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 submitted 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-authority, etc. The chaotic diversity of new consensus protocols gave also room for endless reviews, overviews and comparisons [9]–[15]. 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 [16]–[19]. 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], [20]. 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 [20]. 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 [21]. 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], [13]. 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, sparser and unpredictably less reliable.

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 [22]. 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 [13].

As pointed out by [13], [22] 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-authority 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
  $BlockHeader \leftarrow Transaction\ Merkle\ Tree\ Root$
    $\mid Hash\ of\ the\ last\ Block$
    $\mid Timestamp$
    $\mid Other$
  `/* the preceding zero bits in` *target* `depict the mining difficulty */`
  while $Hash(BlockHeader \mid nonce) \geq target$ **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], [13].

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 [12], [23]. 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 computing that proof. 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 [12], [13].

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].

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

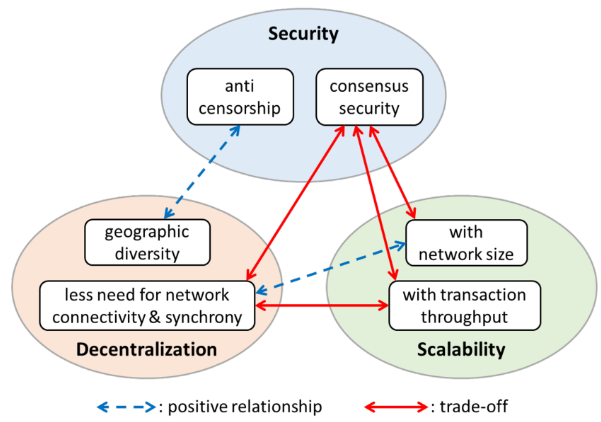

Fig. 1: From [9], the blockchain consensus trilemma and its inner relations.

of leading the block generation and proposal, and consequently winning the 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, affect the network decentralization and security.

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 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 the form of a random sequence of stakeholders, 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* [24], *Ouroboros*

Praos [25] protocols or *Snow White* [26]. 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 deteriorates 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 [25].

Algorithm 2 PoSBlockGeneration

function
  $BlockHeader \leftarrow Transaction\ Merkle\ Tree\ Root$
    $\mid Hash\ of\ the\ last\ Block$
    $\mid Timestamp$
    $\mid Other$
  `/* 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 [27] and the new Ethereum Casper [28] 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. Proof-of-Something Else

Idealized and inspired by Proof-of-Stake, extending or adapting Proof-of-Work became a popular trend in the blockchain community. The main idea is to replace the computational power with some other resource, that is more scarce, or more valuable, or more verifiable, or more traceable, etc., or to combine multiple resources, or even to add extra requirements to pure Proof-of-Work. From Proof-of-Meaningful-Work, or Proof-of-Luck, to Proof-of-Work-Time, going through Proof-of-Humanity, or Proof-of-Personhood, there are countless and exotic variations of these mechanisms, and the list is growing [14], [29]. Not that every one of them has a big potential for entirely solving the permissionless consensus problem, but each one of them may tackle different use cases where consensus needs to be reached, and where different resources are available to make the agreement happen [14], [15].

1) Proof-of-Space: established itself as a popular alternative among those. The goal is to replace computational power with storage space, and if possible give it a meaningful usage. Proof-of-Retrievability is a cited example of this type of protocols, where the storage space is used to store a large amount of relevant or public data, and the retrieval of that data is used as a proof of the storage usage [30]. Despite the reciclability of the storage space, as a resource that could be a good alternative to computational power, this algorithm has its own fatal flaws, depending mostly on a central dealer for randomness as well as for consensus fairness. Other reported Proof-of-Space variants may not even attempt to store/retrieve any meaningful data.

2) Proof-of-Authority: is another well known derivative, which, in effect, is a form of Proof-of-Stake, where the stake is the identity of the entity, that has been verified, scrutinized and approved by the network, giving it the right to participate in the closed consensus group. The consensus protocol, if not a BFT protocol, can follow the style of a Proof-of-Work protocol, depending on the requirements of the network - a choice that may shape the fault tolerance, decentralization, security and performance of the network [9].

