JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

Byzantine Fault Tolerance Consensus Protocols: A Review

[1]S. K. Alaramma, [2]Adamu Adamu Habu, [3]Maishanu, M., [4]S. W. Mustapha, [5]M. K. Dauda
[1,3&4]Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria,
[2&5]University of St Andrews, Jack Cole Building, North Haugh, St Adrews, KY16 9SX, UK

ABSTRACT
Distributed consensus systems are an innovative means of establishing consensus between multiple parties concerning some piece of information, such as ownership of an asset. Consensus is recorded by cryptographically signing data, thus proving its authenticity. Blocks of signed data can be linked to previous blocks, thereby enforcing a temporal chain of blocks, giving rise to the term blockchain. A Byzantine Fault Tolerant (BFT) based consensus ensures that the blockchain network will continue to operate even in the presence of malicious or failure nodes. BFT consensus has received considerable attention in the last decade due to its promising application in blockchains where classic BFT-based protocols suffer from the scalability and network throughput issues despite its great potentials. Several researches were conducted to improve on BFT to address these shortcomings but still there is more to be done. In this paper, we present a review and analysis of structure, strengths and limitations of the Byzantine Fault Tolerant consensus used in blockchain systems, with a specific focus on how these protocols establish consensus. Despite the attempts to improve on the BFT, this review reveals that the existing BFTs still didn't meet certain performance requirement needed for developing scalable and adaptable BFT based consensus mechanism.

ARTICLE INFO
Article History
Received: January, 2025
Received in revised form: February, 2025
Accepted: May, 2025
Published online: June, 2025

KEYWORDS
Blockchain, BFT, Consensus Mechanism

INTRODUCTION

A blockchain is a distributed and tamper-resistant database that no single entity controls, but can be shared and accessed by all. New records (called blocks) can be added to the existing blocks as long as the new block is approved by all in the network. Also, once blocks are recorded, it is not feasible to modify or erase them. Blockchains have been designed to work on an unreliable network with adversarial entities. In a distributed ledger system, consensus represents a state that there is agreement among all the participants regarding the same data values (Xiao et al., 2020). Consensus is a procedure that allows participants in decentralized or distributed multi-agent platforms to arrive at a common agreement (Panda, 2019). According to Zhao et al. (2019), consensus has been a problem in distributed computing since the early 1980s. There are several well-established methods by which different nodes in a blockchain network can reach consensus over a new block. Among these well-established methods is the Byzantine agreement method which is based on Byzantine generals' problem.

The problem of byzantine agreement was discussed initially in the context of synchronous systems (Lamport et al., 1982), resulting in a recursive algorithm with a factorial message complexity. The first solution with a practical message complexity was Practical Byzantine Fault Tolerance (PBFT) that, until today, serves as an inspiration for most byzantine fault-tolerant consensus protocols. *PBFT*, similarly to most modern approaches, provides safety in a completely asynchronous setting but requires synchronous phases to guarantee progress (Castro & Liskov., 1999). This is a

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

necessary assumption since, as proven in Fischer et al., (1985), it is impossible for a consensus protocol to fulfil both liveness and safety in the presence of asynchronous network conditions. This is known as the famous Fischer, Lynch and Patterson (*FLP)* impossibility theorem (Fischer et al., 1985). As a result, the concept of partially synchronous protocols emerged (Dwork et al., 1988). While, in this model, periods of instability may occur where messages might be delayed arbitrarily (asynchronous network conditions), there is an unknown Global Stabilization Time (GST) and a known worst-case network latency _ where, after GST, messages arrive within prescribed time. Thus, in this model, safety is always guaranteed, while liveness is only established after GST.

While there are protocols like Honeybadger (Miller et al., 2016) that offer guaranteed liveness even under asynchronous network conditions, they do not provide deterministic safety (i.e., these protocols only provide safety at a very high probability). In addition to that, this group of protocols requires a substantial network complexity ($O(N^4)$). In this paper, we compare the pros and cons of various implementations of BFT based consensus mechanism and proposed five criteria for evaluating their overall performance. We highlight some of the research challenges that the various models need to be addressed for resilient and scalable Byzantine Fault Tolerance Consensus.

The rest of the article is organized as follows. In section II, we briefly discuss the classification of the implementation of BFT. In section III, we discuss the structure of various BFT based models under the classification presented and also their strengths and limitations. In section IV, we provide a comparative analysis of the models. In section V, we put forward some of the open research challenges. Conclusions are drawn in the last section.

CLASSIFICATION OF BFT PROTOCOLS

Generally, BFTs could assume leader-based approaches, where a single process proposes a value for each round of consensus, which all other processes will then agree on. In this context, the existing approaches were classified into subgroups based on the communication pattern that is used by the leader to disseminate the proposal and by the remaining process to exchange their agreement.

In Clement et al., (2009) the following approaches as shown in figure 1 were identified.

1. *All-to-All:* Leader process broadcasts block, all processes broadcast their vote and verify and collect votes of all other processes.
2. *Star:* Leader process broadcasts block. All processes send their vote to the leader that collects and verifies the votes and relays them in a batch once reaching a majority.
3. *Gossip:* Leader process propagates block through gossip. All processes receive, verify, aggregate, and relay aggregates through multiple rounds of gossip.
4. *Committee:* A small subset of processes is randomly selected. The leader sends the proposal to all processes in this committee, committee members exchange their vote internally and verify and collect votes of all other processes within the committee. After the end of consensus, each committee member distributes the result to the remaining processes before a new committee is elected.
5. *Hierarchical:* First, the leader process disseminates a message to a set of cluster primaries. Next, cluster primaries verify and relay the message to their cluster members. Then, all processes send their vote to their respective primary, the primaries aggregate, verify and then exchange their respective aggregates with the other primaries. Finally, after constructing an aggregate consisting of sufficient votes, each primary relays this to their cluster members and persist the result.
6. *Tree:* Leader process sends a value to their child processes in the tree, child

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

processes relay to their respective child processes until reaching the leaf nodes. Next, all processes verify, aggregate, and relay the votes up the tree back to the leader. Finally, the leader collects the last aggregate and then restarts the distribution process.

