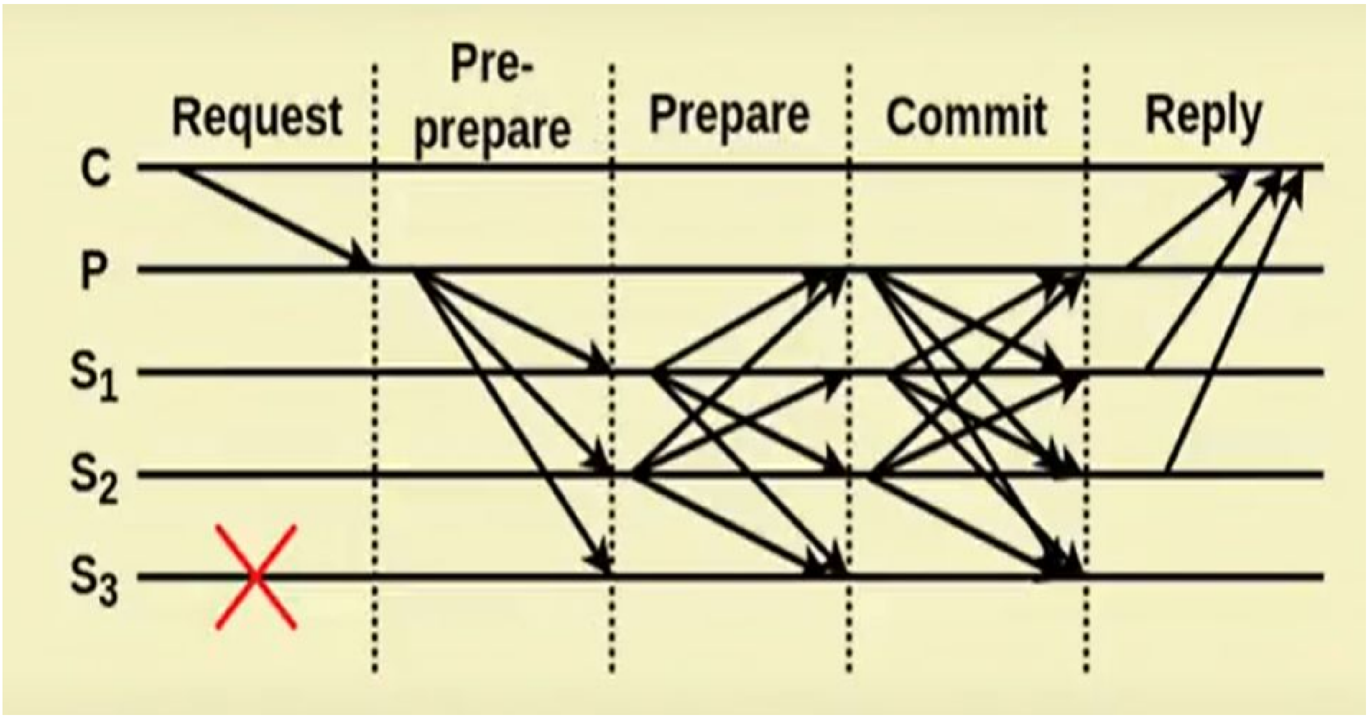
PBFT

Practical Byzantine Fault Tolerance

Client sends request to leader.

Leader sends to the secondary/ backups

If we receive more than m+1 same replies then choose that.



Bad scaling and only one thing controls all.

When a system is violated by an entity by creating and using many false network identities to flood the network and crash the system. This is Sybil attack.