Journal of Network Intelligence ©2024 ISSN 2414-8105 (Online)
Taiwan Ubiquitous Information Volume **9**, Number **1**, February **2024**

Improved PBFT Algorithm Based on Credit Evaluation

Zheng-Yi Tang

School of Computer Science and Mathematics
Fujian University of Technology
Xuefu South Road, Fuzhou City, Fujian Province, 350118, China
Key Laboratory of Hunan Province for Mobile Business Intelligence
Hunan University of Technology and Business
Yuelu Road, Changsha City, Hunan Province, 410205, China
tangzy84@126.com

Tian-Wei Ma

School of Computer Science and Mathematics
Fujian University of Technology
Xuefu South Road, Fuzhou City, Fujian Province, 350118, China
mtw8704@gmail.com

Jin-Shui Wang*

School of Computer Science and Mathematics
Fujian University of Technology
Xuefu South Road, Fuzhou City, Fujian Province, 350118, China
Key Laboratory of Hunan Province for Mobile Business Intelligence
Hunan University of Technology and Business
Yuelu Road, Changsha City, Hunan Province, 410205, China
wangjinshui@fjut.edu.cn

Jia-Huan Huang

School of Computer Science and Mathematics
Fujian University of Technology
Xuefu South Road, Fuzhou City, Fujian Province, 350118, China
huangjiahuan6@163.com

*Corresponding author: Jin-Shui Wang

Received April 6, 2023, revised July 25, 2023, accepted September 30, 2023.

Abstract. *As blockchain technology advances rapidly, its core component, the consensus algorithm, has been widely applied. Despite the widespread use of the PBFT algorithm in consortium blockchains, it has several issues, such as arbitrary selection of the primary node, insensitivity to node network changes, high communication overhead, and low consensus efficiency. To address these issues, an improved Byzantine fault-tolerant algorithm (DC-PBFT) is proposed. Firstly, a dynamic node adjustment mechanism is established to enable the system to respond to node joining and leaving. Furthermore, a hierarchical analysis model is constructed to score the comprehensive strength of nodes, and reliable nodes are selected as consensus nodes according to the scores. The VRF algorithm is introduced to elect backup primary nodes and a primary node to ensure the randomness and unpredictability of the primary node. Finally, the PBFT consensus process is simplified to reduce communication overhead between nodes. Experimental findings demonstrate that the DC-PBFT outperforms the PBFT algorithm in fault tolerance, communication overhead, latency, and throughput, boasting increased consensus efficiency and stability. As a result, it is suitable for consortium blockchain with high transaction volume.*

Keywords: Blockchain; Consensus algorithm; Byzantine fault-tolerant algorithm; Consortium blockchain; VRF algorithm

1. **Introduction.** In 2008, the emergence of Bitcoin [1] not only set off a wave of digital currency but also brought new technical ideas to distributed systems. Blockchain [2] originated from Bitcoin and integrates technologies such as cryptography [3], consensus algorithm, P2P network, and smart contract [4]. Its essence is a distributed ledger, where each block body is equivalent to a ledger, and the transaction data on the chain is recorded in each distributed ledger. Blockchain is a data structure that is closely connected according to the actual order. The blocks in the chain represent the transmission transactions between users in chronological order. Cryptography is the foundation of blockchain technology, and it plays a key role in IoT devices and cloud computing [5], healthcare environments [6], and secure smart home environments [7], etc., helping blockchain technology to ensure information security in blocks. Currently, the application areas of blockchain include digital currencies, financial transactions, supply chain [8], and information security. Blockchain technology has greatly improved many research work. such as, Chen et al. [9] developed a novel medical data sharing plan that uses blockchain to integrate the resources of each hospital and provide a secure distributed environment to avoid single-point attacks. Mei et al. [10] proposed a secure and effective blockchain-enabled privacy-preserving authentication scheme, which supports unconditional anonymity and data batch integrity verification while greatly simplifying key management issues. Combining the characteristics of the blockchain, Chen et al. [11] proposed a new signature exchange protocol called DFSE, which improves the fair transaction problem in metaverse.

Blockchain can be divided into three types based on their openness and application scenarios: public blockchain, consortium blockchain, and private blockchain. In public blockchain, nodes can freely join or exit at will without additional verification. It usually using the proof-of-work (POW) [12] consensus algorithm, which is utilized by Bitcoin, realizes complete decentralization. But it requires a lot of computing power, leading to resource waste and low consensus efficiency. Private blockchain is typically applied within enterprises which is a closed system. The write permissions of each node are controlled internally. Consortium blockchain is used for several organizations or enterprises with mutual interests, where the blockchain is maintained jointly by the consortium. Each member node's joining or exiting requires an audit. Although member nodes have the same goal, there may be Byzantine nodes among them, which lead them to distrust each other. So the commonly used consensus algorithms include Practical Byzantine Fault Tolerance (PBFT) [13, 14], Paxos [15] , Raft [16], and Gossip [17].

In 2015, the Linux Foundation launched the open-source project Hyperledger, which utilizes the PBFT algorithm as its consensus mechanism. While this algorithm has proven effective in addressing Byzantine problems in distributed systems within consortium blockchain. However, it also has many limitations. Firstly, as the algorithm necessitates all nodes in the consortium blockchain to participate in the consensus, an increase in node count leads to a significant rise in consensus delay and network bandwidth requirements. Moreover, the classic PBFT algorithm also cannot dynamically sense node addition or departure. Secondly, the primary node is selected based on the view number and the node counts. Malicious nodes have a chance to become the primary node, which affects system security and causes frequent view changes. Finally, the three-phase protocol in the PBFT algorithm requires multiple rounds of communication between consensus nodes, leading to low communication efficiency and poor scalability. Consequently, the PBFT algorithm is not well-suited for scenarios with numerous nodes. To address these issues, this paper proposes an improved Byzantine fault-tolerant algorithm, with the main focus of the research outlined as follows:

(1) A mechanism for dynamically adjusting nodes has been designed, enabling nodes to join and exit the network freely. The system can dynamically sense and make corresponding process adjustments to ensure that the new nodes will not affect the consensus process.

(2) The Analytic Hierarchy Process (AHP) is employed to assess node credibility, with nodes classified into two categories: backup nodes and consensus nodes based on their ranking. In the consensus nodes, the Verifiable Random Function (VRF) is utilized to select the backup primary node, and the highest-scoring backup primary node becomes the primary node. Once primary node responds with timeout or behaves maliciously, the highest-scoring backup primary node will assume the role of the new primary node immediately, avoiding triggering view change.

(3) The consensus protocol is optimized by excluding backup nodes from the consensus process and simplifying the commit phase, thereby reducing communication complexity.

The rest of the paper is organised as follows. Section 2 reviews related research on improved PBFT algorithm. The following section discusses the consensus process of the PBFT algorithm. Section 4 outlines the proposed improvements to the PBFT algorithm, which includes the dynamic node adjustment mechanism and trust mechanism. In section 5, the performance test and security analysis of PBFT and DC-PBFT algorithms are carried out respectively. The last section draws conclusions and analyzes direction of future work.

