Delft University of Technology student report

Comparative Study of Byzantine Fault Tolerant Consensus Algorithms on Permissioned Blockchains

April 2020

Isitan Gorkey
EEMCS
TU Delft
Delft, Netherlands
I.Gorkey-1@student.tudelft.nl

Chakir El Moussaoui
EEMCS
TU Delft
Delft, Netherlands
C.ElMoussaoui@student.tudelft.nl

Vincent Wijdeveld
EEMCS
TU Delft
Delft, Netherlands
V.A.E.Wijdeveld@student.tudelft.nl

Erik Sennema
EEMCS
TU Delft
Delft, Netherlands
E.J.Sennema@student.tudelft.nl

Supervised by
Zekeriya Erkin
EEMCS
TU Delft
Delft, Netherlands
Z.Erkin@tudelft.nl

Supervised by
Miray Aysen
EEMCS
TU Delft
Delft, Netherlands
M.Aysen@tudelft.nl

***Abstract*—Blockchain is a technology that is in use increasingly. Although owing its common use to the cryptocurrencies there is more to the blockchain technology than just the monetary use. One of these uses is by smaller groups of participants under the control of a central authority, for instance at a company. Such private blockchains use permissioned blockchain consensus algorithms as the participants need the permission of the authority to be able to join the system. This paper will give an overview of the blockchain technology, investigate permissioned and permissionless blockchain, and focus on permissioned blockchains to analyze it in terms of, e.g. trust models between the nodes, incentives, number of nodes & parties involved, and scalability regarding the number of transactions.**

***Index Terms*—Blockchain, Byzantine Fault Tolerance, Consensus Algorithms, Cryptocurrency, Permissioned**

I. Introduction

Blockchain is a technology that is being used in increasingly more platforms [1]. The idea of blockchain was first introduced by Haber and Stornetta in 1990 [2], the first conceptualization of it did not happen until 2008 when Nakamoto applied blockchain as the underlying technology of Bitcoin [3]. Bitcoin made it possible to make safe monetary transactions without the need for a central authority such as a bank, which remarked the strength of blockchain technology. In the following years, blockchain technology has been used in more fields from payment processing to online voting.

The main research question of this paper is: How the consensus algorithms that are used in permissioned blockchains compare against each other. The subquestions investigated to answer this question are: How do consensus algorithms compare against each other in the means of incentives, number of nodes & parties involved, scalability regarding the number of transactions, and trust models between the nodes. The process of answering the research question consists of first giving an overview of the blockchain. Then the consensus algorithms used in blockchain technologies will be discussed followed by the trust models between nodes, the incentive mechanisms, and scalability analysis. Lastly, the comparison of permissioned consensus algorithms will be discussed and a conclusion will be drawn to the paper.

II. An overview of blockchain

Blockchain [4] is a sequence of blocks, where each block keeps a list of transactions. Each block consists of a header and a body. The header contains the following information about the block which can be seen in Figure 1:

- Block version: indicates the set of block validation rules to follow.
- Merkle tree root hash: the hash value of the Merkle tree root, which is recursively derived from the hash value of the transactions in the block, as can be seen in Figure 1.
- Timestamp: the time passed in seconds from January 1, 1970 until the creation of the block (universal time).
- nBits: the target threshold of a valid hash. The generated hash needs nBits or a lower amount of 0's at the beginning of the generated hash.
- Nonce: a 4-byte field, which usually starts with 0 and increases for every hash calculation.
- Previous block hash: a 256-bit hash value that points to the previous block.

The hash of the previous block is contained in the block header, a block has only one previous block. The first block of a blockchain is called the genesis block. The genesis block has no previous block.

