

Retraction

Retracted: Fault Tolerance Byzantine Algorithm for Lower Overhead Blockchain

Security and Communication Networks

Received 10 October 2023; Accepted 10 October 2023; Published 11 October 2023

Copyright © 2023 Security and Communication Networks. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

- (1) Discrepancies in scope
- (2) Discrepancies in the description of the research reported
- (3) Discrepancies between the availability of data and the research described
- (4) Inappropriate citations
- (5) Incoherent, meaningless and/or irrelevant content included in the article
- (6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.






The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

- [1] 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 1855238, 9 pages, 2022.

Research Article

Fault Tolerance Byzantine Algorithm for Lower Overhead Blockchain

Riyad Almakki ¹, Lulwah AlSuwaidan ¹, Shakir Khan ¹, Abdul Rauf Baig ¹,
Samad Baseer ² and Manmohan Singh³

¹College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia

²Department of Computer System Engineering, University of Engineering and Technology Peshawar, Peshawar, Pakistan

³Department of Computer Science and Engineering, IES College of Technology, Bhopal, India

Correspondence should be addressed to Shakir Khan; sgkhan@imamu.edu.sa

Received 9 February 2022; Revised 28 March 2022; Accepted 29 April 2022; Published 20 May 2022

Academic Editor: Mukesh Soni

Copyright © 2022 Riyad Almakki et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new algorithm for practical Byzantine fault tolerance (PBFT), called score-PBFT or S-PBFT, is proposed to solve the problems of high communication overhead and low algorithm efficiency. This algorithm is based on the characteristics of the consortium chain. The scoring mechanism for nodes is added. All the nodes are broken up into consensus nodes, candidate nodes, and early nodes. To make sure the consensus nodes are as reliable as possible, the nodes are changed dynamically based on how each node is behaving. Improved: the election method for the controller node has been changed. The node's score and behaviour are used as the election basis to make the algorithm more stable. In this paper, we want to improve the consensus protocol's execution process, cut down on how many nodes are involved in the consensus process, simplify it, and make it more efficient. Results show that, when compared with the PBFT algorithm, the S-PBFT algorithm has a shorter consensus delay, less communication overhead and throughput, and better consensus node reliability.

1. Introduction

In 2008, a researcher dubbed “Satoshi Nakamoto” suggested the creation of a digital currency called Bitcoin [1]. Bitcoin can successfully resolve the Byzantine “double-spending” problem associated with previous digital currencies [2]. The broader issue [3] swiftly garnered significant attention from people from all walks of life. Blockchain technology has progressively reached the public consciousness as a result of the popularity and growth of digital coins. A blockchain is a decentralized ledger that is structured in the form of a chain [4]. It is a combination of peer-to-peer networks, cryptography, a consensus method, and smart contracts, among other technologies. It is decentralised, unchangeable, and safe, with features such as communal maintenance and reputation management.

At the moment, blockchain technology's application scenarios have moved beyond the early banking sector to include nonfinancial sectors such as health care, the Internet

of Things, supply chain management, and edge computing [5]. As a distributed system, blockchain must guarantee that all nodes achieve a consensus on a consistent state. Thus, the consensus algorithm is a critical component of blockchain technology since it directly influences the system's performance and scalability [6]. Blockchains can be classified as public, consortium, or private chains based on their deployment techniques. The public chain's consensus process is mostly based on proof of X (PoX) [7], such as proof of work (PoW). This algorithm effectively solves the consensus problem in the blockchain system, but it consumes a large amount of network computing power and has a low overall efficiency [8]; Zhang et al. [9] proposed a proof-of-stake (PoS) mechanism, incorporating the concept of coinage based on the difficulty of mathematical calculation, which reduces computing power consumption and improves equity-proven efficiency. Because consortium and private chains require authorization to join the blockchain network, they are often referred to as permission chains [10]. The

licensing chain relies heavily on distributed consensus methods, such as practical Byzantine fault tolerance (PBFT) [11] and its optimization algorithm, the Paxos algorithm [12], and the raft algorithm [13], among others. The PBFT algorithm is a Byzantine state machine method that can withstand an assault from a specified number of hostile nodes. It increases the system's stability and availability via the state machine replication method. Because the PBFT method is more effective at resolving the Byzantine generals problem that distributed systems encounter, it has become the de facto consensus algorithm for permission chains.