2. **Related Work.** Currently, many researchers have proposed improvement methods for the shortcomings of consensus algorithms. Gao et al. [18] proposed selecting a group of primary nodes from the consensus nodes based on trust values, replacing a single primary node, which reduces the probability of view change and optimizes communication complexity. However, it is vulnerable for attackers to add a great number of nodes to forge trust scores, leading to the group controlled by the attacker and disrupting the consensus process. Zhang et al. [19] combined ring signatures technology with consensus algorithms and used associated ring signature to ensure the privacy of members in the consortium blockchain. However, the encryption and decryption process of ring signatures increases the time delay, leading to a decrease in consensus efficiency. Liu et al. [20] improved response mode. He combined DPoS with PBFT to optimize the composition of consensus nodes and improve algorithm performance. However, due to limited network bandwidth,

consensus efficiency is still affected by network congestion. Fang et al. [21] introduced a rating mechanism to classify nodes into three categories, simplifying the confirmation phase in the consistency protocol, and reducing communication overhead. Although the performance of the algorithm has been optimized, the sending and confirmation of messages are controlled by the primary nodes, weakening the multi-center feature of the consortium blockchain. Aublin et al. [22] achieved the reliability of consensus through resource redundancy. And introduced the concept of random numbers which weakens the authority of the primary node. Nevertheless, the algorithm's application scope is limited. When the primary node broadcasts information, the consensus nodes require a lot of time to verify the response, resulting in low efficiency.

Tang et al. [23] introduced a trust rating mechanism and simplified the pre-prepare phase to make the algorithm suitable for high-frequency trading scenarios. Li et al. [24] grouped the nodes into different layers and limited the intra-group communication to reduce communication complexity. Xie et al. [25] elected the primary node based on the Probabilistic Language Term Set (PLTS) to improve consensus efficiency. Jiang et al. [26] proposed a scalable Byzantine fault tolerance algorithm based on a tree topology network (STBFT), which can take different steps to reach consensus according to the abnormal situation of the system. Wang et al. [27] combined the ideas of DPoS and PBFT to design a local blockchain network consensus algorithm (LBNC) to achieve on-chip consensus, which increases the number of nodes that the system can accommodate by processing transactions in parallel with multiple shards. Wang et al. [28] proposed an improved practical Byzantine fault-tolerant consensus mechanism VPBFT based on Verifiable Delay Function (VDF), which improves the problems of unreasonable selection of main nodes and high transaction latency. However, these improvement methods mostly focus on improving consensus efficiency and increasing scalability, and rarely consider fault tolerance and dynamic adjustment of node status.Therefore, according to the improvement measures of the above researchers, this paper proposes a credit evaluation based PBFT algorithm.

3. **PBFT algorithm.**

The Byzantine Generals Problem refers to the consistency protocol problem. It arises when the generals of the Byzantine army are required to reach a unanimous decision about whether to launch an attack against an opposing army, while some of the generals are traitors. When the identity of the traitors is known, the task of the loyal generals is to agree on a course of action without succumbing to the influence of the traitors, thus forming the Byzantine Generals' Problem. The PBFT Algorithm was developed to resolve this problem in distributed systems. It can handle up to f Byzantine nodes in a network of N nodes. The system is considered safe and reliable when f is less than $N/3$. Figure 1 shows the operation of the PBFT algorithm.

1. Request phase: The client initiates this phase by sending a request to the primary node.
2. Pre-prepare phase: Upon receiving the request, the primary node broadcasts it to the replica nodes. The replica nodes then enter the pre-prepare phase where they agree on the request's order and create a digest of it.
3. Prepare phase: In this phase, the replica nodes send a prepare message to each other to validate the request's digest and confirm their agreement on its order.
4. Commit phase: Once a node receives prepare messages from $2f$ replica nodes (f being the number of faults the network can tolerate), it enters the commit phase. In this phase, nodes send a commit message to each other to confirm their agreement on the request's content.
5. Reply phase: Once a node receives commit messages from $2f + 1$ replica nodes, it executes the request and sends a final result to the client.

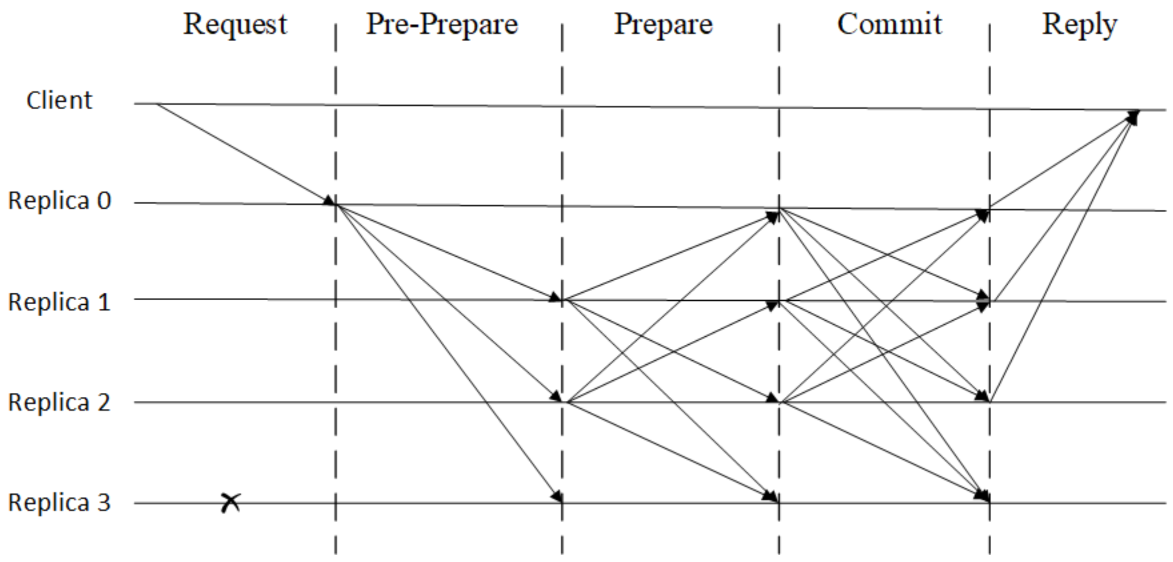

FIGURE 1. PBFT algorithm consensus flow chart

4. **Improve the consensus mechanism.** In order to provide a consensus algorithm that can dynamically adjust nodes and has high scalability, low latency and high throughput, so that it can adapt to the high transaction volume environment of the consortium blockchain. This paper proposes an improved PBFT algorithm based on dynamic adjustment and credit election– DC-PBFT. Its main flow is illustrated in Figure 2.

The DC-PBFT algorithm evaluates the initial scores of the nodes through several evaluation indicators and then categorizes them according to their ranking. The top-ranked nodes are chosen as consensus nodes to take part in the consensus process. Within the consensus nodes, the VRF algorithm is employed to select the backup primary node and the primary node, which solves the problem of arbitrary selection of the primary node and easily triggering view change, thus ensuring consensus efficiency and security. After consensus is complete, the primary node evaluates the node's honesty and adjusts their scores. After every 50 rounds of consensus, the node classification will be readjusted to ensure that more reliable nodes participate in the consensus.

4.1. **Node dynamic adjustment.** Due to the inability of the PBFT algorithm to dynamically perceive the joining or leaving of each node, it has significant limitations in a high transaction volume environment. This paper designs a node dynamic adjustment mechanism to address this problem.

4.1.1. *Dynamically add nodes.* Consensus nodes and backup nodes are two types of nodes in the network. To avoid affecting the consensus process, newly added nodes are first classified as backup nodes. The joining of new nodes needs to go through four phases, which are illustrated in Figure 3:

Join-Request phase: The new node sends a joining message to all nodes in the system, including the identity information, IP address, timestamp, and signature of the new node.

