BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Journal of Computer Science Research
https://ojs.bilpublishing.com/index.php/jcsr

REVIEW

A Review of Consensus Protocols in Permissioned Blockchains

Nenad Zoran Tomić*

University of Kragujevac, Serbia

ARTICLE INFO

Article history
Received: 26 February 2021
Accepted: 8 March 2021
Published Online: 20 April 2021

Keywords:
Permissioned blockchain
Consensus protocols
Byzantine Fault Tolerance
Crash fault tolerance

ABSTRACT

Consensus protocols are used for the distributed management of large databases in an environment without trust among participants. The choice of a specific protocol depends on the purpose and characteristics of the system itself. The subjects of the paper are consensus protocols in permissioned blockchains. The objective of this paper is to identify functional advantages and disadvantages of observed protocol. The analysis covers a total of six consensus protocols for permissioned blockchains. The following characteristics were compared: security, trust among participants, throughput and scalability. The results show that no protocol shows absolute dominance in all aspects of the comparison. Paxos and Raft are intended for systems in which there is no suspicion of unreliable users, but only the problem of a temporary shutdown. Practical Byzantine Fault Tolerance is intended for systems with a small number of nodes. Federated Byzantine Fault Tolerance shows better scalability and is more suitable for large systems, but can withstand a smaller number of malicious nodes. Proof-of-authority can withstand the largest number of malicious nodes without interfering with the functioning of the system. When choosing a consensus protocol for a blockchain application, one should take into account priority characteristics.

1. Introduction

Blockchain technology enables distributed management of large databases. Its functioning was explained for the first time in the Bitcoin cryptocurrency manifesto, in late 2008 [1]. Given its characteristics, blockchain soon came out of the shadow of cryptocurrencies and found application in the broader electronic business context. This technology allows participants to execute transactions in order to enter new data in the public ledger. A transaction is any instruction that leads to a change in the state of the system. The public ledger consists of a series of blocks, which contain records of performed transactions [2]. The content of each block depends on the content of the previously entered blocks, because the new state depends on the previous state and the changes brought by the transactions. Data is entered into the public ledger without third party mediation [3]. As there is no trust among participants, it is necessary to provide a mechanism by which the entered data will be checked and confirmed. This mechanism is called a consensus protocol.

Blockchain technology itself is new, but its foundations are previously known technologies and methods, such as asymmetric cryptography, timestamping, Merkle tree, hash functions and smart contracts. Asymmetric cryptography is used to sign executed transactions. Transactions are timestamped to avoid double spending by creating confusion about the order. Hash functions are used to

**Corresponding Author:*
Nenad Zoran Tomić,
University of Kragujevac, Serbia;
Email: ntomic@kg.ac.rs

DOI: https://doi.org/10.30564/jcsr.v3i2.2921

prevent subsequent changes to the contents of blocks embedded in the public ledger. Therefore, the hash value of the previous block is entered in each new block, which prevents the change of their content [4]. Blockchain technology enables the implementation of smart contracts, as an electronic document that is executed on the basis of a programming code [5]. Consensus algorithms themselves are not a new technological solution. Their foundations were laid by Lamport (1978) and Schneider (1990) in the desire to formulate algorithms tolerant to a certain kind of faults [6] [7].

The subjects of the paper are consensus protocols in permissioned blockchains. The objective of this paper is to identify the functional advantages and disadvantages of observed protocols. The paper will be divided into three sections. The section one will present the key features of blockchain technology. Special attention will be addressed to the difference of the blockchain systems according to the degree of openness for participants. The section two analyzes the principles of functioning of the observed protocols individually. The final section identifies advantages and disadvantages of all protocols through a comparative analysis of their key characteristics.