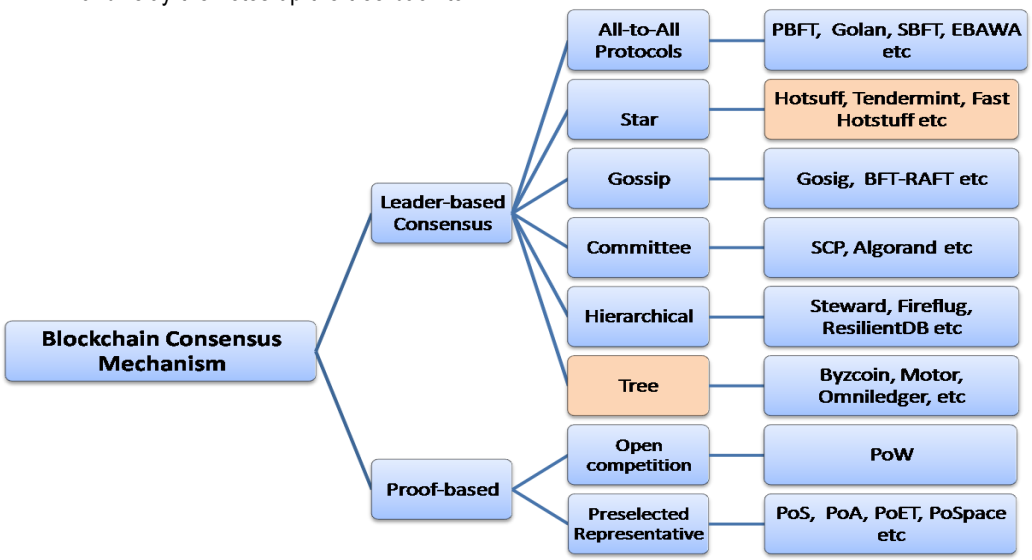

Figure 1: Taxonomy of Blockchain Consensus Protocols

EXISTING BYZANTINE FAULT TOLERANT CONSENSUS

In the following, we present the most dominant proposals based on the classification above.

Byzantine Fault Tolerant Consensus

In the following, the most dominant proposals based on the classifications above are presented.

1. **Hydrachain**. Hydrachain is among the consensus mechanism with low overhead cost. As an extension of BFT, each round of consensus is initiated when two-third (2/3) of the participating nodes vote in the previous round and round robin method is utilised in selecting the prosper of the block in each round. Clearly, this implementation of BFT heavily rely on a set of validators to confirm sequence of transactions and proposed block to be added are presented with a quorum of verifiable signatures. This however, constitute the major drawback of this approach (Sasson et al., 2014).
2. **Practical BFT (PBFT).** *PBFT* is a leader-based protocol where one of the participants fulfils the unique role of the *leader* (in this case N_0). The protocol starts with the client sending transactions to the leader. If the leader is correct and is not suspected to be faulty, it will propose a value as input to consensus in the *pre-prepare* phase. This may either be a single transaction or a batch of transactions (i.e. a block). Next, each process verifies the proposal of the leader and, if the proposal follows the previously agreed-upon rules, each process broadcasts their vote with the proposed value in the prepare phase. At the end of this phase, each process verifies the received prepare messages, and if a full byzantine quorum was collected ($2f + 1$ prepare messages), it locks the value and broadcasts a commit message to notify the other

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

processes. After receiving *2f + 1* commit messages, each process can be certain that the majority of processes (at least f +1 honest processes) did receive a full quorum in the prepare phase. They persist the agreed-upon value and notify the client about the result. If any of these phases fails (i.e., a timeout is reached), a new leader is selected through a view-change protocol. The view change protocol works very similarly to the consensus protocol. After the timeout is reached, each process determines the next leader deterministically (i.e., rotating the leader in round-robin fashion) and broadcasts the selection in a view-change message. After receiving *f + 1* view-change messages, any process that did not run into the timeout yet also determines a new leader and broadcasts its decision. This way, correct processes that were lagging behind may catch up to the remaining processes. Finally, after receiving a byzantine quorum of view-change messages (*2f + 1*), the elected process assumes the role of the leader and restarts the protocol. This algorithm results in a cost of $O(n^2)$ for each view-change, and, as there is a possibility of f subsequent leaders, it might have to be repeated f times. As such, in the worst case, this results in a cost of $O(n^3)$.

3. **Modified Practical Byzantine Algorithm (mFBA).** Is a combination of BFT and PoS implemented in BOSCoin (He et al., 2018) with the PoS incorporated to handle the parts that relate to the governance framework. It is based on hierarchical structure and BOSCoin as a cryptocurrency operator with its PoS, allows participants to lock their assets (coins) over a period of time. The frozen processes are incentivize based on the number of coins they freeze while still maintaining high level of security and integrity. Even with the high level of security, sometimes malicious nodes forging the blockchain are detected and the corresponding locked currency is forfeited leading to serious issues reaching consensus (Smorgunov, (2018); Miller et al., (2016)).
4. **Honey Badger BFT (HB BFT).** As an extension of classical BFT without the need of synchronicity, HB BFT used buffer to tackle deficient network bandwidth to allow a sequence of transaction to be recorded when they reach consensus. Unlike some BFT variants, this can be seen as an approach that served as transaction processing gateway with asynchronous network model through the underlying network or timing propositions. The initial implementation using gossip communication has shown that HB BFT can handle up to 20,000 transactions per second for network with less than 40 nodes (Miller, et al., 2016).
5. **Steward.** Steward is based on hierarchical approach where servers are grouped in hierarchic groups (or clusters), and each cluster acts as a logical unit and individually creates a signature aggregate representing the decision of the group. As such, one process of each group may then use this signature aggregate when attempting to achieve consensus with the remaining group leaders on the global level. This approach requires $O(n^2)$ (where *n* represents the group size) messages within each group and $O(k)$ messages on the global level (where *k* represents the number of groups). As a result, this very successfully reduces the communication complexity and distributes the load significantly. However, in comparison to *PBFT*, *Steward* offers significantly lower resilience guarantees, where, instead of having a global failure limit, failures are limited on a local level.

