SINERGI Vol. 25, No. 3, October 2021: 381-392
http://publikasi.mercubuana.ac.id/index.php/sinergi
http://doi.org/10.22441/sinergi.2021.3.015

REDUCING OVERHEAD OF SELF-STABILIZING BYZANTINE AGREEMENT PROTOCOLS FOR BLOCKCHAIN USING HTTP/3 PROTOCOL: A PERSPECTIVE VIEW

Nur Arifin Akbar[1*], Andi Sunyoto[1], M. Rudyanto Arief[1], Wahyu Caesarendra[2]
[1]Department of Informatics Engineering, Universitas Amikom Yogyakarta, Indonesia
[2]Faculty of Integrated Technologies, Universiti Brunei Darussalam, Brunei

Abstract
Today, there is a tendency to reduce the dependence on local computation in favor of cloud computing. However, this inadvertently increases the reliance upon distributed fault-tolerant systems. In a condition that forced to work together, these systems often need to reach an agreement on some state or task, and possibly even in the presence of some misbehaving Byzantine nodes. Although non-trivial, Byzantine Agreement (BA) protocols now exist that are resilient to these types of faults. However, there is still a risk for inconsistencies in the application state in practice, even if a BA protocol is used. A single transient fault may put a node into an illegal state, creating a need for new self-stabilizing BA protocols to recover from illegal states. As self-stabilization often comes with a cost, primarily in the form of communication overhead, a potential lowering of latency - the cost of each message - could significantly impact how fast the protocol behaves overall. Thereby, there is a need for new network protocols such as QUIC, which, among other things, aims to reduce latency. In this paper, we survey current state-of-the-art agreement protocols. Based on previous work, some researchers try to implement pseudocode like QUIC protocol for Ethereum blockchain to have a secure network, resulting in slightly slower performance than the IP-based blockchain. We focus on consensus in the context of blockchain as it has prompted the development and usage of new open-source BA solutions that are related to proof of stake. We also discuss extensions to some of these protocols, specifically the possibility of achieving self-stabilization and the potential integration of the QUIC protocol, such as PoS and PBFT. Finally, further challenges faced in the field and how they might be overcome are discussed.

Keywords:
Blockchain;
Byzantine Agreement;
PBFT;
Proof of Work;
QUIC protocol;
Self-stabilization;

Article History:
Received: January 5, 2021
Revised: April 15, 2021
Accepted: May 17, 2021
Published: September 29, 2021

Corresponding Author:
Nur Arifin Akbar
Department of Informatics Engineering, Universitas Amikom Yogyakarta, Indonesia
Email:
nur.1233@students.amikom.ac.id

INTRODUCTION

This first section provides a background to the agreement in distributed systems that are prone to failures. After that, blockchain is presented and how it utilizes distributed agreement. Finally, the section is concluded with the aim of this paper. Distributed systems are a fundamental part of many services used Today, and applications often require these systems to be resilient to failures and malicious actors. These systems utilize different sets of protocols to maintain an agreement amongst themselves. Being able to reach an agreement or consensus is necessary as the system is expected to perform coordinated tasks, such as removing misbehaving nodes from the network. Misbehaving nodes, meaning nodes that are not functioning correctly, exist due to the nature of distributed systems, where networks may fail, and malicious nodes may be introduced to the system. In order to provide resilience against malicious behavior, nodes are considered to be able to act Byzantine.

That is, they might exhibit arbitrary or malicious behavior, which may hinder protocol progression. Agreement protocols designed to function correctly with Byzantine nodes are referred to as the Byzantine Agreement (BA) protocols. Additionally, to further strengthen the system's fault tolerance, it can be designed to be self-stabilizing. A self-stabilizing system guarantees that it will always end up in a legal execution, such as exhibiting correct behavior concerning the system requirements, regardless of what state the system might be starting in, a desirable property production system.

In this paper, we will investigate the earlier introduced challenges on the blockchain. Blockchain, and its most well-known application, Bitcoin [1], is a clear example of how distributed systems in practice may reach an agreement. In Bitcoin, nodes must agree on the set of valid transactions and their order in the underlying ledger. What might result from its popularity, Bitcoin has spawned several potential successors utilizing other BA protocols. Altogether, this cryptocurrency ecosystem safeguards immense amounts of money enabled by decades of research in the field of distributed systems and Byzantine fault-tolerance (BFT), a field that is still rapidly evolving.

A well-known, popular blockchain application is a cryptocurrency, a distributed system that imposes strict requirements: most significantly. It is crucial to avoid what is referred to as double-spending. Users can spend their money more than once, which would naturally undermine the entire payment system. In order to avoid this and other similar problems, robust agreement protocols are used. Thereby, the nodes providing the service can agree on the system's current state, which for a blockchain is the underlying ledger of blocks. These agreement protocols need to be resilient and tolerate Byzantine faults, which is achieved through designing BFT protocols.

In this survey paper, we will provide an overview of three consensus protocols with varying applications. We discuss their limitations and whether or not self-stabilization could be introduced to target these weaknesses. As this may result in increased performance overhead, we will finally discuss the extent to which the current internet network stack could limit the performance of distributed blockchain applications. The QUIC protocol will be introduced and discussed as a solution in this context to reduce this overhead increase. The following section presents the theory needed to understand the survey, and after that, Section 3 describes the consensus protocols central to this paper. Section 4 discusses the differences and similarities between these protocols and how some improvements could be introduced. Section 5 describe previous work that combines UDP+TCP for IP protocol experiment. Finally, Section 6 concludes our findings.

THEORY

This section presents the theory for the different areas covered by this paper and aims to introduce these topics. First, blockchain is presented before the area of fault tolerance is discussed. After that, the section is concluded by a presentation of self-stabilization and the QUIC protocol.

Blockchain

When designing algorithms for distributed systems, such as blockchain, nodes' need to agree on a particular task that often arises; they might have to agree on the current state of the underlying ledger, e.g., to appoint a single leader. The process of reaching an agreement is non-trivial in distributed systems, and this problem was introduced by Pease et al. [2]. It is a well-studied problem in computer science that essentially outlines difficulties that arise when a set of nodes wish to agree on a specific task, where a subset of nodes might exhibit faulty behavior.