2. Blockchain Characteristics

There are three types of participants in blockchain systems: nodes, full nodes, and miners. Nodes are participants that can send or receive transactions, but do not participate in consensus building nor keep a copy of the public ledger. In addition to participating in transactions, full nodes also store a copy of the public ledger. Miners are full nodes that participate in consensus building and embed new blocks in the public ledger. Generally, participants do not know each other and do not trust each other. This means that each blockchain must have a built-in protocol for reaching consensus in trustless environment where there is no third party to confirm data authenticity [8]. The protocol determines which participants can create blocks, how consensus is reached and whether there is a reward.

When performing a transaction, the sender applies the selected hash function to it and signs the resulting record using a private key. The signature authenticates the sender. Miners need to confirm the integrity of the transaction and the participant who sent it. This means that the digital signature should correspond to the sender's signature, i.e. hash value of the transaction should correspond to the one signed by the user. After confirmation, one of the miners (depending on the algorithm) packs the transactions into a block and suggests a new block to the other miners [9]. The new block contains the hash value of the previous block, the timestamp, and a list of included transactions. Other miners check whether the size of the block is within the allowed values, whether it follows the previous block according to the timestamp, as well as all hash values.

Sharing a public ledger of transactions among participants and signing transactions creates the conditions for overcoming the problem of mistrust. After reaching a consensus, the block is embedded in the public ledger, which is available to everyone and shows the current state of the system. This eliminates the need for an intermediary in transmitting and storing data [10]. Once recorded in the public ledger, transactions are irreversible. Merkle tree technology is used to connect blocks, so any attempt to change the content of a previously performed transaction leads to a change in the content of all subsequent blocks. Attempting to systemically change a series of blocks would require enormous computing power regardless of consensus protocol, an investment that can hardly be justified by benefits.

According to their openness to participants, blockchain systems are divided into permissioned blockchains and permissionless blockchains. They differ fundamentally in terms of access to the system and the role that the user can perform. Permissionless blockchains have open access. Each user can become part of the network and act as a node, full node or miner, as all roles are available [11]. Because of these characteristics, these blockchain systems are often referred to as public.

In permissioned blockchains, there is a clear separation of roles. Miners are always known and predetermined [12]. There are differences in terms of the capabilities that other users may have. For some systems, membership is open, but nodes can only send and receive transactions. For others, each user must receive a special invitation to become a node. In such systems, all users are known and identified in advance. Due to these characteristics, such systems are referred to as private or consortium blockchains in the literature. However, it should be borne in mind that higher centralization compared to permissionless blockchains should not mean that one institution is the full owner of the system. For any business application in which there is a trusted institution, it is better to use some other database technology than blockchain.

3. Types of Consensus Protocols in Permissioned Blockchains

3.1 Byzantine Fault Tolerant Protocols

Most of the protocols for reaching consensus in permissioned blockchain systems are based on solving the problems of Byzantine generals. The problem describes

DOI: https://doi.org/10.30564/jcsr.v3i2.2921

difficulties in reaching an agreement in conditions of mutual distrust among decision makers [13]. One can imagine that several divisions of Byzantine army attack the enemy city. A unilateral attack of a single division cannot lead to victory. But if the generals reach a consensus on the timing of the simultaneous attack, the city will be conquered. The problem is that generals cannot communicate directly, but solely through couriers. There may be two problems.

The first problem is the possibility that some of the generals are traitors and deliberately send contradictory messages. Another problem is the possibility for some of the couriers to change the content of the messages, either because they are traitors, or because the enemy intercepts them and replaces them with their own couriers. Therefore, it is necessary to devise a mechanism for reaching consensus, so that:

a. All loyal generals adopt the same plan (traitors, if any, can do what they want).

b. Traitors cannot lead loyal generals into adopting a wrong plan.

The problem that participants in permissioned blockchains face is similar to the problem of Byzantine generals. There are a finite number of known participants, but it is not possible to say with certainty which of them are loyal and which are traitors. Therefore, consensus algorithm must be able to allow decision-making even when some participants are unreliable. In addition to this type, crash faults also occur, when, due to technical, or problems of some other nature, the decision-making process is slowed down or stopped. Therefore, consensus algorithms in permissioned blockchains are divided into crash fault tolerant and Byzantine fault tolerant.

