# The Units of Permissionless Consensus: Towards Mobile and Edge Computing

Eduardo Ribas Brito
eduardo.ribas.brito@ut.ee

***Abstract*—Disclaimer: This paper is a work in progress. It is not a final version and it is not intended to be submited yet to the Distributed Systems Seminar. The content of this paper is subject to change.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

## I. Introduction

LONG has been the time when consensus started to be defined as a fundamental problem of distributed systems [1]–[3]. Generally, consensus means reaching an agreement between multiple parties in the potential presence of faulty individuals. As per multi-agent systems, interacting over computer networks, consensus is thought to be the result of a coordination effort, such that those parties agree on some value at a given moment. Achieving consensus implies that the system shall be reliable and fault-tolerant. However, the consensus problem has been limited by some assumptions on the networks. The well-known "secure Byzantine-Fault-Tolerant multiparty consensus systems" that have been designed over the years are usually meant to work only with a set of known participants, faulty or not [4]. The other side of the coin is the permissionless consensus challenge, consisting of achieving agreement in an environment where the participants are unknown and untrusted [5], [6]. Plus, there are other intrinsic particularities of this type of networks, for example, their openness, and the lack of any kind of central authority. This adds another layer of complexity to the problem, as the participants are not only unknown and untrusted but can also join or leave the network at any time, freely choosing if they want to participate in the consensus protocol or not. Nevertheless, the problem of permissionless consensus can still be seen as a special case of the more general consensus and can still be formalized in the same way. In this paper, we will focus on consensus in permissionless systems, especially in the context of blockchain networks. We will reason about its meaningfulness, ultimately by trying to identify the units that may underpin consensus. We will also discuss the current state-of-the-art, with a particular interest in the shift of the consensus layer of these distributed networks to mobile and edge computing environments, for which computationally expensive consensus algorithms are impractical and unfair.

## II. Related Work

### A. Classical Consensus

The establishment of a definition for the problem of reaching agreement in a distributed system was pioneered by Lamport et al. in [1]. The authors defined consensus as the problem of agreeing on a single value among a set of processes, in the presence of faulty entities. The first consensus algorithms were designed for synchronous systems, where the communication between the processes is reliable, and the delay is bounded. However, these initial attempts failed to cover the different types of faulty behavior. Along with the establishment of the famous Byzantine Generals Problem, the first solutions, not only for dealing with treason, but also for unreliable communication channels, or any other kind of arbitrary Byzantine behavior, were also proposed by Lamport in [2], [3]. The solution was a synchronous mechanism that used a set of leaders to reach consensus. Multiple practical implementations and optimizations to this solution have been proposed in the literature.

### B. Asynchronous Byzantine Consensus

The first practical asynchronous consensus algorithm was later proposed by Castro and Liskov in [4]. And naturally, after that work, many other asynchronous consensus algorithms have appeared. However, all of them are based on the assumption that the number of faulty processes is less than a certain threshold. Additionally, the assumption of a known set of participants is also made, as well as their roles in the consensus protocol. These are very strong assumptions that limit the challenges that can be addressed, for example, the impossibility to know the participants beforehand as they may participate anonymously, or dynamically.

### C. Permissionless Consensus

The advancements of the internet more than potentiated the revolution and what we now call the permissionless consensus problem was finally born. Without forgetting the previous attempts, the first practical permissionless consensus algorithm was proposed by Nakamoto in [5]. It is a proof-of-work consensus protocol that resembles a "replicated state machine" where the independent participants reach agreement not only about transactional values, but also about their order - naturally forming the underlying structure of what is now known as a blockchain. "Proof-of-work is essentially one-CPU-one-vote" and this is the novelty introduced by Bitcoin [7], [8]. The focus shifted for decentralized systems and after proof-of-work many other consensus mechanisms have been proposed, based on different consensus units, like proof-of-stake, proof-of-space, proof-of-burn, etc. The chaotic diversity of new consensus protocols gave also room for endless reviews, overviews and comparisons [9]–[14]. The authors of

these surveys often put multiple dimensions into comparison, like fault tolerance, scalability, or energy consumption, and, among those, some focused their efforts on mechanisms that may work in resource-constrained networks [15]–[18]. This paper will try to identify the common conclusions from these comparisons, while looking into the state-of-the-art and novel approaches for running permissionless consensus protocols in mobile and edge devices.