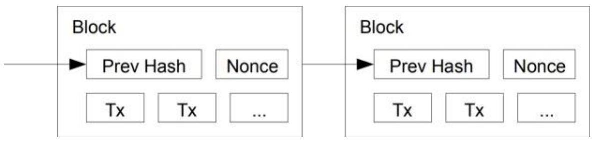

Figure 1. Blockchain Hash Mechanism

A standard, much-discussed usage of distributed systems is blockchain. A blockchain is a distributed record-keeping system structured as a chain of blocks, as seen in Figure 1. Each block contains a hash of the previous block, which acts as a link to its predecessor in the chain, and some transaction data. Transactions are broadcasted in the system, and once in a while, some node has created a block with the transactions that this node has received.

Despite efficiently broadcast messages to the whole network, a gossip protocol is often used. Each node contacts a select few other nodes that it will communicate with, and they, in turn, communicate with some other nodes. When a message is sent, it is quickly delivered to the network, and eventually, all nodes will have a consistent view of the global state.

Blockchain's properties as a database with no central authority make it optimal for systems where trust plays a central part, such as transaction systems. Bitcoin is one example of a blockchain and possibly the most well-known and widespread. However, many more so-called cryptocurrencies and applications exist in other domains, such as smart contracts [3].

As with many distributed systems, one of the main challenges is that of distributed agreement. Specifically, the blockchain agreement is vital since the network needs to agree upon the next block to be added to the chain. Furthermore, the block contains the transactions which are to be permanently committed to the ledger, and should faulty transactions be allowed therein. Thus, the blockchain would be unusable in practice.

A fork can be started if the system decides on two separate blocks with an identical predecessor. Some protocols allow for forks due to the system's often asynchronous nature as long as they converge to a single truth, while others require a final decision on each block. Forks are unwanted since they split the truth, and as a result, a coin may be spent in both blocks, and one of these blocks has to be invalidated to keep the record consistent and prevent double-spending.

Each block must be chosen correctly and honestly. If there is a flaw in the consensus protocol, an attacker could add invalid blocks giving them an advantage and destroying the system's trust. For example, suppose an attacker could add an incorrect block and have it verified by the system. In that case, they could create arbitrary forks that would render the whole system unusable since it would no longer maintain a single truth, i.e., consensus. Furthermore, the adversary could ignore transactions, and it would be impossible for a user to distinguish this case from the case where the transaction has simply not yet reached the node creating the block.

Threshold Adversarial Model

It is common to define the relationship between correct and faulty nodes in a distributed system using an adversarial threshold model, as outlined by Schmid [4]. The adversary is assumed to control a predefined number of nodes and can use them to break the system and hinder protocol progression. In a blockchain, the ultimate goal for an adversary might be to be able to double-spend or ensure that the wrong fork gets chosen as the system progresses. To easily be able to resonate about these thresholds, a notation clearly outlining the threshold is needed. This model introduces two sets of nodes, namely n and f, which represent the following, n is the total number of nodes in the system and f is the total number of nodes controlled by the adversary.

Expressing the nodes in the system in this way enables the usage of thresholds based on these two sets of nodes. For example, when an adversary controls a minority of the nodes, that would be expressed as $n > 2f$.

Byzantine Fault Tolerance

Protocols that are resilient to failures are said to be fault-tolerant, the desired property of distributed systems. An even stronger resilience guarantee is referred to as Byzantine fault-tolerance, or BFT, which means that the system can run properly even when there are Byzantine nodes present in the system. In the Byzantine agreement, the system can agree on the desired task even when there are Byzantine nodes present.

BFT protocols often use the adversarial threshold model to express the system requirements to correct and Byzantine nodes. As proven by Dolev [5], in order for a distributed system to be BFT, at most one-third of the nodes may be Byzantine. This constraint expressed using the threshold adversarial model results in $n >= 3f + 1$.

Self Stabilization

Self-stabilization is a system property that provides an even stronger fault tolerance since the system can recover from faults without human intervention, which is often desirable for distributed systems. However, some terms need explanation in the context of self-stabilization. Whenever the system executes concerning the system requirements, it is said to be in a legal execution. If the system is in a legal execution, it will continue executing correctly in the absence of faults and thus stay in its legal execution. However, the system might be put into an arbitrary state with stale information and no guarantees other intact program code due to transient faults. A self-stabilizing system is guaranteed to converge to a legal execution and exhibit correct behavior within a bounded number of execution steps, starting in an arbitrary system state [6]. This is a stricter fault model than Byzantine fault-tolerance, yet highly desirable since the system may recover from more severe failures as long as no transient faults occur during the recovery period.

QUIC Protocol

QUIC, a protocol introduced and developed by Google and now standardized by IETF, aims to lower the latency and overhead while implementing the main properties of TCP, TLS, and HTTP. QUIC could thereby provide increased performance for many applications which currently use part of this network stack.