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

6. **SUMERAGI**. SUMERAGI is a BFT implementation reaching consensus even in the presence of faulty node inspired by BChain and it's the consensus algorithm used for Hyperledger Iroha (Duan et al., 2014). Nodes participating in consensus are divided into different groups in which the first group consist of *2f + 1* (where *f* is the number of faulty nodes) nodes while the remaining nodes formed the second group. Committee based approach is used and the first group are responsible for validating the transaction to be added to the chain while the second group verify the signature and content of the block before the block is persisted. When all is done in the correct order, the blockchain is updated accordingly (Borran and Schiper, 2010).
7. **Scalable BFT** (**SBFT**). SBFT is an extension of BFT proposed to enhance the scalability of the blockchain network by using application. In a network consisting of *3f + 1* nodes, SBFT achieve consensus when *2f + 1* block validators validate the transactions within 24 hours interval. Result of experiment conducted with SBFT show that it demonstrated robustness in the propagation of message to active replicas (Jiang and Lian, 2019).
8. **Golan** is an optimization of PBFT where each process creates a single aggregated signature. Therefore, the processes only have to broadcast and verify a single signature each round instead of the whole set of signatures. This strategy significantly reduces the bandwidth load of the subsequent rounds and results in a lower computational burden on all processes. However, based on the above algorithm, in both SBFT and PBFT, the leader clearly has more work than the remaining participants, as it needs to collect the requests from clients, broadcast the batch of transactions, and also run the same protocol as the remaining participants. One way to alleviate the load on the leader is by rotating the role of the leader among all participants in every consensus instance. Another reason to rotate the leader regularly is to reduce the potential influence of a faulty leader process (Golan et. al., 2019).
9. **Tendermint**. Tendermint rely on star topology with a protocol that communicate with the participating nodes through a selected leader (Dwork et al., 2018). Tenermint used three stages (pre-vote, pre-commit and commit) to reach consensus. At each stage, each node needs to have its asset frozen. Nodes are however, incentivised or penalised based on their roles in the validation tasks when a proposal for need new block is executed. For a block to be added to the blockchain, a minimum of 2/3 of the participating nodes need to vote in that direction at the commit stage (Zheng et al., 2018; Buchman, 2016).
10. **Fireplug**. Fireplug reduces the cost of the architecture by requiring the leaders to attach a proof containing the signature of enough other leaders when sending decisions to their local cluster members. It runs a byzantine fault-tolerant protocol in the global group where a majority of leaders have to sign each decision before updating the client with the result. By deploying several local group representatives at the global level, it is possible to run the system with less than $3f + 1$ data centers overall. Nevertheless, while it may tolerate faults on any level of the hierarchy and local group members may detect a wide range of byzantine leadership behavior, a majority of byzantine leaders may slow down the system significantly. As such, the actual maximum number of simultaneous faults at the global level is restricted to $k = 3f + 1$ where k presents

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

the number of replicas in the global group, which is unsuitable for highly adversarial environments since additional replicas in the local groups do not increase the worst-case resilience of the system (Neiheiser et al., 2018).

11. **Istanbul BFT**. Istanbul BFT used the typical three stage BFT implementation procedure of pre-prepare, prepare and commit to reach consensus among nodes that constitute the network with a gossip topology. Round robin scheduling is used to select nodes randomly for block prosper where each round requires *2f + 1* votes from block validators as a sign of block verification to proceed to the next round. At the commit stage, data is broadcasted to all nodes when *2f + 1* votes are ensured to finalize block persistence in the network (Moniz, 2020).
12. **Spinning** and its extension *EBAWA* allow multiple instances of consensus to run in parallel (with different leaders) optimistically, further improving the performance of the system. Another strategy consists in carefully selecting the node depending on the network conditions to play the role of the leader (Veronese et al., 2009; Veronese et al., 2010).
13. **ResilientDB**. ResilientDB is a more recent approach which combines hierarchical algorithms similar to Steward or Fireplug within a blockchain. In addition to the hierarchical approach, which distributes the load among the groups, ResilientDB leverages sharding to utilize the left-over resources to boost the system throughput significantly. Nonetheless, similar to the other hierarchical approaches, it comes with restrictive failure assumptions, as a majority of honest nodes is required within each sub-group (Rahnama et al., 2020).
14. **Leader Free Byzantine (LFB).** In a leader-based BFT consensus mechanisms, the leader is responsible for broadcasting block proposal and the final verification of the votes. LFB was proposed to be a deterministic asynchronous consensus protocol that ensure that messages are transmitted by node selected randomly and security and liveness of asynchronous protocol is maintained (Borron and Schiper, 2010).
15. **Byzcoin**. Byzcoin leverages a binary tree and uses *collective co-signing* (a cryptographic scheme) to collect two quorums through a tree requiring three round-trips up and down the tree for each quorum. This approach, therefore, results in a significant latency overhead which affects the potential maximum throughput. In addition, Byzcoin uses verifiable random functions to construct a random tree each round, and if within a given time-frame no consensus is achieved, it falls back to an all-to-all scheme (Kokoris-Kogias et al., 2016).
16. **YAC**. YAC was proposed to achieve consensus even in the presence of *f* faulty nodes in *3f + 1* nodes participating in the network. It is a decentralised BFT with the ability to overcome the problems associated with the leader based BFT in message propagation and collation. When a number of tests were conducted to evaluate the performance of YAC in Hyperledger Ioha, the results illustrated that YAC has potential for scalability but the voting stage has to be modified to reduce the exposure of the voting nodes to systemic failure (Muratov et al., 2018).
17. **Acher**. In Archer, where the system can perform a view change, not only when the current leader is faulty but also when another node could exhibit better performance than the current leader. Being careful in selecting the leader, or rotating the leader, can help the protocol but does not combat the inherent scalability issues. As at some point in

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

the protocol, one or more processes have to broadcast a given value to all processes (bandwidth bottleneck) and receive and verify *n* signatures (CPU bottleneck) (Eischer & Distler, 2018).