## III. The need for permissionless consensus

### A. Permissioned vs Permissionless

Following the line of the previous sections, it has been reasoned about the need for permissionless consensus when there are already well known and established consensus protocols that work in trusted environments [4], [19]. However, even those protocols have their own limitations, not only in terms of trust, fault-tolerance, centrality, permissions, or bottlenecks, but also in terms of scalability [19]. Trying to put some effort on decentralization, Byzantine-Fault-Tolerant consensus protocols are not known for their performance when the network grows in size. The more participants there are, the more messages need to be exchanged between them, and the more time it takes to reach consensus, even if assuring deterministic finality [20]. This is a problem that is not only related to the number of participants, but also to the communication fashion and bandwidth, and the computational capacity of the devices that participate in the consensus protocol. Summing up, the need for permissionless consensus is then justified by the fact that permissioned protocols are not compatible with the requirements of the new generation of distributed systems, especially in the context of blockchain networks. These requirements include dealing with today's sparse networks of anonymously and dynamically participating devices, without interrupting consensus and while battling Sybil attacks [9]. Fundamentally, the permissionless consensus problem is the need for a consensus protocol that can be run in a distributed and decentralized environment, where the participants are unknown and untrusted, and where the network is bigger, sparse and unpredictably unreliable.

### B. Allowances and goals

Technically, permissionless environments allow for larger networks that depict lower connectivity between the participants. Operationally, everything is expected to happen in an asynchronous or partially synchronous fashion, and the number of transactions is expected to be smaller than the permissioned counterparts. Nevertheless, participation is free, and the governance is not centralized, but rather distributed and public. The identity of the participants is secured or semi-secure as it often relies on pseudonymity for protecting the nodes' identity, while enabling full transparency in regard to the rest of the network's content and operation [21]. The goal of permissionless consensus, as generally for any consensus protocol, is to reach agreement on a single value, or a set of values. However, due to the nature of the protocols, the values that are agreed upon end up establishing the serialization of the transactions that are executed in the network, and so establishing time consciousness and total order of the events.

As pointed out by [21] and then referenced by [9], similarly to the permissioned counterparts, permissionless consensus protocols aim at achieving the following properties: Termination, Agreement, Validity, and Integrity. Without going into a lot of details, of these properties, Agreement and Integrity are the most important ones, as they are the ones that guarantee the correctness of the consensus protocols. Termination and Validity are generally related to any classic consensus problem.

### C. The building blocks of permissionless consensus

Also described in [9], very concisely, the way to achieve an operating protocol, as seen in the mainstream blockchain networks, is by first generating the agreeable value, in this particular case, a block and its proof, disseminating the information to the rest of the network, followed by the eventual validation and acceptance of the block by the rest of the nodes. This is the moment when consensus is reached. Nevertheless, during the whole process, a fair and somewhat predictable incentive mechanism is also needed, that rewards the participants for their honest effort in reaching consensus, and punishes the ones that are not behaving correctly. These incentives are of major importance in this very context of permissionless consensus. These building blocks form the basis of the inner functioning of Bitcoin itself [5], and are replicated with some variations in the other permissionless networks [6]. For example, there are some networks that do not have a proof-of-work mechanism, but rather a proof-of-stake, or a proof-of-space, or a proof-of-burn mechanism, etc. when it comes to the generation of the block and its existential and later verifiable proof. All the other pillars are generally the same [9]. The next sections will try to give a more detailed overview of these block generation proof units.

## IV. Proof-of-Work as a reference

### A. The block generation

**Algorithm 1** BlockGeneration

```
function
    BlockHeader ← Transaction Merkle Tree Root
        | Hash of the last Block
        | Timestamp
        | Other
    /* the preceding zero bits in target
  depict the mining difficulty */
    while Hash(BlockHeader | nonce) ≥ target do
        Increment nonce
    end while
    return new block
end function
```

In the classical Nakamoto consensus protocol, the generation of a block, to be proposed for further network agreement, complies with the unit of computational work needed to create, or rather find, a verifiable proof of the effort spent

on assembling the block [5]. This essentially requires brute forcing the search for a cryptographic hash value for the aggregation of the block information with a nonce, such that this hash value satisfies a difficulty threshold (See Algorithm 1), which gets adjusted dynamically over time, to maintain the network overall requirement for the block generation interval [9].

There are a couple of particularities that need to be mentioned. First, the block generation is of probabilistic nature, as the nonce is a random value, and the hash function is a one-way function. Allied to this, increasing the hashing power is not an easy task, which consequently allows for tackling Sybil attacks, despite the permissionless and pseudonymity nature of the network. Plus, the incentive mechanism also plays a role in this regard, by naturally incentivizing honest participation. The block generation interval, forged by the adjustment of the difficulty threshold, is also of major importance because it reduces the fork probability and allows for the timely propagation of proposed blocks, guaranteeing consensus in the presence of an honest majority of the hashing power [22]. This probabilistic finality shifts the 1/3 BFT threshold to a 1/2 threshold.