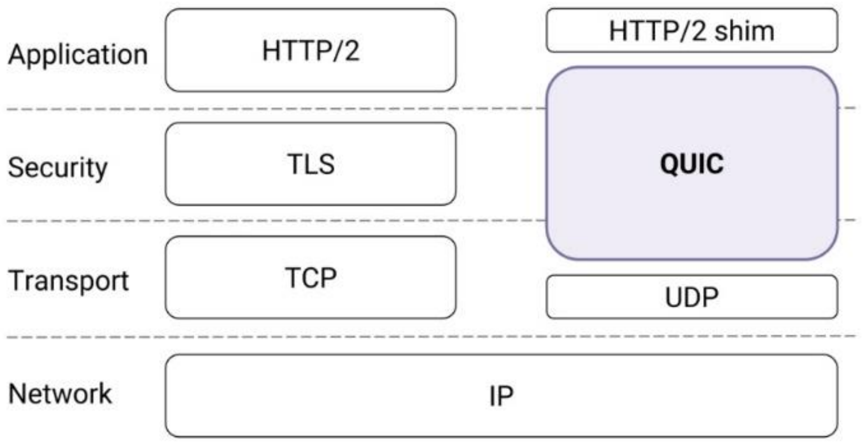

Figure 2. HTTP/3 QUIC Protocol

As outlined in Figure 2, QUIC runs on top of UDP. As a result, it is possible to maintain backward compatibility with the current internet infrastructure while significant changes can be made to the QUIC protocol itself. In essence, QUIC is not too unlike the current HTTP/2 stack as it contains many TLS 1.3 and TCP features. However, using a thin UDP packet for transport, it is possible to encrypt many of the fields that otherwise would be in plaintext for TCP, as these fields are placed in the encrypted QUIC payload instead. Moreover, as the underlying UDP protocol is session less, QUIC can enable a customized session initiation procedure that dramatically lessens the initial setup time than TCP. Specifically, for TCP, the handshake takes three Round-Trip Times (RTT), which is improved to one RTT for QUIC, and even down to zero if the server has been communicated earlier. Another benefit with the use of the minimal transport layer protocol UDP is that most of the protocol stack can reside in user space. Thereby, it is much easier to customize the QUIC protocol than if vital pieces were implemented in kernel space, as would be for TCP. However, this move to user space comes with a slight CPU overhead.

Additionally, a QUIC session can consist of multiple data streams with its buffer, which can have some benefits for lossy links [7]. Finally, with QUIC, multipath routing can be introduced more efficiently, proposed with the modified protocol MPQUIC [8]. This modification to QUIC provides much-increased performance compared to the earlier multipath TCP implementation (MPTCP), both as the handshake for each connection is faster and the protocol itself is more flexible [9].

In practice, QUIC performs worse than HTTP/2 in low latency networks. However, the protocol excels in lossy or high latency environments where it outperforms HTTP/2 in both speed and latency [10][11].

STATE OF THE ART AGREEMENT PROTOCOL

This section is intended to summarize three of the leading state-of-the-art agreement protocols by providing high-level descriptions of the various protocols. First, an overview of Practical Byzantine Fault Tolerance (PBFT) is presented before Nakamoto Consensus , and BA iis discussed. The reason for selecting these three protocols, in particular, is that PBFT was the first protocol to show practical usage of BFT protocols, Nakamoto Consensus is used in the popular cryptocurrency Bitcoin, and BA is used by the first pure Proof-of-Stake (PoS) blockchain Algorand.

Practical Byzantine Fault Tolerance

Another application of the agreement, as part of a state machine replication algorithm, agreeing on the current state, is Practical Byzantine Fault Tolerance (PBFT), introduced by Castro and Liskov. The first work showed practical usage of BFT algorithms for asynchronous systems, where up to one-third of the number of nodes are allowed to act Byzantine. In addition, they showcased the performance of their protocol by implementing a Byzantine fault-tolerant network file system. Their evaluation results showed a mere 3% performance degradation for their Byzantine fault-tolerant implementation, emphasizing that BFT systems can be used in practice.

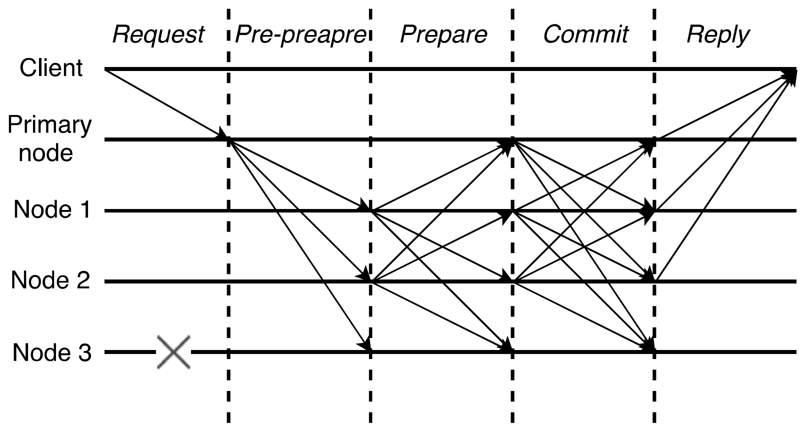

Figure 3. PBFT Consensus

PBFT makes use of a three-phase commit protocol orchestrated by a leader to agree on the ordering of requests. Even though the leader is mainly responsible for system progression, quorums are used throughout the agreement of re- quests. Then the leader makes sure that enough nodes agree on what operation should be executed next. This means that a certain number of nodes need to agree on a task before the system continues, and the size of this quorum is set to make sure that the Byzantine nodes cannot corrupt the system. PBFT is proven to provide safety when at most a third of the total number of nodes are Byzantine, which can be expressed as n>3f+1 using the adversarial threshold model introduced.

One usage of quorums is whenever a client sends a request, and it waits for $f + 1$ identical responses in order to make sure that the result obtained has been carried out by correct nodes . Should the leader be suspected as faulty by enough correct nodes in the system, a leader replacement process is triggered, which elects a new leader through a predefined scheme.

Nakamoto Consensus

Nakamoto Consensus is the agreement protocol used in Bitcoin. It uses a concept called Proof-of-Work (PoW) to ensure that it is challenging to modify a record retroactively. Proof of Work is a consensus that maintains block in the chain contains data and the hash from the previous block linking them together, as shown in Figure 4. No hash will be accepted; it needs to start with a predetermined number of zeroes defined by the protocol. The block's hash is created with a one-way hash function that uses the previous block's hash, the current block's transactions, and a random nonce. The only way to find this hash, in practice, is to try a large number of nonces, which is quite tricky and requires much guessing before the correct hash is found. By making sure that work is needed to find the next hash, this concept is called a PoW. The difficulty can be tuned by adjusting the number of zeroes required; the more that is needed, the fewer hashes will exist, and thus the difficulty to find the nonce increases. The difficulty depends on the number of nodes in the network so that the expected time required to find the next block is kept nearly constant.