3) Proof-of-Elapsed-Time: is another alternative to *PoW*, especially targeting the intensive mining issue, that has been proposed by Intel. The idea is to simulate the mining process by using a randomized time delay, which is proportional to the amount of work that would be required to mine the block. The time delay is then used to pace the block generation and proposal, and the block acceptance can be performed using a BFT protocol [31]. This solution, however, is not a viable alternative to other, permissioned and permissionless, approaches, by holding strong and centralized hardware requirements, as the computations require a trusted execution environment (TEE) to establish the trust among the participants. Some efforts have been made to combine the TEE feature with Proof-of-Stake, in order to achieve a more decentralized and secure protocol, but the same hardware requirements are still a major issue.

4) Proof-of-Location: may be an alternative for envisioning a future where the consensus is achieved by taking advantage of the dynamism and easiness of movement, placement and tracking of smaller and more location-versatile devices, such as mobile phones, IoT devices, etc. The idea behind this approach is to use the location and movement of these devices as a proof for the consensus. It relies on the location validation by close proximity, for example, using low-range communication channels, either by proving the location of the device at a certain point in time, or by proving their proximity to other devices. Incentive mechanisms may play an important role to prevent location spoofing, collusion, etc. [12]. These protocols are getting increased attention, especially in the context of IoT, with the provision of decentralized, permissionless and autonomous networks of low-powered devices that can be used for providing location verification services [14].

Consensus	Scalability	Finality	Adversary Tolerance	Communication Model	Accessibility	Agreement	Incentives	Centralized	Cost
Zab [54]	High	Determinstic	N/A	Asynchronous	Permissioned	Vote-based	✓	-	↑↑
PoW [55]	High	Probabilistic	<=25%	N/A	Permissionless	N/A	N/A	-	↑↑
PoL [56]	High	Determinstic	<50%	Asynchronous	Permissioned	Capability-based	✓	-	↑
PoE [57]	Moderate	Probabilistic	N/A	N/A	N/A	Capability-based	✓	-	↑
PoB [58]	High	Deterministic	<51%	N/A	Permissionless	Vote-based	-	✓	↓
PoT [59]	High	Probabilistic	<25%	N/A	Permissionless	N/A	N/A	-	↑
PoSpace [60]	High	Probabilistic	<25%	N/A	Permissionless	Vote-based	N/A	-	↓
PoExistance [61]	N/A	Deterministic	N/A	Synchronous	Permissinless	N/A	✓	-	N/A
PoM [62]	High	N/A	N/A	N/A	Permissionless	Vote-based	✓	-	N/A
PoAu [63]	N/A	Deterministic	N/A	Synchronous	Permissioned	Vote-based	N/A	-	N/A
PoAp [64]	N/A	Probabilistic	N/A	Partially Synchronous	Permissionless	Vote-based	✓	-	↓
PoKH [65]	N/A	N/A	N/A	N/A	Permissioned	Vote-based	N/A	-	N/A
PoI [66]	N/A	N/A	N/A	Synchronous	Permissionless	Capability-based	✓	✓	N/A
PoCredit [67]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PoPlay [68]	Very High	Deterministic	51%	Synchronous	Permisisonless	Vote-based	✓	-	↓
PoF [69]	N/A	N/A	N/A	N/A	N/A	Vote-based	N/A	-	↓
Flash Consensus [70]	N/A	Deterministic	N/A	Synchronous	N/A	Vote-based	-	-	N/A
PoCooperation [71]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Obelisk [72]	High	Deterministic	50%	Synchroous	Permissionless	Capability-based	✓	-	N/A
PoValue [73]	High	N/A	N/A	N/A	Permissionless	Vote-based	✓	-	N/A
PoDisintegration [74]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PoLearning [75]	N/A	Probabilistic	51%	N/A	Permisisonless	Vote-based	✓	-	↓
PoEligibility [76]	N/A	N/A	<30%	Asynchronous	N/A	Vote-based	N/A	✓	N/A
PoRep [77]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PoVote [78]	N/A	Deterministic	<50%	Synchronous	Permissioned	Vote-based	✓	-	↓
PoPF [79]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PoIndividuality [80]	N/A	Probabilistic	N/A	N/A	Permissionless	N/A	Vote-based	N/A	↑↑
PoPersonhood [81]	High	Deterministic	N/A	N/A	Permissionless	Vote-based	✓	-	↓
SIEVE [82]	N/A	Probabilistic	N/A	Partially Synchronous	Permissioned	Capability-based	N/A	-	↓↓
PoStake [83]	High	Probabilistic	<51%	Synchronous	Permissioned/less	Vote-based	✓	-	↑
PoET [84]	High	Probabilistic	Unknown	N/A	Both	Vote-based	N/A	-	↑
Ripple [85]	Low	Deterministic	<20%	Synchronous	Permissionless	Vote-based	-	-	↑
PoL [86]	High	N/A	N/A	Synchronous	Permissionless	Capability-based	N/A	-	N/A
PoCredibility [87]	N/A	Probabilistic	N/A	N/A	Permissionless	Vote-based	N/A	-	N/A
PoHistory [88]	High	Deterministic	N/A	Synchronous	N/A	Capability-based	✓	✓	N/A
PoEnergy [89]	High	Probabilistic	N/A	Asynchronous	Permissioned	N/A	✓	-	↓
PoWP [90]	N/A	Probabilistic	N/A	Synchronous	Permissioneless	Vote-based	✓	-	↑
PoO [56]	High	Deterministic	N/A	Synchronous	N/A	Capability-based	✓	-	↓
Bitcoin-NG [91]	High	Deterministic	50%	N/A	Permissioned	N/A	✓	✓	↓
PoAsset [92]	High	Deterministic	N/A	synchronous	N/A	N/A	✓	-	N/A
PoBelievability [93]	N/A	Probabilistic	N/A	N/A	Permisisoned	Capability-based	N/A	-	N/A
PoT [94]	High	Deterministic	N/A	Asynchronous	Both	Vote-based	N/A	-	↓