### B. Trade-offs and trilemma

Despite the harmony of all the aforementioned clever characteristics, the proof-of-work mechanism has some well known drawbacks. First, the block generation is computationally expensive, and so, the energy consumption associated with it is also high. This is caused by the dynamically adjusted difficulty threshold, which is a function of the block generation interval. For example, in the Bitcoin network, to maintain the 10 minutes interval, the hashing difficulty increases with the increase of the hashing power, and so too the energy needed for it. However, reducing the block generation interval would increase the probability of forks, which results in a more frequent waste of useful work, loosening the security of the network. Increasing the block size would also have the same security effect due to the increased propagation delay.

As pointed in the literature, the design of these consensus mechanisms shall aim for a protocolar choice between a set of properties that form a trilemma (see Fig. 1): Security, Scalability, and Decentralization. Summarizing, relaxing the security requirements may allow for more scalability, both of which, consequently, have hands tied with decentralization. These trade-offs are of practical consideration when defining the network goals and use cases [9].

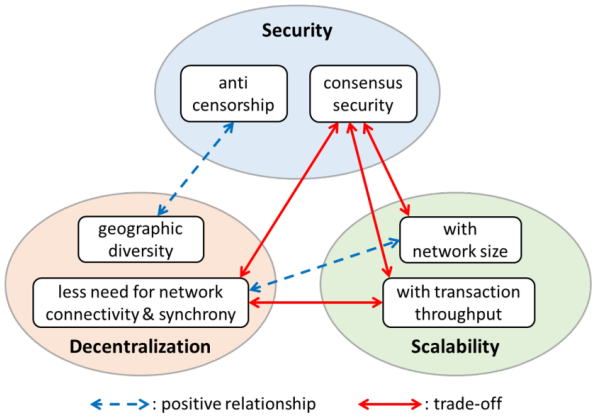


Fig. 1. From [9], the blockchain consensus trilemma and its inner relations.

## V. Alternatives to Proof-of-Work

### A. Proof-of-Stake based protocols

At this point, we can exercise the imagination and extrapolate the previous block generation mechanism to a *Proof-of-Something* pseudo-random competition in which an entity in possession of a higher amount of a certain resource, either computational power, or stake, or certain currency, or a higher amount of storage space, etc., guarantees a higher probability of leading the block generation and proposal, and consequently winning acceptance by the majority. This is the essence of the Proof-of-Stake consensus protocol, as a derivative of the Proof-of-Work mechanism, where a stake is a traceable and verifiable amount of a certain unit, token or currency, that is owned by a certain entity that wishes to participate in the consensus protocol. The stake works as a form of collateral that is used to guarantee the honesty of the entities, in an attempt to reduce the Sybil attack probability. And, generally as in Proof-of-Work, the higher the stake, the higher the probability of leading the block generation and proposal. Proof-of-Stake is, indeed, a more energy efficient consensus protocol, however, this adds up to nothing, especially when compared to the already proved BFT solutions, if centralization is a functional requirement, or unfair advantage is a potential trait. These limitations, along with other security concerns, may push away the participation of resourceless entities, and consequently, the network decentralization.

*1) Chain-based and Committee-based protocols:* Due to the self-contained nature of this type of protocols, where the attestation of the stake is done inside the network, multiple classes of Proof-of-Stake can be distinguished, by the way the meaningfulness of the stake is defined and employed. For example, there are the simplest forms of *PoS* that inherit most of the characteristics of the Proof-of-Work mechanism, but with the stake as part of the block generation and hashing proof (See Algorithm 2). These are commonly referred to as Chain-based *PoS*, inheriting also the fault-tolerance properties of the Nakamoto consensus protocol. Committee-based *PoS* protocols, on the other hand, are more complex, and they are based on the idea of a committee of entities that are selected to lead the block generation and proposal. The committee is formed by a subset of the network entities, and the selection is done based on their stake. Usually, the selection takes a form of a random sequence of stake holders, where the probability of being selected for every spot is proportional to the stake. That sequence is then consciously timed to orderly pace the block generation and proposal, by the current slot leader. Examples of this type of mechanisms are the *Ouroboros* [23], *Ouroboros Praos* [24] protocols or *Snow White* [25]. The longest chain rule is still applied, with probabilistic finality, and the 1/2 threshold is still required or tolerated

for the network to reach consensus. Nonetheless, Committee-based *PoS* protocols may not pair well with decentralization and increase in the committee size, which detereorates the performance of the protocol. Several improvements have been proposed to tackle not only this issue, but also the security of the committees which, smaller, could be more vulnerable to targeted attacks [24].