If an attacker modifies a block, the hash will change, and thus, they would have to change the hash of every subsequent block. However, if many other blocks succeed a block, it is regarded as practically infeasible to modify it, and therefore the integrity of the block is considered safe.

Transactions are broadcasted to every node in the network, and any node can propose to add their block of transactions to the chain. To propose, the node needs to find a PoW for the current set of transactions it has received. When a block is proposed, it is also broadcasted, and nodes receiving it will quit working on finding a PoW for their current block and instead start working on a new block. Before doing this will validate all transactions in the block received so that none of the transactions interferes with the previous. This means that any proposed block is a validation of all previous blocks in the chain. When starting to find a PoW, the nodes choose the end of the longest chain with valid transactions as the predecessor since that block has the most work put into it, and as such, most nodes agree on it. Validating all transactions before finding the PoW makes it improbable for a maliciously created block to be part of the final chain.

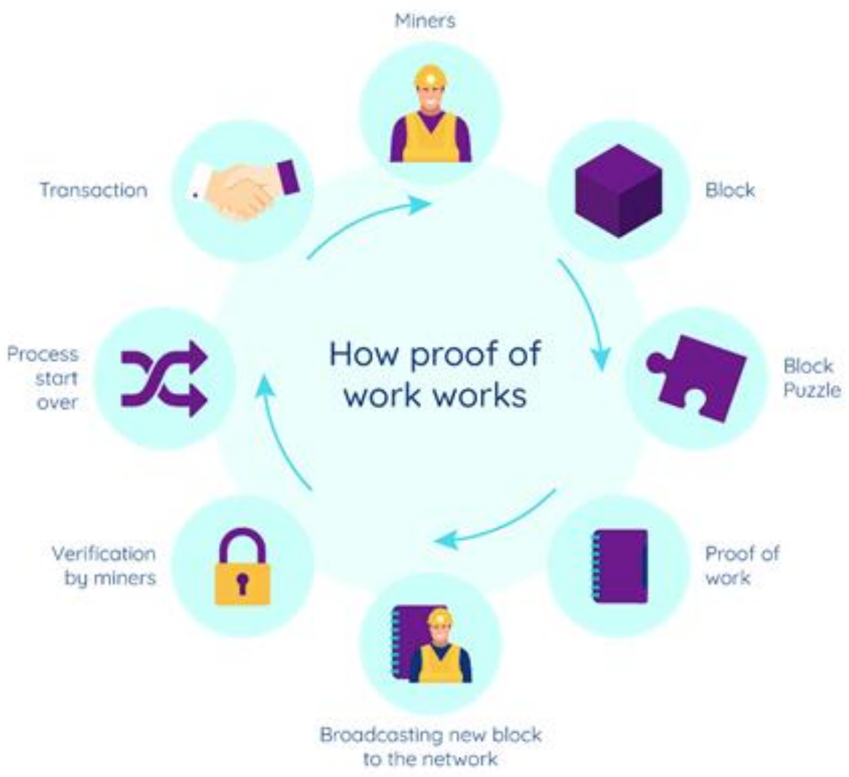

Figure 4. Proof of Work Mechanism [12]

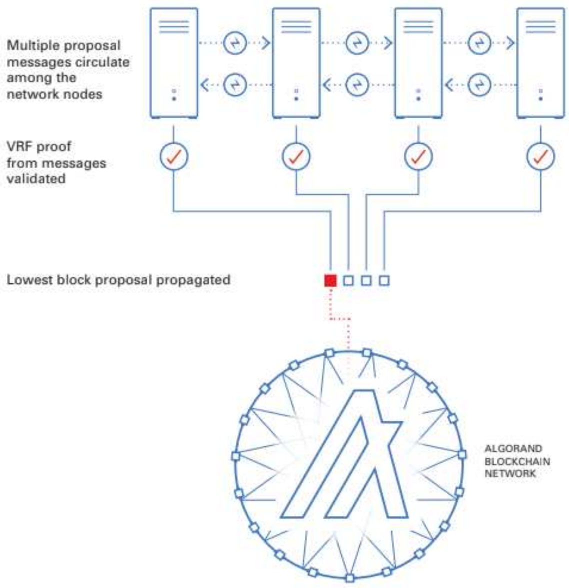
Multiple proposal messages circulate among the network nodes
VRF proof from messages validated
Lowest block proposal propagated
ALGORAND BLOCKCHAIN NETWORK

Figure 5. Byzantine Agreement with Pure Proof of Stack Consensus [13]

However, if an adversary controls more than half of the network's processing power, it can create blocks faster than the honest nodes. This means that it has complete control of the blockchain and can create arbitrary blocks with arbitrary transactions. With an extensive system like Bitcoin, this is unlikely to occur but still possible.

With this in mind, a Byzantine fault-tolerant consensus protocol is less resilient against majority attacks since an attacker needs to gain control of fewer nodes than a substantial majority (one-third is enough). Still, they have other properties that are desired.

Byzantine Agreement (BA)

Instead of relying on PoW for safety, one can use a byzantine agreement protocol called BA, which Gilad et al. in Algorand, as shown in Figure 5. This protocol guarantees that the system will reach consensus given that 2/3 of the stake is possessed by honest nodes. Instead of proposing a block with PoW, the protocol uses cryptographic sortition. This means that with a given random seed, each node can determine if they have been chosen to propose a block or not. The sortition "chooses" a small fraction of nodes each round, and each of these nodes will broadcast their block together with proof of their participation. They participate in the block proposal with a single message, eliminating an attack against a selected node. In the same message, each node also includes a proposal for a new random seed used in the sortition for the next round. Each node computes this seed with a Verifiable Random Function, which uses the current seed and round number as input. This makes it difficult for an adversary to be part of the committee more often to affect the seed [14].

