Analysis of Committeeless Proof-of-Stake protocol: Searching for a better point of operation

Vinícius Peixoto[1], **Marco Aurélio Amaral Henriques**[1]

[1]Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas

v245294@dac.unicamp.br, maah@unicamp.br

Abstract. *This work aims to discuss the currently ongoing efforts towards the implementation of a fully autonomous and decentralized consensus mechanism based on Proof-of-Stake. We focus on the technical challenges arising from adopting a committeeless Proof-of-Stake consensus protocol (CPoS) where the global state of the entire peer-to-peer network is used to converge to a distributed consensus instead of relying on a validation committee. Specifically, we analyse the performance and security tradeoffs of the protocol. Since the CPoS protocol is very sensitive to its configuration parameters, we investigate their full impact on the blockchain performance and propose improvements to previous works in the area.*

1. Introduction

Blockchains are a relatively recent piece of technology that gained significant notoriety over the last decade. They can be defined as shared, encrypted and distributed databases, with the special property of being *immutable and irreversible*: Every piece of data (also known as a *block*) in the database is mathematically linked (through the use of hash functions) to its preceding entry in the blockchain, making it impossible to tamper with any given entry in the database without invalidating all the subsequent blocks.

Aside from integrity guarantees, there is also the advantage of *decentralization*: Blockchains run on top of peer-to-peer networks, where every node has a local copy of the blockchain (either in part or in its totality); this redundancy serves as a guarantee of high availability, as well as fault tolerance [Gao et al. 2018]. Furthermore, blockchains are completely *open* and *transparent*, meaning that any node in the network can read the entire history of the blockchain and is potentially able to insert new data into it.

2. Distributed consensus

The inherent decentralization in peer-to-peer networks gives rise to significant technical challenges concerning the reliability of blockchains. The first of them, which is a common problem in distributed systems, is the need for some sort of synchronization scheme that allows the nodes to write to the blockchain in an organized fashion; that is, only one node at a time may create a new block, and all the peers must agree upon the content and ordering of the blocks so as to maintain consistency between all the distributed copies of the blockchain [Zhang et al. 2020]. The second challenge is the need for *byzantine fault tolerance*: The network must still be able to function even in the face of distrust among peers [Lamport et al. 1982]. Faulty nodes (be it due to genuine malfunction or malicious behavior) must not be able to compromise the network as a whole.

To solve those challenges, several protocols (usually referred to as *consensus mechanisms*) have been developed. A comprehensive survey of the state of the art, as well as a comparison between several different proposals for distributed consensus protocols, can be found in [Lashkari and Musilek 2021]; however, we will briefly discuss two classes of consensus mechanisms that are generally considered to be the most prominent: *Proof-of-Work* and *Proof-of-Stake*.

2.1. Proof-of-Work

Proof-of-Work (PoW) is a consensus mechanism most notable for its use in the *Bitcoin* cryptocurrency. The idea behind it is very simple: In order to guarantee that only one node in the network will create a new block at a time, all peers will compete with each other for the right to create a block by solving a (very) computationally expensive problem. In this sense, the advantage in the race for generating new blocks comes from raw computational power. Verifying the solution to the problem, however, is very easy and cheap. Although this does solve the problem of achieving consensus in the network, this mechanism has the disadvantage of wasting large amounts of energy.

2.2. Proof-of-Stake

Proof-of-Stake (PoS) is an energy-friendly alternative to PoW, having gained a lot of attention with its recent adoption in the cryptocurrency *Ethereum*. The general idea behind all PoS consensus mechanisms is that peers invest a certain amount of currency (*stake*) in order to participate in a lottery; the winner of the lottery earns the right to create a new block. In several implementations of PoS (such as Ethereum, Algorand, Solana, etc.) a *validation committee*, comprised of a number of peers periodically selected at random, is responsible for electing and validating the generated blocks.

Validation committees serve as an additional security measure by allowing a voting process to take place between a small set of trusted validators, but they create various attack surfaces, introduce additional implementation complexity, and reduce the overall decentralization of the blockchain. Recent developments in blockchain security research have shown that the voting process that takes place between validators can be exploited in order to delay the consensus and increase the profits of malicious actors [Neuder et al. 2021], potentially at the cost of a considerably smaller fraction of the total stake in the network [Schwarz-Schilling et al. 2022] than the previous estimates of 33% based on the solution to the byzantine generals problem [Lamport et al. 1982].

3. Committeeless Proof-of-Stake

Committeeless Proof-of-Stake (CPoS) is a proposal for a PoS-based consensus mechanism that attempts to address the shortcomings of previous iterations of PoS by not requiring a validation committee, opting instead to offload the task of verifying new blocks to the nodes in the network. Like most other PoS variants, all peers participate in a lottery in order to earn the right to create new blocks; in CPoS, however, the sortition process is locally verifiable by all peers in the network (through the use of verifiable random functions) and there is a disambiguation criterion in place that ensures that even if several blocks are generated, only one of them will be accepted into the blockchain. A comprehensive description of the consensus protocol can be found in [Martins 2021], but an overview of the main points of interest is provided below.