Fig. 2: From [14], an extensive list of pure proof alternative consensus mechanisms.

C. Other approaches

1) The XRP Ledger Consensus Protocol: powers the XRP Ledger in a low-latency, high throughput, and BFT fashion. The network itself operates somewhat differently from the other blockchain references, and treats transactions as atomic units of the ledger. These transactions are collected by the nodes and relayed to the nodes' UNL (Unique Node List), which is a list of peers that are trusted by that current entity. At every epoch, multiple rounds are needed to filter transactions and vote on the next state of the ledger. This voting happens at the UNLs' level, which are expected to be overlapped by a certain percentage, in order to achieve the artificial vote threshold for globally accepting the transactions. That threshold defines the fault tolerance of the protocol, and is set to 80%, which bounds this protocol to a 1/5 BFT allowance [32], [33]. Despite that limitation, when compared to other BFT protocols, this protocol seeks to trade fault tolerance for performance, by having lower connectivity and less message exchanges. However, it is still expected high synchrony within the UNLs. Given these characteristics, this protocol may be more suitable for permissioned networks, and, in terms of network size, is seen as not as scalable and decentralized as the permissionless ones [9].

2) DAG structures: may diverge from the previous approaches with an envisioned non-linear structure for the result of the consensus attestations. The choice resides in directed acyclic graphs that should expand and physically connect, further in time, the new data appended to the network. The data, which is typically held in the vertices of the graph, can have more or less granularity, either by storing blocks of transactions, or raw transactions directly. This design decision may pose new potential challenges, related to the verification and finalization of the consensus protocol. Currently, those two approaches are then distinguished as *blockDAG* and *txDAG* [9]. The block-based DAG approach diverges from the traditional blockchain structure by flexibly connecting every new block to multiple previous blocks. The rest of the protocol may work in the same way as the traditional Proof-of-Work, however, this new feature yields new security considerations. These protocols are theoretically designed for very high throughput in permissionless environments, but the security trade-offs are still not well-known [34]. In a transaction-based DAG, every vertex holds one transaction, that is connected to previous vertices. All the sibling vertices shall then represent disjoint transactions to successfully deal with history and conflict resolutions. Two well-known examples of this approach are IOTA Tangle [35] and Nano [36]. The former is of particular interest, as it is especially designed for IoT applications, low-powered devices, and micro-transactions. The protocol states that every new vertex needs to be connected to two previous vertices, therefore approving the parent transactions, as an incentive mechanism for honest behavior [9]. The users are the ones to submit transactions, generating and attaching a