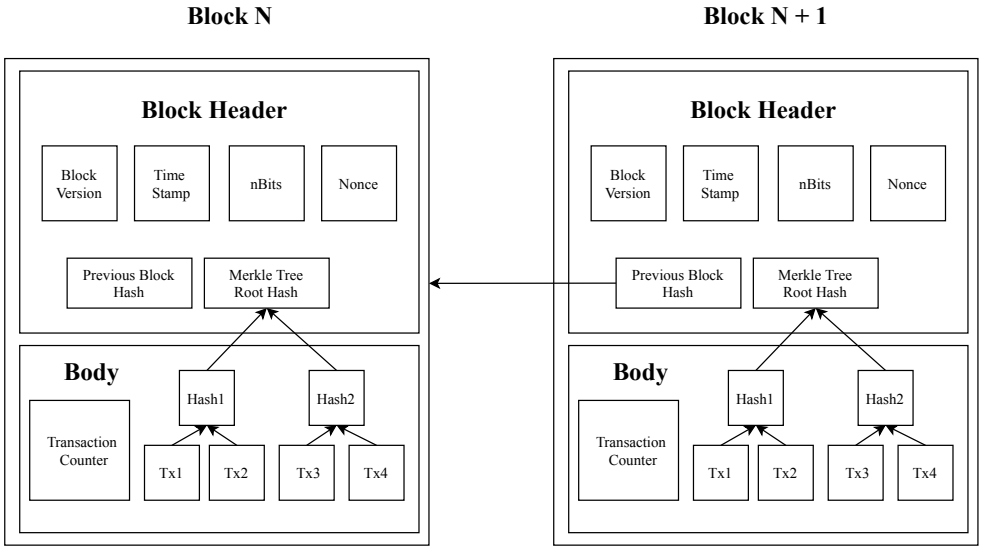

Fig. 1. An example of a block structure in a blockchain

The body consists of a transaction counter and a list of transactions. The maximum number of transactions that a block can contain depends on the maximum block size and the size of the transactions that is set for the specific implementation of blockchain. Bitcoin, for example, has a maximum block size of 1 MB and a block is mined every 10 minutes on average. This results in a maximum processing time of 7 transactions per second. One solution to increase the transaction throughput is to make use of larger blocks. This, however, means that if all blocks are filled the blockchain will grow twice as fast as before. This can make it significantly more costly to run a full bitcoin miner node in the future. Doubling the block size can thus result in a higher barrier of entry for node administrators and lower the number of nodes ran on the network. This could cause centralisation which is undesirable as it is contradictory to the decentralized nature of the system.

A. Permissionless Blockchain

Permissionless blockchains [5] are open for any entity to participate in by running a local instance of the protocol using their digital signature (public key). There is no need for participants to publicize their real identities. The first permissionless blockchain was Bitcoin [3].

There are some advantages to the fully replicated immutable data store of the permissionless blockchains and their open nature, which are [5]:

1) No need to have full trust in any single entity in the network or a third party.
2) Each entity in the network is identified by its digital signature. There is no need for real identities. This makes the permissionless blockchains pseudonymous.
3) Blockchains are fully replicated immutable data stores that aim to ensure integrity and non-repudiation.

Permissionless blockchains do, however, suffer from several drawbacks due to their fully replicated and open nature [5]:

1) They have low performance in terms of the ability to process a high throughput of transaction requests. For example, Bitcoin is currently able to process only about 7 transactions per second (TPS).
2) Data stored in permissionless blockchains is publicly accessible as it is fully replicated among all peers. Without implementing additional measures on top of the original protocol, permissionless blockchains suffer from confidentiality issues.
3) Due to forking, permissionless blockchains with the consensus protocols Proof of Work, Proof of Stake and Proof of Authority never reach consensus finality, that is, they have forks resulting in more waiting time for a block to be deemed valid as they go by one of the chains of the fork [6]. This chain could be chosen by different parameters based on implementation, such as the length of the chain. When forking is possible, transactions committed in dropped chains of the fork might be revoked.

To avoid these drawbacks, permissioned blockchains were introduced.

B. Permissioned Blockchain

Permissioned blockchains have evolved as an alternative to permissionless blockchains. In [7] Vukolić stated that ”this happened in order to address the need for running blockchain technology among a set of known and identifiable participants that have to be explicitly admitted to the blockchain network”.

The permissioned blockchain [7] concept is best used in business applications of the blockchain technology. Even though the nodes do not necessarily trust each other or any

third parties, they are still required to identify themselves, which is usually the case within business circles. The notion of smart contracts [5] was introduced by Ethereum [8] and are used in applications that require trustworthy and decentralized execution of business logic involving multiple entities.

The reason permissioned blockchains provide enhancements over their permissionless counterparts is that they facilitate enterprise-grade use cases [5], such as:

1) The participation in the consensus protocol is limited to a specific group of nodes that require explicit system configuration in terms of permission from the authority, software, and hardware. This results in permissioned blockchains being able to use Byzantine Fault Tolerant (BFT) protocols, which generally have better throughput and transaction latency. [9].
2) Falazi et al. [5] noted that "Permissioned blockchains are generally better in terms of confidentiality since sensitive transactions can be isolated from public access."
3) For permissioned blockchains forks and non-final decisions could be avoided. For example, Byzantine consensus algorithms reach consensus with finality [10].

Note that, permissioned blockchains have similar drawbacks and limitations as their permissionless counterparts that are not desirable for the use of business applications. Also, the enhancements for permissioned blockchains come with a price [7][5]:

1) These enhancements reduce decentralization in the blockchain networks, as user roles and participation is decided by an administrative entity and transaction validation is done by a predefined set of nodes.
2) Generally within applications, the order of execution of transactions is determined by the consensus algorithms. Therefore the execution is sequential. The effective throughput is inversely proportional to the latency of execution and therefore limited by this latency.
3) It is very hard to change the consensus protocol in blockchains. Changing the consensus protocol under different system conditions can be desirable as these protocols are known to exhibit different performances under different conditions.

III. Consensus Algorithms

In [4] it is stated that "blockchain is a trustless distributed ledger which allows transactions to take place in a decentralized manner". To establish any kind of trust within such networks, the nodes need to reach consensus on which blocks containing transactions are accepted into the distributed ledger. In this section, several algorithms will be discussed to reach such a consensus.

A. Proof-Based Algorithms

In this section different proof-based consensus algorithms will be discussed. In proof-based algorithms miners must prove that they can create a new block. The proof has to be mutually confirmable by the other nodes and the proof should be infeasible to forge.

1) Proof of Work

Proof of Work (PoW) was first introduced by Satoshi Nakamoto in the implementation of the Bitcoin network [3].

Within a decentralized network [4], there needs to be some node in the network that records the transactions, creates a block, and sends it to all other nodes in the network for them to attach it to their own blockchain. The created block will then be recorded by all other nodes in the network. This so called block creation is rewarded with a block reward which will be discussed in section V. The easiest and the fairest way to select such a node is random selection. But randomly selecting such a node is vulnerable to attacks. To avoid this, a node must do a lot of work in the form of computer calculations to be selected. The idea behind this is that a node which has done a lot of work would not be likely to attack the network.