The PBFT algorithm is an evolution of the Byzantine fault tolerance method (BFT) [1]. It inherits the benefits of the BFT method, significantly decreases the algorithm's network cost, and gives the technique actual application value. However, there are several drawbacks to employing the PBFT method. To begin, the PBFT algorithm is incapable of dynamically detecting changes in the system's node count. Due to the algorithm's C/S design, restarting the system to add additional nodes has a major influence on the system's daily functioning. Second, in numerical sequence, each node in the PBFT algorithm takes turns serving as the controller node. Third, the selection procedure is rather straightforward, and the absence of node qualification evaluation introduces unknown risks to the system's security. Finally, while the PBFT method decreases the complexity of the BFT algorithm to a polynomial level, the communication load it imposes on distributed systems remains rather large. The system's latency will considerably grow as the number of nodes increases. Increases have an effect on the system's operational efficiency. To address the aforementioned issues, researchers intend to optimize the PBFT algorithm's execution protocol. S-PBFT [14], scalable BFT [15], and EPBFT [16] are all examples of improved PBFT algorithms.

Addressing the shortcomings of the PBFT algorithm in conjunction with the consortium chain's properties, this article presents a better PBFT method (score-based PBFT, S-PBFT) [14]. To begin, the basic PBFT algorithm is enhanced with a scoring system. Each node is assessed based on its operational condition and classified into three categories based on its score: consensus node, candidate node, and reserve node. Second, the confirmation step of the PBFT algorithm's consensus process is simplified. The consensus procedure, which was previously including all nodes, is simplified to involve only the consensus nodes, increasing consensus efficiency and decreasing the algorithm's complexity. Finally, the controller node election approach is enhanced when integrated with the scoring system. The node serves the controller node in S-PBFT with the highest score and significant number among the consensus nodes, ensuring the controller node's reliability to the maximum extent possible and reducing the frequency of triggering the view replacement protocol to improve the algorithm's efficiency.

2. S-PBFT Algorithm

2.1. The Overall Idea of the Algorithm. The PBFT algorithm is a kind of "state machine" Byzantine system. It was used to

achieve the consistency of the states of each node in the traditional distributed system customer request. In the blockchain system, the consensus process of each block is carried out in a strict order without the intervention of the consensus algorithm [14]. In addition, in the alliance chain environment, nodes need to go through a specific identity authentication mechanism when entering the system, such as role-based identity authentication, which can effectively avoid problems such as witch attacks. Therefore, to make the PBFT algorithm better applied to the consortium chain system, combined with the characteristics of the consortium chain, aiming at the deficiencies in the classic PBFT algorithm, the following optimization ideas are proposed:

- (a) In the consensus protocol, exclude the request and respond stages. In the PBFT algorithm, the demand and reply phase represents the interaction between the system node and the client. Simultaneously, the data block is created directly by the controller node during the blockchain consensus process, without the need of a specific client.
- (b) Implement a scoring algorithm to categorize nodes based on their node points. To begin, the starting points (Sib) are assigned to each node based on the alliance chain's identity authentication features and the size of the node's processing capability. Each node's first moments are recorded and then separated into consensus nodes and candidate nodes. Primary nodes are classified into three categories. The consensus node is in charge of concluding the system's consensus procedure. When a client receives more than $f+1$ consistent messages from lawful nodes, the client can decide that the request was properly completed. Thus, the number of consensus nodes is $2f+1$; the number of candidate nodes is not fixed; the number of reserve nodes is indeterminate. The reserve nodes are composed of nodes with low initial points and nodes with errors in the consensus nodes; they do not participate in the consensus process but are required to save the consensus results.
- (c) Simplify the process of selecting controller nodes. The controller node is composed of the consensus node's components. When a controller node encounters a problem, the node is immediately downgraded to a reserve node. Then, among the remaining consensus nodes, the node with the highest scores and highest starting scores is chosen as the controller node.
- (d) Simplify and optimize the consistency protocol. The consensus process is finished in the node interaction stage of the PBFT algorithm, and the confirmation stage's primary purpose is to enable each node to comprehend the state of the remaining nodes. In conventional distributed systems, the confirmation phase serves as a mechanism for confirming the system's status. Each node can obtain information about the consensus status of the other nodes and

verify if the system has attained consensus. Each consensus block can act as a checkpoint in the system of alliance chains. After the interaction step is complete, the system can achieve consensus via block synchronization. In comparison with a conventional distributed system, the alliance chain's nodes are more reliable, and the system environment is more stable. Simultaneously, because the enhanced controller node election mechanism and traceability of the alliance chain assure the legality of the block synchronization process, the S-PBFT algorithm simplifies the confirmation procedure.