3.1. Sortition algorithm

The sortition process in CPoS is inspired by Algorand [Gilad et al. 2017] and aims to provide a way for peers to validate blocks quickly and independently from each other, while using as little information as possible. Here the *stake* owned by an individual node is represented by w_i; the *total stake* in the network, $W = \sum w_i$, is defined as the sum of the individual stakes of all nodes.

The algorithm makes use of the binomial distribution to emulate a lottery. If a node's stake w_i is considered to be the number of lottery tickets in possession of the node, and the chance of any given ticket being selected at random in the sortition is p, then the probability that exactly $k \leq w_i$ among the node's tickets are selected is given by the binomial distribution: $B(w_i, k, p) = \binom{w_i}{k} p^k (1-p)^{w_i - k}$.

It follows from the definition above that investing a higher amount of stake into the lottery increases the chances of a node being selected to generate a block (that is, having at least one winning ticket). It can also be shown ([Martins 2021]) that the expected amount of selected tickets among all nodes in the network, denoted here by τ, is given by $\tau = p \times W$. In practice, this parameter is tuned so as to control the average number of successful sortitions in each round, and the value of p follows from its definition. The sortition algorithm itself consists of using the node's identity and the hash of the previous block in the chain to generate a (verifiable) pseudorandom number $q \in [0.0, 1.0]$ and finding the maximum value of k such that $q > B(w_i, k, p)$, representing the number of selected tickets in the sortition. In this way, the pseudorandom sortition process is deterministic and makes it possible for any peer to validate blocks autonomously.

3.2. Block election, confirmation and fork recovery

By design, the sortition algorithm allows for several tickets to be selected within the same round (based on the parameter τ), implying that many valid blocks could be generated simultaneously. For each candidate block, a pseudorandom value, called the *proof hash*, is calculated; the block with the smallest proof hash is then chosen as the next one to be inserted into the chain.

In a realistic CPoS network, the nodes apply the proof hash criterion in real time, reorganizing their local copy of the blockchain upon receiving a block that is better (i.e. has a smaller proof hash) than the one they currently have stored. Since the election criterion is unambiguous, the network will eventually converge to a consensus as long as the blocks with smallest proof hash reach all the peers; honest nodes make an effort to achieve this by propagating new accepted blocks through gossiping.

The block confirmation procedure is probabilistic in nature and relies on estimating the probability of the existence of a conflicting fork in the network. This is achieved by having the nodes validate incoming blocks locally and verifying that the successful sortitions that generated the incoming blocks share the same view of the blockchain with its local copy. In this manner, by repeatedly verifying that a given block is present in the blockchain view of other peers, a node can estimate the chance that a fork exists at that point; when this chance is lower than a predefined threshold, the block is confirmed.

4. Byzantine resistance concerns

Although previous works by Martins have explored theoretical and practical aspects concerning the performance of the CPoS protocol [Martins 2021], one aspect that was ex-

plored only briefly is the resistance of the consensus mechanism to dishonest nodes performing byzantine attacks. The consensus in the protocol is immune to Sybil attacks (where an attacker controls several fake identities within a blockchain for malicious reasons) due to the fact that the chances of being selected to generate new blocks depends solely on the total amount of stake in possession of an agent or organization, regardless of whether it is concentrated in a single node or in several.

However, the protocol seems to be susceptible to another kind of attack: If a malicious organization holds a significant portion of the total stake in the blockchain, the nodes in their control will be frequently selected in the sortition; they can then exploit this by purposefully refraining from generating blocks and reducing the overall number of perceived successful sortitions in a round. Since the block confirmation algorithm utilizes the number of successful sortitions in a round as an estimate of likelihood for the existence of forks, this will have the negative effect of reducing the overall confidence of the nodes in their local blockchain history.

5. Ongoing work

The previous implementation of the CPoS protocol posed several technical challenges that made it difficult to run new tests and expand on previous works; the design of the algorithm also had various minor shortcomings that affected the security and performance of the network. With those concerns in mind, we decided to invest our initial efforts into addressing those issues by refactoring and improving upon the old codebase.

It is currently being fully rewritten from Python 2.7 into Python 3.11, making use of modern build tools such as Poetry (for dependency management), improving on the previous Docker infrastructure in order to achieve reproducible builds, as well as implementing comprehensive unit testing in order to avoid unwanted regressions. The code is open-source and available at `https://github.com/regras/cpos_v2` with detailed build and usage instructions.

Another shortcoming of the previous iteration of the CPoS protocol is that it worked only under a static, fully connected network topology. We improved it by implementing a proper P2P network architecture, with features like peer discovery and availability monitoring, so that nodes may come and go from the network as necessary. This also allowed us to implement more complex topologies for the network, in order to simulate more realistic scenarios.

6. Preliminary experiments and methodology

In order to begin testing the trade-offs of the parameter τ (average number of successful sortitions in each round), we developed two preliminary experiments. All tests were orchestrated with Docker Compose and run on an AMD Ryzen 7 3700X host equipped with 32GB of RAM and running Linux 6.4. The relevant configuration files and build instructions for reproducing the experiments are available on the project's Github repository.