3.1.1 Practical Byzantine Fault Tolerance

Practical Byzantine fault tolerances was formulated by Castro & Liskov (2002) [14]. The algorithm is designed to work in asynchronous systems and to provide liveness and safety. Liveness is reflected in the fact that some consensus will certainly be reached. Safety refers to the ability to reach a valid consensus in a situation where at most $(n\text{-}1)/3$ nodes act maliciously, with n being the total number of nodes participating in the decision-making. If we denote the faulty nodes with f, then the total number of nodes must be $n = 3f+1$.

To prevent misrepresentation and confusion, each node signs messages with its own secret key. Also, each message has an authentication code, and when sent, it is compressed using the hash function. Each node communicates with all other nodes in the system. Nodes can identify each other based on the signature and check if the message was changed during transmission. Before the consensus-building process begins, the nodes are divided hierarchically, with one chosen as the leader and the others as the backup. The role of the nodes changes before each new round of decision-making on a round robin basis. One round of consensus-building consists of four phases. In the first phase, the client sends a message to the leader wanting to change the state of the system. In the second phase, the leader forwards the message to the backup nodes. Backup nodes consider the content of the message and send a response in the third phase. In the last phase, the client collects $f+1$ identical responses from the backup. The selected response represents the attitude of the entire system towards the message sent by the client.

The key advantage of pBFT in relation to all permissionless blockchains protocols is lower computational complexity, and, thus, lower electricity consumption. Also, the throughput is higher than with the mentioned systems. However, pBFT is intended for systems with a small number of participants. Increasing the number of nodes exponentially increases the volume of communication, so application in permissionless blockchains would lead to congestion. Of the known blockchain platforms, *Hyperledger Fabric* and *Zilliqa* use pBFT.

3.1.2 Delegated Byzantine Fault Tolerance

Delegated Byzantine fault tolerance (dBFT) is a modification of the basic form of pBFT (Coelho et al., 2020) [15]. It was proposed during the creation of the NEO blockchain, which, in addition to the cryptocurrency of the same name, offers a code for creating smart contracts. The GAS token is used to execute smart contracts, which users who own NEO cryptocurrency receive as a kind of dividend.

The use of dBFT overcomes the problem of excessive communication due to the increase in the number of nodes. Any user who owns a NEO can vote for one of the nodes, which then become delegates. In order for a node to become a delegate, it is necessary to positively identify itself, to have a stable internet connection and appropriate computer equipment, and to invest 1000 GAS units [16]. The speaker is then randomly selected from among the delegates. The speaker selects the transactions to be included in the new block and sends the proposal to the delegates for confirmation. The block needs to be confirmed by at least 2/3 of the delegates. Otherwise the proposal is rejected and a new random speaker is elected who repeats the process.

The dBFT protocol is criticized for the increased level of centralization. Although in theory this should not be the case, the NEO cryptocurrency has shown that all delegates are also members of the founding consortium. It

DOI: https://doi.org/10.30564/jcsr.v3i2.2921

can be assumed that this was not the idea of the founders, but it should also be borne in mind that the selection of delegates is a great challenge. On the one hand, if no are set, participants can delegate themselves. On the other hand, setting criteria too high reduces the number of delegates and the system becomes centralized. In that case, the blockchain does not fulfill its basic decentralization premise among users who do not trust each other.

3.1.3 Federated Byzantine Fault Tolerance

Federated Byzantine fault tolerance or the Federated Byzantine Agreement (FBA) is also a modification of the basic form of pBFT. In order to overcome the problem of extensive communication, the FBA implies that nodes exchange messages only with nodes they trust [17]. The *Stellar* and *Ripple* cryptocurrencies use the FBA variants.

A set of nodes that one node trusts is called a quorum slice, while a quorum is a set of nodes that trust each other. Based on the mutual trust with other nodes, one node can be part of multiple quorums at the same time. That node represents the quorum intersection. Nodes communicate only within the quorum. Each node collects transactions performed after the last block was embedded and those performed before which so far have not become part of the blocks. The nodes then declare on the validity of the proposed transactions. When they receive the required percentage of positive votes, the transactions are included in the block, which is embedded into the public ledger.