The S-PBFT method streamlines the confirmation process, which previously needed two-to-two contacts for the controller node to directly ascertain the consensus outcome, resulting in an interaction procedure with an $O(M^2)$ complexity and a reduction in system communication to a certain extent. The algorithm's consensus efficiency is increased by eliminating overhead.

2.2. Symbol Representation and Node Composition

- (a) The set of system nodes is M , represented by $\{1, 2, \dots, j\}$, the maximum tolerated several illegal nodes are g , and the number of nodes j in M should not be less than $3g + 1$. Therefore, considering the system consensus efficiency, j is usually $3g + 1$.
- (b) The set of consensus nodes is H , which is represented by $\{1, 2, \dots, h\}$. To enable the algorithm to reach an effective consensus, combined with the consensus principle of Byzantine agreement, the number h of consensus nodes in the S-PBFT algorithm must not be less than $2g + 1$; the candidate node set is I , represented by $\{1, 2, \dots, i\}$; the primary node set is X , represented by $\{1, 2, \dots, x\}$. The sum of the number of candidate nodes hands the number of prepared nodes x is g .
- (c) In the initial state, the consensus nodes are composed of nodes with a score of 9 and above; the candidate nodes are composed of nodes with scores between 8 and 9; and the reserve nodes are composed of nodes with scores of 8 and below.

2.3. Node Scoring Mechanism. The node scoring mechanism is the core of the improvement idea of the S-PBFT algorithm, which lays the foundation for the optimization of controller node election and consensus protocol. The node scoring mechanism of the S-PBFT algorithm is mainly divided into two parts: the initial node integration and the integration adjustment rules. The two parts are combined to analyze the node scoring mechanism.

2.3.1. Initial Node Integration. In the S-PBFT algorithm, the initial points of nodes are given in the following two cases: the score of the nodes in the system when the system is initialized and the score of new nodes when a new node is added.

- (a) *The score of the node when the system is initialized.* The S-PBFT algorithm proposed in this paper is mainly used in the consortium chain environment. The consortium chain is constructed by a certain number of nodes with everyday demands, such as the interbank consortium chain for settlement problems between banks. In this type of blockchain, there is a particular gap in the comprehensive strength between nodes generally, the more muscular the total strength of the node, the more significant the performance and functional advantages of the node, and the better the node stability.

Combined with the above analysis, the S-PBFT algorithm uses the comprehensive strength of nodes as the scoring basis for the initial score. When scoring, all nodes are sorted according to their total power. Considering the demand for node stability and performance in the consensus process, the sorted nodes are divided into consensus nodes according to the ratio of $2fg + 1$, $g/2$, and $g/2$. All nodes vote candidate nodes and preparatory nodes, and the node order. Among them, the score of some nodes of the consensus node should be greater than 9, the score of some nodes of the candidate node should be between 8 and 9, and the score of some nodes of the reserve node should be less than 8.

In the actual application process, the ratio of candidate nodes and reserve nodes can be adjusted according to the needs of the deployment system. Still, the number of consensus nodes shall not be less than $2g + 1$.

- (b) *The score of the node when a new node is added.* In the consortium chain, the node needs to be authorized by the consortium chain before it can join the consortium chain network. At this time, the node needs to be rated. Authorized nodes also vote the size of node points, and the system divides new nodes into corresponding node sets according to the node points. Under normal circumstances, new nodes are not directly classified as consensus nodes to maintain system stability.

2.3.2. Points Adjustment Rules. In the S-PBFT algorithm, nodes have three different states, and nodes in different forms can be converted, but there are certain conversion conditions. The following rules are formulated here:

- (a) Bonus rules: every time a node completes a consensus process, the node with a score of 9 or more will add 0.01, the node with a score of 8 to 9 will add 0.05, and the node with a score of less than eight will add 0.2. In set H , the node integral is up to 10, and the integral will not increase after reaching 10; in the sets I and X , the integral node increase has no upper limit. There are the following formulas:

$$T = \begin{cases} T_p + 0.01, & T_p \geq 9, \\ T_p + 0.05, & 8 \leq T_p < 9, \\ T_p + 0.2, & T_p \leq 8. \end{cases} \quad (1)$$

T_p is the score before the node executes the consensus, and T is after the node performs the agreement.