In the first one, we measured the influence of τ on the blockchain performance, as well as its impact on the amount of data circulating in the network, in a scenario where all peers are honest. For various values of τ, we monitored the changes in the total network throughput (number of confirmed blocks per minute, denoted by Q), average block confirmation delay, total number of messages exchanged in the network and total

amount of data transmitted between the nodes. We set up a network with 25 nodes in which each peer only knows of 5 other random peers in the network, in order to simulate a realistic P2P network scenario where a fully connected topology is impractical.

In the second one, we measured the influence of τ on the blockchain performance, particularly on the average block confirmation delay, in a scenario where adversarial nodes will refrain from broadcasting blocks, thus hijacking the confirmation mechanism. We set up a network with 30 nodes, where 5 of them ($\approx 16\%$ of the total stake) exhibited this malicious behavior. For this experiment we made the P2P network fully connected in order to eliminate the influence of the topology on the results and focus solely on τ itself.

In both tests, the data was collected over the course of 30 rounds and with a round time of 5 seconds, which proved sufficient for the scope of the experiments (due to the high throughput of Docker's internal virtual network). Furthermore, the nodes were configured to generate empty blocks (containing only the header) in order to establish a baseline for the total data usage in the network. Finally, all results were averaged from a total of 10 runs for each configuration of τ.

7. Results and discussion

The tables below contain the results of both preliminary experiments:

Table 1. Relation between τ and blockchain performance/network stress.

τ	Blocks/min	Confirmation delay (rounds)	Total messages	Total data
3	2.41	5.1	2.8×10^3	1.5 MiB
5	3.78	3.2	5.8×10^3	3.1 MiB
7	4.61	2.7	8.0×10^3	4.3 MiB
10	5.87	2.0	9.1×10^3	5.1 MiB

Table 2. Relation between τ and confirmation delay on an adversarial network.

τ	Confirmation delay (rounds)
3	12.4
5	4.8
7	3.2
10	2.3

The data in Table 1 shows that an increase in τ serves to both reduce the block confirmation delay and increase the blockchain throughput. However, this comes at the cost of a noticeable increase in the total number of messages exchanged between the nodes (since more blocks are being generated and passed around). The increase in the total data transit in the network is not to be dismissed, especially considering that once the blocks are no longer empty the data usage figures will spike up considerably. On the other hand, the data in Table 2 shows that the confirmation delay suffers very significantly from the presence of adversarial nodes; however, this effect is mitigated by an increase in τ. This demonstrates that the CPoS protocol relies heavily on the calibration of its parameters in order to strike a balance between network stress and performance/reliability.

The experiment also suggests that a viable strategy for allowing more aggressive tweaking of the τ parameter is having the nodes only broadcast block headers, instead of its full contents, until the network reaches a consensus about the selected block for a given round; only then should the entire block be broadcast to all peers. This would minimize the waste of bandwidth during the beginning of the round, when several blocks from different peers have to make their way around the network, allowing τ to be increased.

8. Conclusions and future work

In real-world applications of blockchains, high throughput and tolerance to network partitions are critical. As such, the issues with byzantine resistance pose significant obstacles that need to be overcome in order to turn the CPoS protocol into a competitive alternative to other consensus mechanisms. Thus, extensive testing is needed in order to tweak the protocol parameters and find a balance between reliability and performance.

For future work, it will be necessary to polish the new implementation of the CPoS protocol, as well as run more comprehensive experiments in order to make the current block confirmation algorithm more robust. It will also be necessary to find new ways to minimize the amount of data exchanged between nodes in order to allow for more aggressive tweaking of the protocol's parameters.

References

Gao, W., Hatcher, W. G., and Yu, W. (2018). A Survey of Blockchain: Techniques, Applications, and Challenges. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–11. ISSN: 1095-2055.

Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. (2017). Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 51–68. Association for Computing Machinery.

Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems, Volume 4, Issue 3*, 4.

Lashkari, B. and Musilek, P. (2021). A Comprehensive Review of Blockchain Consensus Mechanisms. *IEEE Access*, 9:43620–43652.

Martins, D. F. G. (2021). Um novo mecanismo de consenso probabilístico para blockchains públicas. *https://repositorio.unicamp.br/Busca/Download?codigoArquivo=507683*.

Neuder, M., Moroz, D. J., Rao, R., and Parkes, D. (2021). Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders.

Schwarz-Schilling, C., Neu, J., Monnot, B., Asgaonkar, A., Tas, E. N., and Tse, D. (2022). Three Attacks on Proof-of-Stake Ethereum. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, pages 560–576. Springer-Verlag.

Zhang, C., Wu, C., and Wang, X. (2020). Overview of Blockchain Consensus Mechanism. In *Proceedings of the 2020 2nd International Conference on Big Data Engineering*, BDE 2020, pages 7–12. Association for Computing Machinery.