Check phase: The consensus node will verify the identity of new node, check whether the node has any bad or illegal behavior, and broadcast messages to other consensus nodes after the verification is passed. When the consensus node receives $2f + 1$ messages, it proceeds to the next phase.

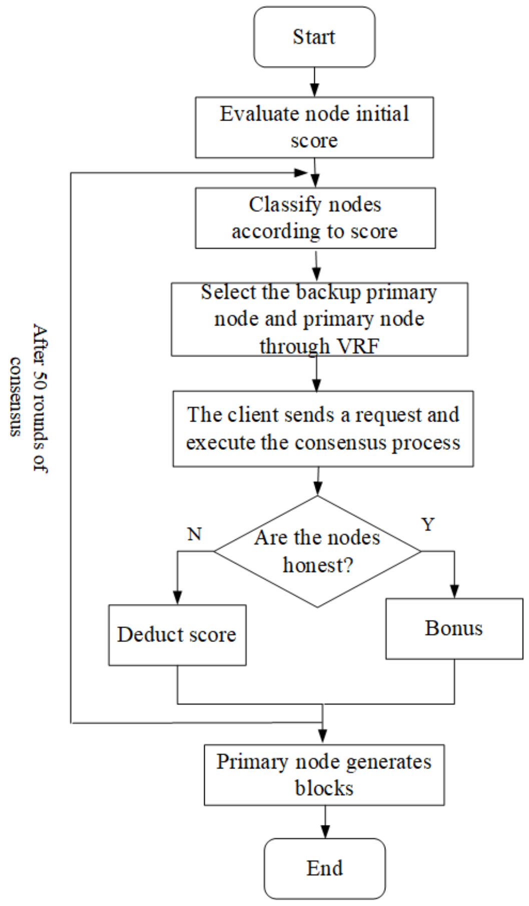
Start
Evaluate node initial score
Classify nodes according to score
Select the backup primary node and primary node through VRF
The client sends a request and execute the consensus process
Are the nodes honest?
N
Y
Deduct score
Bonus
After 50 rounds of consensus
Primary node generates blocks
End

FIGURE 2. Algorithm flow chart

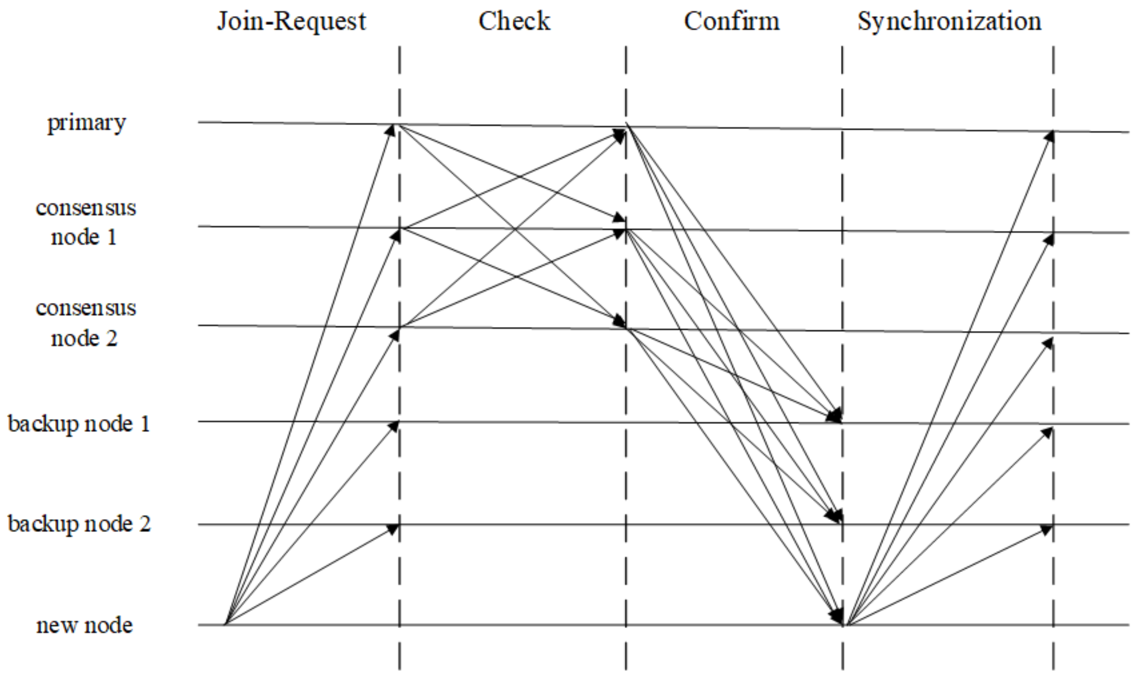

FIGURE 3. Node dynamic addition flow chart

Confirm phase: The consensus node broadcasts the authentication passing information to the backup node and the new node, and the primary node will add log information to the confirmation information so that the new node can synchronize.

Synchronization phase: The new node synchronizes with the network based on the log content broadcast by the primary node. After synchronization, it broadcasts to all nodes. The new node officially joins the system network and is classified as a backup node.

4.1.2. *Node exit.* Consensus node and backup node have different procedures for exiting. The consensus node needs to switch to a backup node before exiting, which can be divided into three phases, as shown in Figure 4:

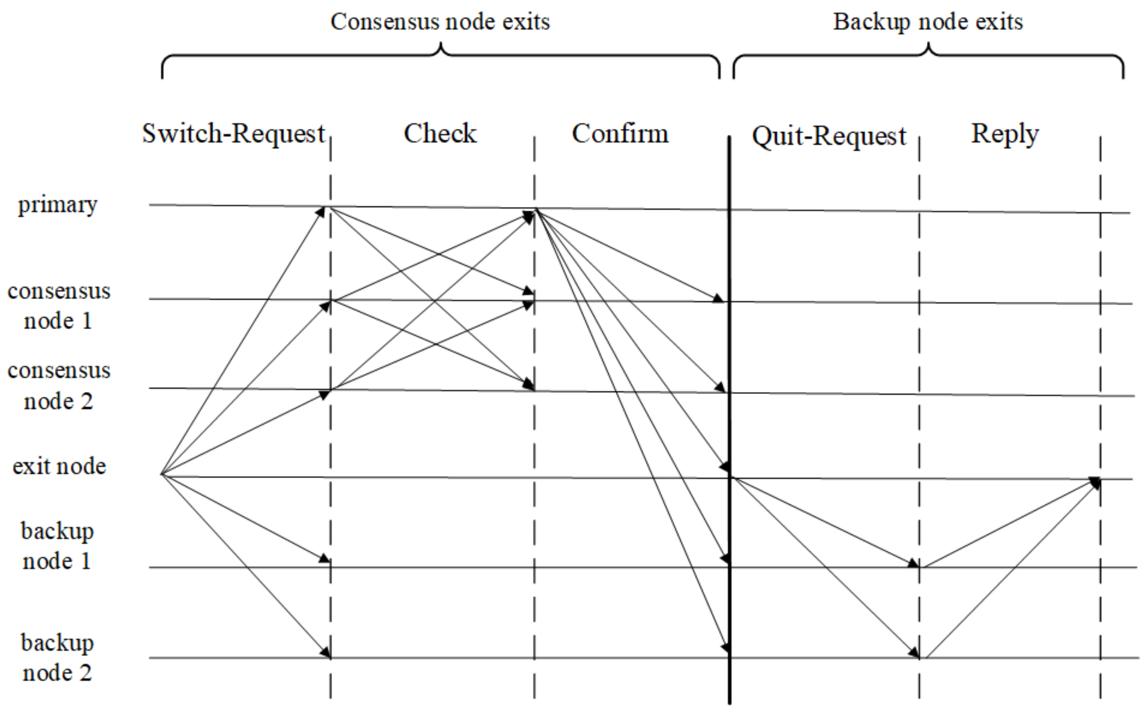

FIGURE 4. Node active exit flow chart

Switch-request phase: The node actively requests to switch to the backup node, and broadcasts the request to all nodes. Before fully confirming the switching of the backup node, it still needs to continue to complete the consensus.