In Bitcoin, each node of the network is calculating a hash value of the block header. This is done by hashing all transactions within a block into a single hash called the Merkle Tree Root Hash. Together with this hash, using the timestamp, the block version, the previous block hash and a random number (nonce) a new hash value is created. The miners change the nonce to get different hash values. In larger mining pools other variables such as timestamps and transactions inside the block can be changed to find a hash value. A certain target value is given by the algorithm in the form of nBits. This target value is the maximum number of 0's the calculated hash needs to start with. Due to the properties of hash functions, it is only feasible to get this sort of hash by randomly trying different values for the nonce, or in the case of larger mining pools, other variables. When a miner finds a hash value meeting this criterion, it broadcasts the block to other nodes. All of the nodes need to validate the generated hash value. If the nodes have confirmed the correctness, then all nodes will append the created block to their own blockchains. It could happen that multiple miners reach the correct target value at roughly the same time. This results in two blocks being sent to the nodes to be validated. The nodes, in turn, create a "fork" of added blocks to their blockchains. They keep adding blocks to the respective chains of the fork. The longer chain is then seen to be the authentic one as there is more work put into it, and the other ones will be disregarded.

Specific to Bitcoin [3], the maximum size of the block is fixed to 1 MB. The difficulty of the calculation is chosen such that it takes 10 minutes on average. Should a node be malicious, by for example adding fake transactions within a block, it must then be able to compete with all other nodes in the network by finding the right target value for several blocks. Since the target value is chosen such that finding it takes 10 minutes on average, it would be inefficient for a

malicious node to alter transactions within a block, as the calculations require intensive computing power. The resources put in would outweigh the benefits.

Implementation of PoW within permissioned blockchains would not be favorable. In permissioned blockchains, the network is not accessible to everyone making the notion of persistency used in PoW not needed. If it is used, it would be inefficient regarding computational power and energy requirements.

2) Proof of Stake

First suggested by an anonymous user on a bitcoin forum [11], then being implemented by PeerCoin [12], the Proof of Stake (PoS) mechanism is an improved version of PoW to reduce the excessive energy consumption. In PoS the miner to forge the next block is selected with a probability proportionate to the amount of its stake. Stake is the monetary value a node is willing to put at stake. Should the miner be fraudulent, then a miner is penalized by taking away a percentage of the staked coins, resulting in a miner with higher amounts of stake being less likely to be fraudulent. It does, however, have its drawbacks since the rich tend to get richer, as they are more likely to be selected [4]. PeerCoin implemented a solution to this notion by not only taking wealth into account when deciding the probability, but also the coin's age expressed in "coin-days". [13]. Coin age is simply defined as $currencyAmount * holdingPeriod$. In a simple example, if Bob received 10 coins from Alice and kept it for 20 consecutive days, it is said that Bob has 200 coin-days.

Implementation of PoS within permissioned blockchains would not be favorable. A miner with the most stake in the network has the highest probability of getting picked to forge a new block. However, such a miner could add transactions which would otherwise not be approved by the authorities within a permissioned blockchain as this could result in the miner to gain more say in the network than the authorities.

3) Proof of Authority

Proof of Authority (PoA) [14] is a hybrid consensus algorithm that is both Byzantine fault tolerant (BFT)[15] and Proof of Stake based. Its eminence is due to the offered performance increase with respect to typical BFT algorithms and toleration to faults. PoA was first proposed by the private networks of the Ethereum [8] ecosystem. PoA is a form of PoS where the stake is not the monetary value, but instead the real identity of the authority and therefor it's reputation. This discourages malicious use of the network, as the authorities can face real-world consequences.

PoA algorithms rely on a set of N trusted nodes called the authorities. The authorities each have a unique id, namely their real identity and at least $\frac{n}{2} + 1$ of these authorities are assumed to be honest. The authorities need to reach consensus to process transactions issued by clients. The PoA consensus algorithms use mining rotation [16]. A mining rotation is a widely used approach to fairly distribute the responsibility of block creation among authorities. In a mining rotation, time is divided into steps, each of which has an authority elected as mining leader who is in charge of proposing new blocks on which distributed consensus is achieved. Election of the mining leader is algorithm specific and will be explained for the following two algorithms.

Parity [17] and VeChain [18] are two blockchain implementations which use PoA algorithms for permissioned blockchains. Parity's consensus algorithm Aura [19] is based on UNIX synchronous time (the time passed in seconds from January 1, 1970), all the authorities in the network are assumed to be synchronous with this time. Each authority is identified by a unique id. The index of the steps are determined by the time and step duration, where the step duration is constant. In each step the leader is determined by $i = stepIndex \mod N$, where i is the leader id. Each authority has a queue of transactions and a queue of pending blocks. The leader i always broadcasts a proposed block which it forms from the transactions in the queue, also when the queue is empty. The leader will be the only one to broadcast a block. All authorities send the received block to the other authorities. The block is accepted when all authorities have received the same block. A voting is triggered to decide if the leader is malicious when the authorities do not accept the proposed block. A majority among the authorities is needed to vote out the leader based on, the leader not having proposed any blocks, more than one or different blocks to different authorities. The leader is then removed from the set of authorities.