In the Ripple protocol, a quorum is a unique node list (UNL). Making a decision implies a positive response from at least 80% of the nodes involved. This means that the maximum number of malicious nodes can be $f \leq (n - 1) / 5$, where n is the number of nodes within a particular UNL. This leads to a strong correctness of the system. In case the number of malicious nodes is higher, the system has a defense plan. As long as their number is $f \leq (4n + 1) / 5$, the system exhibits weak correctness, i.e. it is not capable of validating true transactions, but can prevent malicious nodes from embedding fake transactions [18]. Unlike the previous two protocols, which ensured finality without the possibility of forking, in FBA, the possibility of forking depends on the size of the quorum. If the size of individual quorums does not have a lower limit, then forking is possible. However, even with the lower limit of $0.2 * n_{total}$, where n_{total} is the total number of nodes in the system, forking becomes impossible if the quorums partially overlap.

3.1.4 Proof-of-authority

Proof-of-authority (PoA) is a consensus protocol based on the identification of nodes and their reputation. It was created in 2015 based on the proof-of-stake (PoS), with the proviso that in this case the nodes do not invest the coins they own, but their own reputation [9]. Nodes that embed blocks in the public ledger are called validators. In order to become a validator, node's identity must not only be publicly known, but also confirmed by the notary service. The point of the system is that in case of any undesirable behavior, the validator gains a negative reputation and loses the opportunity to participate in further block creation. Nodes with a satisfactory reputation change in the role of validator on a round robin basis.

Although PoA can also be used in permissionless blockchains, its design is extremely centralistic, because a small group of validators embed blocks. Also, PoA does not ensure anonymity, as one of the key features of a permissionless blockchain. The advantage is that, unlike PoS, it does not favor rich individuals. In practice, PoA is used on the *PoA Network* and *Vechain* trading platforms, as well as on the *Quorum* blockchain platform.

3.2 Crash Fault Tolerant Protocols

The second class of consensus protocols deals with the problems of the imperfect environment in which systems operate. A metaphorical representation of the problems addressed by these protocols is the fictitious local assembly on the Greek island of Paxos. Deputies constantly enter the assembly hall and leave, while staying in the building for a period of time that is not foreseen in advance, nor is it the same for each deputy. Despite the occasional absence of deputies, it is necessary to provide conditions for unhindered legislative decisions. Thus, crash fault tolerant protocols ensure functioning in situations where individual participants are unavailable during a certain period. The assumption is that there are no participants who are malicious, i.e. that Byzantine faults are not a problem that the system encounters.

3.2.1 Paxos

Paxos is not an individual protocol, but a whole family of protocols, created with the aim of solving the problem of absence of participants [19]. The basic system assumptions are as follows: processors perform operations at any arbitrary speed; during work they may experience a crash fault; after recovery, processors can rejoin the protocol; during operation, processors do not attempt to trick the system in any way. In terms of communication, one processor can send messages to any other processor, which takes an arbitrarily long time to be delivered. Messages can be lost, but they are not modified by third parties.

DOI: https://doi.org/10.30564/jcsr.v3i2.2921

Consensus can be reached when the total number of nodes is $n = 2F + 1$, where F is the maximum number of nodes that can experience a simultaneous crash fault. In other words, most of the total number of nodes must always be present.

Nodes are called acceptors in Paxos protocols. They are divided into quorums, as subsets of a set of acceptors, with any two quorums having to be intersected by at least one acceptor [20]. The size of the quorum is such that it includes the majority of the total number of acceptors (and in the event that the acceptors are assigned a certain voting weight, the quorum must include more than half of the votes). The client sends a message requesting a change of state from the system. In order for a decision to be made, all quorum acceptors must receive the same message. The proposer coordinates the distribution of messages and decision-making (especially in case of disagreement). Each system has one prominent proposer, called a leader. Finally, learners execute the decisions made (change the state of the system) and send a response to the client.