PoW proof to it and so guaranteeing probabilistic finality with 1/2 fault tolerance. This solution may indeed pair well with decentralization and scalability, however, there are known security issues related to the easiness of growing malicious parallel branches with little computational effort, and those issues have been addressed by the introduction of an undesirable centralized solution. Other interesting and worth mentioning protocols are the ones that combine the DAG structure with the BFT consensus, such as the Snowflake-Avalanche protocol [37].

VI. Resource-Constrained Networks

A. Characteristics

Resource-constrained networks are a class of networks that are characterized by the limited access to resources, such as energy, bandwidth, storage, etc. These networks are typically seen in IoT applications, where the devices are usually low-powered, with limited computational resources, but are expected to collectively operate with high availability, at least the network as a whole, preferably without the need for frequent maintenance. These networks are also expected to be dynamic, with frequent changes in the network topology, by the unpredictable addition and removal of nodes. The nodes are also expected to be heterogeneous, with different capabilities, and may be connected to the network through different communication channels. Mobility may also be a feature, as the nodes may spatially move around, and may directly pair with different nodes at different moments. The nodes may also be individually unreliable, untrusted, uncoordinated, unmanageable, may fail at any time, and may be disconnected from the rest of the network.

Nowadays, the typical resource-constrained networks follow a centralized architecture with a client-server model, where the network is controlled by a single entity, which is responsible for the management of the nodes, including authentication, authorization, and communication. This single-point-of-failure is also a major scalability bottleneck, and a maintenance burden, as the network grows in number of devices [19].

B. Challenges

Having in mind these characteristics of resource-constrained networks, and the drawbacks of the centralized architecture, it may be postulated the need for a decentralized or permissionless shift that addresses some the most important challenges of these deployments [19]:

- *Scalability*: in terms of the number of available devices - decentralizing may allow for better, faster and less costly maintenance of the network, assuring more availability and reliability and cutting the single-entity dependency;
- *Security*: in terms of the network as a whole - removing the single-point-of-failure may reduce the attack surface, and may also allow or require more flexible and dynamic security policies;
- *Privacy*: in terms of the network data exchange features - decentralizing may allow for a user-centric approach, where the users are in control of the data that is shared with the network;

Permissionless consensus is a promising solution for these challenges, as an enabler for decentralized environments. The use cases for these networks are diverse. Nevertheless, as long as consensus is achieved, the network as a whole can be abstractly seen and used as a trusted entity that runs as a trustless decentralized service. Applications may then be built on top of it, and the network may be used as a platform for other services [18]. However, there are some key aspects that need to be addressed in order to make this approach viable for resource-constrained networks, as we saw in the previous section:

- *Resource consumption*: the consensus protocols should be resource-efficient, for example, in terms of energy, bandwidth, or storage, as the nodes are expected to be low-powered devices, with limited computational, communication, and storage resources;
- *Heterogeneity*: the consensus protocols should account for different device characteristics, as the nodes are expected to be heterogeneous in terms of their capabilities;
- *Fault tolerance*: the consensus protocols should be fault-tolerant, as the nodes are expected to be unreliable, untrusted, uncoordinated, or unmanageable;
- *Openness*: the consensus protocols should be open and trustless, as the nodes are expected to be dynamically added and removed from the network;
- *Security*: the consensus protocols should be resistant to a multitude of new attacks, especially if it turns out to be easy for malicious entities to overcome resource constraints;

The overall theoretical properties of the consensus protocols still need to be taken into account, as the goal is to have a proved functional mechanism that achieves Termination, Agreement, Validity, and Integrity, while playing with the trade-offs imposed by the trilemma (See Fig. 1) [9].

