GitHub Externship
Student - Industry Connect

</>INCUBATE IND.COM

PROPOSAL FOR ZEEVE INC. FOR GITHUB EXTERNSHIP

Flexible consensus algorithm supporting the participation of IOT and Mobile devices

By Aditya Ashutosh

- Email id: adityaashutosh82@gmail.com
- Contact No: +91-9430890922
- LinkedIn: https://www.linkedin.com/in/adityaashutosh
- Application ID: 21-05_Adi240_tfa_259

- **Synopsis**

Blockchain has been the technology of prevalent attention since the beginning of the pandemic era. It still is far away from mainstream adoption but the major boost in the past couple of years in the context of newly coming up startups has been massive and people have now started seeing why it is referred to as technology of the future. Blockchain is all set to disrupt and completely change the way we perceive the societal structure and sense of trust around us.

The ability of creating a decentralized ecosystem has been the prime highlight of blockchain and is certainly its most revolutionary vital feature. At the heart of decentralization, resides the ability to make decisions in a network in a distributed manner which is known as consensus. Without a consensus algorithm, a blockchain would just have been a dumb, immutable database.

Many consensus algorithms have been proposed in literature each having its own performance and security characteristics. One consensus algorithm cannot serve the requirements of every application. It is vital to technically compare the available consensus algorithms to highlight their strengths, weaknesses, and use cases.

Here, I have tried to compare various consensus algorithms available to us and then propose a flexible consensus algorithm for IOT devices and mobiles which might serve as a solution to the cons posed by the existing consensus algorithms.

- **Impact**

  As we have known by now, decentralization is the paramount highlight of a blockchain. A consensus algorithm allows to confirm transactions and operations, without the need of a third party intermediary.
  Therefore, a consensus algorithm helps a blockchain achieve decentralization.
  Therefore, having an effective consensus algorithm helps a blockchain retain its salient highlight and an effective consensus plays a very major role in pacing up the mainstream adoption of the blockchain

- **Available Consensus Algorithms and their overcomings**

Consensus Algorithm	Idea	Pros	Cons
Proof of Work	POW requires miners to compete to solve mathematical puzzles which require considerable computation and the winner gets to add a new block to the blockchain	It has been battle-tested and stands firm even today	Requires very high computation power and thus execution leads to huge energy consumption
Proof of Stake	POS requires participants to put some coins at stake. Higher the stake, higher the probability to mine next block	Energy efficient as it doesn't require as much computation as POW and is very expensive to attack	Rich gets richer and there is a nothing to stake attack possible where nodes can vote for multiple forks, thus exploiting no penalty loophole of POS as a result of which consensus can never be reached. Also, a validator can perform some bias to turn the randomness of execution process into its own favor which is called stake grinding attack
Delegated Proof Of Stake	In DPoS, the stake holders in the system can elect leaders(witnesses) who will vote in their	Energy efficient and faster than PoS	A bit centralized and participants with high stake can promote themselves as delegates

	behalf. This makes it faster than the normal PoS.		
Proof of Elapsed Time	In PoES, each participant node is required to wait for some randomly designated time and the fastest one to complete the waiting time gets to add the block	Low cost of participation and thus more people can participate and it is simple for all nodes to verify that leader was legitimately selected	Even though it's cheap, you have to use specialized hardware. Thus cannot be mass adopted
RAFT Consensus	RAFT achieves consensus via electing leaders. Servers in RAFT can be in 3 states: Leader, Follower and Candidate. Leader is supposed to replicate logs for followers and must respond to them within a fixed interval of time	It was ideated on the complexity of Paxos and thus offers a simpler solution than Paxos and is even implemented in different languages	Usually used in private, permissioned networks only

- **<u>Proposed Consensus Algorithm</u>**

  I propose a consensus algorithm which tries to overcome several shortcomings of the above mentioned consensus algorithms. I have tried to name the consensus algorithm as **STAQUORUM.**
  Below mentioned are some of the salient features of the Staquorum consensus:

  1. Partial PoS: Staquorum makes use of partial PoS. This means that participants who wish to become validators need to put certain currency at stake. Similar to PoS, there can be explicitly mentioned penalties to vote for faulty transactions. To avoid the "rich gets richer" problem, there will be a limit to the amount to be put at stake otherwise if larger players grow past 33% of the network, they can pose a threat to the validity of the network.

  2. Importance-based-selection : A completely randomized pool of validators will be chosen from the entire set of stakeholders. The user from the pool with maximum transaction value in the currency will be prioritized to perform necessary calculations to

add a new block of data. Higher the transaction value of an entity, higher the chances of being given the mining project to the entity are.

3. Dual vote penalty: As PoS deals with nothing-at-stake attack possible, where a voter votes in favor of every fork, hoping to maximize it's rewards, i.e. , guaranteeing that it gets rewards no matter which fork wins or loses. This might pose a situation where nodes might never come to consensus.
   Therefore, to eliminate this nothing-at-stake situation, Staquorum will slash and penalize those actors who try to vote for multiple forks and this will happen if voting takes place using cryptographically identifiable/verifiable signatures.

4. Anti Stake Grinding: POS also deals with stake grinding attacks where users try to manipulate and create bias into the randomness using some unfair computation in their own favour. To avoid that sort of situation, users should deposit their stake well in advance and also we can avoid using any information which can serve as a source data for any sort of randomness.

5. Quorum Slices: After a validator is selected to add a new block after determining its importance and activity in the network, it proposes a new block into the network. Now, here we can use the concept of explicit voting. Since there is a strong possibility of a Sybil attack in a blockchain, there might be some nodes we cannot weigh our trust on.
   Staquorum allows us to choose our own set of trusted nodes instead of having a central party choose them for us. This set of users we can trust is called a quorum slice. Each node in the network can be a part of multiple quorum slices where he chooses whom to trust and can be chosen by someone as a trusted node.
   When multiple quorum slices overlap, we form quorum intersections and thus it results in a larger quorum. This quorum (if not disjoint) finally makes the network reach consensus.
   This is highly decentralized and can work with very low latency and flexibility towards trust.

6. Sharding: It is a popular process in CS even used in ETH 2.0 in which the entire database is split horizontally to spread the load. This can help reduce network congestion and increase the scalability of the network by increasing transactions per second by creating new chains known as “shards”.
   With shard chains running, now everyone can afford being in the network and running a node and thus higher network participation and higher scale of decentralization.
   Each shard will have its own set of validators which will be randomly chosen for the opportunity to add a new block as described before.
   That’s all how **Staquorum** might function