Introduction to consensus

Diego Olivier Fernandez Pons
diego.olivier.fernandez.pons@gmail.com
TZ Ventures - Tezos Combinator

October 2, 2020

Distributed databases

Distributed databases

Distributed databases are **everywhere**

- Search engines (Google, Bing)
- Online edition of documents (Google Docs, Office 365)
- Bank accounts
- RAID storage

Distributed databases

The architecture of distributed databases is extremely diverse

- Banking systems
  - don't keep a copy of all data in each node : your bank account only exists in the database (node) managed by your bank
  - don't expect nodes to fail in an irrecoverable way
  - work with very heterogeneous nodes (each bank)
- RAID
  - mirrors all the data on multiple hard drives (RAID ≥ 1)
  - expects nodes (hard drives) to just fail and be replaced with new (blank) ones

Distributed databases - Properties

Problems in distributed databases

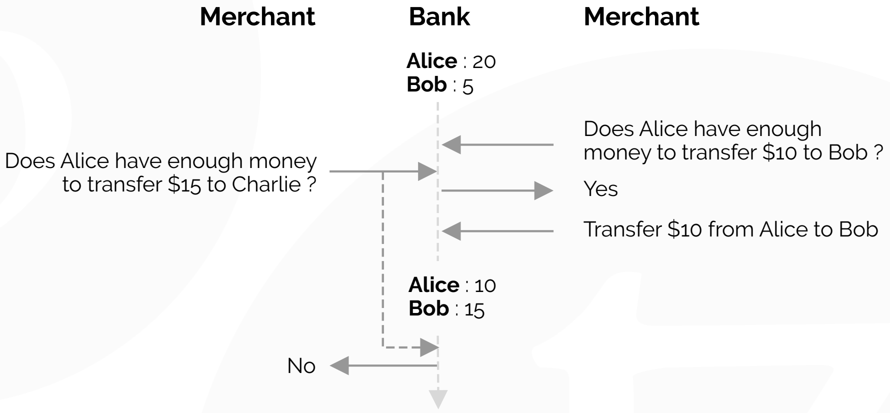
- **Consistency** : the system has to always return the latest version of the data
- **Availability** : the system always answers to requests
- **Reliability** : the request to the system have to succeed
- **Capacity** : the system has to accommodate a high volume of data and requests
- **Performance** : the system has to answer quickly

Distributed databases - Serialized access

Keeping data consistent in a database requires the **serialization of operations**.

- In a (central) database, a **lock** is put on the data, allowing only one process to access to it. The process guarantees the sequential and consistent handling of the data.
- In a distributed database, a **leader** is elected for a limited time, to manage the database in a sequential way.

Distributed databases - Serialized access



Merchant
Bank
Merchant
Alice : 20
Bob : 5
Does Alice have enough money to transfer $10 to Bob ?
Does Alice have enough money to transfer $15 to Charlie ?
Yes
Transfer $10 from Alice to Bob
Alice : 10
Bob : 15
No

Consistency of simple arithmetic operations (bank accounts) requires serialization.

Distributed databases - Consensus

The problem consisting in all nodes agreeing on a piece of information (who is the leader, what transaction to execute, etc.) is called the **consensus problem**.

Many consensus algorithms have been created

- BFT (Lamport, 1982)
- Paxos (Lamport, 1989)
- pBFT (Castro & Liskov, 1999)
- Raft (Ongaro, 2014), simplification of Paxos
- Tendermint (Buchman, 2016), simplification of pBFT

Distributed databases - Consensus

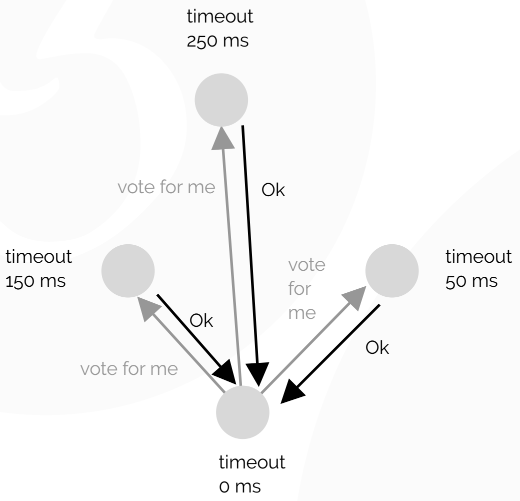
Most consensus algorithms for databases are based on **voting**