When a proposer receives a request from a client, they formulate a message, assign it a certain number p, and distribute it to the acceptors within the quorum. The number p serves only to determine the chronological order of the message and must be greater than the numbers of all messages that the acceptors have processed so far. This activity is called *Prepare*. Upon receipt, the acceptors check the number p and if that number is greater than the numbers of all previous messages, they send a positive response. This process is called Promise. If they have accepted a message in the previous period, they shall also submit to the proposer the number of that message q (where $q < p$) and the value of w accepted by that message. Since Paxos is intended for asynchronous systems, participants may be at different stages of the process, so it is possible that the proposer does not know that the message q is accepted at all. In case the number of the message p is less than the number of any of the previous messages, the acceptors can send a negative response to the proposer, or not respond at all. This concludes the first phase.

If it receives a sufficient number of positive responses, the proposer in the second phase submits to the acceptors the full content of the message p, which, in addition to the number, also contains the proposed value $v = x$. In case the acceptors inform them that the message q has been accepted in the meantime, the proposer can decide to modify the proposal before sending, and for the proposed value to be $v = y$. A request for verification and acceptance is sent to the acceptors with the message. Acceptors accept the message (p, v) if in the meantime they have not promised to consider only the message under the number r, $r > p$. [21]. When they accept the message, they inform the proposer and the learners (although the role of the learner is often played by the proposer itself).

This case is a general form of the Paxos algorithm. As it is a family of protocols, this means that the general case has a large number of modifications into specific forms. Individual cases vary significantly in terms of purpose, and, therefore, have different performance, which further complicates the comparison of this protocol with other ones.

3.2.2 Raft

Raft is a consensus protocol with resilience to crash faults, proposed by Ongaro & Ousterhout (2014) [22]. It is a modification of the Paxos algorithm, in which nodes can exist in one of three states: leader, followers or candidates. Time is divided into arbitrarily long sections, called terms. During one term, the same node always plays the role of the leader. They receive requests for transactions from clients, check them and index them, in order to maintain insight into chronological order. The transactions are then packed in blocks, which are forwarded to followers. The task of the followers is to send the leader an acknowledgment of the correctness of the block and to replicate the block, i.e. to take over the information it contains.

When requests for new entries stop arriving, the node waits for a certain period, known as an election timeout. This means that the old leader has lost its role and that elections for a leader for the next term are taking place. The node becomes a candidate, gives itself a vote and sends messages to the rest of the network asking for support for its candidacy. If it gets the support of most other nodes, it will be elected leader in the coming term. It then sends a message to all other nodes informing them of its election. Also, it can happen that, while waiting for votes, the node receives a message from another node claiming to have been elected leader. In that case, the node compares its term index with the term index of the node that claims to be elected. If its term index is lower, it must recognize the other node as the leader. However, if its term index is higher, it can reject the message of the node claiming to be the leader and continue to collect votes. A higher term index value indicates that this node is aware of the last changes that have occurred. If no candidate collects enough votes, the whole process is repeated with a randomly assigned short pause. The node assigned the shortest pause will initiate a new vote.

The key difference between Paxos and Raft algorithms is that Raft allows only the best updated nodes to take on the role of the leader, while with the Paxos algorithm it can be any node [23]. The well-known blockchain plat-

DOI: https://doi.org/10.30564/jcsr.v3i2.2921

forms, *R3 Corda* and *Quorum*, use Raft as a consensus protocol.

4. Comparing Protocols

Permissionless blockchains are easy to compare. They all have the same purpose, but show significant differences in terms of required investments, throughput and scalability. In addition, a large number of papers compare these protocols in some respect. The situation with permissioned blockchains is somewhat more complex. First of all, the systems are divided into two large groups, which differ according to the level of security they provide. Differences in purpose undoubtedly lead to differences in performance. However, the problem that exists in the literature is the uneven terminology in this area. Some papers that generally compare consensus protocols for permissioned blockchains actually compare the performance of the platforms applying them [24].