18. **Federated Byzantine Fault Tolerance (FBFT)**. This implementation of BFT has been used by a number of cryptocurrencies blockchains prominent among them are Stellar and Ripple as their underlying consensus mechanisms. FBFT is among the class of BFT with high fault resistance and ability to scale as the number of nodes increases. It uses hierarchical topology communication method and for a block of transaction to scale in FBFT, 80% of the nodes must generate its unique node list and vote (Mazieres, 2015; Huang et al., 2019).
19. **Motor**. Motor is a continuation of Byzcoin, which restricts the tree to a depth of two to decrease the actual latency cost, and uses bls signatures that do not require several round-trips to construct a single quorum. It constructs trees in a similar manner as Byzcoin, but instead of falling back to an all-to-all scheme, in the presence of failures, it slowly falls back to a star topology within $\frac{N}{m}$ steps (for a fanout of *m*). As such, in the worst case, it requires $f + 1 * \frac{N}{m}$ steps to find a robust configuration (Kokoris-Kogias, 2018).
20. **Hashgraph**. Hashgraph or Asynchronous BFT Hashgraph is an asynchronous BFT utilising star topology. In this protocol, participating nodes are allowed to communicate with one another and perform synchronization. In this however, consensus can be reached by restriction that only messages sent by honest nodes will be propagated. Hashgraph make the network robust and ensure that the network is preserved even when 1/3 nodes in the network behaving maliciously. Special data structures are used to store transactions with cryptographic security (Behl et al., 2014; Jeon et al., 2018).
21. **Optimistic Protocols**. Optimistic protocols are based on the observation that in fault-free runs, it may be possible to update the client about the result earlier. To achieve this, they run a combination of two sub-protocols: an optimistic sub-protocol that only provides termination in fault-free runs and a recovery sub-protocol that needs to run when faults occur (e.g., somehow inconsistencies are detected, or consensus is not achieved). The recovery protocol typically follows the structure of the original *PBFT* protocol, while the optimistic protocol typically uses fewer messages and/or fewer communication rounds. Thus, in failure-free runs, optimistic protocols can be more efficient but in faulty runs they are more expensive because both sub-protocols are executed sequentially (Eischer & Distler, 2018).
22. **Stellar Consensus Protocol (SCP)**. In SCP, participating nodes are divided into two sets where each set handles a portion of the consensus protocol. A quorum of nodes making the first set is responsible for reaching the consensus while the other set ensure agreement is reached on the state of the network. Validator nodes are used to reach agreement without the need to trust the whole network. SCP use tree topology and nomination protocol to verify agreements and feed node with the nomination ballots (Mazieres, 2015; Sankar et al., 2017).
23. **WHEAT**. *WHEAT* is an extension of optimistic approach and the optimistic sub-protocol follows the same structure of *PBFT* but is executed only among a subset of *f + 1* participants. Although the message complexity is still quadratic, namely O(n^2), in practical terms, it is

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

more efficient than a protocol where at least *3f + 1* nodes need to participate. If the optimistic protocol succeeds, the result is propagated to the remaining processes in the background; otherwise, a recovery protocol involving all nodes is executed (Sousa & Bessani, 2015).

24. **Proof of Quality of Service (PoQ)**. PoQ is a combination of BFT and proof-based mechanisms used to improve consensus. It relies on all to all approach and quality of service where each node in the network is nominated based on its region as a representation of the node's quality of service in the region. Classical BFT is used to perform the election of the representative. Experimental results obtained from series of tests carried out with PoQ show that it significantly improves the throughput of BFT with an equal opportunity for node participation in the network (Yu et al., 2019).
25. **Eplis**. Eplis is a BFT compliant consensus protocol based on tree topology and incorporates geo-scale implementation of locality of communication threads while moving to the direction of speedy agreement. Eplis utilises large size of replica by connecting coin ownership with dynamic command as oppose to the representation of coin support in the locked asset. This is used to ensure high throughput for the network and to maintain serial mechanism similar to the one obtained in SBFT and Hashgraph (Garg et al., 2019).
26. **Omniledger**. Omniledger is the final evolution of Byzcoin and Motor and attempts to improve the throughput significantly through sharding by making each sub-tree a shard of the system that decides on a given subset of transactions and deploys a directed acyclic graph to deal with conflicting transactions. However, as each shard is smaller than the overall system size N, Omniledger trades resilience against additional throughput and, as it leverages acyclic graphs to achieve cross-shard consensus, it requires significantly longer finalizing a given block (Kokoris-Kogias et al., 2019).
27. **Zyzzyva**. Among the major issues with most BFT implementations that Zyzzyva address is that of three phase commit protocol necessary before committing a block. Zyzzyva proposed a speculative mechanism as a substitute for the three-phase commit. The speculative mechanism involves view change, three levels of agreements and a checkpoint where fast pattern follows a primary node to send its request to all the replicas for speculative agreement. Two ways pattern combining the replica and the client is utilised for coherent response and *2f + 1* responses are required as with other BFTs to persist block of transactions to the blockchain networks (Kotla et al, 2010).
28. **HotStuff** is a recent algorithm that uses a communication pattern based on a starnetwork (Abraham et al., 2018). HotStuff consists of three rounds where all communication runs through the leader that collects and distributes aggregated signatures. The protocol has a linear communication cost but takes eight communication steps to terminate (4 phases of two steps each). This process is shown in Figure 2. The protocol is very similar to PBFT. In the first step, a client sends transactions to the current leader. The leader, similarly to PBFT, disseminates a proposal in a prepare message. Finally, if the proposal is valid (i.e., consists of valid transactions), processes send their vote to the leader as a response. After aggregating *2f + 1* votes the leader broadcasts the result in the pre-commit step. Processes verify the set of votes (i.e., the set of signatures or the

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

signature aggregate) and, if valid again, signal the leader their agreement. The leader again aggregates *2f + 1* votes (this time over the pre-commit) and notifies the remaining processes in a commit message. In turn, after receiving a correct commit message, processes lock the result and again notify the leader. In the final step, the leader aggregates *2f +1* commit response messages and broadcasts the decision. The remaining processes, after receiving a valid decision message (consisting of *2f + 1* valid commits) then persist the result. To compensate for the additional number of communication steps, HotStuff allows up to four steps of different instances of consensus to run in parallel. This is displayed in Figure 3. More precisely, the first step of the n+1th instance of consensus is executed in parallel with the second step of the nth instance of consensus, and so forth. It is important that the conduit mechanism is done precisely in this manner, since in a blockchain scenario, the n + 1th instance depends on the nth instance. Despite its apparent benefits in terms of message complexity, algorithms that use a communication pattern based on a star network have a significant drawback: the protocol becomes limited by the physical capacity of the leader node both in terms of bandwidth and CPU.