Check phase: The consensus node broadcasts to other consensus nodes to confirm receipt of a switch request.

Confirm phase: Upon receiving $f+1$ confirmation requests, the primary node broadcasts the node i switching success information to all nodes and switches it to a backup node. Upon receiving the information, each node updates its local log accordingly.

The exit of the backup node is divided into two phases:

Quit-request phase: The backup node sends an exit request to all nodes.

Reply phase: The node that received the request updates the log. The content includes: deleting the information of the exit node, updating the overall number of nodes, and changing the view number to $v+1$. After updating, broadcast a reply message to the exit node. The exit node that receives the reply officially exits the network.

4.2. **Trust mechanism.** This paper proposes the DC-PBFT algorithm. The nodes' overall strength is evaluated using the AHP, which results in their classification into backup nodes and consensus nodes according to their respective scores. Then the backup primary node is selected from the consensus nodes through the VRF algorithm. The backup primary node with the highest score will be the primary node. The remaining backup primary nodes not only require participating in the consensus but also need to use the same view to record the pre-generated blocks. When the primary node responds with timeout or acts maliciously, the backup primary node with the highest score is selected to take over which will not trigger view change. The nodes do not need to ensure that $2f + 1$ nodes believe $prepared(m, v, n, i)$ is true through the commit phase.

4.2.1. *Node initial score.* The consortium blockchain is composed of members from a specific industry, such as insurance, banking, and securities. For this type of blockchain, there are certain differences in the Generated Blocks (GB), Transaction Processing Time (TPT), Social Reputation (SR), Historical Reliability (HR), and Processed Transactions (PT) of different member nodes. Therefore, the comprehensive strength of nodes can be evaluated according to multiple indicators. The paper argues that the stronger the nodes' overall strength, the higher their security and stability, and the less likely they are to behave maliciously. The initial score calculation steps are as follows:

(1) Determine the regulatory node. The system assigns numbers 1, 2, ..., N to each node in the consortium blockchain, and then generates a random number between these numbers. The node with the assigned number corresponding to the random number is selected as the regulatory node.

(2) Rating. The node scores other nodes in the chain on the five indicators of "GB, PT, TPT, SR, and HR". The scoring range is 1-10. If there is no intersection with some nodes, the score is directly set to 5. After scoring, the results will be stored in the local log and sent to the regulatory node. After the regulatory node collects all node data, calculate the mean value according to Equation (1).

$$\hat{x}_i^k = \frac{\sum_{j=1}^{N} x_{j\to i}^k}{N} \tag{1}$$

Where $\hat{x}_i^k$ is the mean value of the k-th index for the i-th node. For the k-th index, $x_{j\to i}^k$ is the score given by the j-th node to the i-th node. N is the total number of nodes.

(3) Build hierarchical model [29]. The target layer is the initial score of the node, and the criterion layer is divided into "GB, TP, TPT, SC, HR" five indicators, the solution layer is each node in the consortium blockchain. The hierarchical structure is shown in Figure 5.

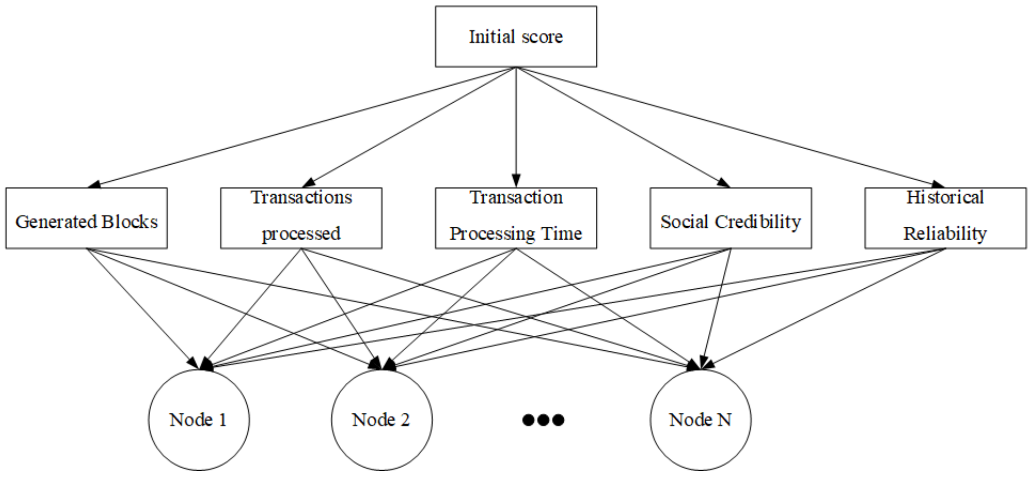

FIGURE 5. Hierarchy model

(4) Establish a comparison matrix. It is formed by comparing five indicators in pairs, using the 1-5 point scale method, and using a_{ij} to represent the scale. As shown in Table 1.

TABLE 1. Comparison matrix

$i \backslash j$	GB	PT	TPT	SC	HR	Weight
GB	1	5	3	2	3	0.4121
PT	1/5	1	1/2	1/3	1/3	0.0682
TPT	1/3	2	1	1/2	1/2	0.1181
SC	1/2	3	2	1	1	0.2078
HR	1/3	3	2	1	1	0.1938

The higher the value of a_{ij}, the more important the index i is than the index j. If a_{ij} equal to 1, the index i and the index j are equally important. After generating the matrix, the weight allocation is checked for its validity using Equation (2-3). If $CR < 0.1$, it is determined that the weight allocation meets the consistency requirements. Otherwise the data in the matrix needs to be adjusted.

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{2}$$

$$CR = \frac{CI}{RI} \tag{3}$$

Where λ_{max} is the largest eigenvalue, CR is the consistency ratio, CI and RI are the consistency index and random consistency index, respectively. n is the order of the matrix.

(5) Compute the initial score. The regulatory node will calculate the initial scores of each node based on the Equation (4).

$$\hat{X}_i = \frac{\sum_{k=1}^{5} \hat{x}_i^k \cdot w_k}{5} \tag{4}$$

Where $\hat{X}_i$ is the initial score of the i-th node. w_k is the weight of the k-th indicator.

After obtaining the initial scores, the regulatory node will classify the nodes according to the scores, and send the scores and the data calculated in each phase to other nodes. If there are nodes that have doubts about the final results, they can ask all nodes for their own scores to check. If there is a node malicious score, the regulatory node will check the log information to determine whether the node's score is honest. If the regulatory node behaves maliciously and tampers with the data, it will be removed from the consortium blockchain once confirmed, and the system will recalculate the initial scores according to the above process.