When blocks have been proposed by the selected nodes (called the committee), a vote is started. Each voting step starts, as the block proposal, with sortition. Only a selected few can vote in each step, and they vote by broadcasting their selection of the block that should be added, which it will choose depending on the priority of the blocks received. This is where PoS is used; priority is determined depending on the stake of the block's sender [15]. A user with a higher stake is deemed more honest since they have invested more in the system. Again, this is a single message containing the vote together with proof of participation. Since all blocks might not reach all nodes, the committee members might vote on different blocks due to the gossiping protocol and not reach a consensus. Therefore, this is repeated until some threshold of committee members reach consensus.

The sortition is needed to prevent a committee member from being targeted by a malicious actor and make it difficult for an adversary to affect their election. If the committee members were chosen in advance, it would be possible for malicious users to prevent them from sending the message. Preventing everyone from sending messages is infeasible, but targeting a single node could be done relatively quickly with some Denial of Service attack. The protocol cannot, and should not, guard a specific node against an attack but rather protect the protocol and system as a whole. Therefore, each node can verify whether they are in the committee or not, and they only need to send one message to participate. Each node needs its private key to verify the participation, and this makes targeted attacks very difficult.

Since there is no need for processing power to create a block (as in Nakamoto Consensus), there needs to be another mechanism to prevent an adversary from creating multiple clients and increase the probability of being in the committee in each step. As mentioned before, BA uses the concept of PoS, which has previously been found difficult to use in practice. Stakeholders can try to invalidate a branch without losing any resources, among other reasons [16]. While many attempts to use PoS have tried to replace PoW directly, BA uses it only for sortition. The PoS is, as such, not responsible for the safety of the protocol. It should be noted that the probability of being selected may be heavily increased should a malicious actor invest a significant amount of money in the system, which is acknowledged by Gilad et al.

PBFT laid the foundation for continued work, and as outlined by Natoli et al., blockchain solutions such as PeerCensus [17] and ByzCoin [18] utilize agreement protocols based on PBFT. These protocols are designed to function in partially- synchronous environments [19] rather than asynchronous such as PBFT.

DISCUSSION

This section starts by comparing Bitcoin and Algorand, which use two of the surveyed protocols. After that, the possible introduction of self-stabilization to the surveyed protocols is discussed before a more practical approach to optimize using the transport layer network protocol QUIC is presented.

Algorand & Bitcoin

The availability and widespread usage of Bitcoin make it difficult for any other cryptocurrency to enter the market. However, with its flaws, there are many opportunities for new protocols and systems such as Algorand, which tackles several of the more discussed problems of Bitcoin. As previously mentioned, Bitcoin uses PoW, and while this is a suitable mechanism for safety and immutability, it requires many resources. According to a study conducted by De Vries, the Bitcoin network consumes at least 2.55 gigawatts (at the time of publication) which is comparable to Ireland's consumption, which is about 3.1 gigawatts [20]. This has, of course, raised a debate whether or not it is a practical system, and it has received much criticism as other, primarily non-blockchain systems, use a fraction of that power [21].

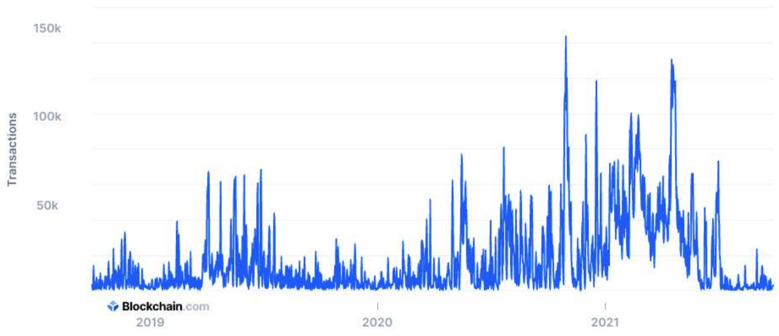

Figure 6. Unconfirmed Transaction at 2009-2021.

Algorand solves this issue by not using PoW while still being able to retain distributed properties and safety. It also solves other issues with Bitcoin, namely transaction throughput. The maximum size of a block is 1MB in Bitcoin, making scaling a bit difficult since the block time has to be constant. Experiments carried out by Gilad et al. suggests that Algorand has a throughput that is 125 times higher than Bitcoin. As can be seen in Figure 3, Bitcoin has, at times, big problems with throughput. In this diagram, the number of unconfirmed transactions peaks at over 17 000, but historically this number has reached over 80 000. Since block size and block time are both fixed, this might look even worse in the future if the use of Bitcoin grows further. Another situation where Algorand shines compared to Bitcoin is in confirmation speed: a transaction can be confirmed in the order of minutes. This is important if the system is ever intended to be used for financial transactions. Although it is still prolonged compared to current card transactions, it is much faster than Bitcoin, which needs several hours to confirm a transaction [22]. For Bitcoin, six blocks are deemed necessary since an entity with 40% computing power has less than a 50% chance of succeeding in an attack. Each block takes on average ten minutes to complete, but a single block can take much longer as some have taken more than sixty minutes [23]. The Unconfirmed Transaction from 2009-2021 is depicted in Figure 6.

One thing that favors Bitcoin is its simplicity; with some knowledge in cryptography, it is trivial to understand why it works and is safe. Algorand, on the other hand, is quite complex, and one can not expect many users to understand it fully. Instead, the users have to trust the community to verify its security which many people do not mind, given that they already trust the third party to handle their money [24].

With this in mind, Algorand seems like a very potent substitute or addition to Bitcoin. However, a thorough review and performance comparison is still needed, and it remains to be seen how well it works in practice under the same conditions.

Self-Stabilization of Agreement Protocols

This section outlines ideas and suggestions relevant to introducing the self-stabilization property to two agreement protocols: BA and PBFT. We chose not to look at Nakamoto Consensus as well in this regard because introducing more overhead as part of the self-stabilization would most likely not be worth it due to poor performance. As we have shown earlier, Bitcoin already suffers performance problems, and other blockchain technologies seem more promising going forward. We, therefore, chose to focus on the other two protocols [25].