29. **Ouroboros**. Ouroboros is an extension of *Zyzzyva* where, instead of relying on the clients to detect inconsistencies, the results are published on the blockchain and are, if necessary, solved through forks. Thus, similarly to PoW blockchains, there is a possibility that blocks are revoked later on. Therefore, *Ouroboros* also does not offer deterministic finality (Kiayias et al., 2017).
30. **Dual Byzantine Fault Tolerance (DBFT).** In BFT all participating nodes are required to contribute to the appending of new block to the blockchain network. The restriction is relaxed in DBFT as only a subset of the nodes in the network are saddled with the responsibility of validating transaction and generating new blocks. In DBFT, some randomly selected nodes called the bookkeepers propose the next block while all the other nodes in the network determine through votes who will be the bookkeepers. Only block proposal that secured 66% votes from the bookkeepers are added to the chain in the network (Biswas et al., 2019).
31. **Gosig**. In Gosig, the leader election is done by executing verifiable random functions. This approach decreases the risks of potential attacks on the elected leader as the leader is only known at the moment of the consensus and not ahead of time. While there is a chance of no or multiple leaders being elected in the same round, requiring a new election, this approach is still advantageous in highly adversarial environments. Protocols as *PBFT* or *Spinning* where the leader is always known beforehand can suffer denial of service attacks on their leaders, which can affect the liveness of the system. Additionally, the communication is optimized by relying on epidemic message propagation (gossiping) between the replicas and applying multi-signatures to improve the bandwidth usage. However, at the moment of writing, byzantine fault-tolerant gossip still requires at least multiple rounds of $O(N * log(n))$ messages, which is more than O(N) of the star-based approach (Li et al., 2020).
32. **Proof of Previous Transactions (PoPT).** PoPT is a BFT compliant consensus protocol that use PBFT mechanism for node selection from the set of participating nodes in the network. It corporate accounting-based scheme

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

with impartial accounting scheme to mitigate the issue of wrong incentive allocation. PoPT increase the chance of more than a single account regime where each accountant administers a transaction differently leading to simultaneous block persistence in the blockchain. When used in ledger, the protocol has shown a high level of throughput and scalability (Xiang et al., 2019).

33. **Kauri**. Kauri was proposed to be an extension of HotStuff with the aim of improving its robustness with the use of all-to-all communication pattern. Kauri is deterministic consensus mechanism that enables the participating nodes in the network to democratically carryout a decision in relations to the consensus. Even in the presence of failed coordinator, Kauri provides termination and scalability in a partially synchronous setting (Crain et al., 2018).
34. **Dynamic BFT**. Dynamic BFT allows replicas to communicate with the client simultaneously without any need to detect instability in the network that will hurt performance and decline throughput. Dynamic BFT is a leader based BFT that employs speculation like Zyzzyva to address resilience issues associated with BFTs using novel proposed technique called double-response. Results of experiments conducted to evaluate the performance of Dynamic BFT revealed that it outperforms several mechanisms like PBFT and Zyzzyva in the area of fault tolerance, security and liveness of the blockchain network (Zhang et al., 2019).
35. **Diversity of Opinion BFT (DoOBFT)**. DoOBFT employs the use of two-level agreement to reach consensus. At the first level, classical BFT is used as consensus algorithm to increase the chances of reaching agreement even in the presence of faulty nodes. In the second level, verifiers of the block are randomly chosen to reduce the chances of collision and ensure integrity of the blockchain network. The two levels must be executed for the complete consensus leading to block addition be performed (Puthal et al., 2019).
36. **BFT Smart.** BFT Smart is an extension of PBFT with modular support and node reconfiguration. When the modularity and reconfiguration is added to the mix, it outperformed its predecessors PBFT (Castro and Liskov, 1999) and Upright (Clement et al., 2009) in a number of arears including fault tolerance and throughput. Since its inception in 2017, several improvements have been added to BFT smart to increase transactions processing, security and range of application it can serve (Bessani et al., 2014).

Table 2: Summary of the different consensus protocols.

Protocol	Author(s)	Resilience	Network Model	Load	Finality	Scalability	Topology	Latency	Recovery
Byzcoin	(Kokoris et al., 2016)	N/A	Partially	Yes	N/A	Low	Star	N/A	No
Omniledger	(Kokoris et al., 2018)	N/A	Synchronous	Yes	Deterministic	Moderate	Star	N/A	No
SCP	(Baliga, 2017)	<50%	Partially	Yes	N/A	Low	Tree	Low	Yes
BFT Smart	(Bessani et al., 2014)	N/A	Synchronous	No	Deterministic	Moderate	All-to-All	N/A	No
Steward	(Yu et al., 2019)	< 33%	Synchronous	Yes	Instant	Low	Star	N/A	No