4.2.2. *Node allocation and update.* In the DC-PBFT algorithm, consensus nodes and backup nodes are two types of nodes with different roles in the consortium chain. A small subset of consensus nodes is chosen to serve as backup primary nodes and primary nodes, as shown in Figure 6.

The backup nodes are used to receive and synchronize the data broadcast by the primary node, but they do not engage in the consensus process. For newly added nodes, they are directly assigned to the backup nodes to maintain the consensus process and prevent

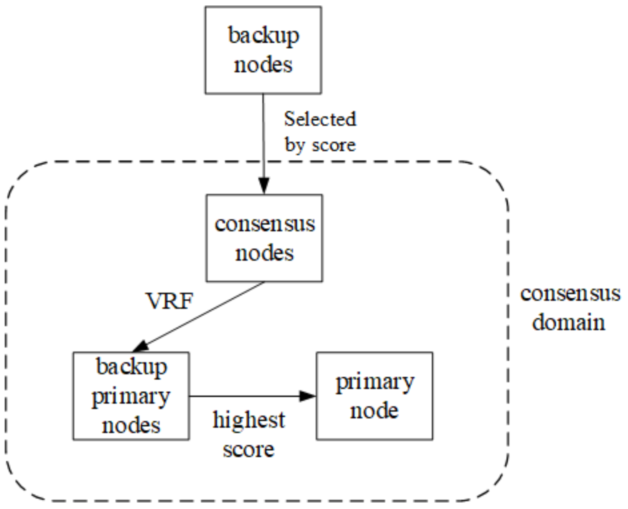

Figure 6. Hierarchy model

it from being affected. The consensus nodes participate in the block consensus, are responsible for receiving the transaction information transmitted by the primary node and verifying the received information to prevent any malicious behavior from other nodes.

The primary node handles the client's request by receiving it, assigning a sequence number, and packaging the transaction into a pre-generated block. In addition to participating in the consensus, the backup primary node also needs to package transactions into its own pre-generated blocks. If the primary node responds with timeout or behaves maliciously, the highest-scoring backup primary node will assume the role of the new primary node, completing the consensus without triggering a view change.

Nodes are ranked based on initial scores, and the top $a(0 < a \leq 1)$ of nodes will be placed in the consensus domain. a represents the ratio of the number of nodes in the consensus domain to the total nodes. Nodes outside this domain are designated as backup nodes, and the consensus domain must have at least $N \geq 3F + 1$, where F is the number of Byzantine nodes within the consensus domain. In the context of consortium blockchain applications, the number of consensus nodes is set at half the total node count. But in the actual application process, the proportion of consensus nodes and backup nodes can be adjusted according to network node count and credibility.

After each consensus is completed, the primary node will broadcast the consensus result to all nodes and update the node's scores. For nodes whose broadcast messages during the consensus process are consistent with the final message, the scores will be increased by 0.01, and if they are inconsistent, the scores will be reduced by 0.05. If the primary node fails or is successfully challenged by other nodes, its scores will be reduced by 5 and will be directly demoted to the backup node, while the score of the challenging node is increased by 0.02. After every 50 rounds of consensus, the backup nodes and consensus nodes will be readjusted, and new backup primary nodes and primary node will be selected. The above process is shown in algorithm 1. In this way, the number of nodes participating in the consensus can be reduced, the blockchain consensus process can be accelerated and significantly enhances the network's dynamism and robustness.

4.2.3. *Primary node selection.* The traditional PBFT algorithm selects the primary node according to the view number and node count. The identity of the primary node is easily predicted by malicious nodes, which will be attacked and destroy the stability of the network. The DC-PBFT algorithm uses VRF [30] to select the primary node among the

Algorithm 1: Update scores

```
Input: Node scores $X_i$, Consensus result $R$, Broadcast message set $M$
Output: Updated scores $X_i'$
if primary node completes consensus then
    broadcast $R$ to all nodes
    for each $m_i \in M$ do
        if $m_i == R$ then
            $X_i' \leftarrow X_i + 0.01$;
        else
            $X_i' \leftarrow X_i - 0.05$;
        end
    end
    if primary node response timeout or is successfully challenged by node i then
        $X_p' \leftarrow X_p - 5.00$ // $X_p$ is the score of primary node
        backup node $\leftarrow$ primary node
        $X_i' \leftarrow X_i + 0.02$
        re-select new primary node from backup primary nodes
        add the backup node with the highest score to the consensus domain
    end
    return $X_i'$
end
if consensus rounds == 50 then
    readjust the node classification according to the $X_i'$
end
```

consensus nodes. The VRF ensures that malicious nodes cannot know the identity of the primary node before the consensus starts, which enhances the security of the algorithm. The selection process is as follows:

(1) All consensus nodes execute the primary node selection algorithm. The input of the algorithm is the node private key SK and the random seed s. The output is hash value and zero-knowledge proof, as follows:

$$s = hash(PK, block - height) \tag{5}$$

$$result = VRF\_Hash(SK, s) \tag{6}$$

$$(r, p) = VRF\_proof(SK, s) \tag{7}$$

Among them, PK is the public key, $block - height$ is the height of the current block, r is a random number, and p is a zero-knowledge proof.

(2) The node judges whether it is a backup primary node according to the Equation (8).

$$\begin{cases} \text{yes}, & \dfrac{\text{hash}(r)}{2^{\text{hashlen}}} < \lambda \\ \text{no}, & \dfrac{\text{hash}(r)}{2^{\text{hashlen}}} \geq \lambda \end{cases} \tag{8}$$

Among them, λ is the threshold, and its value range is dynamically adjusted according to the number of nodes. $hashlen$ is the length of the output result of the SHA-256 algorithm.

(3) Nodes that meet the threshold requirements broadcast $(r, p, yes, rank)$ to other nodes, where $rank$ is the initial score ranking. The nodes that receive the message will verify according to Equation (9).

$$\frac{hash(r)}{2^{hashlen}} < \lambda \quad \&\& \quad Verify(r, s, p, PK) \tag{9}$$

Upon successful verification, the node will compare the rank of the backup primary node and choose the highest-ranked node as the primary node. Subsequently, the node sends an acknowledgment message to the main node. After receiving $f + 1$ acknowledgment messages, the backup primary node confirms its role as the primary node and initiates the consensus process by packaging transactions into a new block.

4.3. **DC-PBFT consensus process.** After scoring the nodes, divide the consensus domain and start to execute the 4-phase consensus protocol of DC-PBFT algorithm, as shown in Figure 7.