The VeChainThor [18][20] consensus algorithm which is used by the VeChain cryptocurrency also depends on a set of N trusted nodes. VeChain has 101 authorities that are authorized by the VeChain Foundation. The rotation mechanism depends on an active set of authorities. The authority that needs to build and broadcast a block is each time step determined by the deterministic pseudo-random process (DPRP), this uses the timestamp and the number of the block. Due to the pseudo-random properties of the DPRP, the order of authorities that need to generate the blocks will not be deterministic, this is desirable for a system-security point of view.

Unlike Practical Byzantine Fault Tolerance (PBFT), PoA has a preference for availability over consistency. This is not desirable for business applications that favor strong data integrity guarantees.

B. Byzantine Fault Tolerance-based Algorithms

The Byzantine Generals problem [21] is a problem in computer science that describes the difficulty of multiple nodes in a distributed system reaching consensus. The analogy with Byzantine generals goes as follows: multiple

Byzantine generals have surrounded a city. Each general has its own army. The challenge is that the generals must reach a consensus on how to attack the city. If they do not reach consensus their siege of the city will be unsuccessful. The generals must communicate with messengers, however, these messages are not reliable as they might be intercepted or they may fail to reach to the other general. This means that it is impossible to reach consensus in this way. The same problem of nodes communicating with one another and having to reach a consensus occurs in blockchain networks. Nodes might not be trusted or the network can be faulty. For this reason, certain blockchain systems have implemented different consensus algorithms to overcome these challenges.

Byzantine Fault Tolerance is the ability of a distributed network to function appropriately, such that the network correctly and consistently reaches consensus despite bad actors either propagating incorrect information or failing to send information at all [15].

1) Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT) is one of the first solutions to the Byzantine Generals problem [15]. The PBFT model works by providing a practical Byzantine state machine replication that accepts malicious nodes. PBFT offers both liveness and safety provided at most $\frac{n-1}{3}$ out of a total n replicas are simultaneously faulty. Liveness means clients eventually receive replies to their requests. Safety means the replicated service satisfies linearizability, that is, it behaves like a centralized implementation that executes operations at one time. PBFT works under the following assumptions:

- The distributed system is asynchronous: The network may fail to deliver messages, delay them, duplicate them, or deliver them out of order.
- Faulty nodes may behave arbitrarily, subject to the restriction that node failures are independent. To assure this each node should run different implementations of the service code and operating system, have separate administrators and root passwords.
- Strong adversary can coordinate faulty nodes, delay communication, or delay correct nodes to cause the most damage to the replicated service. However, an adversary cannot delay correct nodes indefinitely. Also, adversary and the faulty nodes under its control are computationally bound so that with high probability it is unable to subvert cryptographic techniques that are used between the nodes to prevent spoofing and replays, and to detect corrupted messages.

The nodes in a PBFT network, which are called replicas, consist of a primary node, called the leader, and the rest, are the backup nodes. All nodes constantly communicate with each other trying to reach a consensus state. For each broadcast, every node has to prove that the message came from a specific peer node using public-key signatures, and the integrity of the message using message authentication codes (MAC).

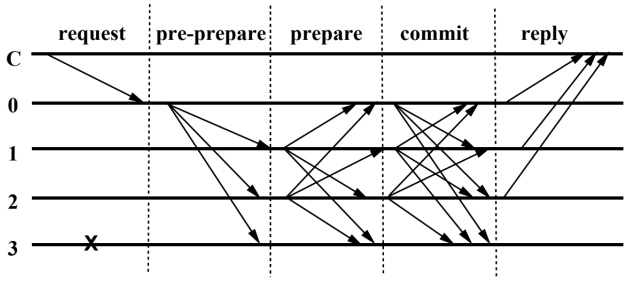

Fig. 2. PBFT algorithm stages. C represents the Client. 0 represents the leader node and the rest of the numbers represent the backup nodes. [15]

Each round of consensus in PBFT algorithm works as described in the following steps:

1) A client sends a request to the leader to invoke an operation. This starts a three-phase protocol to atomically multicast the request to the replicas. The three phases are pre-prepare, prepare, and commit.
2) In the pre-prepare stage the leader multicasts the request to the backups.
3) In the prepare stage, the backups that accept the pre-prepare message send an acknowledgment message to all other nodes.
4) After the nodes are prepared, in the commit stage, the nodes send the commit message to all other nodes. If a node receives valid commit messages from more than $\frac{n-1}{3}$ nodes, then they carry out the client request and send the reply to the client.
5) The client waits for $\frac{n-1}{3} + 1$ replies from different replicas with the same result to ensure it has the correct result, that is, matching replies from one more node than the faulty number of nodes. This is the result of the operation.

An illustration of this process can be seen in Figure 2 where $node_3$ is the faulty node.

The requirements for the nodes are that they start in the same state and that they are deterministic i.e., the execution of an operation in a given state and with a given set of arguments must always produce the same result. The model only performs efficiently with a small group of nodes since nodes are constantly communicating, therefore it is a consensus algorithm that is suitable for permissioned blockchain systems.

Zilliqa [22] has a consensus algorithm based on PBFT. PBFT uses MAC for authenticated communication between the nodes. As MAC requires a secret key shared between all pairs of nodes, the complexity of agreeing on the same record is $O(n^2)$. To improve efficiency, Zilliqa replaces MAC with digital signatures to effectively reduce the communication overhead to $O(n)$.