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

Protocol	Author(s)	Resilience	Network Model	Load	Finality	Scalability	Topology	Latency	Recovery
Dyna BFT	(Zhang et al., 2019)	N/A	Asynchronous	Yes	N/A	N/A	Hierarchical	Low	No
Kauri	(Crain et al., 2018)	<33%	Partially	Yes	Probabilistic	Low	Tree	High	Yes
ResilientDB Layer	(Rahnama et al., 2020)	<33%	Synchronous	Yes	Probabilistic	Low	N/A	High	No
PoPT	(Xiang et al., 2019)	N/A	Partially	Yes	Deterministic	Low	Gossip	N/A	No
Gosig	(Li et al., 2020)	N/A	Synchronous	Yes	Instant	Moderate	Star	N/A	No
Ebawa	(Veronese et al., 2010)	N/A	Synchronous	No	N/A	Moderate	Star	High	Yes
DBFT	(Biswas et al., 2019)	N/A	Asynchronous	Yes	Deterministic	Low	Committee	N/A	No
Ouroboros	(Kiayias et al., 2017)	N/A	Synchronous	Yes	Deterministic	Low	Hierarchical	N/A	No
Zyzzyva	(Kotla et al, 2010)	< 33%	Synchronous	Yes	Deterministic	Low	Gossip	N/A	No
HB BFT	(Miller, et al., 2016)	N/A	Asynchronous	No	N/A	N/A	Star	N/A	Yes
mFBA	(Smorgunov, (2018)	N/A	Synchronous	No	Deterministic	N/A	N/A	Low	No
Algorand	(Gilad et al., 2017)	<=50%	Partially	Yes	Probabilistic	Low	Committee	N/A	No
DoOBFT	(Puthal et al., 2019)	N/A	Synchronous	No	N/A	Moderate	All-to-All	N/A	Yes
Hydrachain	(Sasson et al., 2014)	< 33%	Synchronous	No	Instant	High	Gossip	Low	Yes
Steward	(Amir et al., 2010)	N/A	Synchronous	Yes	Deterministic	Low	Star	Low	No
SUMERAGI	Borran & Schiper,	N/A	N/A	No	Deterministic	N/A	Gossip	High	Yes
Eplis	(Garg et al., 2019)	N/A	Synchronous	No	Deterministic	High	Tree	N/A	Yes
PoQ	(Yu et al., 2019)	<=50%	Synchronous	No	Immediate	Low	Committee	High	No
WHEAT	Sousa & Bessani,	N/A	Synchronous	Yes	N/A	Moderate	Tree	High	No
HoneyBadger	Eischer & Distler, 2018	N/A	Asynchronous	No	N/A	Low	All-to-All	N/A	No
Fireplug	(Neiheiser et al., 2018)	N/A	Synchronous	Yes	Deterministic	Moderate	All-to-All	N/A	Yes
SBFT	(Jiang and Lian, 2019)	< 33%	Asynchronous	No	Deterministic	High	Star	Low	No
Hashgraph	(Jeon et al., 2018)	N/A	Synchronous	No	N/A	Low	N/A	N/A	Yes
Motor	(Kokoris-Kogias,	N/A	Synchronous	No	Deterministic	Moderate	Gossip	Low	No
HotStuff	(Abraham et al., 2019)	N/A	Partially	No	Deterministic	Low	Star	N/A	Yes
FBFT	(Huang et al., 2019)	< 33%	N/A	No	Immediate	High	Hierarchical	N/A	Yes
Acher	(Eischer & Distler,	N/A	Synchronous	Yes	N/A	Low	N/A	Low	Yes
PBFT	(Castro et al., 2019)	N/A	Synchronous	No	Deterministic	Low	All-to-All	Low	Yes
YAC	(Muratov et al., 2018)	N/A	N/A	Yes	N/A	High	N/A	Low	No
LFB	(Borron and Schiper,	N/A	Synchronous	Yes	Deterministic	N/A	Tree	N/A	No
Spinning	(Veronese et al., 2009)	< 33%	Asynchronous	No	Deterministic	Moderate	Star	N/A	No

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

Protocol	Author(s)	Resilience	Network Model	Load	Finality	Scalability	Topology	Latency	Recovery
Golan	(Golan et. al., 2019)	N/A	Synchronous	Yes	Deterministic	Low	Tree	High	Yes
Istanbul BFT	(Moniz, 2020)	N/A	N/A	No	Deterministic	N/A	Gossip	N/A	Yes
Fireplug	(Neiheiser et al., 2018)	N/A	Asynchronous	Yes	N/A	Low	Hierarchical	Low	No
Tendermint	(Buchman, 2016)	< 33%	N/A	Yes	N/A	Low	Star	N/A	Yes

COMPARATIVE ANALYSES OF THE EXISTING BFT PROTOCOLS

In this section, we present five performance criteria and we use the criteria to conduct a comparative analysis of existing BFT protocols. As can be seen, table 1 summarizes up the discussion about the different consensus protocols. In the Table above, the third column specify the resilience of a given approach to attack. This is shown as the percentage of overall nodes the approach can tolerate to be faulty. The fourth column specifies the network protocol an approach in communication with participating nodes. The fifth column specify how the approach distributes the computational and bandwidth load among a set of nodes. The sixth column defines if a system fulfils deterministic finality (i.e., a block is irreversible after a specific number of steps).

Then, in the seventh column, the level of scalability a given approach displays is specified. The network topology a given approach utilises is given in the eighth column. Different levels of pipelining which fulfils the condition of providing high throughput independent of the network latency, are applied to a number of approaches and in the nineth column, the level of pipelining used is specified. Finally, if an approach can recover deterministically in a linear number of configuration steps is specified in the last column. While traditional protocols like PBFT, EBAWA, and HotStuff provide deterministic finality, N = 3f + 1 resilience, and quick recovery, in either protocol, one or more processes have to broadcast the block and process and verify N signatures, which is a major bandwidth and CPU bottleneck, as such they do not offer any load-balancing properties. In addition, in a geo-replicated scenario, only Ebawa can compensate for the inherent latency overhead by running multiple consensus rounds and steps asynchronously in parallel. Nonetheless, Ebawa assumes non-conflicting values, which is not the case in the blockchain environment where each block extends the previous one.

Early hierarchical protocols offer the load balancing and deterministic finality the previous protocols lack, but at a tradeoff as they have to reduce their resilience significantly. In addition to that, none of these approaches achieves high throughput in a high latency setting, and only multi-layer offers a reconfiguration algorithm that allows quick recovery. Modern committee-based approaches like SCP and Algorand may reduce the cost significantly by having only a committee run the consensus. However, this is done at the cost of deterministic finality. In addition to that, neither approach deals with the inherent latency cost of a geographical deployment resulting in vast idle times.

Finally, existing tree-based approaches offer load balancing and deterministic finality but again lack compensation for geographical deployments. Omniledger attempts to compensate this partially through sharding but gives up a large part of their resilience in the trade.

CONCLUSIONS AND FUTURE WORK

Byzantine Fault Tolerant (BFT) consensus has received considerable attention in the last decade due to its promising application in blockchains. In this review several state-of-the-art BFT based protocols that have been proposed in the literature to address the scalability and throughput issues inherent to classical BFT. Despite their contribution, none of these proposals have provided a comprehensive solution to the issues. Thus, the result of our review has shown that even though there is tremendous improvement over the classical BFT, the existing BFT consensus mechanisms analysed still need improvement. The existing protocols were

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

evaluated based on carefully crafted criteria proposed and none of them passed all the criteria. This indicates that a gap exists in this area of research. To fill this gap, a suitable analytical framework is required for studying how these implementations of BFT provide solution to the inherent problem. We plan to develop a scalable and low latency BFT consensus mechanism that will meet all the criteria in our future research.

REFERENCES

Abraham, I., Gueta, G., and Malkhi, D. 2018. Hot-Stuff the Linear, Optimal-Resilience, One-Message BFT Devil. In: *CoRR* abs/1803.05069. arXiv: 1803.05069. url: http://arxiv.org/abs/1803.05069.