Database consensus - Raft

Raft is a typical, simple and popular consensus algorithm for databases

- A node that wants to perform an operation on the database sends it to the current leader for processing. After some time it will receive confirmation the operation was executed
- If there is no current leader, the node contacts other nodes asking to be elected the leader

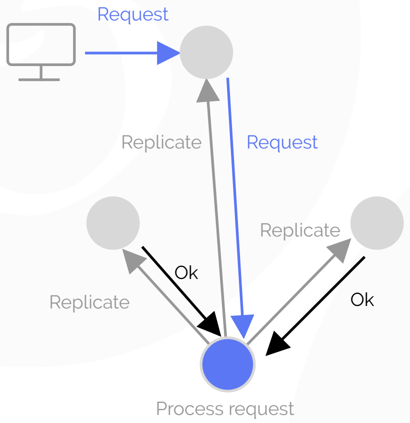
Database consensus - Raft



Election of a leader in Raft.

If there are multiple nodes requesting to be elected and none gets the majority, the nodes wait a random amount of time before asking again

Database consensus - Raft



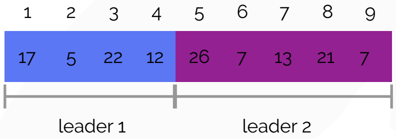
Replication in Raft

Client requests are saved uncommitted by the leader, sent to all peers for confirmation, and committed upon reception a majority of OKs. Then client then gets a confirmation.

Database consensus - Raft

Log structure in Raft

Raft synchronizes an append-log entry. Messages have of the form $log[i] = v$ which makes them idempotent.



If messages received by a node from the leader don't match their current log (e.g. the node crashed and recovered), they work together to resynchronize.

Database consensus - Raft

Recovery in Raft

Raft also requires some recovery measures

- A heartbeat is sent by the leader to prove it hasn't crashed
- If nodes don't hear from the leader, they start a new election round after a timeout.
- Nodes can only vote for a leader that has a longer history than them. As a result only values that are committed by a majority of nodes survive a leader crash

Blockchains

Blockchains

A blockchain is a distributed database with an **open set of nodes**

- Anyone can join or leave the database at any point
    - Like internet
- No hypothesis is made on the honesty of a group of nodes
    - Messages intercepted or ignored
    - Answering false information
    - Deliberate and coordinated attacks

Blockchains

Having a set of open nodes creates requirements over the architecture

- **Data redundancy** : If a node that has the only copy of a piece of data leaves, the system loses its consistency
- **Local verification of state**: because anyone can send any fake message, the state of the database needs to be double-checked in each node

Blockchains - Delta representation

Blockchain represent data with a **delta representation**

- **Initial state** : an initial state (genesis) is shared by all participants by an external mean (e.g. news paper)
- **List of past deltas** : each new participant receives from other nodes the list of the past deltas to apply to the initial state to obtain the latest state
- **New deltas** : periodically new deltas are sent to all participants for synchronization