Since none of the observed protocols are based on intensive computation, no high initial investment is necessary. Protocols are not competitive as proof-of-work and related protocols are, so there is no energy load. They still differ a lot in scalability and throughput. Also, there is a significant difference in terms of mutual trust between nodes. Having in mind the representative research in this field, the comparison was made on the basis of the following characteristics: security, mutual trust, throughput and scalability. Table 1 presents the results of the comparison.

Table 1. Comparative analysis of the presented consensus protocols in permissioned blockchains

Protocols Characteristics	**pBFT**	**dBFT**	**FBA**	**PoA**	**Paxos**	**Raft**
Security	Byzantine if f < 33.3%	Byzantine if f < 33.3%	Byzantine if f < 20%	Byzantine if f < 49%	Only from crash fault	Only from crash fault
Mutual trust	Based on node selection	Nodes choose who to trust	Flexible trust	Based on identity	Complete in terms of good intentions	Complete in terms of good intentions
Throughput	Moderate	High	High	Low	Moderate	Moderate
Scalability	Limited	High	High	Low	Limited	Limited

Source: author, according to the broad literature

The first four presented protocols protect the system from Byzantine faults. They differ from each other in the proportion of malicious nodes that the system can tolerate without losing normal functioning. Paxos and Raft assume that all nodes have already been checked and known, and that they are not characterized by malicious behavior. Therefore, they do not provide protection against Byzantine faults, but only from a temporary crash fault. Thus, the level of security they guarantee is significantly lower.

Nodes do not know each other in permissionless blockchains, so there is no basis for trust. With permissioned blockchains, the nodes do not have to know each other either, but the very fact that the membership is permissioned gives some degree of trust. Thus, with pBFT, mutual trust is based on the fact that the nodes have been allowed to enter the system. That is why each node communicates with all other nodes, because it trusts their decisions. With dBFT and FBA, the situation is somewhat different, so the nodes choose who they trust. This is especially pronounced with FBA and the so-called flexible trust, meaning that, regardless of the total number of nodes, the participants make a decision only with those who they trust. With PoA, trust is based on a previously irrevocably established identity and the responsibility it entails. With the Paxos and Raft protocols, nodes have complete mutual trust in terms of goodwill, but they do not trust each other in terms of prompt response to tasks.

One of the key problems of permissionless blockchains is that, due to protocol design in a trustless environment, they cannot achieve throughput at the level of payment card companies. In order for a single protocol to be considered an adequate basis for a financial or other business system, it must allow the processing of a large number of transactions. All protocols analyzed in this paper show significantly high throughput, but there are differences between them. Thus, with pBFT, on a sample of 10,000 transactions on the Hyperledger Fabric platform, a throughput of about 200 transactions per second (TPS) was proven [25]. dBFT is said to be able to support around 4000 TPS [26], though specific data on the NEO cryptocurrency blockchain shows that this number is 4 times lower. Also, the data shows throughput of about 4000 TPS with FBA [27], though theoretical assumptions claim that this number could be over 10,000. PoA has a slightly lower throughput, of about 80 TPS.

There are large differences in terms of scalability of the observed protocols. It was shown that a pBFT network with 40 nodes takes only 4 seconds to confirm a block of 10,000 transactions, but that with an increase in the number of nodes to 200, than time increases to 26 seconds and continues to grow exponentially [28]. From the above it can be concluded that pBFT is a good solution for small and controlled systems, but that due to the need for all nodes to communicate with each other it is not a good choice for large systems. It was stated that the NEO blockchain can process a block within 20 seconds, when the number of nodes ranges from 7 to 1024 [26]. Thus, with the growth of the system, dBFT becomes a better choice than pBFT, but it also has scalability limits. FBA is designed to maintain high scalability despite the involvement of thousands of nodes. However, this has not been confirmed in practice, because the Ripple and Stellar cryptocurrency blockchains have only 130 and 136 nodes, respectively. At the same time, they maintain a very stable scalability. PoA has a stable scalability which means 5-8 seconds required to validate the block even in case of increasing the number of nodes to over 1000. The problem that Paxos and Raft face in this context is the need for all communication to go through one node (leader), which leads to a bottleneck.