C. Related Work

There have been several attempts to address the challenges of resource-constrained networks, and to understand the limitations of the existing consensus protocols in this context. A common conclusion is that the existing protocols are not suitable for these networks, due either to the high resource consumption, or to the centralization of various aspects of the protocol, or also to high communication overheads, that are translated to high latency and low throughput [19], making these protocols unsuitable for resource-constrained applications. Other concerns are the high space consumption, to store the older blockchain information, or the security issues, as some attempts to overcome the resource constraints may lead to new vulnerabilities, which may be exploited by malicious and more powerful entities.

A thoroughly mentioned approach is the IOTA Tangle protocol [35], which presents a friendly design for resource-constrained networks, infinitely scalable, and with low latency. However, there are still known security and centralization limitations, as presented in the sections before [17]. Another concept that has been explored is sharding [18], [19], which is a technique that allows for the partitioning of the network into smaller sub-networks, that are then managed by different entities, validating transactions in parallel. This approach may be used to overcome the scalability bottleneck, but it may also introduce new issues that are still to be researched. Frankly, there are indeed some well established protocols that, all in all, can be adapted to networks with resource-constrained devices that are not necessarily short on the resources needed to run the protocols. For example, a lot of *PoS* related protocols may be adapted to resource-constrained networks, where the stake is the only considerable resource that the nodes need. There are a multitude of projects that are currently exploring this approach, for example, with the Ethereum transition to *PoS* [28]. However, this does not address the hypothesis of taking advantage of the very nature of these limited environments, as simply choosing a BFT protocol that is resource-efficient may also put us far apart from the ultimate decentralization and scalability goal of other and fairer means of consensus participation. An interesting idea is based on the use of *Proof of Location* for coordinating the consensus. The easiness of movement and targetted location deployment of a lot of these low-power devices may be used to achieve a decentralized consensus protocol, where elected nodes may be somehow chosen based on their location, either by proximity, or by some other witnessing mechanism. Variations and extensions of this idea, with different and wide use cases, have been seen in [38], [39]

VII. Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec auctor, nisl eget aliquam aliquam.

VIII. Acknowledgements

This work was supported by the 2022 XRP Ledger Trust Scholarship Program for Master's Students at the University of Tartu.

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] Christopher Natoli, Jiangshan Yu, Vincent Gramoli, and Paulo Esteves-Verissimo. Deconstructing blockchains: A comprehensive survey on consensus, membership and structure. *arXiv preprint arXiv:1908.08316*, 2019.

[13] 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.

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

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

[16] 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.

[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] Jorge Peña Queralta and Tomi Westerlund. Blockchain for mobile edge computing: Consensus mechanisms and scalability. In *Mobile Edge Computing*, pages 333–357. Springer, 2021.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

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

[29] Blockchain consensus encyclopedia. https://tokens-economy.gitbook.io/consensus/chain-based-proof-of-work. Accessed: 2022-10-22.

[30] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597, 2007.

[31] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, and Cian Montgomery. Sawtooth: an introduction. *The Linux Foundation*, 2018.

[32] David Schwartz, Noah Youngs, Arthur Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5(8):151, 2014.

[33] Brad Chase and Ethan MacBrough. Analysis of the xrp ledger consensus protocol. *arXiv preprint arXiv:1802.07242*, 2018.

[34] Yonatan Sompolinsky and Aviv Zohar. Phantom. *IACR Cryptology ePrint Archive, Report 2018/104*, 2018.
[35] Serguei Popov. The tangle. *White paper*, 1(3), 2018.
[36] Colin LeMahieu. Nano: A feeless distributed cryptocurrency network. *Nano [Online resource]. URL: https://nano. org/en/whitepaper (date of access: 24.03. 2018)*, 4, 2018.
[37] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies. *Available [online].[Accessed: 4-12-2018]*, 2018.
[38] Foamspace Corp. Foam white paper. Accessed: 2022-11-06.
[39] Scott Scheper Arie Trouw, Markus Levin. Xyo white paper. Accessed: 2022-11-06.