2) Federated Byzantine Agreement

PBFT requires all nodes to agree upon the same set of trusted validators, this means that acceptance to the system will be centralized[23]. To prevent centralization, Federated Byzantine Agreement (FBA) was introduced and implemented in the Stellar Consensus Protocol (SCP) [23]. Instead of having a central list of trusted validators, each node has its own set of validators that they trust. A transaction is only accepted by a node when the majority of a trusted set of validators agree on a transaction. All sets overlap with one another, meaning that the nodes are included in the set of trusted validators of multiple nodes. This system of overlapping sets ensures that the network is distributed, such that no central authority can decide what nodes are to be trusted. Nodes can only become part of the validation process when they are included in the trusted set of other nodes.

Stellar Consensus Protocol

In Stellar, a node's trusted set of validators is called the quorum slice. The total overlapping set of quorum slices is in turn called the quorum. Transactions are added to the ledger in voting rounds. Each transaction must pass through 2 voting rounds: a nomination protocol and a ballot protocol. During the nomination protocol candidate values are proposed to be agreed upon by the quorum through a federated vote. Each node votes for a single value and listens for votes from its peers until it finds a quorum that will accept the vote. After the nomination protocol, the ballot protocol is run. A ballot is a <Counter, Value> pair. The counter is how often a certain ballot has been run and the value is the value upon which is being voted. Now the nodes run a federated vote to decide on whether or not the value should be committed or aborted. If the nodes cannot reach a consensus for a certain value, the nodes are stuck, this is called a blocked state. The blocked state is resolved by incrementing the counter on the ballot and re-executing the ballot protocol. Ballots with a higher count have priority over ballots with lower counts. Through this system, FBA can provide liveliness by removing and avoiding all blocked-states.

Ripple Protocol Consensus Algorithm

Ripple is a network that has been designed to facilitate the transfer of funds between banks and international institutions [24]. Ripple makes use of a distributed ledger system where all Ripple nodes keep track of all funds [25]. Stellar has quorum slices as mentioned in section III-B2 and the Stellar Consensus Protocol section. Likewise, in Ripple, all nodes keep track of their own so called Unique Node List (UNL) [23]. If 80 % of nodes in the nodes UNL agrees with a candidate set of transactions, those transactions are added to the ledger. According to the Ripple white paper, as long as no more than 20% of nodes are faulty, the ledger will remain correct.

The Ripple Protocol Consensus Algorithm (RPCA) works with frequent voting rounds to decide what transactions are added to the ledger. The RPCA works in 4 steps each round:

1) All valid transactions that the node has received are put into the candidate set of the node.
2) All nodes collect the candidate sets of the nodes in their UNL and votes on each transaction.
3) The transactions that receive sufficient votes are moved to subsequent rounds. Transactions that receive insufficient votes are either discarded or included in the candidate set of the next round.
4) For transactions to be added to the ledger at least 80% of the nodes UNL must agree on a transaction. The ledger is closed when all transactions that receive a minimum of 80% of the votes in a UNL are added to the ledger.

Ripple is a permissioned blockchain system [26]. The network has high expectations of trust (80% of the network must be trustworthy) this is only realistically possible in a controlled and permissioned environment.

FBA was designed for decentralized systems, this means that no single authority decides what nodes are allowed to join the consensus process. However, to become part of the consensus process, other nodes must include the new node in their quorum slice. The decision of whether or not nodes are allowed to add a node to the consensus process could still be decided by a central authority. Therefore it is possible to use FBA in both permissioned and permissionless blockchain networks, it is most suitable for systems that require many validators.

3) Delegated Byzantine Fault Tolerance

Delegated Byzantine Fault Tolerance (DBFT) is a consensus algorithm used in NEO blockchain [27]. DBFT enables large-scale participation in consensus through proxy voting. Proxy voting means participants can delegate their votes to representatives every round, and a selected group of representatives reach consensus between themselves in the PBFT manner. In NEO ecosystem, these representatives are named bookkeepers.

Since there is delegation, and the consensus is reached between a small number of representatives, the algorithm works efficiently compared to PBFT. DBFT is an improved version of PBFT, therefore as PBFT, it is suitable for permissioned blockchain systems. DBFT is preferred for permissioned blockchains that have many nodes as it scales well.

IV. Trust models between nodes

Before people and organizations make use of a blockchain network they must have faith in the network's ability to correctly maintain and validate transactions. This section will discuss how the trust between nodes in a blockchain is managed in different consensus algorithms suited for private blockchains.

In PBFT, all nodes that are participating in the validation process of the network must communicate with all other

nodes in the network. PBFT algorithms are only functional when less than $\frac{1}{3}$ of the network is untrustworthy. Therefore, the nodes have to trust that other nodes in the blockchain are not malicious, essentially putting their trust in the organization that controls access to the network.

In DBFT, nodes vote on what validation nodes they deem trustworthy. This way, nodes have more control over which nodes they trust.

In FBA based algorithms all nodes can partake in the consensus process. Each node has its own list of trusted validators. This way all nodes are fully able to control what nodes they trust and what nodes they do not deem trustworthy. In the Stellar whitepaper [23], this is called flexible trust. Flexible trust is the opposite of the trust model of PBFT, instead of trusting all nodes in the network that are granted access by a central regulator, each node makes an individual consideration of what nodes to trust.