Abraham, I., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M., Tamir, O., and Tomescu, A. 2019. SBFT: A Scalable and Decentralized Trust Infrastructure. In: *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. P. 568–580.

Amir, Y., Danilov, C., Dolev, D., Kirsch, J., Lane, J., Nita-Rotaru, C., Olsen, J., and Zage, D. 2010. Steward: Scaling Byzantine Fault-Tolerant Replication to Wide Area Networks. In: *IEEE Transactions on Dependable and Secure Computing* 7.1 (2010), P. 80–93.

Amsden, Z. 2020. *The Diem Blockchain*. url: https://developers.diem.com/docs/technical-papers/the-diem-blockchain-paper/ (visited on 07/02/2023).

Artur, B., Ermol'ev, Y. M. and Kaniovskii, Y. M. (1983). A Generalized Urn Problem and its Applications. In: *Cybernetics* 19.1 (1983), pp. 61–71.

Baliga, A. 2017. Understanding Blockchain Consensus Models. In: *Persistent* 4 (2017), P. 1–14. url: https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf (visited on 08/02/2023).

Block'tivity Team. 2022. Block'tivity. In: (2022). url: https://blocktivity.info/ (visited on 08/02/2023).

Boneh, D., Lynn, B., and Shacham, H. (2004). Short signatures from the Weil pairing. In: *Journal of cryptology* 17.4 (2004), P. 297–319

Brown, R. G., Carlyle, J., Grigg, I., and Hearn, M. 2016. Corda: an Introduction. In: *R3 CEV, August* 1 P. 15. url: https://blockchainlab.com/pdf/cordaintroductory-whitepaper-final.pdf (visited on 09/02/2023).

Buchman, E. 2016. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains. PhD thesis. (cit. on P. 1, 19).

Cachin, C., Guerraoui, R., and Rodrigues, L. (2011). *Introduction to Reliable and Secure Distributed Programming*. 2nd. Springer, 2011 P. 16 and 34.

Castro, M., and B. Liskov, B. 1999. Practical Byzantine Fault Tolerance. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. OSDI '99. New Orleans, Louisiana, USA: USENIX Association, 1999, P. 173–186.

Cebe,M. (2018). Block4forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles. IEEE Commun. Mag. **56**(10), 50–57 (2018)

Clement, A., Wong, E., Alvisi, L., Dahlin, M., and Marchetti, M. 2009. Making Byzantine Fault

Tolerant Systems Tolerate Byzantine Faults. In: *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. NSDI'09. Boston, Massachusetts: USENIX Association, 2009, P. 153–168.

Cohen, B. and Pietrzak, K. (2020). *The chia network blockchain*. 2020. url:https://www.chia.net/assets/ChiaGreenPaper.pdf (visited on 15/02/2023).

Dwork, C., Lynch, N., and Stockmeyer, L. 1988. Consensus in the Presence of Partial

Corresponding author: S. K. Alaramma
✉ skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

Synchrony. In: *Journal of the ACM* 35.2 (Apr. 1988), P. 288–323.

Eischer, M., and Distler, T. 2018. Latency-Aware Leader Selection for Geo-Replicated Byzantine Fault-Tolerant Systems. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and NetworksWorkshops (DSN-W)*. 2018, P. 140–145.

Ferdous, M.S. (2020). Blockchain consensus algorithms: a survey. arXiv (2020)

Fischer, M. J., Lynch, N. A., and Paterson, M. S. 1985. Impossibility of Distributed Consensus with One Faulty Process. In: *Journal ACM* 32.2 (Apr. 1985), P. 374–382.

Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. (2017). Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In: *Proceedings of the 26th Symposium on Operating Systems Principles*. SOSP '17. Shanghai, China: Association for Computing Machinery, 2017, P. 51–68.

Golan, G., Gueta, I. A., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M., Seredinschi, D.A., Tamir, O., & Tomescu, A. 2019. SBFT: A Scalable and Decentralized Trust Infrastructure. In: *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2019, P. 568–580.

Gramoli,V. (2020): From blockchain consensus back to Byzantine consensus. Futur. Gener. Comput. Syst. **107**, 760–769 (2020). https://doi.org/10.1016/j.future.2017.09.023

Grigg, I. 2017. EOS, An Introduction. In: (2017). url:https://iang.org/papers/EOS_Una_Introduccio%5C%CC%5C%81n-IanGrigg-Trad_MAM+AIH-20180717.pdf (visited on 07/02/2023).

Herlihy, M. (2019) Blockchains from a distributed computing perspective. Commun. ACM **62**(2), 78–85 (2019)

Kiayias, A., Russell, A., David, B., and Oliynykov, R. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In: *Advances in Cryptology – CRYPTO 2017*. Springer International Publishing, 2017, P. 357–388.

Kokina, J. (2017). Blockchain: emergent industry adoption and implications for accounting. J. Emerging Technol. Account. **14**(2), 91–100 (2017)

Kokoris-Kogias, E., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., and Ford, B. 2016. Enhancing

Bitcoin Security and Performance with Strong Consistency via Collective Signing. In: *USENIX Security*. Austin (TX), USA, 2016, P. 279–296.

Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., and Ford, B. 2018. OmniLedger: A

Secure, Scale-Out, Decentralized Ledger via Sharding. In: *2018 IEEE Symposium on Security and Privacy (SP)*. P. 583–598.

Kokoris-Kogias, E. 2019. Robust and Scalable Consensus for Sharded Distributed Ledgers. In: *IACR Cryptology ePrint Archive* 2019 (2019), P. 676.

Kotla, R., Alvisi, L., Dahlin, M., Clement, A., and Wong, E. 2010. Zyzzyva: Speculative Byzantine Fault Tolerance. In: *ACM Transactions on Computer Systems* 27.4 (2010), P. 1–39.

Lamport, L., Shostak, R., and Pease, M. 1982. The Byzantine Generals Problem. In: *ACM Transactions on Programming Languages and Systems* 4.3 (1982), P. 382–401.

Larimer, D., Scott, N., Zavgorodnev, V., Johnson, B., Calfee, J., and Vandeberg, M.2016. Steem: An Incentivized, Blockchain-Based Social Media Platform. In: *March. Self-published* (2016). url: https://steem.com/SteemWhitePaper.pdf (visited on 07/02/2023).