5. Conclusions

Comparing consensus protocol performance has three limitations. First of all, although all the observed protocols are intended for permissioned blockchains, not all have the same purpose. The clearest clasification is into those protocols that protect the system from Byzantine faults and those that protect only from crash faults. However, there are differences within these groups as well. Thus, pBFT is intended for systems with a small number of nodes, while FBA is intended for systems with a large number of nodes. If performed outside the intended environment, these systems can show significant deviations in terms of performance. Comparison under conditions that favor one group or one separate protocol yields results that do not reflect their actual usability.

Furthermore, observed protocols were not tested evenly in practice. It is worth mentioning that dBFT and PoA actually have limited application, and that their performance should therefore not be taken as completely reliable. Some protocols can be analyzed in parallel on the examples of a larger number of cryptocurrencies and/or business platforms, while others have an application that is sufficient only for conditional results. Finally, most protocols in practice show deviations from the theoretically stated optimal results. This problem is related to the previous one and speaks of insufficient testing in different conditions and on different platforms.

Having in mind the above and the results of the comparison, it can be concluded that no protocol shows absolute dominance in all aspects of the comparison. When choosing a consensus protocol for a blockchain application, one should take into account priority characteristics.

References

[1] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system [R]. 2008, retrieved from https://bitcoin.org/bitcoin.pdf.

[2] Zheng, Z., Xie, S., Dai, H.N., Chen, X. & Wang, H. An overview of blockchain technology: architecture, consensus and future trends [C], IEEE 6th international congress on Big data, Honolulu, HI, 2017: 557-564 https://doi: 10.1109/BigDataCongress.2017.85.

[3] Belotti, M., Božić, N., Pujolle, G. & Secci,S. A vademecum on blockchain technologies: when, which, and how [J], IEEE Communication, Surveys & Tutorials, 2019, 21(4): 3796-3838.

[4] Oliveira, M.T., Reis, L.H.A., Medeiros, D.S.V., Carrano, R.C., Olabarriaga, S.D. & Mattos, D.M.F. Blockchain reputation-based consensus: A scalable and resilient mechanism for distributed mistrusting applications [J], Computer Networks, 2020,179: 107367. https://doi.org/10.1016/j.comnet.2020.107367.

[5] Szabo, N. Formalizing and Securing Relationships on Public Networks [J]. First Monday, 1997, 2(9). https://doi.org/10.5210/fm.v2i9.548.

[6] Lamport, L. Time, Clocks and the Ordering of Events in a Distributed System [J]. Communications of the ACM, 1978, 21(7): 558-565.

[7] Schneider, F. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial [J]. ACM Computing Surveys, 1990, 22(4): 299-319.

[8] Bamakan, S.M.H., Motavali, A. & Bondarti, A.B. A survey of blockchain consensus algorithms performance evaluation criteria [J], Expert Systems with Applications, 2020, 154: 113385. https://doi.org/10.1016/j.eswa.2020.113385.

[9] Ismail, L. & Materwala, H. A Review of Blockchain Architecture and Consensus Protocols: Use Cases, Challenges, and Solutions [J], Symmetry 2019, 2019, 11: 1198. https://doi:10.3390/sym11101198.

[10] Zheng, Z., Xie, S., Dai, H.N. & Wang, H. Blockchain challenges and opportunities: A survey [J]. International Journal of Web and Grid Services, 2018, 14(4): 352-375.

[11] Lin, I.C. & Liao, T.C. A survey of blockchain security issues and challenges [J], International Journal of Network Security, 2017, 19(5): 653-659. https://doi:10.6633/IJNS.201709.19(5).01.

[12] Wang, X., Zha, X., Ni, W., Liu, R.P., Guo, Y.J., Niu, X. & Zheng, K.. Survey on blockchain for internet of things [J], Computer Communication, 2019, 136: 10-29. https://doi.org/10.1016/j.comcom.2019.01.006.

[13] Lamport, L., Shostak, R., & Pease, M. The Byzantine Generals Problem [J]. ACM Transactions on Programming Languages and Systems, 1982, 4(3): 382-401. https://doi.org/10.1145/357172.357176.

[14] Castro, M., Liskov, B. Practical Byzantine Fault Tolerance and Proactive Recovery [J]. ACM Transac-

DOI: https://doi.org/10.30564/jcsr.v3i2.2921

tions on Computer Systems, 2002, 20 (4): 398-461. https://doi:10.1145/571637.571640.

[15] Coelho, I. M., Coelho, V. N., Araujo, R. P., Yong, Q. W. & Rhodes, B. D. Challenges of PBFT-Inspired Consensus for Blockchain and Enhancements over Neo dBFT. Future Internet, 2020, 12(8): 129. https://doi:10.3390/fi12080129.

[16] Manoppo, M. Delegated Byzantine Fault Tolerance Consensus Mechanism [J], Medium, 2018, June 15.

[17] Yoo, J., Jung, Y., Shin, D., Bae, M. & Jee, E. Formal Modeling and Verification of a Federated Byzantine Agreement Algorithm for Blockchain Platforms [C], 2019 IEEE International Workshop on Blockchain Oriented Software Engineerin, 2019,. https://doi.org/10.1109/IWBOSE.2019.8666514.

[18] Schwartz, D., Youngs, N. & Britto, A. The Ripple Protocol Consensus Algorithm [R], 2018, retrieved from: https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[19] Lamport, L. The part-time parliament [J], ACM Transactions on Computer Systems. 1998, 16(2): 133-169. https://doi:10.1145/279227.279229.

[20] García-Pérez Á., Gotsman A., Meshman Y. & Sergey I. Paxos Consensus, Deconstructed and Abstracted [M]. In: Ahmed A. (ed.) Programming Languages and Systems. ESOP 2018. Lecture Notes in Computer Science, 2018, 10801: 912-939, https://doi.org/10.1007/978-3-319-89884-1_32.

[21] Lamport, L. Paxos made simple [J], ACM SIGACT News. 2001, 32(4): 51-58.

[22] Ongaro, D. & Ousterhout, J. In Search of an Understandable Consensus Algorithm [C]. Proceedings of the 2014 USENIX Annual Technical Conference (USENIX ATC 14), Philadelphia, PA, USA, 2014, 305-319.

[23] Howard, H. & Mortier, R. Paxos vs Raft: Have we reached consensus on distributed consensus? [C], 7th Workshop on Principles and Practice of Consistency for Distributed Data (PaPoC '20), April 27, 2020, Heraklion, Greece, 2020, https://doi.org/10.1145/3380787.3393681.

[24] Polge, J., Robert, J. & Le Traon, Y. Permissioned blockchain frameworks in the industry: A comparison [J], ICT Express, in press, 2020, https://doi.org/10.1016/j.icte.2020.09.002.

[25] Nasir, Q., Qasse, I.A., Abu Talib, M. & Nassif, A.B. Performance analysis of hyperledger fabric platforms [J], Security and Communication Networks, 2018, vol. 2018. https://doi.org/10.1155/2018/3976093.

[26] Xian, M. NEO White paper [R], 2018, retrieved from: https://github.com/neo-project/docs/blob/3c5530197768d98a1abf7eeed0119a8f4e99e7cc/enus/whitepaper.md.

[27] McCaleb, J., Crain, B.F., Couture, S. & Roy, M. Soundcloud [R], 2017, retrieved from https://soundcloud.com/epicenterbitcoin/eb-128.

[28] Jalalzai, M.M., Busch, C. & Richard, G.G. Proteus: A scalable bft consensus protocol for blockchains [C], 2019 IEEE International Conference on Blockchain, 2019, 308-313.