In PoA [14] based algorithms validation nodes have to prove they are trustworthy by using their real identity. The majority of the validation nodes are assumed to be honest. A validation node can be found malicious by a voting of the majority of the validation nodes.

V. Incentive Analysis

To motivate participants to mine blocks and hence keep the system running, cryptocurrency and smart contract blockchains have incentive rewards implemented, to motivate parties to participate in the system. Rewards can be given for the mining blocks. Many BFT algorithms do not give direct financial rewards for participating in the, but rather reward the nodes through non-economic benefits. This will be discussed in the following section.

A. Proof of Authority - Parity

In Parity's [14][17] consensus algorithm, nodes or authorities are required to give their real identity and therefore their reputation is at stake. Parity is a client for permissioned setting of Ethereum and can be used by business applications and for smart contracts. Using these applications is the incentive to join.

B. Practical Byzantine Fault Tolerance - Zilliqa

Zilliqa [22] which uses a modified version of PBFT incentives miners by requiring the sender of each transaction to pay some "gas price" upfront. Gas the is the smallest unit of computation. The Gas price is the amount that the sender is willing to pay per unit of gas for computations incurred in the transaction processing.

C. Delegated Byzantine Fault Tolerance - NEO

In NEO [27], GAS is the token for the network's resource control. The NEO network charges GAS for the operation and storage of tokens & smart contracts, thereby creating economic incentives for bookkeepers and preventing the abuse of the resources by charging for resource usage.

D. Federated Byzantine Agreement - Ripple and Stellar

In Ripple and Stellar, there are no direct economic rewards for participating in the validation process, unlike in PoW [28], [29]. However, there are a few other benefits for parties that are actively making use of the network to run their own validation nodes. Most importantly being independent of third parties to be able to participate in the network. This allows companies to have a trusted entry point to the network. There are also other benefits such as more up-to-date information about the network, customized triggers and full control of what data is stored. It is important to note that there is less need for economic incentive for hosting a validation node as the costs of running a server are low compared to a mining node in PoW.

VI. Scalability analysis

When comparing the scalability of different permissioned blockchains, similarly to Vukolić [9], this paper distinguishes 3 important categories: the number of nodes to reach consensus, the number of clients, and the number of transactions per second (TPS). The number of consensus nodes is the number of nodes that are actually taking part in the consensus process and deciding what transactions get accepted into the ledger. The number of transactions is the maximum throughput of the consensus algorithm in real world applications. In this section, implementations of PBFT, FBA, DBFT, and PoA consensus algorithms are compared in terms of the specified criteria.

A. Number of Validating Nodes & Block Validation Time

Validating nodes are the nodes that handle the requests made by the client nodes.

In the case of PBFT consensus, it is desirable to keep the number of validating nodes low to keep the transaction throughput high. As described in section III-B1 algorithms the complexity of agreeing on the same record is $O(n^2)$. From practical tests, as done in [30], it can be seen that the validation time for a block size of 10,000 transactions, goes from 4 seconds for 40 nodes to 26 seconds with 200 Nodes. For most applications a validation time this high may not be practical for use cases that require faster validation, and other consensus algorithms would be better suited.

DBFT algorithms were designed to overcome the limitation of PBFT algorithms, which can only achieve consensus in networks with a limited number of peers in a faster manner. In the NEO implementation of DBFT, validation time is just 15-20 seconds [27]. The NEO implementation of DBFT has a minimum of 7 and a maximum of 1024 nodes [31]. The number of validation nodes must be kept low compared to PoW and FBA to keep the transaction speed high.

TABLE I
SCALABILITY OF CONSENSUS ALGORITHMS PREFERRED FOR PERMISSIONED BLOCKCHAINS

Implementation	Number of validating nodes	Transaction throughput	Block Validation Time
Hyperledger Fabric v1.0 - PBFT	less than 200	200 TPS	4-26s
NEO - DBFT	7-1024	up to 1000 TPS	15-20s
Stellar - FBA	1000s of nodes	4000 TPS	5-6s
Ripple - FBA (RPCA)	1000s of nodes	1500 TPS	3-5s
Parity - PoA	1000s of nodes	80 TPS	5-8s

FBA was created to allow for thousands of validation nodes whilst still keeping transaction speed high. Validation of a transaction takes 5-6 seconds in the Stellar network [32]. Currently, the Stellar network has 136 validating nodes [33], however, there is no theoretical limit to the maximum size of the network. The ripple consensus network functions similarly to Stellar and is also capable of thousands of validators at the same time. However, currently, the number of active validators on the main Ripple network is approximately 130 [34]. It takes these validators about 3-5 seconds to validate a block of transactions [24].

In PoA [14] [35] algorithms there is a set of trusted nodes that validate the transactions on the blockchain network. Due to the constant transaction processing of Parity, the throughput remains constant when the number of nodes in the network is increased. The block sizes in the network are determined by the step duration, this is the time the leader has to build and propose a block. The typical step duration for Parity is between 5-8 seconds.

B. Number of Transactions

In tests of PBFT implementations in Hyperledger Fabric v1.0, the number of transactions per second (TPS) given 10,000 transactions was 200 TPS [36].

DBFT implementations can support up to 10,000 TPS [27]. However, in the current practical application in the NEO network, the TPS is closer to 1,000.