Blockchains - Processing of updates 1/8

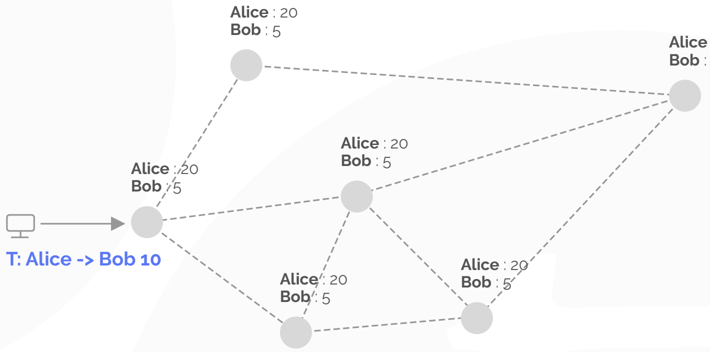


Figure: An application connects to a node and creates a transaction

Blockchains - Processing of updates 2/8

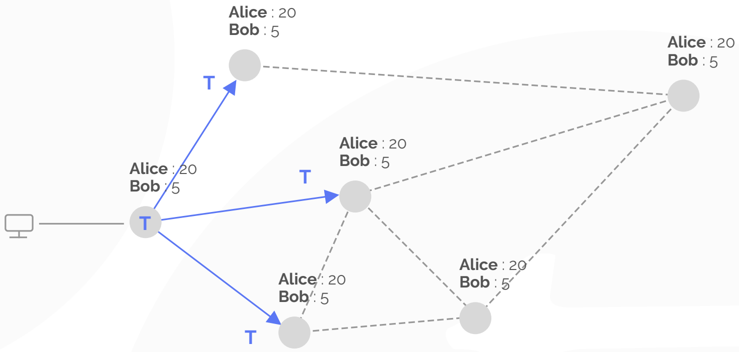


Figure: All peers are informed of the transaction and put in their pool

Blockchains - Processing of updates 3/8

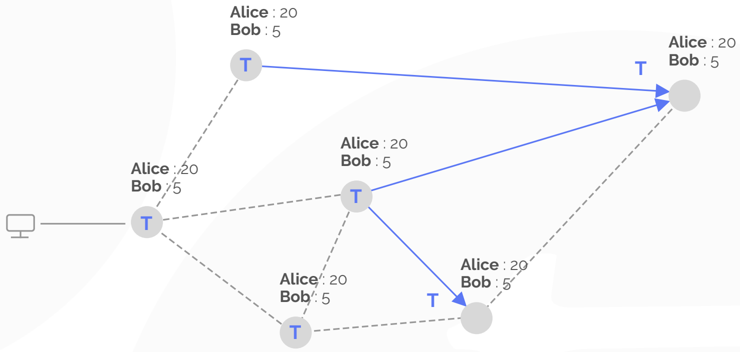


Figure: The nodes inform their own peers

Blockchains - Processing of updates 4/8

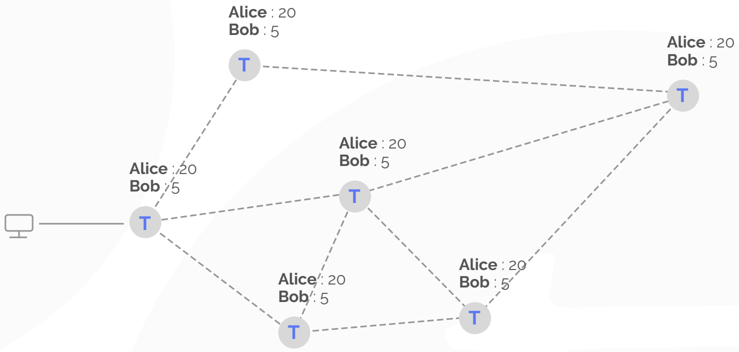


Figure: At some point all the nodes know about the transaction

Blockchains - Processing of updates 5/8

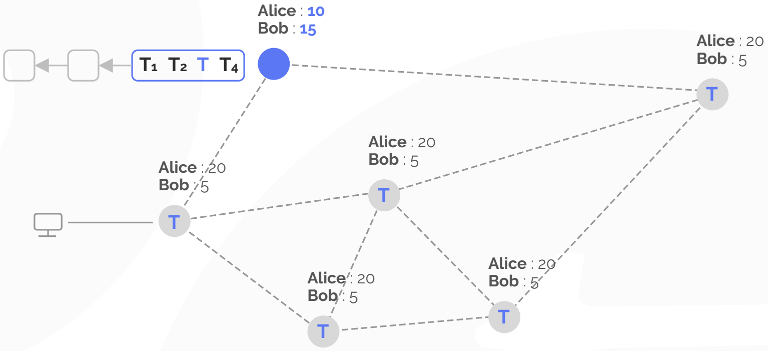


Figure: A node is randomly selected to execute all transactions in the pool

Blockchains - Processing of updates 6/8

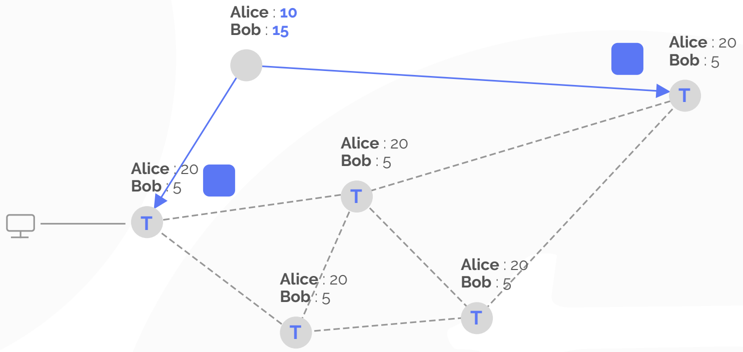


Figure: The new block of deltas is sent to the node peers

Blockchains - Processing of updates 7/8

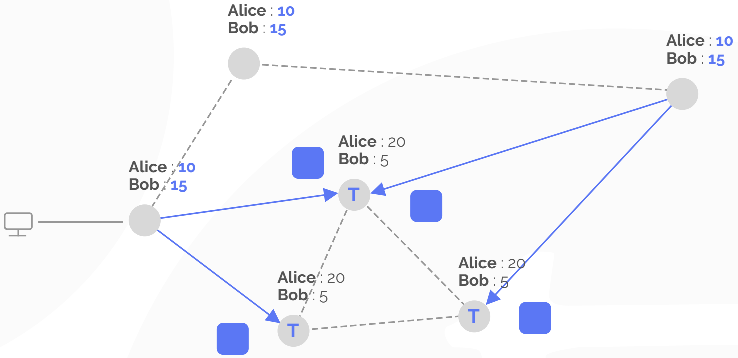


Figure: The new block of deltas is sent to from peer to peer

Blockchains - Processing of updates 8/8

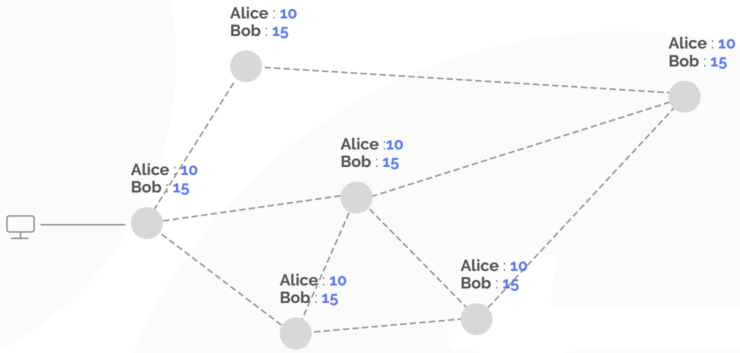
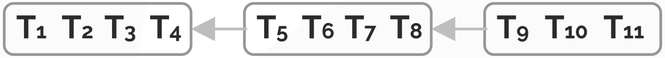


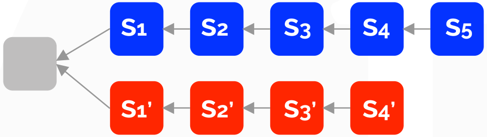
Figure: At some point all the nodes know about the block of deltas

Blockchains - Chain of blocks (of deltas)