One could argue that introducing self-stabilization to the protocol would be beneficial to make BA even more resilient to failures and malicious behaviour. Other self-stabilizing Byzantine Agreement protocols exist [26], such as the one introduced by Daliot and Dolev [27]. Self-stabilization often comes with some form of overhead, though, so the added resilience and robustness will most likely yield a performance degradation of some kind. Further work presented by Daliot and Dolev suggests a way of "converting" Byzantine Fault-Tolerant (BFT) protocols to self-stabilizing versions while only introducing an overhead of O(f t) communication rounds, where f t is the number of actual faults [28]. BA might be extended using some of the techniques presented in and make it even more resilient to failures and malicious behavior while not adding too much communication overhead. Naturally, this added resilience would need to be efficient enough not to slow down the system too much since one of the main selling points of the blockchain using BA, Algorand, is its speed and efficiency.

PBFT is a well-known BFT protocol and has consequently laid the foundation for continued research based on PBFT and self-stabilization.

Dolev et al. introduced the first self-stabilizing BFT replicated state machine based on failure detectors inspired by PBFT [29][30]. Since it is based on self-stabilizing failure detectors, this solution can provide self-stabilizing BFT with weaker synchrony requirements than when using, for example, clock synchronization. A preliminary evaluation of the protocol carried out by Niklasson and Petersson [31, 32, 33] validated the self-stabilizing property of the system. The evaluation also showed promising results for practical usage for certain types of state machines, further indicating that adding the self-stabilizing criteria to a system such as PBFT is practical and desirable. Given that the evaluation of the protocol introduced by Dolev et al. was performed as a Master's Thesis and considered preliminary, a more full-blown evaluation of the system would be interesting. Should the results from such an evaluation be promising, the barrier for introducing self-stabilization to this kind of system could be lowered.

HTTP/3 Approach

We believe that the current internet protocol stack, most significantly TCP, is holding back some blockchain implementations. TCP is beholden to its legacy specifications, which can not be changed due to the plethora of middleboxes and other networking devices that may drop unknown protocols. Therefore, TCP's issues will remain; TCP has a slow handshake, its proposed multipath implementation is complicated and inefficient, and the protocol itself is inflexible and slow to modify. Here, QUIC shows great promise, which is outlined in the next section.

It is possible to replace the underlying protocol stack To increase the performance of consensus protocols. Here we will look at TCP, which is widely used in blockchain protocols such as Bitcoin and Ethereum.

In peer-to-peer networks, such as the consensus protocols looked at in this survey, the expected behavior is to talk to many different nodes very briefly. For BA, communication between nodes often includes only a single message. Here QUIC could provide a noticeable improvement as the initial handshake requires RTT instead of three compared to TCP. Moreover, QUIC outperforms TCP in environments with high loss, high latency, and high congestion, which may arise in peer-to-peer networks due to the geographically sparse distribution, and low-cost nodes may be used [34].

As mentioned in previous section, multipath routing in QUIC is much faster than MPTCP. We believe that this addition could provide better performance when sending large datasets, such as when the client is initially synchronizing (downloading) the blockchain ledger. This lowers the difficulty of entering the network, which might encourage additional users to join, making the system increasingly robust, as more honest nodes would be present [35]. QUIC has another benefit; namely, it is more configurable than TCP - both as it implements more features above the transport layer protocol (in this case, UDP) and as it runs in userspace. Thereby, blockchain-specific implementations can more easily be created. Although this move to user space comes at a CPU overhead, possibly not an issue in modern devices.

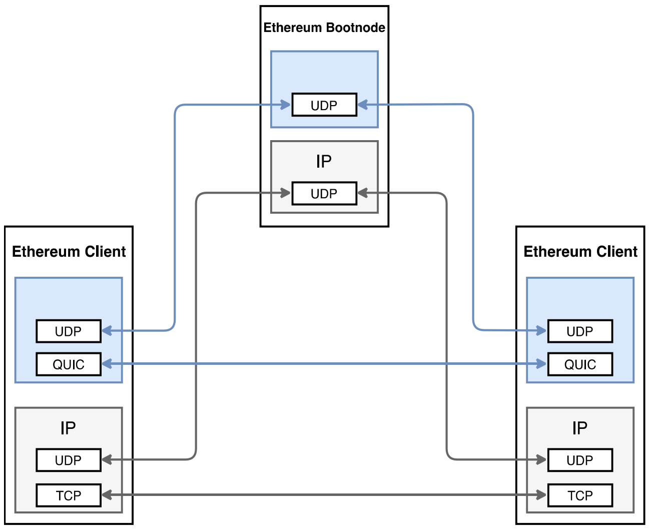

Figure 7. Implementation of QUIC Protocol at Ethereum Ledger using PoW mechanism

Consensus Comparison

According to section 3, we can compare that some consensus works as listed in Table 1.

Table 1. Consensus Usage

Name	PoW	BA (PoS)	PBFT
Usage	Ethereum, Bitcoin	Algorand	ByzCoin
Fault Tolerance	50%	50%	33%
Power Consumption	Large	Less	Negligible
Type	Probabilistic-finality	Probabilistic-finality	Absolute-finality

RELATED WORK

Previously, a study was conducted by Aleksandar Vorkapic in order to build some proof of concept to secure communication networks in BGP routing and blockchain networks [36]. It has several VM for the different regions, which are expected to have a more secure platform by combining UDP and TCP as QUIC protocol did for Ethereum that use a Proof-of-Work mechanism called SCION project [26], as shown in Figure 8 and listed in Table 2 and Table 3.

Table 2. IP Performance

IP	Germany	USA	South Korea
Avg RTT (ms)	28.781	126.005	287.131
Std dev RTT (ms)	1.395	8.947	15.597
Avg hops	15.2	18	18.6
Std dev hops	1.4	0.0	0,9
Packet loss (%)	1.4	0.0	0.0

Table 3. SCION Performance

SCION	Germany	USA	South Korea
Avg RTT (ms)	62.568	150.734	301.898
Std dev RTT (ms)	3.246	6.213	6.247
Avg hops	7.7	7.7	8.8
Std dev hops	0.5	0.9	0.7
Packet loss (%)	0.0	0.0	0.0

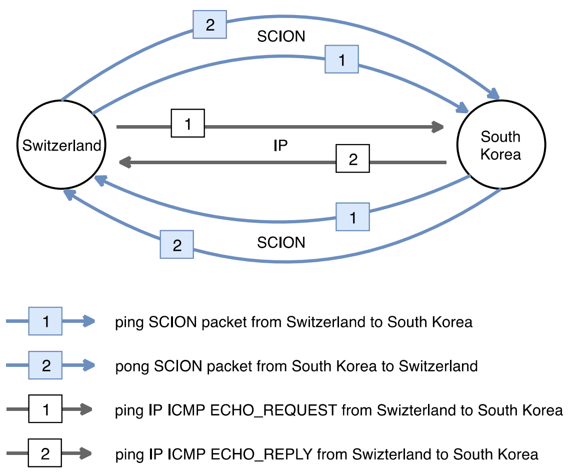
SCION
2
1
Switzerland
1
IP
2
South Korea
1
2
SCION
1
ping SCION packet from Switzerland to South Korea
2
pong SCION packet from South Korea to Switzerland
1
ping IP ICMP ECHO_REQUEST from Switzerland to South Korea
2
ping IP ICMP ECHO_REPLY from Switzerland to South Korea

Figure 8. Scion vs. ordinary IP Protocol

Implementation of Covid-19 Program

Refer to Figure 7, we implemented blockchain protocol on our product called PlasmaNation [36], which enables Patient that suffer from covid-19 to get some Plasma Donation from anyone who just gets recovered from this disease. This site uses smart contracts to enable an incentivized mechanism for successful donors by participating in some business sponsors. In addition, it will enable a speed recovery for covid-19 economic using fair mechanism whereas vaccines were not available during the day of the beginning phase of a virus outbreak.

CONCLUSION

This survey has presented three state-of-the-art protocols for reaching Byzantine Agreement: PBFT, Nakamoto Consensus, and BA (PoS). Both Nakamoto Consensus and PBFT are widely used Today; Bitcoin utilizes Nakamoto Consensus, and PBFT has laid the foundation for various other projects, ranging from BFT protocols to cryptocurrencies. BA is used in the PoS cryptocurrency Algorand. A comparison of applications that use BA (Algorand) and Nakamoto Consensus (Bitcoin) has been presented, along with ideas of making protocols more resilient to failures through self- stabilization. The comparison between BA and Nakamoto Consensus showed that Algorand is a serious contender even though more thorough performance evaluations are needed.

A more practical approach to optimize the protocols with the introduction of QUIC has also been discussed by the SCION project. SCION resulted slightly slower than ordinary IP-Based Performance for Proof of Work consensus. However, there has still needed more practical investigations of the protocols and optimization possibilities for another consensus, such as BA (using PoS) and PBFT, which remain unexplored.

When looking at the improvement ideas presented in Sections 4.1 and 4.2, it becomes clear that more thorough, in-depth research is needed to investigate the possible performance gains. For instance, comprehensive performance evaluations of these research areas are needed to fully grasp the proposed solutions' validity. If it were possible to add self-stabilization to blockchain solutions while keeping the performance high enough for practical usage, that would naturally be very desirable and make blockchain even more attractive for various applications. It could also be interesting to investigate further how much performance could be increased in popular agreement protocols if QUIC were introduced. Given that it reduces the number of round-trips needed, some algorithms might be re-designed to utilize this and consequently exhibit better performance fully.

Furthermore, the development of multipath transport layer protocols may be put into practice with QUIC. Where the userspace protocol enables a solution to much easier be put into production. The benefits of multipath routing only become apparent when much data traverses the connection, thereby showing benefits only for some blockchain implementations. Finally, QUIC is a reasonably new protocol and might need to be researched further. For instance, there might be environments where this protocol is currently not allowed, and can the increased CPU overhead be an issue for blockchain applications.

REFERENCES

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2009

[2] M. Pease and L. Lamport, “Reaching Agreement in the Presence of Faults,” *Journal of the AssoctaUon for Compmmg Machinery*, vol. 27, no 2, pp. 228-234, 1980

[3] D. Wood, “Ethereum: A Secure Decentralised Generalised Transaction Ledger,” *Undefined*, 2014.

[4] R. W. Ahmad, H. Hasan, I. Yaqoob, K. Salah, R. Jayaraman, and M. Omar, “Blockchain for aerospace and defense: Opportunities and open research challenges,” *Computers & Industrial Engineering*, vol. 151, ID: 106982, 2021, doi: 10.1016/j.cie.2020.106982

[5] D. Dolev, “The Byzantine generals strike again,” *Journal of Algorithms*, vol. 3, no. 1, pp. 14-30, 1982, doi: 10.1016/0196-6774(82)90004-9

[6] S. Dolev, *Self-Stabilization*, MIT Press, UK, 2000

[7] A. Langley et al., “The QUIC Transport Protocol: Design and Internet-Scale Deployment,” *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, August 2017, pp. 183-196, doi: 10.1145/3098822.3098842v

[8] Q. De Coninck and O. Bonaventure, “Multipath QUIC: Design and Evaluation.”, *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, 2017, pp. 160-166, doi: 10.1145/3143361. 3143370

[9] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe and R. Steinmetz, "Multipath QUIC: A Deployable Multipath Transport Protocol," *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1-7, doi: 10.1109/ICC.2018.8422951

[10] G. Carlucci, L. De Cicco, and S. Mascolo, “HTTP over UDP: An experimental investigation of QUIC,” in *Proceedings of the ACM Symposium on Applied Computing*, 2015, vol. 13-17-April-2015, pp. 609–614, doi: 0.1145/2695664.2695706

[11] C S. Cook, B. Mathieu, P. Truong and I. Hamchaoui, "QUIC: Better for what and for whom?" *2017 IEEE International Conference on Communications (ICC),* 2017, pp. 1-6, doi: 10.1109/ICC.2017.7997281

[12] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling Byzantine Agreements for Cryptocurrencies,” *Proceedings of the 26th Symposium on Operating Systems Principles*, October 2017, pp. 51-68, 2017, doi: 10.1145/3132747. 3132757

[13] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” *Proceedings of the Third Symposium on Operating Systems Design and Implementation,* New Orleans, USA, 1999, pp. 1-14

[14] S. M. Fati, A. Muneer, N. A. Akbar, and S. M. Taib, “A Continuous Cuffless Blood Pressure Estimation using Tree-Based Pipeline Optimization Tool,” *Symmetry*, vol. 13, no. 4, pp. 686, 2021, doi: 10.3390/ sym13040686

[15] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” in *Lecture Notes in Computer Science*, vol. 10401, Springer, Cham 2017

[16] I. Bentov, A. Gabizon, and A. Mizrahi, “Cryptocurrencies without Proof of Work,” Clark J., Meiklejohn S., Ryan P., Wallach D., Brenner M., Rohloff K. (eds) *Financial Cryptography and Data Security*. FC 2016. Lecture Notes in Computer Science, vol

9604, Springer, Berlin, Heidelberg, doi: 10.1007/978-3-662-53357-4_10
[17] C. Decker, J. Seidel, & R. Wattenhofer, “Bitcoin Meets Strong Consistency,” *ACM International Conference Proceeding Series*, 04-07-January-2016. Retrieved from http://arxiv.org/abs/1412.7935
[18] E. K. Kogias et al., “Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing,” *Conference of USENIX Security Symposium 2016*, 2016
[19] C. Natoli, J. Yu, V. Gramoli, and P. Esteves-Verissimo, “Deconstructing Blockchains: A Comprehensive Survey on Consensus, Membership and Structure,” *arXiv*, Aug. 2019. S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2009
[20] A. de Vries, “Bitcoin’s Growing Energy Problem,” *Joule*, vol. 2, no. 5. pp. 801–805, May 2018, doi: 10.1016/j.joule.2018.04.016
[21] C. S. You, J. S. Yeom, and B. C. Jung, “Cooperative maximum-ratio transmission with multi-antenna relay nodes for tactical mobile ad-hoc networks,” *ICT Express*, vol. 6, no. 2, pp. 87–92, Jun. 2020
[22] NN, *Blockchain Charts*, [Online]. Available: https://www.blockchain.com/en/charts/mempool-count?timespan=24h. [Accessed: 14-Apr-2021]
[23] L. Wang, X. Shen, J. Li, J. Shao, and Y. Yang, “Cryptographic primitives in blockchains,” *Journal of Network and Computer Application*, vol. 127, pp. 43–58, Feb. 2019
[24] C. V. Helliar, L. Crawford, L. Rocca, C. Teodori, and M. Veneziani, “Permissionless and permissioned blockchain diffusion,” *International Journal of Information Management*, vol. 54, Oct. 2020
[25] D. Berdik, S. Otoum, N. Schmidt, D. Porter, and Y. Jararweh, “A Survey on Blockchain for Information Systems Management and Security,” *Information Processing & Management*, vol. 58, no. 1, 2021, doi: 10.1016/j.ipm.2020.102397
[26] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982
[27] A. Daliot and D. Dolev, “Self-stabilizing Byzantine Agreement,” *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, 2009, pp. 143-152, doi: 10.1145/ 1146381.1146405C
[28] S. You, J. S. Yeom, and B. C. Jung, “Cooperative maximum-ratio transmission with multi-antenna relay nodes for tactical mobile ad-hoc networks,” *ICT Express*, vol. 6, no. 2, pp. 87–92, Jun. 2020
[29] A. Daliot and D. Dolev, “Self-stabilization of Byzantine protocols,” in *Lecture Notes in Computer Science*, vol. 3764 LNCS, pp. 48–67, 2005, doi: 10.1007/11577327_4
[30] S. Dolev, C. Georgiou, I. Marcoullis, and E. M. Schiller, “Self-stabilizing Byzantine Tolerant Replicated State Machine Based on Failure Detectors,” In: Dinur I., Dolev S., Lodha S. (Eds) *Cyber Security Cryptography and Machine Learning*. CSCML 2018, vol 10879. Springer, Cham, doi: 10.1007/978-3-319-94147-9_7
[31] S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services,” *ACM SIGACT News*, vol. 33, no. 2, pp. 51-59, Jun. 2002
[32] N. A. Akbar, A. Sunyoto, M. Rudyanto Arief and W. Caesarendra, "Improvement of decision tree classifier accuracy for healthcare insurance fraud prediction by using Extreme Gradient Boosting algorithm," *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, 2020, pp. 110-114, doi: 10.1109/ICIMCIS51567.2020.9354286
[33] A. Langley et al., “The QUIC Transport Protocol: Design and Internet-Scale Deployment,” *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, August 2017, pp. 183-196, doi: 10.1145/3098822.3098842v
[34] R. Wasim Ahmad, H. Hasan, I. Yaqoob, K. Salah, R. Jayaraman, and M. Omar, “Blockchain for aerospace and defense: Opportunities and open research challenges,” *Computers & Industrial Engineering*, vol. 151, ID: 106982, 2021, doi: 10.1016/j.cie.2020.106982
[35] S. Zhang and J. H. Lee, “Analysis of the main consensus protocols of blockchain,” *ICT Express*, vol. 6, no. 2, pp. 93–97, 2020, doi: 10.1016/j.icte.2019.08.001
[36] R. Wasim Ahmad, H. Hasan, I. Yaqoob, K. Salah, R. Jayaraman, and M. Omar, “Blockchain for aerospace and defense: Opportunities and open research challenges,” *Computers & Industrial Engineering*, vol. 151, ID: 106982, 2021, doi: 10.1016/j.cie.2020.106982