When looked at FBA implementations, the active stellar network has reached up to 4,000 TPS [37] however there have also been tests claiming over 10,000 TPS [38]. Currently, according to the Ripple company, the TPS of the network is around 1500. Ripple, however, claims that the number of TPS could easily be expanded to support tens of thousands of TPS [39]. It is hard to define an upper limit, because in practice the maximum TPS is bottlenecked by different factors, such as the implementation of the core software, used hardware, and internet latency.

PoA is limited by hardware and not the consensus algorithm itself. To keep the system working under the hardware limitations of the network, blockchains using PoA enforce transactions in a block to not surpass a certain threshold. The threshold can be many things ranging from the total monetary amount of transactions to the number of transactions in the block. The implementations hardcode this amount based on their network's hardware limitations. For Parity [35], the latency and throughput remain constant when transaction rates are increased beyond 40 TPS. Parity has a maximum constant client request rate of 80 TPS.

VII. DISCUSSION

Each consensus algorithms has its advantages and drawbacks. They were made to tackle different kinds of problems and are applied to different fields, from business applications to payment processing. Each prioritizes some aspect of scalability and reliability. For permissioned systems, it is seen that Proof of Authority (PoA) (in section III-A3) and Byzantine Fault Tolerant (BFT) (in section III-B) based systems are the most applicable. Proof of Work (PoW) (in section III-A1) and Proof of Stake (PoS) (in section III-A2) based systems require additional work to guarantee safety in permissionless systems. In permissioned systems there is an assumption of trust between the nodes, therefore other performance aspects such as transaction throughput can be prioritized.

In terms of the number of clients involved in the network, there is no notable difference between the different consensus algorithms. Each has the ability to support thousands of clients with an upper bound set by their hardware.

The difference between the BFT consensus algorithms lay within the number of validating nodes and the throughput of the transactions. As can be seen in Table I, PBFT and DBFT are both restricted in the amount of validating nodes. PBFT is suitable for smaller circles where there are a small number of validating nodes, and a high transaction throughput is not the priority, like in business applications. DBFT is less centralized compared to PBFT since the delegates are chosen by all participants as stated in section III-B3. DBFT scales better in terms of the number of nodes since a consensus is reached between a number of delegates, that is, less than the number of all nodes in the network. DBFT has high transaction throughput since it can reach consensus with a small number of delegates, which makes validation time lower, hence allowing higher volumes of transaction at unit time compared to PBFT.

FBA is the consensus algorithm that steers furthest away from a centralized network structure. FBA can support up to thousands of validating nodes making it much more scalable

in terms of validating nodes. Allowing to reach consensus with a smaller group in a similar manner to DBFT, FBA also allows high volumes of transaction throughput.

PoA can support up to thousands of validating nodes. In the means of transaction throughput, it is bound by the hardware.

There are different trust models between nodes among the consensus algorithms to encourage participants to trust the network. In PBFT and PoA, there is an assumption that the majority of the validating nodes are trustworthy. In DBFT and FBA, however, each node has a set of nodes that it decides to trust on.

To motivate participants to mine blocks there are different incentives among the consensus algorithms. In Parity which implements PoA, the incentive is for each block to get their transactions validated. In Zilliqa which implements PBFT, and NEO implementing DBFT, the incentive is to obtain some reward in exchange for creating a block. Lastly in Ripple and Stellar which implement FBA, there is no direct incentive other than having the benefits of independence of the third parties and having full control over the data.

VIII. Conclusion

This paper has touched upon the notion of blockchain and the distinctions between permissioned and permissionless blockchains. PoW, PoS, PoA, PBFT, FBA, DBFT consensus algorithms were then concisely explained, with additional information on consensus algorithms that were suspected to be beneficial for permissioned blockchains. Thereafter the consensus algorithms have been analyzed in terms of incentives, trust between nodes, and scalability. From research, it has been found that PoA, PBFT, DBFT, and FBA based consensus algorithms are the most suitable ones for permissioned blockchains.

For most permissioned blockchains the main incentive to participate and maintain the system is having access to the system. Most permissioned blockchains are designed to facilitate transactions between organizations. Having access to these transactions is enough incentive to actively participate. Also, it should be noted that the costs associated with running permissioned blockchains is much lower than that of PoW blockchains.

Trust between nodes in a permissioned blockchain differs per consensus algorithm. In PoA based blockchains trust stems from the knowledge of a node's real world identity. In PBFT and DBFT based blockchains, the trust comes from the fact that the nodes participating in the validation process have been approved by a central authority. Finally, trust in FBA based systems is decided by the individual nodes, through their own set of nodes that they trust.

Scalability in permissioned systems differs strongly between the different consensus algorithms. In PBFT and DBFT based systems, the number of nodes participating in the validation process must remain low compared to FBA and PoA. These algorithms are often not applicable with more than a thousand validation nodes. FBA and PoA based systems can support many thousand validation nodes. However, in practice, there are often are not that many validation nodes. In terms of transaction throughput, there is strong variation between implementation, however, this is more likely to be implementation specific rather than algorithm specific.

When comparing different consensus algorithms, we see that this is a non-trivial task since each of them has its own benefits and drawbacks. As seen in the examples of implementations of blockchain systems as touched upon in this paper, many of them use approaches where consensus algorithms have been tailored for their specific use cases and needs.

Future research could focus on a number of things. It is difficult to fully and in detail compare the performance of permissioned blockchains. A research complying with this problem would consist of comparing multiple different permissioned systems on comparable hardware and network conditions. Only in this manner would it be possible to define an accurate ranking of the performance of permissioned blockchains. Another possible area of future investigation would be the detailed comparison of the associated security risks of different implementations of consensus algorithms in permissioned blockchains.

In conclusion, architects of permissioned blockchain systems should look into their system needs and tailor the consensus algorithms that are close to their needs to work optimally for their system.

References

[1] Market Research Future, *Blockchain Technology Market Research Report - Global Forecast to 2023*, 2018. [Online]. Available: https://www.marketresearchfuture.com/reports/blockchain-technology-market-1708 (visited on 03/20/2020).

[2] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*, Springer, 1990, pp. 437–455.

[3] S. Nakamoto, "Bitcoin whitepaper," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[4] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, Jun. 2017, pp. 557–564. DOI: 10.1109/BigDataCongress.2017.85.

[5] G. Falazi, M. Hahn, U. Breitenbücher, F. Leymann, and V. Yussupov, "Process-based composition of permissioned and permissionless blockchain smart contracts," in *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*, IEEE, 2019, pp. 77–87.

[6] G. Falazi, V. Khinchi, U. Breitenbücher, and F. Leymann, "Transactional properties of permissioned blockchains," *SICS Software-Intensive Cyber-Physical Systems*, pp. 1–13, 2019.

[7] M. Vukolić, "Rethinking permissioned blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, 2017, pp. 3–7.

[8] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[9] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International workshop on open problems in network security*, Springer, 2015, pp. 112–125.

[10] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," *arXiv preprint arXiv:1707.01873*, 2017.

[11] Anonymous, *Proof of stake instead of proof of work*, 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=27787 (visited on 03/15/2020).

[12] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[13] L. 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)*, IEEE, 2018, pp. 1545–1550.

[14] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain," in *Italian Conference on Cyber Security (06/02/18)*, Jan. 2018. [Online]. Available: https://eprints.soton.ac.uk/415083/.

[15] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the 3rd Symposium on Operating System Design and Implementation*, 1999, pp. 173–186.

[16] J. Garzik, "Public versus private blockchains," *BitFury Group, San Francisco, USA, White Paper*, vol. 1, 2015.

[17] Anonymous, *Parity technologies*. [Online]. Available: https://www.parity.io.

[18] X. Liu, G. Zhao, X. Wang, Y. Lin, Z. Zhou, H. Tang, and B. Chen, "MDP-Based Quantitative Analysis Framework for Proof of Authority," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Oct. 2019, pp. 227–236. DOI: 10.1109/CyberC.2019.00046.

[19] Anonymous, *Aura*. [Online]. Available: https://wiki.parity.io/Aura.

[20] Anonymous, "VeChain: Development plan and whitepaper," 2018. [Online]. Available: https://cdn.vechain.com/vechainthor_development_plan_and_whitepaper_en_v1.0.pdf.

[21] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[22] Anonymous, *The ZILLIQA technical whitepaper*, 2017. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf (visited on 03/20/2020).

[23] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, vol. 32, 2015.

[24] Anonymous, *Ripple.com*. [Online]. Available: https://ripple.com/.

[25] D. Schwartz, N. Youngs, A. Britto, *et al.*, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, 2014.

[26] R. Henry, A. Herzberg, and A. Kate, "Blockchain access privacy: Challenges and directions," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 38–45, 2018.

[27] M. Xian, *NEO White Paper*, 2018. [Online]. Available: https://github.com/neo-project/docs/blob/3c5530197768d98a1abf7eeed0119a8f4e99e7cc/en-us/whitepaper.md (visited on 03/14/2020).

[28] Anonymous, *Technical FAQ*. [Online]. Available: https://xrpl.org/technical-faq.html.

[29] Anonymous, *Administration - stellar core: Stellar developers*. [Online]. Available: https://www.stellar.org/developers/stellar-core/software/admin.html#why-run-a-node.

[30] M. M. Jalalzai, C. Busch, and G. G. Richard, "Proteus: A scalable bft consensus protocol for blockchains," in *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 308–313.

[31] Anonymous, *A Statement from NEO Council*, Mar. 2018. [Online]. Available: https://neo.org/blog/details/3067.

[32] Anonymous, *Stellar network dashboard*. [Online]. Available: https://dashboard.stellar.org/.

[33] Anonymous, *Stellar network visibility*. [Online]. Available: https://stellarbeat.io/quorum-monitor.

[34] Anonymous, *XRP Charts*. [Online]. Available: https://xrpcharts.ripple.com/#/validators.

[35] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1085–1100.

[36] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Communication Networks*, vol. 2018, 2018.

[37] J. McCaleb, B. F. Crain, S. Couture, and M. Roy, *Soundcloud*, 2017. [Online]. Available: https://soundcloud.com/epicenterbitcoin/eb-128.

[38] B. P. Eha, *How barclays aims to bring a billion unbanked into the fold*, Jan. 2017. [Online]. Available: https://www.americanbanker.com/news/how-barclays-aims-to-bring-a-billion-unbanked-into-the-fold.

[39] Anonymous, *XRP Ledger*, Sep. 2017. [Online]. Available: https://xrpl.org/blog/2017/decent-strategy-update.html.