The blockchain name was given because of the transactions being executed by blocks that point towards the previous block



This was done to easily identify if there are concurrent blocks being broadcast in the network



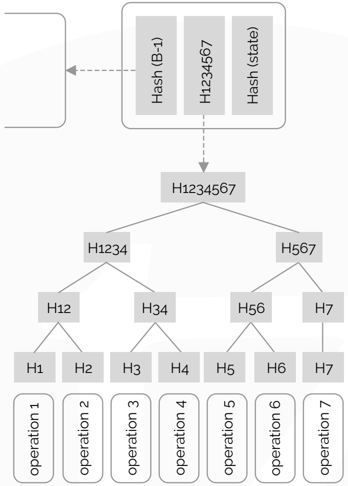
Blockchains - Hashes and lazy data transfer

To save bandwidth, blockchains don't exchange data (transactions, blocks, states) but **hashes** (of transactions, blocks, states).

If a node doesn't have the original object (e.g. transaction) it requests it to the node that send the hash. This **lazy data transfer** mechanism ensures an object is only sent once.

Blockchains - Block

A block usually contains the root of the Merkle tree of the transactions executed, the hash of the previous block, and a hash of the final state after applying the transactions



Consensus in blockchains

Consensus - Sybil attacks

Consensus in blockchains cannot be based on voting.

A **Sybil attack** is the creation by an attacker of a large number of identities (nodes) that can vote in the consensus algorithm and thereafter bias it in the sense desired by the attacker.

The only known solution to sybil attacks is **collateral-based voting** : base the voting system on something valuable and difficult to fake electronically.

Consensus - Nakamoto consensus

Satochi Nakamoto is the first person to succeed in designing a collateral-based consensus algorithm with **Bitcoin** (2008)

The core idea of the Nakamoto consensus is

- Make participants pay to vote (the amount of money of each participant is independent of the number of nodes they create)
- Reimburse them for their participation if they follow the consensus algorithm (do not reimburse them if they deviate)

However, the voting in Nakamoto consensus is much more indirect than in database consensus.

Consensus - Nakamoto consensus

We will divide the Nakamoto consensus in 3 steps

- **Collateral** : form of value used in the algorithm
- **Sortition** : selection of a leader in charge of creating the next block
- **Consensus** : selection of the current state in the history of blocks received

Consensus - Nakamoto consensus / collateral

Collateral is **electricity**

- Nodes are required to burn electricity and compute something
- Other nodes have to be able to check that the electricity was burnt by checking the result of the computation
  - The solution of an NP-complete problem requires an exponential time to find but can be checked in polynomial time

The problem chosen by Nakamoto to burn electricity is hash reversal

Consensus - Nakamoto consensus / sortition

The leader is any node that can **reverse a given hash**

- There can be multiple leaders at the same time. Ties will be solved by the consensus part of the algorithms
- The hash to be reversed is the hash of the next block + predefined suffix $h^{-1}(block + suffix)$ and is sent with the block to other nodes
- Other nodes can probabilistically check a node has burnt the electricity required to participate

The constants are adjusted for a block to be created every 10 minutes

Consensus - Nakamoto consensus / consensus

The current block is the block with **the longest chain**

- Every time a block is added to a chain, its block producer spent electricity to reverse its hash
- The longest chain is the chain that contains the most used electricity / used computing power / used computing time

It is almost impossible to create a parallel chain of same length (it would take the same amount of computing power as the real chain, meaning it would cost almost the same amount of money and take almost the same amount of time)

Consensus - Nakamoto consensus / reimbursement

To reimburse participants for their expenses, Nakamoto embedded a currency (a list of accounts with 'money') in the database.

- Block producers add to their own account a predefined number of "virtual coins" every time they create a block
- Operations on the database can be done only if the creator of the operation has "virtual coins" to pay for the operation fees
- Block producers can then sell the "virtual coins" they created for themselves when they created a block, to people that want to use the database

Consensus - Nakamoto consensus / punishment

The fact that both the **proof of money expenditure** and **reimbursement** are in the block makes the algorithm completely local.

Actors that misbehave are punished by ignoring their blocks.

Nakamoto consensus is a voting system

Nakamoto consensus is a voting system

- **Uniqueness** : A given amount of electricity can only "vote" / create a block once (possibly zero if it loses the hash reversing race)
- **Proportionality** : Statistically the number of "votes" / blocks created by a node is proportional to the total amount of electricity the node has spent
- **Outcome** : The outcome of the vote is the current chain

Nodes "vote" by spending money to add blocks to the chain of their choice

Nakamoto consensus is a voting system

If a vote was organized among block producers to choose which block they want as the current one

- Each block producer would vote for the chain containing the largest amount of money to be reimbursed to them
- The chain with the largest amount of money to be reimbursed would get the most votes

In other words the longest chain would be chosen

Nakamoto consensus converges

It makes no economic sense to choose as current node any node other than the head of the longest chain

- The longest chain has the highest probability to survive
- Any block added to a chain that is not the longest has a negative expectancy (cost money and is likely not to be ever reimbursed as the chain will just be ignored)

All **economically rational** actors chose the longest chain as the current state of the database

Nakamoto consensus security

It is very difficult to rewrite the history of a blockchain with Nakamoto consensus. It requires a **51% attack**

In Nakamoto consensus, the space of hashes is explored in parallel by the miners. It is not a perfect partition, but rather a random sampling. Nevertheless, the **net computing power of the network** is significantly larger than the one of a miner.

In order to rewrite the history of the database, an attacker would need more computing power than the net computing power of the network, to create blocks in < 10 minutes, in order to create more blocks and reveal a longer chain, making the network swap.

Beyond Nakamoto consensus

Limitations of Nakamoto consensus

- Consumption of electricity
    - Has reached unacceptable levels even for a very modest usage of the database
- Cycle of exchanges
    - Using electricity as a collateral means that block producers need to enter a perpetual cycle of exchanges money $\rightarrow$ electricity $\rightarrow$ virtual currency $\rightarrow$ money which makes the system fragile to exchange rates
- Speed
    - Simple solutions like decreasing the time between blocks by making the hash reversal problem don't work (because it generates an order of magnitude more simultaneous leaders)

Removing the electricity collateral

Due to the limitations of Nakamoto consensus, new algorithms known as Proof-of-Stake (PoS) tried to replace the electricity with the virtual coin as collateral

- **Peercoin** (King et Nadal 2012) : idea of using the virtual currency as **collateral** and first implementation
- **Chains of Activity** (Bentov et al. 2014) : basic algorithms for virtual currency collateral including **sortition** (*follow the satoshi*) and clear analysis of attacks to be prevented
- **Slasher** (Buterin, 2014) : **punish** nodes that deviate from the algorithm with destruction of their collateral

Removing the electricity collateral

The combination of Peercoin, CoA and Slasher algorithms prevents all the main attacks and provides a basic PoS algorithm

- Nothing at stake $\Rightarrow$ Slasher punishment
- Long range attack $\Rightarrow$ CoA checkpoints
- Stake grinding $\Rightarrow$ CoA follow-the-satoshi

However, no clean document was published explaining the merge of these algorithms created a basic PoS algorithm at the exception of the white paper of Tezos (2014)

Removing the electricity collateral

Modern PoS algorithms implement the basic PoS algorithm with minor changes

- Tezos **Emmy** (Goodman 2014) basic PoS algorithm with follow-the-satoshi sortition, simple vote by committee
- **Ouroboros** (Kiayias et al. 2019) basic PoS algorithm with VRF sortition, simple vote by committee

Collateralizing database algorithms

Once the basis of PoS were established new algorithms tried to collateralize database consensus algorithms

- **Tendermint** (Kwon 2014, Buchman 2016) : basic PoS algorithm with round-robin sortition, BFT-style vote by committee
- **Algorand** (Chen et Micali 2016) : basic PoS algorithm with VRF sortition and BFT-style vote by committee

Collateralizing database algorithms

Modern Nakamoto-based and BFT-based PoS algorithms are only superficially different (committee simple voting vs BFT voting)

In some sense it shows that the solution to the Sybil attack problem (Nakamoto consensus, its PoS ”simulation”) is independent from the BFT consensus problem.

But because Nakamoto consensus is an ”all in one” algorithm, it reuses the same tool (hash reversal) to solve various orthogonal problems.

Polling algorithms

In 2018, Emin Gun Sirer published the **Avalanche** algorithm (under the pseudonym Team Rocket and claiming it was the most groundbreaking work since Satoshi by an equally unknown genius... to reveal later it was him).

The idea is that consensus can be achieved by each node polling an increasing number of neighbours until the margin of error of the poll is low enough (and swapping accordingly).

The polling family of algorithms is **leaderless**

Blockchain consensus algorithms

3 families of blockchain consensus algorithms have emerged

- Patched versions of Nakamoto Consensus
  - Tezos (Emmy, Emmy+), Cardano (Ouroboros)
  - They remove the hash reversal and try to patch accordingly
- Patched versions of database consensus algorithms
  - Cosmos (Tendermint), Algorand (BFT), Tezos (Tenderbake)
  - They add collateral to database consensus to avoid Sybil attacks
- Polling algorithms
  - AVA (Avalanche)
  - Each node does a poll on an increasing number of neighboring nodes until the margin error of the poll is small enough

Proof of Stake

Proof of Stake

Proof of Stake (PoS) algorithms are algorithms that replace the electricity collateral with virtual coins.

> *Slasher in a lot of ways, although not all, makes proof of stake act like a sort of simulated proof of work.*
> *Vitalik Buterin, Oct 2014*

PoS algorithms behave like Nakamoto consensus (PoW, Bitcoin) without the hash reversal, and patched accordingly.

Proof of Stake - Template

PoS algorithms require

- **Collateral** : A freeze / unfreeze mechanism to hold the coins hostage and destroy them in case of misbehavior
- **Sortition** : A mechanism to select the next block producer in a way that is random and proportional to the amount of coins owned by each participant
- **Consensus** : A mechanism to select the current head (the block on top of which all nodes will continue building the chain) and a way to punish the nodes that deviate from the consensus

Tezos

Tezos (Goodman, 2014) introduced **Emmy**, the first PoS algorithm to put all the elements of Peercoin, Chain of Activity and Slasher together (freeze / unfreeze, sortition, accusations).

Tezos also added the concepts of delegation (inspired by BitShares), inflation, governance, a virtual machine designed to facilitate formal verification of smart contracts, and the use of functional languages (OCaml) to write the core software and open the possibility of formally proving the code.

Tezos

- **Delegation**: if you don't want to create blocks, you can delegate your **right to create blocks** to someone else
- **Inflation**: in Bitcoin, creation of new coins (inflation) is just a way to jump-start the system, and stops after a while. Tezos sees inflation as a fundamental economic mechanism.
- **Governance**: the ability to vote for changes in the code to avoid technical separations of the community

Tezos - Emmy (2014)

Tezos Emmy (2014) works as follows

- **Collateral** : Tokens are frozen for a given time. All claims of misbehavior need to happen in that window
- **Sortition** : A precomputed random calendar of block producers (and backups) is generated from a random snapshot of the amount of coins each node has
- **Consensus** For each level / round, a committee of 32 nodes is randomly selected (using the calendar system) to vote for the block they believe is the current one. All other blocks are invited to follow the committee majority, but are free to do what they want as long as they choose only one block.

Tezos - Emmy (2014) - Punishment

There are 3 attempts to cheat in Emmy

- Bias the random calendar
    - **Failure to disclose seed**
- Being part of a committee and vote for more than 1 block
    - **Double endorsement**
- Not wanting to select only one head, and when it is your turn to create a block, create multiple different blocks
    - **Double baking**

In Nakamoto consensus, all these problems are solved by hash reversal which costs money (= collateral) hence economically rational actors don't try to cheat

Tezos - Emmy (2014) - Punishment

Any node that notices a violation can create a **accusation** (with proof) and trigger a punishment

- 1/2 of the coins hostage are destroyed,
- 1/2 of the coins hostage are awarded to the accuser

(if all the coins hostage are awarded to the accuser, it could team with the violator and denounce after a timeout to try to recover all the coins before someone else does)

Tezos - Emmy (2014) - Liveness

Liveness is achieved with a list of backups for the block producer. The rank of the backup in the list is called its **priority**

- If after 1 minute the backup hasn't seen a block from the node producer for level *n* it can create the block (and successively)

Any block from the k-th backup that arrives before the k-th minute is invalid.

Tezos - Emmy (2014) - Liveness

Emmy doesn't use the block producer of node $n + 1$ as backup for node n because it wants to encourage node producers to be present in the network.

- The calendar is known 1 week advance
- You want to discourage nodes to be present just for the minute they need to bake and disconnect after that

In Emmy, there are **block steals** (a block producer of priority > 0 may be the selected producer for a block and collect the reward)

Tezos - Emmy (2014) - Liveness

The committee vote is implemented in a way that doesn't block the advancement of the algorithm

- Votes are broadcasted as separate operations (each member announces he votes for bloc B on level n)
- The votes for level n are included in the block of level $n + 1$
- The recommendation function (score / fitness) is based on the number of votes of the previous block

Tezos - Emmy (2014) - Recommendation

The score function (fitness) is the recommendation done to the nodes about which chain they should consider as current (just like Nakamoto's longest chain)

- The number of endorsements of the previous block
- The priority of the node producer
- The level (round) of the block

Tezos - Emmy (2014) - Summary

In summary Emmy is

- A **calendar** of leaders, backups and committee (from CoA)
- Rules stating nodes have to chose one head / **create only one block** and committee members **vote only once** (from Slasher)
- A freeze / unfreeze and accusation based **punishment mechanism** to avoid deviations (from Slasher)
- Endorsement **votes** for the current block by the committee
- A **recommendation** for all nodes to follow the committee

Tezos - Emmy (2014) - Summary

Tezos tends to see Emmy as a template algorithm (like Paxos) where some implementation details can be changed without fundamentally changing the algorithm

- Sortition algorithm (follow-the-Satoshi, VRF)
- Details of calendar (one "vertical" backup per round, next leader backup for previous leader)
- Vote of the committee (simple vote, BFT)
- Blocking or non-blocking committee vote
- Recommendation function

Algorand

Algorand (Micali 2016) introduced the idea of using verifiable random functions (Micali at al 1999) to do sortition on-the-fly

- BFT is used inside of the committee instead of simple voting
- Sortition is done with VRF
- Punishment is not implemented yet
- They keep claiming the algorithm is forkless, to after that explain (in the same paper) it isn't

The algorithm of Algorand is essentially the same as Emmy

Algorand

As discussed, at a very high level, a round of Algorand ideally proceeds as follows. First, a randomly selected user, the leader, proposes and circulates a new block. (This process includes initially selecting a few potential leaders and then ensuring that, at least a good fraction of the time, a single common leader emerges.) Second, a randomly selected committee of users is selected, and reaches Byzantine agreement on the block proposed by the leader. (This process includes that each step of the BA protocol is run by a separately selected committee.)

Algorand

The agreed upon block is then digitally signed by a given threshold (T_H) of committee members. These digital signatures are circulated so that everyone is assured of which is the new block. (This includes circulating the credential of the signers, and authenticating just the hash of the new block, ensuring that everyone is guaranteed to learn the block, once its hash is made clear.)

Chen et Micali - Algorand (2017) section 4 page 22

Ouroboros

Ouroboros are various PoS algorithms published between 2016 and 2019

- The authors of Ouroboros claim to be ”the first blockchain protocol of its kind with a rigorous security analysis”

The variants of Ouroboros are essentially the same as Emmy (and most are not implemented)

Tezos - Emmy+ (2019)

Emmy+ solves some limitations of Emmy

- Minor flaw discovered during the review of the algorithm done by the INRIA before the mainnet launch in 2018
- Minor annoyances for block producers
- Simplification of the analysis of the algorithm

Emmy also introduces some new fundamental ideas

Tezos - Emmy+ (2019) baker annoyance

In Emmy, a block producer for level n needs to decide between

- Wait till more endorsements arrive for level $n-1$ to increase the score of the block it has produced
- Broadcast the block to maximise the probability his block will be seen by the committee at level n before a block (potentially) made by his backup if he waits too much

Tezos - Emmy+ (2019) convergence reasoning

In Emmy, the fitness of a block (recommendation to nodes) is a combination of

- The number of endorsements of the previous block
- The priority of the node producer
- The level (round) of the block

Which makes it difficult to reason on it - for instance proving it makes no economic sense to deviate from the recommendation (just like in Nakamoto consensus)

Tezos - Emmy+ (2019) propagation speed

Emmy+ introduces the idea that nodes shouldn't select the current chain based on a complex fitness function but based on **time** (the first block that arrives)

The committee vote is not used to establish a numeric recommendation, but to **delay the blocks** in such a way the most approved one is propagated quicker in the network.

Tezos - Emmy+ (2019) propagation speed

The main idea behind Emmy+ is to emulate the behavior of Nakamoto Consensus where **the minority chain advances slower** than the majority chain

This temporal behavior protects Nakamoto consensus from attacks (as attackers need to have enough computing power to produce blocks quicker than 10 minutes)

- In Emmy+, attackers need to have enough votes in the committee to delay the other chains

Tezos - Emmy+ (2019) implementation

Emmy+ was implemented within the existing Emmy framework for simplicity

- The fitness function is the level of the block
- A block is valid after $(32 - e) * T_0$ seconds of its production time (which still follows the k-priority * 1 minute rule)

There is no obvious way to add a punishment for broadcasting a block before its validity time (it would require a heartbeat giving a random unique information at precise times). Therefore punishment for these violations hasn't beed added.

Tezos - TenberBake (2020)

TenderBake is a variation of Tendermint to make it closer to Emmy (it could be considered the Tendermint $\rightarrow$ Emmy approach, as opposed as Emmy $\rightarrow$ BFT)

TenderBake won't be deployed in Tezos as is it considered inferior to Emmy+. Instead research is orienting towards a **finality gadget** like in Ethereum's Casper on top of Emmy+

Exporation of the space of consensus algorithms

Problems to be solved

The space of Emmy and Emmy+ like algorithms is huge, there are many variants that need to be implemented and simulated in order to chose the best combination of features.

Our problem is to quickly prototype, simulate and compare (in realistic conditions) many of such variants.

Examples of variants

- Today, Emmy's calendar computes a leader, a backup list and a committee per level
  - Use leader for level $n+1$ as backup for level n
  - Don't select a leader, let each participant in the committee creates a block and selection of final leader with round-robin
- Today Emmy's committee makes a non-blocking simple vote leading to a probabilistic finalization
  - Blocking vote
  - More complex vote (pBFT, Tendermint, Paxos, Raft)
  - Deterministic finalization
- Today Emmy+ doesn't penalize for violating the delays rule
  - Introduction of a heartbeat + Slasher style penalty