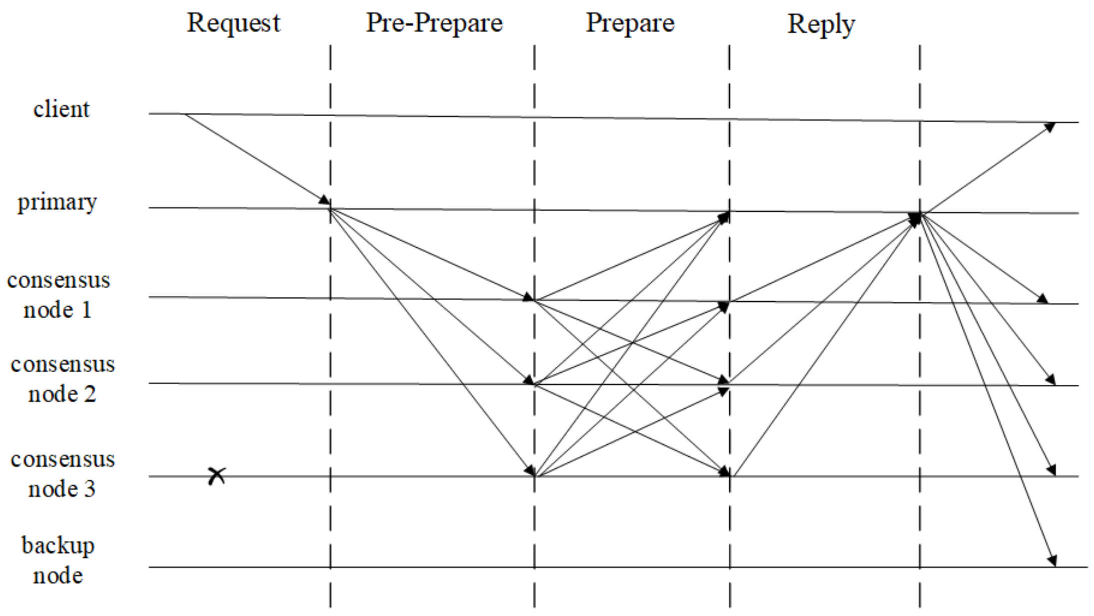

FIGURE 7. DC-PBFT algorithm consensus flow chart

1. Request phase: The client sends a request message $< REQUEST, o, t, c > \delta_c$ to the primary node, where o represents the operation, t represents the timestamp, c represents the client, and δ_c represents the client's signature.

2. Pre-prepare phase: In this phase, the primary node will assign a sequence number n to the request message, and broadcast the pre-prepare message to the consensus node. The content of the pre-prepare message is $<< PRE - PREPARE, v, n, d, x > \delta_p, m >$, among them, d is the request digest, x is the initial score, δ_p is the primary node's signature, and m is the details of the request.

3. Prepare phase: In this phase, the backup primary node will use the same view to temporarily record the pre-generated blocks. If the primary node responds with timeout or behaves maliciously, the highest-scoring backup primary node will assume the role of the new primary node immediately. The node that receives the pre-prepare message will verify it. The verification content includes: whether the signature of the message is forged; whether the request sequence number n is between the waterline h and H;

whether the sequence number n is assigned to only one request in the current view. After the verification, the node will broadcast a prepare message $< PREPARE, v, n, d, i, x > \delta_i$ to other members in consensus domain.

4. Reply phase: Once consensus nodes received more than 2/3 prepare messages, it will send a reply message $< REPLY, v, t, c, i, x, r > \delta_i$ to the primary node, where r represents the reply result of node i. If the primary node receives $f + 1$ identical execution results, including the same r and t in the message, it means that the consensus is reached. The primary node will broadcast the consensus result to the client and all nodes.

5. **Experiment analysis.**

The experimental analysis of the PBFT and DC-PBFT algorithms from four perspectives: fault tolerance, communication overhead, consensus delay, and throughput. We have simulated the performance of the PBFT and DC-PBFT algorithms using Go, and created clients and nodes with different addresses locally by opening multiple ports. The experimental environment is Intel Core i5-7300HQ CPU, Windows 10 operating system, 16GB memory, and 256GB SSD.

5.1. **Fault Tolerance Analysis.**

The rate of Byzantine fault tolerance plays a crucial role in determining the system's stability, as the number of Byzantine nodes directly impacts the success of the consensus. In a network with a total number of nodes of N, $(N-1)/3$ is the maximum number of Byzantine nodes allowed in the PBFT algorithm. Section 4.2.2 shows that the DC-PBFT algorithm selects the top $a(0 < a \leq 1)$ of nodes in the initial integral ranking as consensus nodes, allowing it to tolerate $(aN-1)/3$ Byzantine nodes within the consensus nodes, where a is the proportion of consensus nodes. Assuming that all nodes outside consensus domain are Byzantine nodes, then DC-PBFT algorithm can tolerate $(3N - 2aN - 1)/3$ Byzantine nodes. The maximum tolerance ratios of the two algorithms for Byzantine nodes are shown in Equation (10):

$$F = \frac{\frac{3N-2aN-1}{3}}{\frac{N-1}{3}} = \frac{3N - 2aN - 1}{N - 1} \tag{10}$$

For consensus nodes with different proportions, a visualized three-dimensional graph has been generated generated using Python, as shown in Figure 8:

Among them, the value of a are $0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$ respectively. And N ranges from 10 to 100, increasing by 0.1 for each experiment. The results show that for values of a less than 1, the F value is always greater than 1, indicating that the DC-PBFT algorithm consistently has a higher fault tolerance. This demonstrates that the system employing the DC-PBFT algorithm is more stable.

5.2. **Communication overhead.**

Communication overhead pertains to the average number of times a consensus node performs information interaction. Assuming there are N nodes in the network, and the probability of primary node failure is P. The communication times of each phases in the PBFT algorithm are $(N-1)$,$N(N-1)$,$N(N-1)$, N respectively. The communication times during a view change upon primary node failure is $N(N-1)$, so the communication overhead of the PBFT algorithm is $2N^2-1+P(N^2-N)$.

The DC-PBFT algorithm selects $aN(0 < a \leq 1)$ nodes to complete consensus. Then the communication times for pre-prepare, prepare and reply are $(aN-1)$, $aN(aN-1)$ and $(aN-1)$, respectively. After reply phase, the primary node needs to disseminate N consensus message to all nodes and client. When the primary node fails, the backup primary node with the highest score broadcasts $(N-1)$ election certification message to all nodes. The remaining nodes confirm $(N-1)(N-1)$ interactively, and then reply $(N-1)$

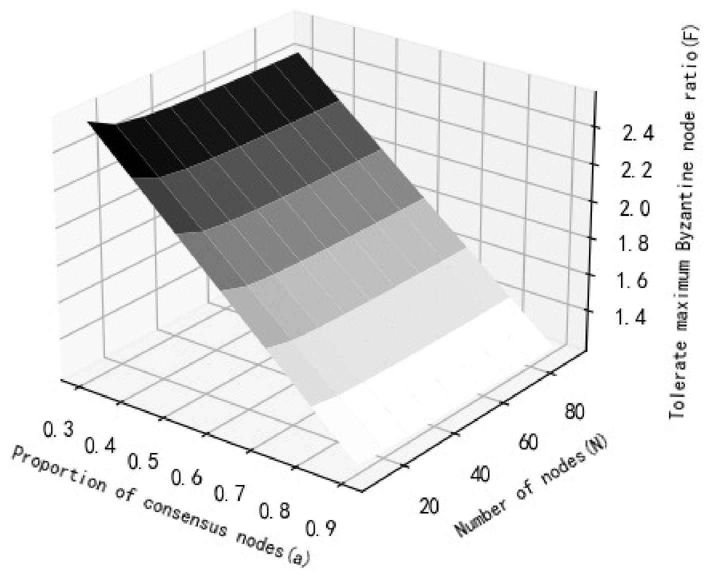

FIGURE 8. Comparison of fault tolerance between DC-PBFT and PBFT

to the new primary node. The total communication times is $a^2N^2+N(a+1)+P(N^2-N)$. The communication overhead ratio is shown in Equation (11):

$$C=\frac{a^2N^2+N(a+1)-2+P(N^2-1)}{2N^2-1+P(N^2-N)} \tag{11}$$

The DC-PBFT algorithm sets a to 0.5, and the visualized three-dimensional graphics of Equation (11) are obtained through Python, as shown in Figure 9:

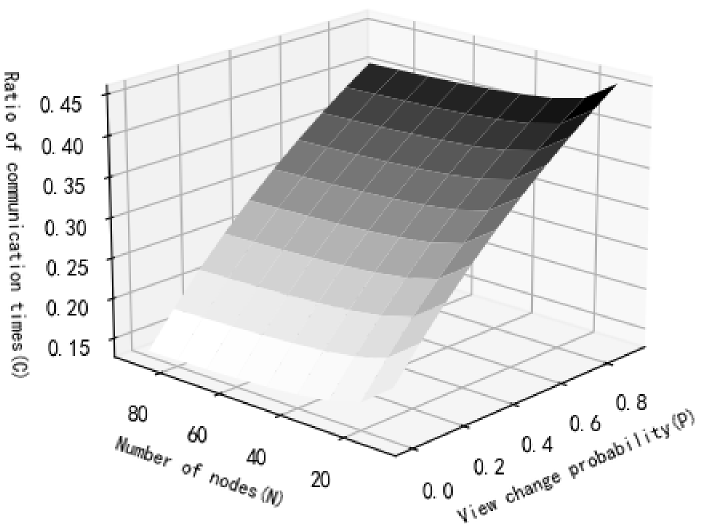

FIGURE 9. Comparison of communication overhead between DC-PBFT and PBFT

Regardless of the variations in the number of nodes and the primary node's failure probability, the figure shows that the ratio of communication times is always less than 1.

This indicates that the DC-PBFT algorithm has fewer communication overhead, which can better improve consensus efficiency.

5.3. **Consensus Latency.** Consensus latency signifies the duration from the instant a client submits a transaction request to the primary node until the client obtains the transaction confirmation. Latency serves as a crucial metric for assessing the performance of a consensus algorithm. A shorter delay corresponds to a quicker consensus completion and heightened system efficiency. In this experiment, the node count (N) escalated from 4 to 49. For the DC-PBFT consensus algorithm, the consensus node count was configured at N and $0.5N$. The consensus latency for both PBFT and DC-PBFT algorithms underwent multiple tests, with the average value representing the final outcome. The results are displayed in Figure 10:

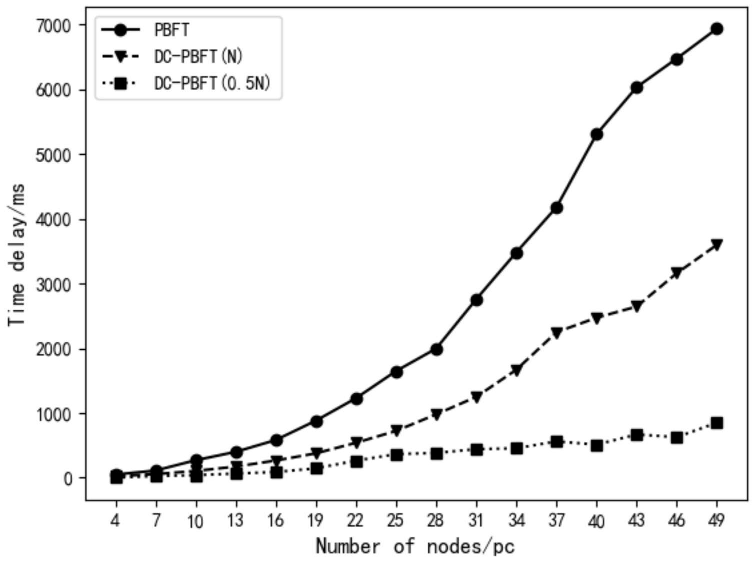

FIGURE 10. Comparison of consensus delay between DC-PBFT and PBFT

The experimental findings indicate that the DC-PBFT algorithm outperforms the PBFT algorithm concerning delay. This superiority is primarily attributed to the DC-PBFT algorithm's optimization of the consensus process, which decreases communication volume and the quantity of consensus nodes. As the number of nodes expands, the DC-PBFT algorithm with a consensus node number of $0.5N$ has a more gradual growth trend and higher stability, and is suitable for large consortium blockchain environments with multiple members.

5.4. **Throughput.** Throughput, a crucial metric for evaluating a system's transaction processing capacity, denotes the quantity of transactions executed per time unit. In blockchain applications, it reflects the total number of transactions from transaction sending to transaction confirmation within a certain period of time under certain network conditions, expressed as TPS. In this experiment, the consensus node count ranges from 5 to 50 in increments of 5. For different numbers of nodes, 1000 transaction requests are sent for each test, and the time to complete the transaction is recorded under different numbers of nodes. Subsequently, the number of transactions finalized per second is calculated. Figure 11 illustrates the results.

The figure illustrates that as the consensus node count escalates, the throughput of both algorithms progressively declines. The DC-PBFT algorithm's average throughput is

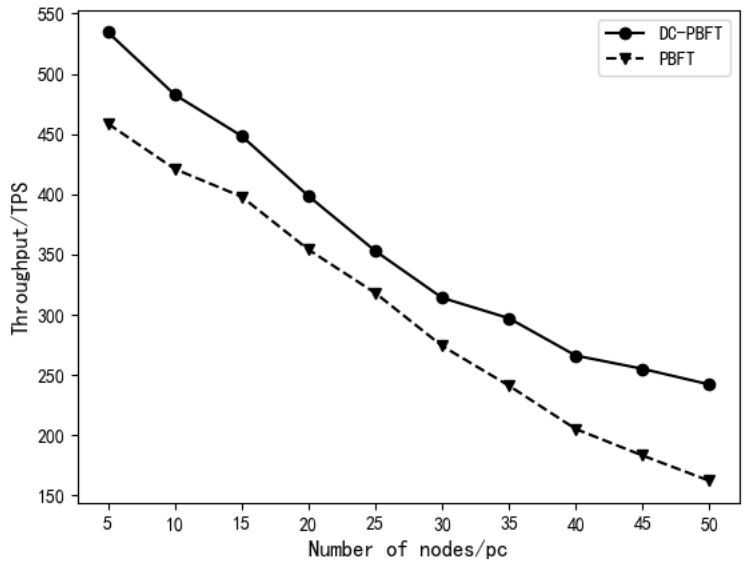

FIGURE 11. Comparison of throughput between DC-PBFT and PBFT

359 TPS, while the PBFT algorithm's is 301 TPS, with the DC-PBFT algorithm consistently surpassing the PBFT algorithm in throughput. When the node quantity exceeds 40, the declining trend of the DC-PBFT algorithm is more gentle. Therefore, the DC-PBFT algorithm can better improve transaction efficiency in the consortium blockchain environment.

6. **Conclusions.**

To address the excessive redundancy of the three-phase protocol, the inability of the system to dynamically sense node joining and quitting, and the overly arbitrary primary node selection method. This article proposes DC-PBFT algorithm. Firstly, the DC-PBFT algorithm designs a node adjustment mechanism, enabling the system to dynamically recognize nodes joining and leaving. Secondly, the algorithm uses a trust mechanism to score and classify nodes and introduces a VRF algorithm to select the primary and backup nodes in the consensus nodes, ensuring the unpredictability of the primary node selection. Finally, the three-phase protocol is improved by selecting reliable nodes for consensus participation and simplifying the communication mechanism. Experimental outcomes demonstrate that the DC-PBFT algorithm is superior to the PBFT algorithm in many aspects such as fault tolerance, delay, and throughput, and has significantly improved consensus efficiency and system stability. DC-PBFT algorithm is more suitable for consortium blockchain with high transaction volume. In future research, the node scoring rules will be more detailed, the consensus protocol will be optimized, and the consensus efficiency and stability will be further improved. Explore how to make PBFT safe and correct under the asynchronous network model, that is, in the presence of message loss, out-of-order or delay. And extend the PBFT algorithm to new application scenarios, such as the Internet of Things or edge computing.

Acknowledgment. This work is partially supported by Natural Science Foundation of Fujian Province (No. 2020J01877), and Open Research Fundation of Key Laboratory of Hunan Province (No. 2015TP1002). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized business review*, 2008, [online]. Available: http://www.bitcoin.org/bitcoin.pdf.

[2] Y. Yuan, and F.-Y. Wang, “Blockchain: The State of the Art and Future Trends,” *Acta Automatica Sinica*, vol. 42, no. 4, pp. 481–494, 2016.

[3] T. M. Fernandez-Carames, and P. Fraga-Lamas, “Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks,” *IEEE Access*, vol. 8, pp. 21091–21116, 2020.

[4] W. Zou, D. Lo, and P. S. Kochhar, “Smart contract development: Challenges and opportunities,” *IEEE Transactions on Software Engineering* , vol. 47, no. 10, pp. 2084–2106, 2019.

[5] T.-Y. Wu, F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, “Rotating Behind Security: An enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture,” *EURASIP Journal on Wireless Communications and Networking* , vol. 2023, pp. 36, 2023

[6] T.-Y. Wu, Q. Meng, L. Yang, S. Kumari, and M.P. Nia, “Amassing the Security: An Enhanced Authentication and Key Agreement Protocol for Remote Surgery in Healthcare Environment,” *Computer Modeling in Engineering and Sciences* , vol. 134, no. 1, pp. 317–341, 2023

[7] T.-Y. Wu, Q. Meng, Y.-C. Chen, S. Kumari, and C.-M. Chen, “Toward a secure smart-home IoT access control scheme based on home registration approach,” *Mathematics* , vol. 11, no. 9, pp. 2123, 2023

[8] C.-Y. Jiang, and C. Ru, “Application of blockchain technology in supply chain finance,” in *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)* , IEEE, 2020, pp. 1342–1345.

[9] C.-M. Chen, X.-T. Deng, S. Kumar, S. Kumari, and S.K. Islam, “Blockchain-based medical data sharing schedule guaranteeing security of individual entities,” *Journal of Ambient Intelligence and Humanized Computing* , 2021. [Online]. Available: https://doi.org/10.1007/s12652-021-03448-7

[10] Q. Mei, H. Xiong, Y.-C. Chen, and C.-M. Chen, “Blockchain-enabled privacy-preserving authentication mechanism for transportation cps with cloud-edge computing,” *IEEE Transactions on Engineering Management* , 2022. [Online]. Available: https://doi.org/10.1109/TEM.2022.3159311

[11] J.-H. Chen, H. Xiao, M.-C. Hu, C.-M. Chen, “A blockchain-based signature exchange protocol for metaverse,” *Future Generation Computer Systems* , vol. 142, pp. 237–247, 2023

[12] M. Jakobsson, and A. Juels, “Proofs of work and bread pudding protocols,” in *Secure Information Networks: Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS’99)* , Springer, 1999, pp. 258–272.

[13] L. Lamport, R. Shostak, and M. Pease, “The Byzantine generals problem,” *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[14] M. Castro, and B. Liskov, “Practical Byzantine fault tolerance,” in *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, 1999, pp. 173–186.

[15] L. Lamport, “Paxos made simple,” *ACM SIGACT News (Distributed Computing Column)*, vol. 32, no. 4, pp. 51–58, 1999.

[16] D. Ongaro, and J. Ousterhout, “In search of an understandable consensus algorithm,” in *TAnnual Technical Conference.* 2014, pp. 305–319.

[17] Z. J. Haas, J. Y. Halpern, and L. Li, “Gossip-based ad hoc routing,” in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE, 2002, pp. 1707-1716.

[18] S. Gao, T.-Y. Yu, J.-M. 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.

[19] L.-S. Zhang, “Research on blockchain consensus algorithm based on Byzantine fault tolerance,” M.S. dissertation, University of Electronic Science and Technology of China, 2020.

[20] X.-F. Liu, “Research on performance improvement of blockchain based on dynamic byzantine fault authorization tolerance consensus algorithm,” M.S. dissertation, Zhejiang University, 2017.

[21] Y.-B. Fang, C.-M Zhou, S. Li, Y.-F Song, N. Gao, and T. Liu, “Improvement of practical Byzantine fault algorithm in alliance blockchain,” *Computer Engineering and Applications*, vol. 58, no. 3, pp. 135–142, 2022.

[22] P. L. Aublin, S. B. Mokhtar, and V. Quéma, “Rbft: Redundant byzantine fault tolerance,” in *2013 IEEE 33rd International Conference on Distributed Computing Systems.* IEEE, 2013, pp. 297–306.

[23] S. Tang, Z.-Q. Wang, J. Jiang, S. Ge, and G.-F. Tan, “Improved PBFT algorithm for high-frequency trading scenarios of consortium blockchain,” *Scientific Reports*, vol. 12, no. 1, pp. 4426, 2022.

[24] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and A.-I. Muhammad, "A scalable multi-layer PBFT consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2020.

[25] M.-Y. Xie, J. Liu, S.-Y. Chen, G.-X. Xu, and M.-W. Lin, "Primary node election based on probabilistic linguistic term set with confidence interval in the PBFT consensus mechanism for blockchain," *Complex & Intelligent Systems*, 2022. [Online]. Available: https://doi.org/10.1007/s40747-022-00857-9

[26] W.-X. Jiang, X.-X. Wu, M.-Y. Song, J.-W. Qin, and Z.-H. Jia, "A Scalable Byzantine Fault Tolerance Algorithm Based on a Tree Topology Network," *IEEE Access*, vol. 11, pp. 33509–33519, 2023.

[27] Q. Wang, J.-W. Ma, and J.-X. Luo "A blockchain sharding scheme in edge computing," *Chinese Journal on Internet of Things*, vol. 7, no. 2, pp. 1–13, 2023.

[28] C.-D. Wang, X. Jiang "Improved practical Byzantine fault-tolerant algorithm based on verifiable delay function," *Journal of Computer Applications* , 2023. [Online]. Available: 10.11772/j.issn.1001-9081.2022111708

[29] A. Rodríguez, F. Ortega, and R. Concepción, "A method for the evaluation of risk in IT projects," *Expert Systems with Applications*, vol. 45, pp. 273–285, 2016.

[30] S. Micali, M. Rabin, and S. Vadhan, "Verifiable Random Function," in *40th Annual Symposium on Foundations of Computer Science (cat. No. 99CB37039)*, IEEE, 1999, pp. 120–130.