Leonardos, S. (2020). PREStO: A Systematic Framework for Blockchain Consensus

Corresponding author: S. K. Alaramma
✉ *skalarammane@gmail.com*
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

Protocols. IEEE Trans. Eng. Manage. **67**(4), 1028–1044 (2020)

Li, P., Wang, G., Chen, X., Long, F., and Xu, W. (2020). Gosig: A Scalable and High-Performance Byzantine Consensus for Consortium Blockchains. In: *Proceedings of the 11th ACM Symposium on Cloud Computing*. SoCC '20. Virtual Event, USA: Association for Computing Machinery, 2020, P. 223–237.

Li, W., Feng, C., Zhang, L., Xu, H., Cao, B., & Imran, M. 2021. A Scalable Multi-Layer PBFT Consensus for Blockchain. In: *IEEE Transactions on Parallel and Distributed Systems* 32.5 (2021), P. 1146–1160.

Lina A., Noureddine L., & Mohamed A. 2020. Local bitcoin network simulator for performance evaluation using lightweight virtualization. In 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), P. 355–360. IEEE, 2020.

Liu, Y., Liu, J., Zhang, Z., Xu, T., and Yu, H. (2019). Overview on consensus mechanism of blockchain technology, J. Cryptol. Res. 6 (4) (2019) P. 395–432.

Michael M., Maria L. P., and Robert D. L. 2009. Performance of t12 and t8 fluorescent lamps and troffers and led linear replacement lamps caliper benchmark report. Technical report, Pacific Northwest National Lab.(PNNL), Richland, WA (United States).

Miller, A., Xia, Y., Croman, K., Shi, E., and Song, D. 2016. The Honey Badger of BFT Protocols. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, P. 31–42.

Moon, J. W. (1983). On level numbers of t-ary trees. In: *SIAM Journal on Algebraic Discrete Methods* 4.1 (1983), P. 8–13.

Neiheiser, R., Presser, D., Rech, L., Bravo, M., Rodrigues, L., and Correia, M. 2018. Fireplug: Flexible and Robust N-Version Geo-Replication of Graph Databases. In: *2018 International Conference on Information Networking (ICOIN)*. 2018, P. 110–115. doi: 10.1109/ICOIN.2018.8343095.

Neiheiser, R., Matos, M., and Rodrigues, L. 2021. Kauri: Scalable BFT Consensus with Pipelined Tree-Based Dissemination and Aggregation. In: *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. SOSP '21. Virtual Event, Germany: Association for Computing Machinery, 2021, P. 35-48.

Nicholson, W. L. (1956). On the Normal Approximation to the Hypergeometric Distribution. In: *The Annals of Mathematical Statistics* 27.2 (1956), P. 471–483. url:http://www.jstor.org/stable/2237005

Panda, S. (2019). Study of blockchain based decentralized consensus algorithms. In: TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), pp. 908–913 (2019). https://doi. org/10.1109/TENCON.2019.8929439

Pilkington, M. 2016. 11 Blockchain technology: principles and applications. In: *Research Handbook on Digital Transformations*. P. 3-5.

Rahnama, S., Gupta, S., Qadah, T. M., Hellings, J., and Sadoghi, M. 2020. Scalable, Resilient, and Configurable Permissioned Blockchain Fabric. In: *Proc. VLDB Endow.* 13.12 (Aug. 2020), P. 2893–2896.

Sousa, J. and Bessani, A. 2015. Separating the WHEAT from the Chaff: An Empirical Design for Geo-Replicated State Machines. In: *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*. 2015, P. 146–155.

Stathakopoulou, C., David, T., and Vukolic, M. 2019. Mir-BFT: High-Throughput BFT

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

for Blockchains. In:*CoRR* abs/1906.05552. arXiv: 1906.05552.

Steinbeck, D. 2019. Crypto Snippets: The Stellar Blockchain just Crashed—This is why Nodes Matter. In: *Cryptolawinsider*. url: https://cryptolawinsider.com/stellar-crash/ (visited on 08/02/2023).

Veronese, G. S., Correia, M., Bessani, A. N., and Lung, L. C. 2009. Spin One's Wheels? Byzantine Fault Tolerance with a Spinning Primary. In: *2009 28th IEEE International Symposium on Reliable Distributed Systems*. 2009, P. 135–144.

Veronese, G. S., Correia, M., Bessani, A. N., and Lung, L. C. 2010. EBAWA: Efficient Byzantine Agreement for Wide-Area Networks. In: *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*. 2010, P. 10–19.

Wan, S., Li, M., Liu, G., and Wang, C. (2019). Recent Advances in Consensus Protocols for Blockchain: A Survey. Wireless Netw. **26**(8), 5579–5593 (2019). https://doi.org/10.1007/s11276-019-021 95-0

Wood, G. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger. In: *Ethereum project yellow paper* 151, P. 1–32. url: https://files.gitter.im/ethereum/yellowpaper/VIyt/Paper.pdf (visited on 06/02/2023).

Wu, W., and Gao, Z. (2020). An Improved Blockchain Consensus Mechanism Based on Open Business Environment. In: IOP Conference Series: Earth and Environmental Science, p. 012043. IOP Publishing (2020)

Xiao, Y. (2020): A Survey of Distributed Consensus Protocols for Blockchain Networks. IEEE Commun. Surv. Tutor. **22**(2), P. 1432–1465 (2020). https://doi.org/10.1109/COMST.2020.296 9706

Yin, M., Malkhi, D., Reiter, M. K., Golan-Gueta,G., and Abraham, I. 2019. HotStuff: BFT consensus with linearity and responsiveness, in: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, on, Canada, July 29 - August 2, 2019, 2019, P. 347–356.

Zhang, C. (2020). Overview of Blockchain Consensus Mechanism. In: Proceedings of the 2020 2nd International Conference on Big Data Engineering, New York, NY, USA, pp. 7–12. Association for Computing Machinery (2020). https://doi.org/10.1145/3404512.3404522

Zhao, W. (2019). On Consensus in Public Blockchains. In: Proceedings of the 2019 International Conference on Blockchain Technology, New York, NY, USA, pp. 1–5. Association for Computing Machinery (2019). https://doi.org/10.1145/3320154.3320162.

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria