International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(5s) 2026 pp. 458-473
https://doi.org/10.70917/ijcisim-2026-2745
Article

CE-PBFT: A Cluster-Enabled Practical Byzantine Fault Tolerance Consensus Protocol for Consortium Blockchain

Kamal Kant[1], Udai Shanker[2], Sarvesh Pandey[3], Birendra Kumar Sharma[4]

[1] Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology (MMMUT), Gorakhpur, Uttar Pradesh, India.
Email: kkamal2544@gmail.com
[2] Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology (MMMUT), Gorakhpur, Uttar Pradesh, India.
Email: udaigkp@gmail.com
[3] Department of Computer Science, Mahila Mahavidyalaya (MMV), Banaras Hindu University (BHU), Varanasi, Uttar Pradesh, India.
Email: pandeysarvesh100@gmail.com
[4] Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology (MMMUT), Gorakhpur, Uttar Pradesh, India.
Email: bkscs@mmmut.ac.in
Corresponding Author:Kamal Kant, kkamal2544@gmail.com

Abstract: The fundamental component of blockchain systems, the consensus mechanism, is essential to maintaining the synchronized copy of the ledger amongst peers. The popular Practical Byzantine Fault Tolerance (PBFT) consensus technique has considerable consensus latency and low throughput as the number of nodes in the network increases. A solution to this issue is suggested in this work: the Cluster-Enabled PBFT (CE-PBFT) consensus mechanism. Through an analysis of nodes' interactions with one another, it looks at how nodes behave inside networks. The activity of the node determines whether it belongs in the consensus or non-consensus cluster. In contrast to nodes in the consensus cluster, members of the non-consensus cluster take part in the consensus-building procedure. CE-PBFT is a more energy-efficient solution than PBFT since it makes use of its leader selection approach to select the most reliable node instantly. For a higher peer count, CE-PBFT is robust and scalable because nodes within a cluster can dynamically migrate to another based on their behavior over time. In comparison to PBFT and its version, the suggested method reduces the message communication count by up to 48.4%, resulting in increased throughput, as demonstrated by both the theoretical analysis and the simulation trials.
Keywords: Blockchain; Transparency; Privacy; Leader Selection; Cryptography, Consensus

1. INTRODUCTION

First of all, Satoshi Nakamoto [1] first proposed the concept of blockchain technology (BT] in 2008. Due to the prime characteristics of BT including persistence, anonymity, auditability [2] and decentralization, people all over the world have shown their interest in it; therefore, it has got its extensive uses in numerous domains such as promoting the development of the Internet of Things, coming up with a new idea for the security of smart cities [3], controlling access for secure manufacturing system [4] [5], ensuring information security in healthcare [6], coming up with a new way to run a crowdsourcing service and likewise many more. The persistence refers to the characteristic of the data stored in the blockchain to be permanent and immutable. The blocks containing data are interlinked to form a chain; the hash chaining makes tampering infeasible and detectable. Once data is added to a blockchain, it becomes a permanent part of it and cannot be altered. Anonymity ensures that the true identity of a user or the origin of a

transaction is not easily traceable or identifiable by the other users of the network. Auditabil-ity is the ability to track and verify the complete history of transactions that have taken place on a blockchain network. Every transaction on a blockchain network is recorded in a block, which is then added to the chain of blocks. Decentralization refers to the distribution of power and control across a network of nodes (computers) that jointly participate in maintaining and verifying the integrity of the blockchain. Rather than relying on a central authority or a single entity for the validation of the transactions and maintaining the ledger, a decentralized blockchain relies on a network of participants, who have a copy of the ledger and collaborate to validate new transactions & add them to the blockchain. The incorporation of fundamental technologies like as distributed consensus mechanisms, digital signatures based on asymmetric cryptography, and cryptographic hashes makes decentralization possible. In blockchain, the execution of the transactions can occur without need of control by any single person or group. As data is present at multiple nodes, trust in the blockchain network can be established without relying on any one node. A distributed ledger system, which is based on BT today, is being utilized for maintaining records. The chain keeps expanding as more blocks are added to it. Therefore, one can also see the blockchain as an expanding collection of blocks.

The primary objective of the blockchain is to ensure data and computation transparency; however, initially blockchain was not designed to work as traditional database management systems armed with transactional management. However, a blockchain as a decentralized where data—such as payment repositories, public or private key information etc. and private or public key information—is recorded about transactions. Therefore, now one can say that blockchain is a list of transactions that keeps growing. It consists of blocks with several transactions and a hash pointing to the unique predecessor block in each. It can achieve a consensus in an open, distributed & trustless framework and is secure, hard to fake and easy to be checked [7]. Also, without consensus, it is impossible to add anything to the blockchain as it ensures that everyone agrees on the same truth making the decentralized environment more reliable than open channels. Additionally, the usage of consensus algorithms also impacts the speed, safety, and scalability etc. As a result, the process of consensus is the key to the evolution of the blockchain since it now has all the features of a distributed database, including efficiency, security, decentralization, and transparency. Thus, a well-designed consensus protocol for blockchain systems is today's most important need because through a consensus mechanism, all entities involved reach a consensus regarding the legitimacy of the entries, by creating a trusted and secure environment for open communication.

Three different kinds of blockchains exist, i.e., public blockchain (permissionless), private block-chain (permissioned), and consortium blockchain (hybrid between permissioned and permissionless).

- Public Blockchain: You can participate in it and it is open to everyone. Ethereum and Bitcoin are a couple of examples.

- Private blockchain: Businesses and organizations frequently utilize this closed network, which is only available to authorized users, for internal operations and data management.

- Consortium blockchain: It is controlled by several organizations that have come to an agreement to cooperate and share data on the blockchain. It is a cross between above blockchains. In fact, it is a confined set of pre-approved nodes enabling more efficient and secure consensus algorithms. It necessitates concentrating on strengthening privacy, resolving scale concerns, guaranteeing interoperability with other systems, and devel-oping efficient governance frameworks.

Blockchains for consortiums are frequently used in industry-specific applications that demand solutions that are specifically designed to fulfil unique needs. Most study is focused on consortium block-chains because of the special features and difficulties of this network. It is essential that this technology be researched in order to properly use consortium blockchains in corporate settings, where effective-ness, data security, and cooperation among dependable and validated partners are crucial for successful adoption. Consortium blockchain systems employ Byzantine Fault Tolerant (BFT) consensus algorithms, such as Practical BFT (PBFT) [8]. Distributed systems that contain rogue nodes are vulnerable to several types of failures, including Byzantine faults, wherein nodes behave deliberately or randomly. For this reason, PBFT was created to provide a practical, efficient, and safe method of achieving consensus in distributed systems. Byzantine faults can lead to unstable system performance, degraded security, and inaccurate judgements reached by consensus. This means that a consensus mechanism that can successfully tolerate Byzantine errors is required to preserve the security and integrity of distributed systems. For distributed systems with attack-prone nodes, byzantine fault tolerance is crucial.

This work presents CE-PBFT, a novel consensus technique for consortium blockchain, as a solution to the three issues of message transmission, throughput, and latency mentioned above. Data transmission between network nodes

is referred to as message communication. Transactions, blocks, and other network-related information can be included in messages. Each peer in a blockchain network is assigned a default weight on its registration; this weight dynamically changes with each consensus decision that eventually follows a block append operation. Moreover, the proposed algorithm optimizes the communication flow in the throughout, thereby enhancing the system's efficiency. In a nutshell, our major contributions in this work are outlined below.

1. The trustworthiness of each node is computed by considering its role in the overall process. Thus, it proposes the selection of the most reliable nodes to participate in and oversee the entire consensus process.

2. Based on the choice of a more trusted node, it also cuts down the time it takes to complete transactions and the amount of communication needed to reach a consensus.

To give an overview of the paper's structure, in Section 2, different consensus techniques are explored. In particular, PBFT and its variations are covered in Section 3. In Section 4 provides the system models and threat assumptions. Section 5 provides an explanation of the suggested consensus algorithm, and Section 6 provides a performance evaluation. The work is finally concluded in section 7, which also suggests options for future research.

2. THEORETICAL REFERENCE

The blockchain system's security, throughput, availability, and other important metrics are all directly impacted by a consensus algorithm's effectiveness [9]. Therefore, design of a promising consensus algorithm is a key activity for the blockchain system [10]. Numerous consensus algorithms were presented in the recent past for appending a block. These algorithms can be differentiate based on how they arrive at a final decision to achieve consensus. One of the such categorizations di-vides consensus algorithms into proof-based methods, which comprise the first category. In order to commit transactions, a node in this group must outcompete the others and demonstrate its superior qualification. Delegated Proof of Stake (DPoS) [11], Proof of Work (PoW) [12], Proof of Stack (PoS) [13], Proof of Burn (PoB) [14], Proof of Activity (PoA) [15], Proof of Importance (PoI) [16], Proof of Luck (PoL) [17], Proof of Elapsed Time (PoET) [18], and Proof of Authority (PoA) [19]. This includes algorithms, among other things. Voting-based algorithms make up the other type since their commitment is contingent on which committed result receives the greatest number of votes. Paxos [20] [21], Practical Byzantine Fault Tolerance (PBFT) [22] [23] [24], Redundant Byzantine Fault Tolerance (RFBT) [25], Ripple protocol consensus algorithm (RPCA) [26] [27], BFT-SMART [28] [29] [30], Raft [31], Stellar consensus protocol (SCP) [32] [33], workload-based randomization Byzantine fault tolerance protocol (WRBFT) [34], HotStuff [35] etc. belong to this category. Thus, it is possible to classify consensus algorithms into two categories: based on proof and on voting.

PoW, PoS, PoA, PoI, PoL & PoB etc. consensus algorithms fall in permissionless blockchain; however, Paxos, Raft, PBFT, RFBT, RPCA, SCP, BFT-SMART, HotStuff, BFT-RAFT, WRBFT etc. are the consensus algorithms utilized in permissioned blockchain. Typically, consortium blockchains employ consensus algorithms that are better adapted for permissioned environments. These algorithms prioritize effectiveness, scalability, and participant trust. Here are some of the consensus algorithms typically employed in consortium blockchains: PBFT, RBFT, scalable and location-based PBFT [36], PoA, DPoS, CPBFT [37], Reputation-Based Byzantine Fault-Tolerance (RB-BFT) , Scalable Practical Byzantine Fault Tolerance (S-PBFT) [38], GPBFT [1], Workload-Based Randomization Byzantine Fault Tolerance Protocol (WRBFT), Extensible-PBFT (EPBFT) [40], T-PBFT [41], Egalitarian Practical Byzantine Fault Tolerance (e-PBFT) [42], SBFT [43] etc.

The capacity of a network to remain operational and accessible to the users is referred to as its avail-ability. The PBFT algorithm can improve the availability of a blockchain network because it is more fault-tolerant than PoW consensus algorithms. By allowing multiple transactions to be processed simultaneously, this algorithm can increase the throughput of a blockchain network. And for business purposes it is better than alternative consensus algorithms and effectively addresses the computing waste issue commonly associated with traditional blockchain consensus algorithms. While it achieves a reasonable balance between availability and security, it is also not without limitations. The number of nodes in a network based on PBFT determines the maximum throughput because every node must process and validate every transaction. The PBFT performance significantly decreases as the number of nodes rises because message complexity increases. It also suffers from issues of high energy consumption & limited transaction throughput due to the communication level being elevated as the nodes aim to authenticate all the information circulating on the network. Nevertheless, the algorithm encounters further scalability challenges when numerous nodes are involved in reaching a consensus as there is no flexible method for consensus nodes to enter in or exit from the system. Furthermore, the presence of malicious or faulty nodes in the consensus process is evident also, which hinders the system's security, availability, and scalability. Although the BFT-based consensus algorithm offers some

degree of protection against malicious nodes, it remains vulnerable as achieving the correct consensus requires the absence of more than one-third of the malicious nodes. In the following ways, PBFT can affect the speed, throughput, and availability of a blockchain network: These restrictions impair the consortium blockchain's functionality and increase the difficulty for people to embrace it. In [36], For location-based, scalable consensus technique called G-PBFT is proposed. To reduce consensus overhead & ensure system security, a trusted fixed node is selected as the consensus participant. Fixed nodes, which possess more computational power than mobile ones & are less likely to become malicious, are preferred over PBFT. However, this approach significantly reduces the decentralization behavior of the system. CPBFT consensus [37] algorithm was proposed to enhance credit via credit improvement. The original C/S structure was replaced with a pear-to-pear framework; thus, the consensus process was simplified. Here, a credit factor was also introduced. A voting mechanism was used to choose the master node based on the node's previous actions, and it influenced the possibility of being selected as master node. Experiments show that in comparison to the PBFT algorithm, the CPBFT algorithm reduces data trans-fer and enhances throughput. However, upon thorough analysis, it has been determined that the current selection of the master node is inadequate and the latency issue remains unresolved. A proposal in the form of RB-BFT has been given in [2] to utilize a hashing algorithm to group the nodes that exhibit consistency resulting in reduced communication between nodes and decreased net-work communication complexity. This approach can enhance network scalability. However, it doesn't offer a way to recognize Byzantine nodes.

In [43], SBFT, which stands for Scalable Decentralized Trust Infrastructure for Blockchains, im-proved the performance of the PBFT algorithm. But SBFT has limited promotion dimensions and unique access points. Unfortunately, the PBFT consensus method and its variations are vulnerable to various attacks on the master node and are not very effective at locating and removing the bad nodes from the blockchain. To tackle these challenges, a new algorithm called RB-BFT has been proposed. The overall system works similar to the PBFT process and relies on a three-stage broadcast protocol, but with novel mechanisms and applications. The RB-BFT algorithm replaces PBFT's view mechanism with a fixed period for achieving quick consensus generation. It also assigns a credit value to each node, which reduces the risk of system failure due to view switching or malicious behavior. By evaluating nodes based on their credit value, the discourse power of higher-credit nodes can be elevated, the participation of lower-credit nodes can be minimized, and the main node's failure risk can be reduced [45]. Analysis reveals that reputation systems are susceptible to Sybil attacks, in which a hacker assumes many identities to increase their power within the network. It would not be scalable to support large networks because the reputation system requires a significant amount of computation and communication between nodes. It has limited fault tolerance power because it relies on reputation scores to determine the weight of each node in the network. It is also not able to resist collusion between nodes with similar reputations. Collusion can lead to a compromised network and a breakdown in consensus. The S-PBFT algorithm [40] is also a variant of the traditional PBFT algorithm that aims to improve issues of its scalability and efficiency by introducing a node scoring mechanism that enables nodes to be classified based on their reliability & behavior. By doing so, in a distributed system, S-PBFT can lower communication costs and enhance consensus process performance overall [46]. However, the charging of penalties is uncategorized resulting in a serious impact on node scoring performance.

A protocol called Extensible-PBFT (EPBFT) was introduced in [38] with the aim of streamlining the consensus process by assigning a master node to lead the process and allowing nodes to participate in it only on a chosen basis. A method for choosing consensus nodes using verifiable random functions (VRF) has been added to improve it. With this modification, the algorithm operates better on networks with fluctuating conditions. While this approach reduces communication complexity, the random selection of nodes and the dominant power of the master node result in excessive centralization, which weakens the system's security. Additionally, the protocol's method of analyzing only the master node's down-time fails to protect against a Byzantine master node. In summary, one can say that the EPBFT prioritizes the scalability and ease of communication (security) of the entire system at the expense of decentralization. The T-PBFT algorithm [41] proposes a method to analyze the node trust in a network by examining transaction history between nodes. This allows the formation of a consensus group with the help of high-quality nodes. Reduction in the frequency of view changes via this group by replacing a single major node. The core group's resilience can further be strengthened by utilizing group signatures and mutual supervision. Essentially, by introducing a reputation evaluation procedure, this strategy reduces the likelihood of malevolent behavior and view switching by identifying nodes that are likely to become master nodes. The selection of the master node is based just on transaction behavior, ignoring any hid-den traits the node may have, which bears hazards even though this evaluation procedure makes the system more complex. Egalitarian Practical Byzantine Fault Tolerance (e-PBFT) [42] is an alternative solution that disperses power equally throughout the system's nodes and does away with the central node mechanism in the consensus process. However, there are no sufficient measures in place to prevent malicious behavior by the nodes in the system.

To address the issue of low efficiency caused by abnormal network conditions while simultaneously increasing the complexity of consensus communication is Honey Badger BFT [47]. However, it also suffers from lack of backward compatibility meaning the algorithm is not back-ward compatible with previous versions of the protocol. It requires a high level of expertise in cryptography and distributed systems to understand and implement correctly.

DPFT was proposed with the introduction of a double-response mechanism. Additionally, a self-conflict checking mechanism was implemented to address issues with view changes and enable smooth performance drops under typical circumstances [48]. Another location-based and scalable consensus system that has been suggested is the G-PBFT protocol. It is not entirely focused on the issue of malicious nodes because its primary purpose is for Internet of Things blockchain applications. By using an era switch mechanism and location-based endorser election, the protocol reduces network overhead and increases the efficiency and scalability of the consensus process [39]. Task-Dependent Randomization The new Byzantine fault tolerance protocol known as Workload-based randomization Byzantine Fault Tolerance Protocol (WRBFT) was introduced in [34]. By taking their workload into account, this protocol uses a novel method to choose consensus-related nodes. The protocol determines each node's workload levels by evaluating its involvement following each consensus round. These variables are used by the protocol to decide which nodes will take part in consensus. Additionally, by merging node workload values with an authentic random function for primary node se-lection, the protocol seeks to reduce the probability of dishonest nodes being chosen as the primary node. This protocol also enhances the PBFT agreement protocol by selecting more trustworthy and reliable nodes. As a result, this approach significantly reduces the time and communication required to achieve consensus [34]. However, it needs improvement in the communication overhead. The communication overhead is still high due to addition of an extra phase: This expands the two-phase PBFT confirmation procedure into a three-phase one, resulting in a four-phase consensus process (Request, Pre-prepare, Prepare, Pre-commit & Commit, Reply). Depending on which primary node is deemed more trustworthy, WRBFT switches out multi-to-multiple connectivity with one-to-multiple communication. There is also an increase in response time. Upgrading the algorithm's dynamics and adaptability can enhance its ability to calculate the node workload value. This statement pertains to the constraints im-posed on the system's capacity to handle a lot of transactions swiftly and effectively.

3. PBFT AND ITS WORKING PROCESS

The three fundamental parts of PBFT are view, primary, and replica. The voting procedure starts with the main node. The system remains the same and retains the previous data if a consensus cannot be reached within the allotted time. In the case of a shift in viewpoint, a new primary node is selected to promote consensus among the nodes. The replica node triggers the view-change protocol when it detects that the current main node is unable to attain consensus within the allowed time. The replica node is essential to guaranteeing the voting process' effectiveness. Moreover, the view rotation mechanism is initiated to substitute the current main node in the case of a failure. The all three-step PBFT working process are shown in Figure 1, where node 0 is the principal node or leader node or master node is initiate process across all nodes. Node 3 is also the Byzantine node, whereas nodes 1, 2, and 3 are replica nodes [43] [49].

1. During the "pre-prepare" phase, the client makes a request to the primary node. The primary node reacts to the request by doing its verification and sending a "pre-prepare" message to the replica nodes. The "pre-prepare" message is authenticated and verified by any replica node that gets it from the primary. After a successful verification, the replica node accepts the request and moves on to the "prepare" phase.

2. The replica nodes disseminate prepared messages among themselves during the "prepare" phase. They also get and authenticate messages generated by other nodes. Upon receiving 2f valid prepared messages from distinct replica nodes, the prepare phase concludes. In this context, f represents the quantity of Byzantine nodes inside the network.

3. In “commit” phase, each node transmits a message to other nodes in order to have it validated. When in "prepare" mode, a node does not reply to the client until it receives 2f+1 commit messages, including its own. The client reaches consensus when it receives f+1 identical replies.

Once more, imagine a system where f of the n consensus nodes is faulty. The message complexity may approach O(n2) depending on the distributed procedure. The number of communication messages in PBFT that clients and nodes must exchange at various phases in order to reach consensus. In specifically, the client sends n messages first. In "pre-prepare", the principal node sends n-1 messages. Duplicates transmit (n-1) * (n-1) messages in "prepare". In commit, primary and clones send n* (n-1). Client receives n messages back. Unanimity takes 2n(n-1) messages.

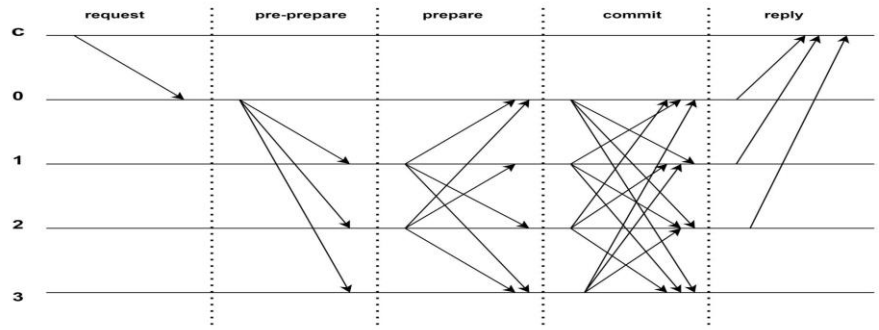

Figure 1 Working Process of PBFT Protocol

4. SYSTEM MODEL & THREAT ASSUMPTIONS

The proposed consortium blockchain operates over a partially synchronous P2P network consisting of N participating nodes. Each node is assigned an initial weight (W) and an execution weight (Wi), where Wi represents its trustworthiness and participation in consensus. A management node calculates the threshold value (THi = Wi − W) to determine consensus eligibility. Nodes exceeding the threshold participate in consensus, while others are excluded. This approach improves communication efficiency, accelerates decision-making, and prevents malicious or unreliable nodes from affecting the consensus process.

System Model:

Nodes: The blockchain network consists of validator and replicator nodes. Communication: Messages exchanged between nodes may experience delays or losses during transmission. Synchronous Network: CEPBFT assumes a synchronous network where messages are delivered within a predefined time limit. Byzantine Faults: The protocol tolerates a limited number of Byzantine nodes that may behave maliciously or attempt to disrupt the consensus process.

Threat Assumptions:

CEPBFT assumes that message integrity is ensured through digital signatures, while Byzantine nodes cannot collude and are limited by finite computational resources. The network supports dynamic membership, allowing nodes to join or leave over time, provided that honest nodes remain in the majority. The consensus process comprises of three phases: Pre-Prepare, Prepare, and Commit. During these phases, nodes exchange signed messages to agree on the order of transactions. Similar to PBFT, the protocol requires 2f + 1 honest nodes to reach consensus with up to f Byzantine faults. The symbols used in the protocol are listed in Table 1.

Table 1 Symbols & Their Description

Symbols	Description
N	The number of system nodes
M	Nodes that took part in the consensus process in terms of number
Ni	The system's ith node
W	A node's base (intial) weight
Wi	Weight of the ith node when operating
THi	The ith node's threshold
bhi	Behavior type of the ith node
bn	Types of behavior
Pn	Penalty

n	Predefined threshold value

5. CE-PBFT CONSENSUS ALGORITHM

Although existing consortium blockchain consensus algorithms have improved over time, they still face challenges in scalability, message complexity, and real-time performance. To address these issues, CE-PBFT classifies nodes based on their behaviour and dynamically assigns them to consensus or non-consensus groups. Only trusted nodes participate in consensus, while others are excluded.

This selective participation reduces communication overhead and improves throughput, scalability, and latency. By combining behaviour-based node selection with Byzantine fault tolerance, even when malicious nodes are present, secure and effective consensus is made possible via CE-PBFT.

5.1 DESCRIPTION OF DIFFERENT SEGMENT OF PROPOSED PROTOCOL

In our proposed protocol, there are three algorithms that are interconnected with each other.

Algorithm 1-It provides an operational summary for determining the behavior of a node. In this sat refer to satisfactory transaction, unsat refers to unsatisfactory transaction, Tr refers to time frame in which transaction can be executed, max refers to the maximum value would be taken, N is a measure of the network's nodes' size. and p1 and p2 are the two categories of penalty in which the nodes lie. In this one node is a transaction with another node in a particular time frame and then we find out the behavior. The node's category is determined based on its activity with other nodes in step 3. The algorithm also determines which penalty category the node belongs to, according to steps 10, 11, and 13. The output of step 10- p1, and step 12-p2, provide sub-inputs to Algorithm 2.

Steps	**Algorithm 1**
1.	for each node i in Nodes
2.	if i has a transaction with node j
3.	$S_{ij} = ((sat_{ij} - unsat_{ij}) / T_r) * 10;$
4.	$S_{ij} = \max(S_{ij}, 0);$
5.	end if
6.	if $S_{ij} = 0$ then,
7.	Set $S_{ij} = 1/N;$ // N is the nodes' size.
8.	end if
9.	if $S_{ij} > 20$, then
10.	execute p1
11.	else
12.	execute p2
13.	end if
14.	end for

Algorithm 2- First, A node's execution weight is determined using the data obtained from algorithm 1. From algorithm 1 we find the penalty category. In this, W is for the initial weight of the node, Wi is for the execution weight of the node, Ni is for the number of nodes, The behavior of the network's participating nodes is referred to as bhi. and RT refers to the response time of a particular node that is in the net-work. In step 1 node execution weight is negative, then it keeps the zero weight. If the node value meets the upper limit or greater than the upper limit, then we cannot change the execution weight of the node in step 4. The category of the node is identified in step 7, and a penalty is applied to its execution weight in step 9 and step 11. The execution weight is then checked to see if it falls within the predetermined range in step 13. If it does, the node is rewarded in step 14. Should the node's reaction time be less than anticipated, an additional reward is given to the node in step 17 of its execution. Finally, the execution weight of the node is calculated. This execution weight becomes the output of algorithm 2 and the input for the next algorithm, algorithm 3.

Step	**Algorithm 2: Trust calculation of Node**
1.	if W_i is negative
2.	set $W_i=0$
3.	else if W_i is equal to the highest possible value (Ceiling value)
4.	W_i remains no change
5.	end if
6.	for each node N_i

```
7.      if N_i has participated in the consensus and bhi is in (P1,P2)
8.          if bhi=P1
9.              W_i=W_i-10              // apply panalty
10.             else
11.             W_i= W_i-15             // apply panalty
12.         end if
13.         if (W_i >0 && W_i <100) then
14.             W_i=W_i+20              // apply Reward
15.         end if
16.     end if
17.     if response time (RT) < 30ms
18.         W_i=W_i+5                   // apply reward for better response
19.     end if
20. end for
```

Algorithm 3, With this, we can decide whether a node should take part in the consensus group or not. In this, THi refers to the threshold value, Wi refers to the execution weight of the node, W stands for the node's initial weight., and Ti refers to the predefined value that is taken to categorize the consensus group. In step 1, first determine the node's threshold value., which is equal to the current execution weight minus the initial weight of the node. Then this value is compared to the predefined value in step 4. It enters the consensus group if its threshold value exceeds the predetermined value in step 5; otherwise, it is excluded from the consensus group in step 7. When a node's threshold value is higher than the predefined value, it goes into the consensus group. It is excluded from the consensus group. So, the nodes that have more trust are only participating in the consensus.

Steps	Algorithm3: Consensus Group
i/p	trust set T, node set Nodes,
o/p	Consensus group
1.	$TH_i= (W_i-W)$ **// calculate threshold value**
2.	Nodes are sorted by T
3.	Nodes do for € $node_i$
4.	If $TH_i >= T_i$
5.	Into the consensus group, include $node_i$
6.	Else
7.	Don't include $node_i$ in the consensus group
8.	End
9.	End

The threshold value of a node can be computed through the given equation, which is also in the algorithm, which is execution weight minus initial weight. This is how the threshold value THi of node Ni is calculated: THi= (Wi-W) ----------(1) Wi = execution weight; W = initial weight, A node with a higher THi is chosen to participate in the message consensus, while a node with a lower THi will not.

5.2 PENALTY SECTION THROUGH BEHAVIOR

The goal of the behavioral study of the consensus process nodes is to ascertain each node's degree of negative weight adjustment. The following defines two probable unacceptable behaviors, NB = {P1, P2}. P1: The node takes some time to process the information it gets. P2: The information obtained by the node was never shared. The node modifies the data and transmits the updated data. if P1 then minus 10), P2 (if P2 then minus 15). R(t)= (sat(i,j)-unsat(i,j))/Tr This decides the bhi (decides the behavior). Where Tr represents the total transactions at the moment of 't', Sat(i, j) and unsat(i, j) represent satisfactory and unsatisfactory transactions between nodes I and J, respectively.

5.3 REWARD SECTION

Based on their activities, the nodes taking part in the consensus process receive rewards. In two scenarios, a node Ni is rewarded by having its weight value increased. First, when its running weight is between 0 and 100 after penalties. Second, when its response time is less than 30 ms.

Algorithm for trust calculation: The functioning model of the suggested protocol is demonstrated using Figure 2 and the "n" number of nodes. Normal nodes and existing nodes are the two categories into which the network nodes can be divided. Normal nodes refer to those that enter the network for the first time, while existing nodes are those

that have already been part of the network. Nodes are additionally divided into two clusters in the CE-PBFT protocol according to their behavior and the rewards and penalties they receive. Nodes that act well during the consensus process and reach the predetermined threshold value are added to the consensus cluster; nodes that behave poorly and fail to reach the predetermined value are added to the non-consensus cluster. This classification is important because it safe-guards the integrity of the network and prevents rogue nodes from taking over the system. Depending on its threshold value, each node has an opportunity to be added to the consensus or non-consensus cluster following each consensus process. When the consensus algorithm is executed and the consensus procedure is complete, the block is appended to the blockchain. This ensures that the network agrees on the blockchain's present status and that every node has a consistent understanding of the system. Assuming a network with 'N' nodes, the PBFT consensus approach involves the participation of all the 'N' nodes. Each node verifies the consensus and broadcasts its vote to the remaining 'N-1' nodes resulting in a total message exchange of 2N*(N-1). On the other hand, CE-PBFT requires fewer message exchanges because only a subset of 'M' nodes is selected to participate in the consensus process. Therefore, T-PBFT requires the exchange of 2M*(N-1) messages.

Figure 3 illustrates how the CE-PBFT consensus mechanism helps a group of nodes agree on a state despite malfunctioning or malicious nodes. There are 8 nodes in this, with node 0 starting the process and nodes 1 to 7 being replica nodes, with node 7 being malevolent. The multi-phase algorithm works as follows: Request: When the principal node receives a client request, the CE-PBFT algorithm separates network users into consensus and non-consensus groups. Exclusive involvement in the consensus process updates consensus group members' trust value. Non-consensus group members have lower trust values since they are not involved in consensus. After each round, each group member's trust value is determined and assigned to the consensus or non-consensus group. Preparation: The leader node prepares a block for each request, assigns a sequence number, and broadcasts it to all nodes. Prepare. A node verifies a pre-prepare message against its blockchain. If the request is genuine, the node tells others to prepare; Commit: If a node receives 2f prepare messages, it commits and f is the maximum number of defective nodes; After 2f+1 commit messages, the node processes the request and replies to the client.

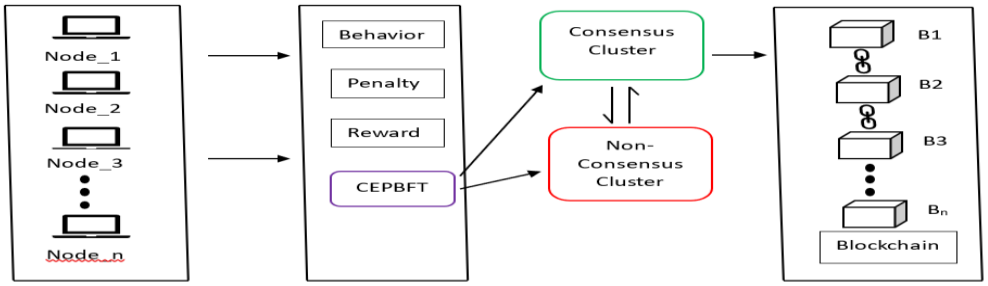

Figure 2 Working Model of CE-PBFT Consensus Protocol

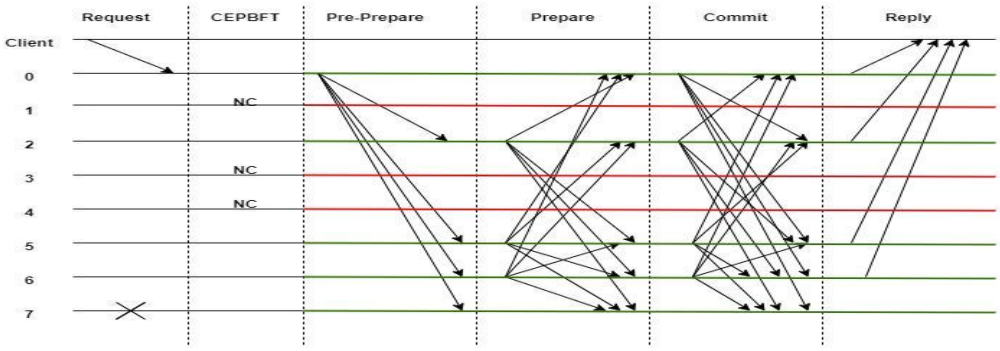

Figure 3 Process of CE-PBFT

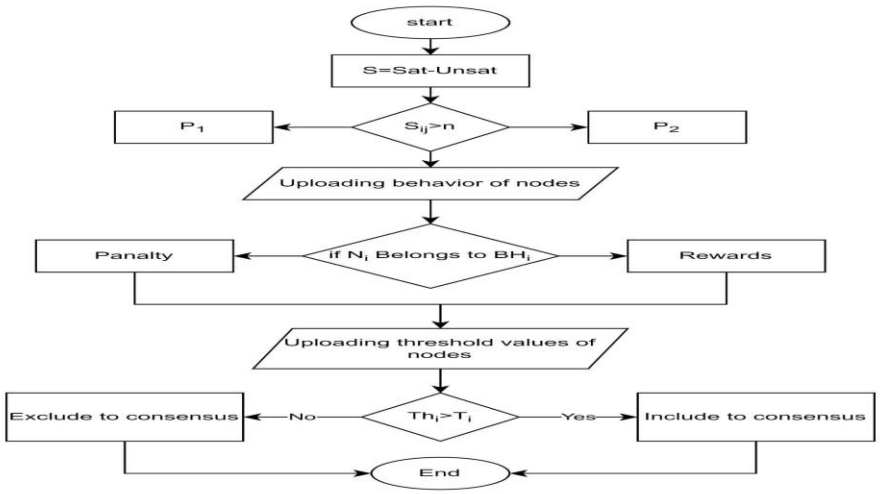

Figure 4 Flow Chart of CE-PBFT Algorithm

5.4 OUR PROPOSED METHOD WORKING

Our proposed algorithm is primarily concerned with locating the trustworthy or representative node, which is the only node that participates in the consensus. So, the proposed algorithms are divided into three steps:

1. Determine a node's past behavior by calculating satisfactory and unsatisfactory transactions to other nodes at a specified timestamp. Nodes can be divided into parts either P1 and P2 in P1 it has less penalty then the P2. Based on its behavior, a penalty is applied during the trust value computation.

2. Use an algorithm 2 that complies with each consensus algorithm calculate each node's trust. By adding a reward based on reaction time and a penalty based on behavior to each node's threshold value, the trust value is calculated. Make use of a mechanism to ascertain whether the node is contributing to the consensus.

3. A node can join the consensus process if its trust value exceeds a threshold. Any node with a trust value below a certain threshold is removed from consensus.

Ensuring that only trusted nodes take part in the consensus process is the basic objective of our goal. This might make the system more dependable and secure. Algorithm 3 operates as follows to determine the consensus group: First, we utilize Algorithm 2 to calculate the execution weight for all nodes in the network. This output is used as input for Algorithm 3. Next, in step 1, we calculate the threshold value of a node as the execution weight minus the initial weight. This threshold value is the overall trust value of the node. In step 3, we compare the threshold value of each node, which was obtained from algorithm 2, with a predefined limit. If the node's thresh-old value exceeds this limit, it is included in the consensus group (stop 4). Conversely, if the threshold value is less than the limit, the node is excluded from the consensus process (step 6). Each stage of the PBFT algorithm has its own set of rules and messages that are exchanged between nodes. If less than one-third of the nodes are faulty or malicious, the algorithms ensures safety (all legitimate nodes agree on the same state) and liveness (requests are executed).

The CE-PBFT consensus algorithm's flow diagram is illustrated in Figure 4. Initially, nodes' behavior is evaluated over a specific time frame, it counts how many transactions between two nodes are both satisfactory and unsatisfactory.. Based on their behavior, nodes are then divided into two categories, P1 (in which there is less penalty) and P2 (in which there is more penalty), indicating a penalty. The execution weight of the node is then determined. If the node falls into the P1 and P2 groups, its execution weight is penalized, but if it is not part of these groups, it receives a reward for its execution weight. If its response time is satisfactory, the node also receives a bonus. The node's total execution weight is then determined. The node's threshold is determined by subtracting the execution weight from the node's initial weight. If the threshold value is higher than the predetermined threshold, the node is added to the consensus group. If the threshold value is below the threshold, the node is excluded from the consensus group and not be able to take part in the process of consensus.

6. PERFORMANCE EVALUTION

Assume the network has N (N4) nodes. C (NC, C4) are consensus nodes. As this study suggests, the CE-PBFT method employs election to select consensus nodes. The experiments in this article use versus code and Node.js on a Windows 10 PC with a 3 GHz Intel i7-9700 processor and 8 GB of RAM. To test CE-PBFT, communication volume, throughput, and transaction latency are compared to PBFT. Table 2 covers all experimental parameters.

Table 2 Experimental Parameters Table

S. No.	Parameter/ Software/ Hardware	Specification/Setting
1	Operating System	Windows 10
2	Memory	8 GB of RAM
3	CPU	i7-9700 processor
4	Node workload value threshold WVlow	00
5	Node workload value threshold WVhigh	100
6	Threshold value of a node	15
7	Initial value of node when enter in the network	30

6.1 COMPARISON OF MESSAGE OVERHEAD

To evaluate CE-PBFT, Figure 5 message complexity is used as a performance metric. Unlike PBFT, where all N nodes participate in consensus and exchange 2N(N−1) messages, CE-PBFT selects only M trusted nodes, reducing communication to 2M(N−1) messages. As the network size grows, CE-PBFT significantly lowers communication overhead and achieves about 45–50% less message complexity than PBFT, resulting in better scalability and efficiency.

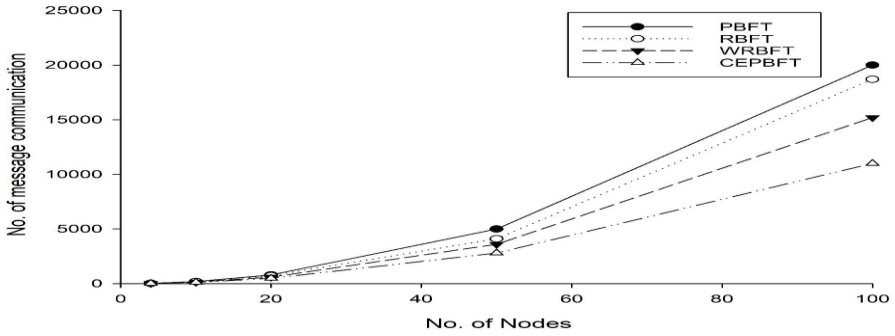

Figure 5 Message Exchange Analysis for Various Algorithms and CEPBFT

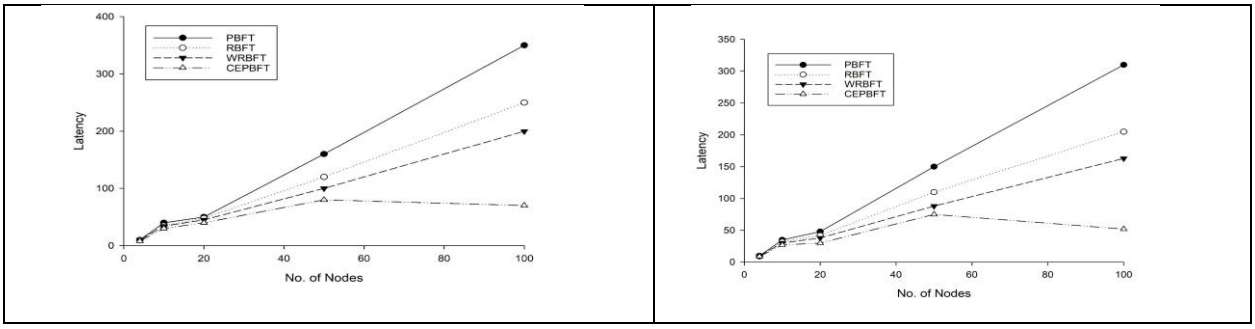

Figure 6 Latency Analysis for Various Algorithms and CEPBFT without byzantine nodes

Figure 7 Latency Analysis for Various Algorithms and CEPBFT 20% byzantine nodes

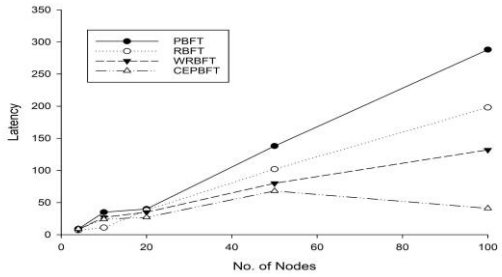

Figure 8 Latency Analysis for Various Algorithms and CEPBFT 30% byzantine nodes

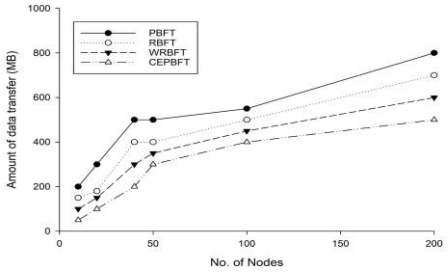

Figure 9 Amount of data transferred versus the number of nodes

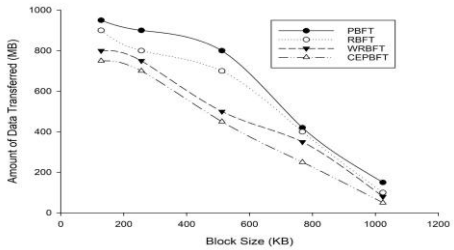

Figure 10 Amount of data transferred versus block size

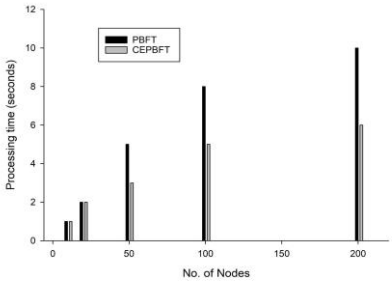

Figure 11 Comparison of processing time and node count

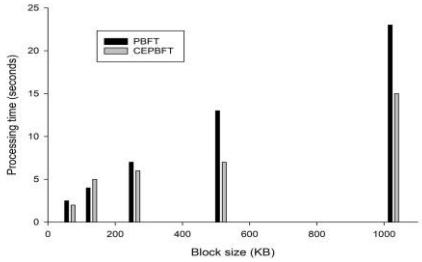

Figure 12 Processing time versus block size

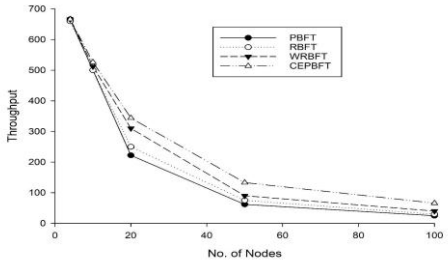

Figure 13 Throughput Analysis for Various Algorithms & CEPBFT without Byzantine Nodes

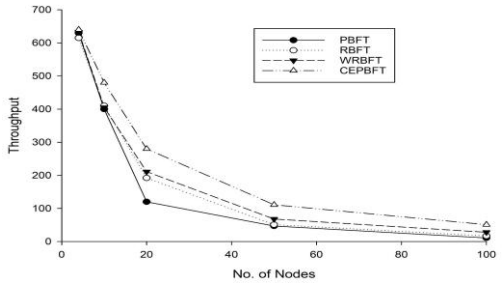

Figure 14 Throughput Analysis for Various Algorithms & CEPBFT 20% Byzantine Nodes

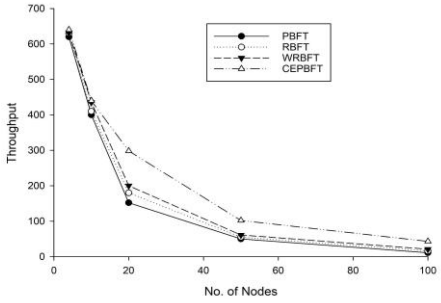

Figure 15 Throughput Analysis for Various Algorithms & CEPBFT 30% Byzantine Nodes

6.2 COMPARISIB OF LATENCY

Latency is an important metric for evaluating consensus performance, as lower latency enables faster agreement among nodes and quicker block confirmation. It is calculated as the difference between the block confirmation time and the proposal time (Delay = Tconfirm − Tpropose). The results shown in **Figure 6** compare the latency of PBFT and CE-PBFT, while **Figures 7** and **8** present the performance under 20% and 30% Byzantine nodes, respectively. Although both algorithms exhibit similar latency for small networks, the latency increases as the network grows. However, CE-PBFT shows a much slower increase than PBFT, resulting in lower latency and better overall performance.

6.3 DATA TRANSFERRED

When comparing the amount of data transferred to CEPBFT, as seen in Figure 9 with its variable number of nodes, the total amount transferred in the proposed system is more. This total is the culmination of all the stages from the request issued to the client to the request returned to the client. Figure 10 shows that as block size grows, so does the number of transactions that it can support. The graphic illustrates how network data transport volume declines with increasing block size. CEPBFT reduces computational and network traffic overhead by performing better than PBFT, RBFT, and WRBFT.

6.4 PROCESSING TIME

The whole amount of time needed for all phases of the CEPBFT, from sending the request to the client to having it returned to them, is the total processing time. With different network and block sizes, Figures 11 and 12 show that CEPBFT performs better in terms of processing time than the PBFT net-work. with the PBFT, nearly every node does data replication and ledger updates; with the CEPBFT, however, this occurs at the consensus group rather than both groups.

6.5 COMPARISION OF THROUGHPUT

Throughput, measured in transactions per second (TPS), is an important indicator of consensus performance. The results shown in Figures 13, 14, and 15 compare the throughput of PBFT and CE-PBFT under different Byzantine node scenarios. While both algorithms achieve similar throughput for small networks, throughput decreases as the number of nodes increases. However, CE-PBFT experiences a much slower decline than PBFT, demonstrating better scalability and higher transaction processing efficiency in larger blockchain networks.

Based on the information provided, it appears that CE-BFT is a consensus algorithm that uses node trust values to ensure the trustworthiness of consensus nodes. Compared to other BFT-based consensus algorithms, The network's consensus node count is decreased via CE-PBFT., which in turn speeds up the time it takes to reach consensus. The graph shows that CE-PBFT has a low communication overhead, a shorter time compared to other BFT-based consensus algorithms. Moreover, CE-PBFT improves PBFT consistency, facilitating communication amongst consistency protocol participants.

Overall, CE-PBFT is an effective consensus algorithm for achieving distributed consensus in a blockchain network. Tabulating results simulations show CEPBFT generates less messages than PBFT. The proposed method

takes 48.4% less messages than PBFT to establish consensus. CEPBFT outperforms PBFT in message exchange substantially. Table 3 contrasts earlier PBFT-enabled devices with CEPBFT.

Table 3 Comparative Summary of Key Similar Works

Algorithms	PBFT	HoneyBedger	WBFT	CE-PBFT (this study)
Master node	Single	Single	Single	single
Probability of View change	Higher	Higher	Higher	Lower
Speed	Higher	Lower	Lower	Higher
Network overhead	Higher	Higher	Higher	Lower
Computing overhead	Higher	Lower	Higher	Lower
Communication complexity	$O(N^2)$	$O(N^2 + N^2 \log N)$	$O(N^2 + N^2 \log N)$	$O(mN)$

7. CONCLUSIONS

The paper discussed the PBFT methodology and its limitations as well as a novel consensus method termed CE-PBFT that overcomes the limitations. CE-PBFT is a hybrid consensus protocol that com-bines BT with representative nodes or trust-based nodes to provide a stable, fair, and trustworthy foundation for system evolution. The study also explained how trust scores are assigned to nodes based on their history and current behavior, and how nodes are rewarded or penalized based on their behavior. It is the goal of the system of incentives and penalties to encourage good behavior and discourage bad behavior. Via trials, the article also contrasted the CE-PBFT algorithm's communication complexity, throughput, and time delay with those of the conventional PBFT, RBFT, WRBFT, and CEPBFT algorithms. The experiment findings show a considerable improvement in communication complexity, throughput, and time delay compared to its competitor alternatives. The paper's conclusion highlights the advantages of CE-PBFT in terms of throughput and node selection, as well as how it may be used to build trustworthy and secure systems in public and open spaces.

8. AUTHOR'S CONTRIBUTION

Kamal Kant: Conceptualization, research design, and manuscript writing, Literature review, data analysis, and manuscript refinement. **Udai Shanker:** Supervision, review, and manuscript structuring.

Sarvesh Pandey: Supervision, review, and manuscript structuring.

Birendra Kumar Sharma: Supervision, review, and manuscript structuring. Example

9. ACKNOWLEDGMENTS

The authors express their sincere gratitude to the Department of Computer Science and Engineering for providing the necessary resources and support during the research. Special thanks to Prof. Udai Shanker, Dr. Sarvesh Pandey and Dr. B. K. Sharma for their invaluable guidance in structuring and refining the manuscript.

FUNDING

This research was conducted without any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

COMPLIANCE WITH ETHICAL STANDARDS

The research has been conducted in accordance with ethical guidelines.

CONFLICT OF INTEREST

The authors declare no conflicts of interest related to this research. If necessary, quote entities that supported the development of the research.

References

1. A. A. Menon, T. Saranya, S. Sureshbabu, and A. S. Mahesh, "A comparative analysis on three consensus algorithms: Proof of Burn, Proof of Elapsed Time, Proof of Authority," in *Computer Networks and Inventive Communication Technologies*. Singapore: Springer, 2022, pp. 369–383.
2. J. Yang, Z. Jia, R. Su, X. Wu, and J. Qin, "Improved fault-tolerant consensus based on the PBFT algorithm," *IEEE Access*, vol. 10, pp. 30274–30283, 2022.
3. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
4. K. Yue, Y. Zhang, Y. Chen, Y. Li, L. Zhao, C. Rong, and L. Chen, "A survey of decentralizing applications via blockchain: The 5G and beyond perspective," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2191–2217, 2021.
5. A. Kumar, K. Abhishek, B. Bhushan, and C. Chakraborty, "Secure access control for manufacturing sector with application of Ethereum blockchain," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 3058–3074, 2021.
6. S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," *arXiv preprint arXiv:1711.03936*, 2017.
7. M. Drijvers, S. Gorbunov, G. Neven, and H. Wee, "Pixel: Multi-signatures for consensus," in *Proc. 29th USENIX Security Symposium (USENIX Security '20)*, 2020, pp. 2093–2110.
8. J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 429–445, 2019.
9. L. Bahri and S. Girdzijauskas, "Trust mends blockchains: Living up to expectations," in *Proc. IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1358–1368.
10. M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proc. 3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, LA, USA, Feb. 1999, pp. 173–186.
11. W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.
12. M. Hu, T. Shen, J. Men, Z. Yu, and Y. Liu, "CRSM: An effective blockchain consensus resource slicing model for real-time distributed energy trading," *IEEE Access*, vol. 8, pp. 206876–206887, 2020.
13. M. Z. F. Yang, W. Zhou, Q. Wu, R. Long, and N. N. Xiong, "Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism," *IEEE Access*, vol. 7, pp. 118541–118555, 2019.
14. K. Kant, S. Pandey, and U. Shanker, "Blockchain transaction processing: Challenges and resolutions," in *Emerging Trends in Computer Science and Its Application*. Boca Raton, FL, USA: CRC Press, 2025, pp. 427–431.
15. S. King and S. Nadal, "PPCoin: Peer-to-Peer Cryptocurrency with Proof-of-Stake," Aug. 2012.
16. S. J. Alsunaidi and F. A. Alhaidari, "A survey of consensus algorithms for blockchain technology," in *Proc. International Conference on Computer and Information Sciences (ICCIS)*, 2019, pp. 1–6.
17. M. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.
18. M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of Luck: An Efficient Blockchain Consensus Protocol," in *Proc. 1st Workshop on System Software for Trusted Execution (SysTEX)*, 2016, pp. 1–6.
19. L. Lamport, "Paxos Made Simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 18–25, Dec. 2001.
20. J.-P. Li and W.-C. Cheng, "Research on consistency of distributed system based on Paxos algorithm," in *Proc. International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP)*, 2012, pp. 257–259.
21. R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative Byzantine Fault Tolerance," *ACM Transactions on Computer Systems*, vol. 27, no. 4, pp. 1–39, 2009.
22. A. Konnov, A. Makarov, M. Pozdnyakova, R. Safin, and A. Salagaev, "Practical Byzantine Fault Tolerance," *Information*, vol. 12, no. 5, pp. 1–24, 2021.
23. M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, Nov. 2002.
24. P. L. Aublin, S. B. Mokhtar, and V. Quéma, "RBFT: Redundant Byzantine Fault Tolerance," in *Proc. IEEE 33rd International Conference on Distributed Computing Systems (ICDCS)*, 2013, pp. 297–306.
25. B. Chase and E. MacBrough, "Analysis of the XRP Ledger Consensus Protocol," arXiv:1802.07242, 2018.
26. A. Bessani, J. Sousa, and E. E. P. Alchieri, "State machine replication for the masses with BFT-SMaRt," in *Proc. 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2014, pp. 355–362.
27. E. E. Alchieri, F. Dotti, O. M. Mendizabal, and F. Pedone, "Reconfiguring Parallel State Machine Replication," in *Proc. IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2017, pp. 104–113.
28. J. Sousa and A. N. Bessani, "From Byzantine Consensus to BFT State Machine Replication: A Latency-Optimal Transformation," in *Proc. 9th European Dependable Computing Conference (EDCC)*, 2012, pp. 37–48.
29. K. Kant, S. Pandey, and U. Shanker, "A Journey from Commit Processing in Distributed Databases to Consensus in Blockchain," in *Proc. IEEE 38th International Conference on Data Engineering (ICDE)*, Kuala Lumpur, Malaysia, 2022, pp. 3236–3240, doi: 10.1109/ICDE53745.2022.00306.
30. D. Mazières, "The Stellar Consensus Protocol: A Federated Model for Internet-Level Consensus," *Stellar Development Foundation White Paper*, 2015.
31. D. Mazières, G. Losa, and E. Gafni, *Simplified SCP*. Stellar Development Foundation, Mar. 2019.

32. B. Huang, L. Peng, W. Zhao, and N. Chen, “Workload-Based Randomization Byzantine Fault Tolerance Consensus Protocol,” *High-Confidence Computing*, vol. 2, no. 3, Art. no. 100087, 2022.
33. M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, “HotStuff: BFT Consensus in the Lens of Blockchain,” in *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, 2019, pp. 347–356.
34. L. Lao, X. Dai, B. Xiao, and S. Guo, “G-PBFT: A Location-Based and Scalable Consensus Protocol for IoT-Blockchain Applications,” in *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 664–673.
35. Y. Wang, Z. Song, and T. Cheng, “Improvement Research of PBFT Consensus Algorithm Based on Credit,” in *Blockchain and Trustworthy Systems (BlockSys 2019)*, Guangzhou, China, 2019, pp. 47–59.
36. Y. Li, Z. Wang, J. Fan, Y. Zheng, Y. Luo, C. Deng, and J. Ding, “An Extensible Consensus Algorithm Based on PBFT,” in *Proc. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2019, pp. 17–23.
37. K. Kant, U. Shanker, S. Pandey, and B. K. Sharma, “Design and performance evaluation of Sharded-PBFT for scalable Blockchain architectures”, *J Integr Sci Technol*, vol. 14, no. 6, p. 1589, May 2026, doi: 10.62110/sciencein.jist.2026.v14.1589.
38. S. Gao, T. Yu, J. Zhu, and W. Cai, “T-PBFT: An EigenTrust-Based Practical Byzantine Fault Tolerance Consensus Algorithm,” *China Communications*, vol. 16, no. 12, pp. 111–123, 2019.
39. L. He and Z. Hou, “An Improvement of Consensus Fault Tolerant Algorithm Applied to Alliance Chain,” in *Proc. IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2019, pp. 1–4.
40. G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, and A. Tomescu, “SBFT: A Scalable and Decentralized Trust Infrastructure,” *arXiv preprint arXiv:1804.01626*, 2018.
41. K. Lei, Q. Zhang, L. Xu, and Z. Qi, “Reputation-Based Byzantine Fault Tolerance for Consortium Blockchain,” in *Proc. IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018, pp. 604–611.
42. R. Almakki, L. AlSuwaidan, S. Khan, A. R. Baig, S. Baseer, and M. Singh, “Fault Tolerance Byzantine Algorithm for Lower Overhead Blockchain,” *Security and Communication Networks*, vol. 2022, Article ID 8735679, 2022.
43. A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, “The Honey Badger of BFT Protocols,” in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016, pp. 31–42.
44. J. Zhang, Y. Rong, J. Cao, C. Rong, J. Bian, and W. Wu, “DBFT: A Byzantine Fault Tolerance Protocol with Graceful Performance Degradation,” *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 114–128, 2023.
45. K. Kant, S. Pandey, and U. Shanker, “Addressing Blockchain Efficiency: A Study on Super Node-Based Consensus Mechanisms,” in *Proc. 1st International Conference on Advanced Computing and Emerging Technologies (ACET)*, Ghaziabad, India, 2024, pp. 1–6, doi: 10.1109/ACET61898.2024.10730600.
46. Z. Xiao, B. Zhu, X. Li, and D. Wang, “Proof of Importance: A Consensus Algorithm for Importance Based on Dynamic Authorization,” in *Proc. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2021, pp. 510–513.
47. W. Chen, L. Xu, S. Li, N. Guo, Z. Li, Y. Yang, and W. Shi, “On Security Analysis of Proof-of-Elapsed-Time (PoET),” in *Stabilization, Safety, and Security of Distributed Systems (SSS 2017)*, Boston, MA, USA, 2017, pp. 282–297.
48. D. De Angelis, S. Aniello, L. Baldoni, R. Lombardi, F. Margheri, A. Sassone, and V. Sassone, “PBFT vs. Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain,” in *CEUR Workshop Proceedings*, vol. 2058, 2018.
49. D. Schwartz, N. Youngs, and A. Britto, “The Ripple Protocol Consensus Algorithm,” Ripple Labs Inc., White Paper, 2014.