**Algorithm 2** PoSBlockGeneration

**function**
  $BlockHeader \leftarrow Transaction\ Merkle\ Tree\ Root$
    $| \ Hash\ of\ the\ last\ Block$
    $| \ Timestamp$
    $| \ Other$
  /* the preceding zero bits in $target$ depict the mining difficulty */
  **while** $Hash(BlockHeader \mid clockTime) \geq target \times stake$ **do**
    Increment $clockTime$ by a constant $tickInterval$
  **end while**
  **return** new block
**end function**

*2) BFT-based protocols:* These aim at achieving deterministic finality, by employing a BFT consensus protocol, not at the block proposal level, but at the block acceptance level. The idea is to have a free flow of blocks proposed by the network, but to have a quorum of entities that are responsible for the block acceptance. The block proposal mechanism can be of any kind, for example, Chain-based *PoS*, or Committee-based *PoS*, or even pure Proof-of-Work. The block acceptance layer then allows for a deterministic acceptance of the blocks, by having a sort of permissioned BFT consensus protocol, with the trivial 1/3 byzantine tolerance. The most well known examples and variations are the Tendermint [26] and the new Ethereum Casper [27] protocols.

Delegated Proof-of-Stake (DPoS) is a variation of the BFT-based protocols, where the consensus group is formed by a subset of the network entities, which are selected by the network, with a voting mechanism based on the delegation of stake. The consensus is then achieved using a typical PBFT protocol, with the usual 1/3 byzantine tolerance.

*B.*

## REFERENCES


[1] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[2] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. Springer, 2019.

[3] Leslie Lamport. The weak byzantine generals problem. *Journal of the ACM (JACM)*, 30(3):668–676, 1983.

[4] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, pages 173–186, 1999.

[5] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[6] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.

[7] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. *Cryptology ePrint Archive*, 2016.

[8] Rafael Pass and Elaine Shi. Hybrid consensus: Scalable permissionless consensus, 2016.

[9] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys and Tutorials*, 22(2):1432–1465, 2020.

[10] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154:113385, 2020.

[11] L. M. Bach, B. Mihaljevic, and M. Zagar. Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550, 2018.

[12] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, 7:22328–22370, 2019.

[13] Bahareh Lashkari and Petr Musilek. A comprehensive review of blockchain consensus mechanisms. *IEEE Access*, 9:43620–43652, 2021.

[14] Sarah Bouraga. A taxonomy of blockchain consensus protocols: A survey and classification framework. *Expert Systems with Applications*, 168:114384, 2021.

[15] Aleksandr Ometov, Yulia Bardinova, Alexandra Afanasyeva, Pavel Masek, Konstantin Zhidanov, Sergey Vanurin, Mikhail Sayfullin, Viktoriia Shubina, Mikhail Komarov, and Sergey Bezzateev. An overview on blockchain for smartphones: State-of-the-art, consensus, implementation, challenges and future trends. *IEEE Access*, 8:103994–104015, 2020.

[16] Junqin Huang, Linghe Kong, Guihai Chen, Min-You Wu, Xue Liu, and Peng Zeng. Towards secure industrial iot: Blockchain system with credit-based consensus mechanism. *IEEE Transactions on Industrial Informatics*, 15(6):3680–3689, 2019.

[17] Kimchai Yeow, Abdullah Gani, Raja Wasim Ahmad, Joel J. P. C. Rodrigues, and Kwangman Ko. Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues. *IEEE Access*, 6:1513–1524, 2018.

[18] Mehrdad Salimitari, Mainak Chatterjee, and Yaser P. Fallah. A survey on consensus methods in blockchain for resource-constrained iot networks. *Internet of Things*, 11:100212, 2020.

[19] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 31–42, 2016.

[20] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, pages 1–10, 2016.

[21] Yang Xiao, Ning Zhang, Jin Li, Wenjing Lou, and Y Thomas Hou. Distributed consensus protocols and algorithms. *Blockchain for Distributed Systems Security*, 25:40, 2019.

[22] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 281–310. Springer, 2015.

[23] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[24] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[25] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *International Conference on Financial Cryptography and Data Security*, pages 23–41. Springer, 2019.

[26] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph, 2016.

[27] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.