Consensus Algorithms For Blockchains

Prof. Andrea Bondavalli

Index

- Introduction
- Algorithms
  - PoW (Proof of Work), EtHash (variant of PoW)
  - PoS (Proof of Stake), PoX (Proof of X), PoET
  - PBFT and variants
- Some Comparison

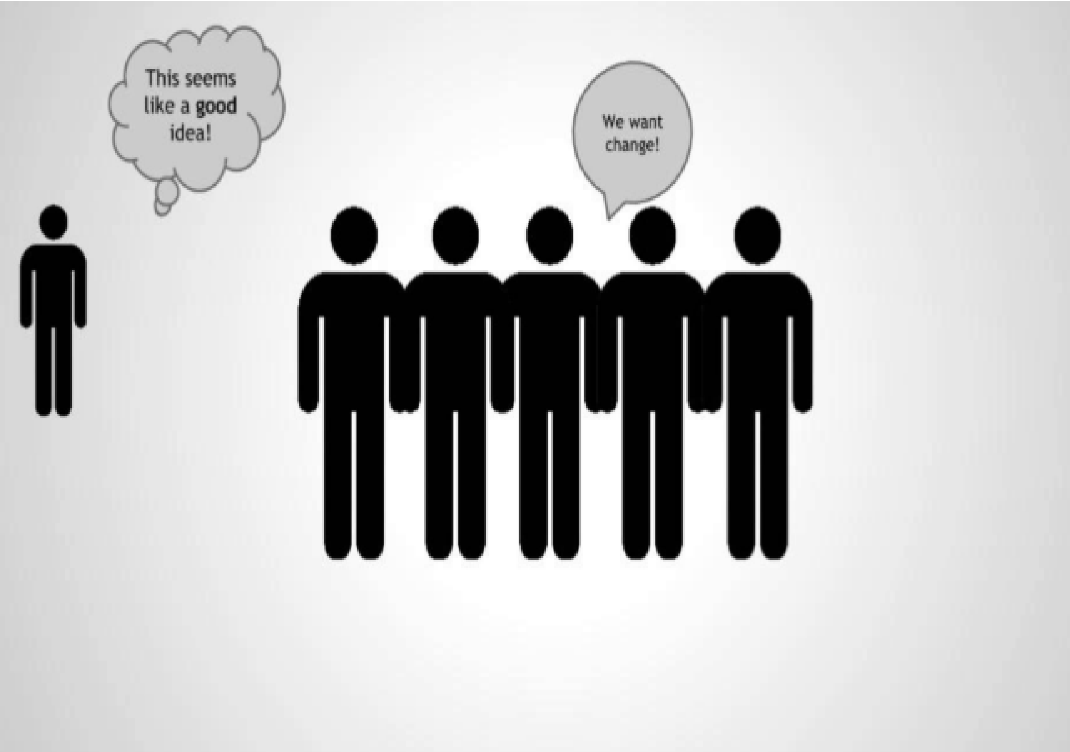
Consensus - Introduction

- **What** is it? Consensus is an old problem…
- It is about how multiples node **agree** on a **value**

- **From Who? It depends!**
- **In which way? Again…. It depends!**
- Permission (all nodes, some nodes)
- Election based, lottery based, BFT based
- PoW, PoX, BFT and variants, Hybrid….

Consensus algorithm - definition

- A **consensus algorithm** is a process in computer science used to **achieve agreement** on a single data value among **distributed processes or systems**.



Consensus algorithm – Introduction 2

Two main properties

- **Liveness**: something good will happen
- **Safety:** nothing wrong happens

➢ Formally defined in these terms:

- ***Termination:*** every correct process eventually (sooner or later) decides;
- ***Uniform integrity****:* every correct process decides at most only one time;
- ***Agreement****:* two correct process do not decide in a different way;
- ***Uniform validity****: if a* correct process decides for a given value *v*, then *v* has been proposed by some process.

Consensus – expected properties

- **Liveness:**
  - Validity: it ensures that if a node broadcasts a message, this will be delivered
  - Agreement: if a message is delivered by a correct node, it will be delivered by **all** correct nodes.
- **Safety**:
  - Integrity: it guarantees that only broadcast message are delivered and they are delivered only **once**
  - Total order: it ensures that all correct nodes deliver messages in the same order **same order**

Consensus in blockchains

- **Consensus in blockchains** refer **to the agreement process**, **not** to the specific choice made
  - It **does not guarantee** that **all (or a majority) nodes agree** to the total order (or on any single message) (but all those that did not fail.....)
  - A number of extensions to consensus protocol **include a validation step**, that checks **if** transactions are valid.

Participants to consensus

- **Two alternatives:**
  - **Permissionless consensus**:
    - All nodes have **same rights**
    - All nodes have **same possibilities**
    - All nodes could participate to the agreement's process **without asking permission to anybody**
  - **Permissioned consensus:**
    - **Some nodes** participate: they have **different rights**
    - In order **to participate** to the consensus process, **nodes need permissions**

blockchains

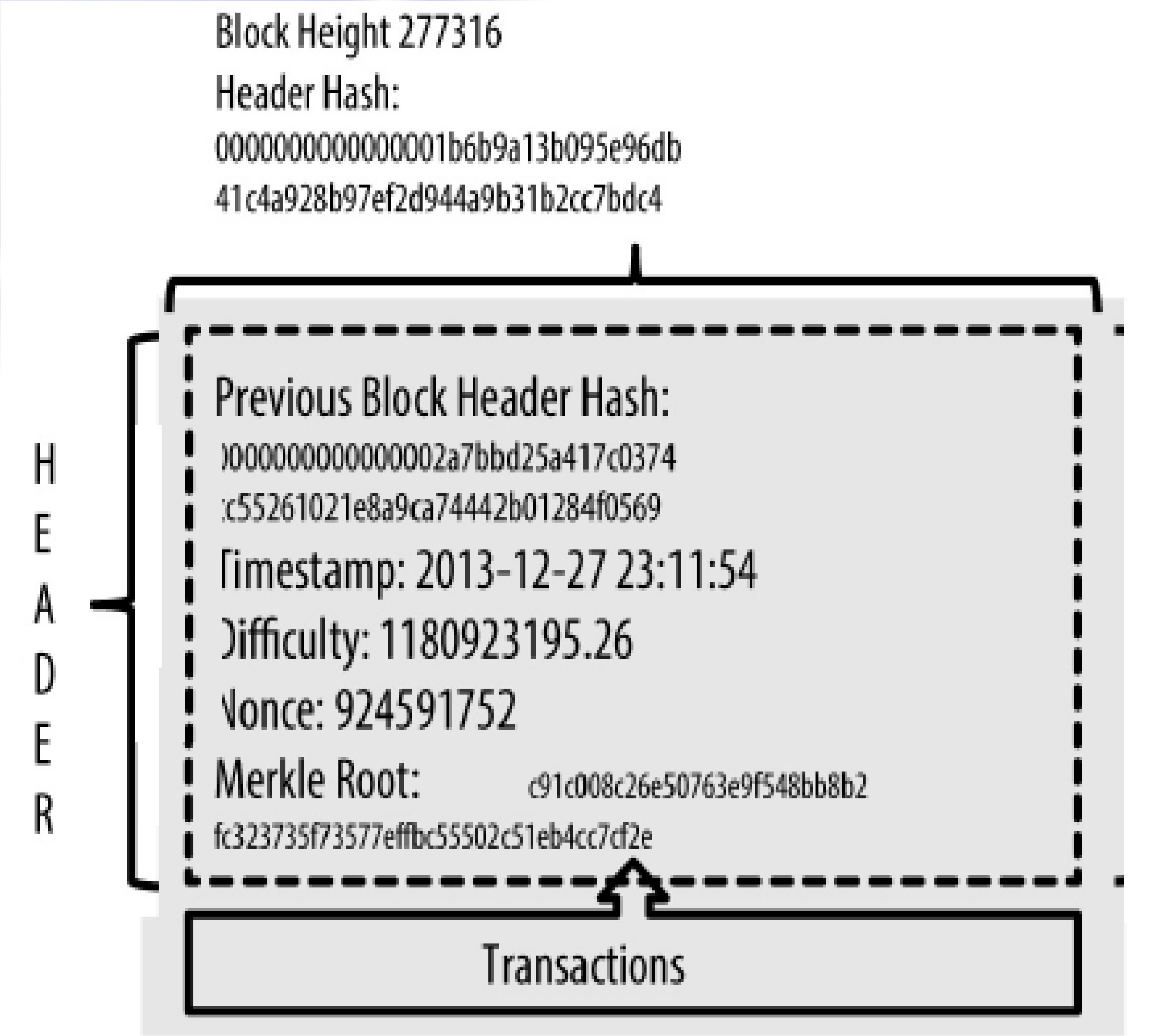
Fact and Terminology in blockchain:

- The process to reach an **agreement** is often called **mining**
- **After the agreement** information is stored in **a chain of blocks** linked one to each other
- Each block is composed by **some** transactions
- **Immutability, non-repudiation, data integrity** are the most important properties
- With Blockchain, study of BFT and its variants had **new impulse** (at the moment more than 700 BFT variants)
- The choice of the consensus protocol impacts on the **security** and **scalability** of **blockchain**

Introduction

- **Concepts:** Block is the base structure of **Blockchain**
  - It is composed by **Header** and **Body**
  - **Header contains** (main fields):
    - hash of previous block
    - timestamp of block creation
    - nonce: pseudo random number used by consensus algorithm
  - **Body contains:** set of validated transactions
- **The network is decentralized!**

Structure of a a block (Proof of Work)



This is a block!

blockchain: Introduction to PoW

- **Mining**: main process with which the block is validated and inserted into a blockchain through the **consensus algorithm.**
- Nodes that participate to the algorithm**: miners!**
- **Distributed ledgers:** registry that contains the chains of blocks **(the Blockchain!)**
- The first consensus algorithm of Bitcoins is named **PoW** (Proof of Work)
- It is **permissionless**!

Consensus in PoW

- **Decentralization permits in all nodes, independently:**
  - To Verify each transaction from client nodes
  - Transactions are aggregated in a block from miners
  - Proof of work: proof that ‘certain work’ is made in order to create a blocks
  - To Verify new blocks and to build the chain
  - To Choose the father of the node in case of fork
- **Transactions**: when done, they go in a pool till they are confirmed. They contain a **fee** for miners!

PoW Steps

- **1. Nodes verify independently that transactions satisfy some criteria (**correct sintax, fees not to low, sum of input > sum of output)
- **2. Miners aggregate transactions in a candidate block:**

mining process

- Nodes add a **coinbase transaction (**generate **Bitcoin** for themselves)!
- Header's block is build
- They find a **nonce** that, put at the end of the block, has SHA256(block + nonce) < **value!**
- **Value begins with a lot of zeros: very difficult!**

Consensus algorithm – blockchain PoW

- The **Value** is set automatically from the network in order that the block is build **in about 10 minutes**!
- The nonce difficulty is **updated each 14 days**
- **After mining: miner send block to all neighbours;** they receive it, validate it and send to their neighbours….
- Each node that receives it, adds it to **its copy of blockchain**!

➢ **3. If the validation process fails:**

- Block is **discharged**
- **Miner that propagated the block has lost time and money!**

Validation of a block

- **4. In case of validation:**
  - If a node receives a block that **follows** the last in the chain: it has to add it to the chain and link it to the previous one.
  - If a node receives a block that is **not** linked to the last one: the node create a **fork,** it calculate the work the two chains and takes **valid** that one with **more work**
  - Other nodes do the same: at the end the valid chain is the **longest one**. If node has chosen the right one, all is ok, otherwise it changes his valid chain
  - If a node **receive** a block that has no father, it send it in a pool… waiting till the father arrives!

PoW problems and advantage

- Number of blocks: **1 every 10 minutes**! High latency…
- Number of transactions: **only 1 MB for each block! Poor performance….**
- **It costs a lot of energy! !!! For 1 transaction more energy than 1 year power in an US house**
- If one node controls the **51%** of the chain work**,** it can mine **what it wants…** (it not easy to reach this condition)
- **But**
- **Completely decentralized**
- **Excellent scalability (node and clients)**
- **Almost immune to Sybil and Spoofing attack**

PoW variant: EtHash

- The algorithm works with a subset of a fixed resource
- The resource is a directed acyclic graph of some GB
- It should find a nonce with which the hash (subset + nonce) < value
- **Each 30.000 blocks** the graph is modified
- The graph should be pre-loaded **before** mining starts**:** if not, there is big delay in finalizing algorithm
- EtHash is used in **Ethereum**

EtHash: Pro and Cons

- Pro:
  - The difficulty is set so to build a block every **15 seconds**
  - Throughput **is higher than PoW** and **latency is lower**
  - Validation needs **fewer resources** to calculate hash
- Cons:
  - The difficulty is **too low**
  - **Countermeasure are needed**: society with big power could easily take the complete control of mining
  - A countermeasure is using the **memory hardness instead of power calculation**

Consensus algorithm – PoS (proof of Stake)

- **Key ideas :**
  - Nodes that want to participate to the mining process has to buy some cryptocurrency (a stake) to have the probability **p** (proportional to the stake) to become a **validator** (to participate to a process)
  - The algorithm selects **with probability p** the validator, in a way that nobody is sure to participate before mining process starts
  - If a selected validator is offline, a new one is chosen

PoS

- **PoS has two way to work: chain-based and BFT-style**

- **The chain-based PoS (permissionless)**
  - The algorithm selects in a pseudo-random way a validator in a time slot (ten seconds)
  - Algorithm assigns to the validator rights to create a block
  - The block must be linked to his father
  - Normally the chain converge into a single chain

PoS

- **PoS has two way to work: chain-based and BFT-style**

- **The BFT-style PoS (permissioned):**
  - Validators are randomly assigned **to propose** a block
  - The agreement of the block is done through a multi-round process
  - **Every validator** sends a vote for some specific block each round
  - At the end of the process validators agree if a block should be part of the chain.

Consensus algorithm – blockchain PoS

- **Problem in the PoS all-chained based**:
  - The validator is ‘invited’ to build as much blocks as it can… in order to have **more probability** to be choosen!
  - This is a **nothing-a-stake** problem! The validator could only gain…
  - **Solution: punish** validators when they generate blocks for nothing: for each block that is generated and not chosen a reward is subtracted!

PoS Pros

- **General:**
- No need for much of energy
- To invite node to participate to the network there is no need to issue many new coins
- It's build to discourage centralized control
- Reduced centralization risk
- Various form of 51% attack are more expensive than in PoW

PoS Cons

- **Chain-Based:**
  - It is easier to build **a complete new blockchain** starting from zero
  - Nothing at stake countermeasure is **every time** necessary
  - **Sybil attack** is easier
- **BFT-like**
  - Liveness denial: If BFT-like is used, a cartel (>34%) **could refuse to finalize** blocks
  - **censorship attack** possible

PoX - Prof of X

- **Identify a lot of algorithms – Proof of something:**
  - Proof of deposit: miners lock an amount of money that can not spend during the mining. Normally, mining voting power is proportional to the coin they have locked.
  - Proof-of-coin-age: miners show possession of quantity of coins and this is weighted by the age of possession. Mining voting power is proportional to this criteria.
  - Proof-of-identity: a miner must show that it knows a private key that corresponds to an approved identity cryptographically linked to a transaction.

PoX - Prof of X

- **Proof of capacity:**
  - Participants vote a new block weighed by their capacity **to allocate** not trivial space of disk
  - They should store **a large segment of a big file** that must be recoverable in case of dealer failure or shutdown
  - The voting power **is proportional** to the space allocated
- **Risk:**
  - Vulnerable to centralization due to participants that **outsourcing** the file storage to an external provider.

PoET (Proof of Elapsed time)

- Proof of elapsed time (permissionless)
- It runs in a **Trusted Execution Environment (TEE)**, like Intel's Software Guard Extension (SGX).
- **Basic ideas**:
  - each node generates a random number to know how much it has to wait before it is allowed to generate a block
  - the random number is based on a distribution F
  - Once that a block is generated, a node must generate a proof of waited activity (helped by SGX)
  - Statistically, it is checked if the node has respected the time distribution to generate a block

PoET (Proof of Elapsed time)

- The algorithm uses a model based on **the leader election (lottery)**: the election must be **random** between all participants and held in a **safe place (TEE)**
- **avoids manipulation**
- **Leader election:**
  - Each validator request a wait time from the code of TEE
  - The validator with the shortest wait time win the lottery and become a leader
  - The function in TEE are designed to **not be tampered**

PoET (Proof of Elapsed time)

- **Block generation:**
  - Whenever a block is generated, it will be verified by other nodes before that is accepted on the system.
  - The check consists in:
    - The node had the shortest time
    - It has waited the designate md time to generate the block
    - It has generated the block in a certain amount of time
- **Security:** it depends on the security of the trusted computing area (not 100% secure).

PBFT (permissioned)

Basics:

- It's the first algorithm based on **BFT** used in blockchain
- It's based on deterministic replicas of a server: if they don't fail, they generate the same result.
- If **f** replicas fail, the algorithm works with **n=3f+1 nodes**

Concepts:

- The protocol works with the primary backup approach:
  - Replicas move in different configurations called views
  - Each view has a primary replica and backup replicas
  - The primary chooses the execution order from client's requests

PBFT -2

- **Concepts:**
  - The primary replica assigns a number to the request and send it to the backups
  - Replica could fail => backups control number assigned to request: in case of problems, they use a time-out and order a change of view
- **Algorithm's steps:**
  - Request phase:
    - The client send request to the primary replica

PBFT - algorithm steps -1

- Request phase:
  - The replica assigns a number to the request if it is able to serve it
- Pre-Prepare phase:
  - The primary sends a pre-prepare message inserting the view **v**, the message **m** and a **digest n**
  - The primary inserts the message in its log

PBFT - algorithm steps - 2

- Prepare phase:
  - the replica accepts a request at condition that….
  - The algorithm is in view **v**, it can verify the message **m** and that **n** is in a certain range
  - The message **m** is accepted by backups, if they have **not** accepted a message with view **v**, sequence **n** and different digest
  - If the request is accepted and a backup has **m** in its log, it sends to all a prepare message

PBFT algorithm steps - 3

- **Algorithm's steps:**
  - Each replica collects messages till it has a message pre-prepare, 2f messages prepare that agree for sequence **n**, view **v** and request **m**.
- **Commit phase**
  - we have the total order in a view **v**, but… **in the others**?
  - Each replica sends a message in which affirms that it has a quorum certificate and add all in a log
  - Each replica collect messages till it has 2f+1 commit messages for the view **v**, message **m** and number **n** from different replicas

PBFT algorithm steps - 4

- **Commit phase**
  - each replica **executes** finally the request, after executing all requests with lower sequence number
- **Reply phase**
  - Replicas send reply to the client
  - Client replies to replicas
  - If the client doesn't reply, replicas send messages again

➢ **Complexity:** quadratic complexity in number of messages exchanged

Optimistic BFT a Variant of PBFT

- Optimistic BFT: it has linear communications complexity in the common case and quadratic only in bad conditions.
- Randomized BFT: it guarantees correctness with very high probability; it is not deterministic.
- XFT: it tolerates up to n/2 byzantine node
- Hybrid BFT: it combines optimistic and deterministic BFT protocols with PBFT.

Algorithm Comparison - table

	PoW	PoS	PoET	PBFT	Stellar
Type	Permissionless	Permissionless and Permissioned	Permissionless and Permissioned	Permissioned	Permissionles
Finalization	Probabilistic	Probabilistic	Probabilistic	Immediate	Immediate
Throughput	Low	High	Medium	High	High
Adversary	<=25%	Depend on the implementation	Not knowed	<=33%	<=33%

Algorithm Comparison – Scalability vs Performance