- (b) Point deduction rules: in sets H and I , if the node has an error in the consensus process, the node's score is directly reduced to 8, and regardless of the original type of the node, it is now converted into a reserve node; in the set X , the node has an error, node integral minus 0.5.
- (c) Node conversion rules: if there is an error in the consensus node, the faulty node will be converted into a reserve node, and the node with the highest score and higher initial score in the set I will be converted into a consensus node, and the node score will become 9; if there is an error in the candidate node, the faulty node is transformed into a spare node, and the node with the highest score and higher initial score in the set X is converted into a candidate node, and the node score becomes 8.

2.4. Master Node Election. A controller node election is required after system initialization or a Byzantine error on the controller node. In the S-PBFT algorithm, the primary node election takes the node points as the main reference, and the node with the highest points and the higher initial points is used as the primary node. Higher holding points indicate that the function of the node is relatively stable shortly, while higher initial points suggest that the node has better performance. Using such a node as the controller node can ensure the stability of the controller node to the greatest extent and reduce the triggering of view transitions, probability and improve system efficiency. The process of controller node election is briefly described below.

Each node in the set H needs to determine the controller node candidate through pairwise interaction during the controller node election process. During the interaction process, the node i number (i), the integral holding value (T_i), and the initial critical value (T_{ib}) are formed into an array (i, T_i, T_{ib}) , which is used as an election voucher (vote) as shown in Figure 1. Then, the node broadcasts its vote to each consensus node. Comparing the vote received by each node with its vote saves the voice with better conditions. The possible comparison results are as follows:

- (a) If the holding point data in the received vote is smaller than the node's holding point data, its voice will be saved.
- (b) If the holding point data in the received vote is equal to the node's holding point data, continue to compare the initial points. If the initial moments of the received vote are higher, save the received voice; otherwise, keep the own ballot. If the initial issues are

also the same, the franchise with the smaller node number is saved.

- (c) If the holding point's data in the received vote is greater than the node's holding point data, it will save its voice. When the comparison of all nodes is completed, each consensus node adds its signature to the final result $(H(i), h, T_g, T_{gb})$, where $H(i)$ is the signature of node i and h is the number of the pending controller node, followed by broadcasting. The election is completed when there is at least $g + 1$ certificate with the same result. The node corresponding to the node number in the final certificate is the controller node, and the election ends. The election process is shown in Figure 1(b). The figure is only the interaction process between node one and other nodes, and the interaction process between other nodes is similar.

2.5. Consistent Protocol. After the controller node election is completed, each node needs to synchronize the state to make the view number, block height, the previous block's hash value, and other data consistent. Then, the new data block can be consensus; the consensus protocol can be executed. A scoring mechanism is introduced in S-PBFT. Multiple screening of consensus nodes is realized by classifying node types and improving the election method of controller nodes. As a result, the reliability of consensus nodes is guaranteed to the greatest extent. The improved consensus protocol execution process is as follows:

When the transaction volume in the alliance chain reaches a certain amount or reaches a specific time interval, the controller node packages the legal transactions in these transactions to generate a data block. Then, the consensus protocol stage is entered, and the consensus is reached on this transaction block.

- (a) S-pre-prepare stage (S-pre-prepare): the controller node generates an S-prepare message according to the content of the data block $\langle\langle S\text{-PRE-PREPARE}, w, m, I(b)\rangle, b\rangle$. Among them, $I(b)$ is the content summary of the block, that is, the block hash value, and b is the block for this consensus. Then, the controller node sends the S-preparation message to all nodes, in which the consensus node needs to verify the content of the message. If the verification passes, it will enter the next stage; otherwise, the view will be changed, and the controller node will be replaced; the candidate node and the reserve node only receive the message that does not provide feedback on the content of the message.
- (b) S-interaction stage (S-prepare): after the consensus node completes the S-prepare message verification, it needs to generate an S-interaction message $\langle S\text{-PREPARE}, w, m, I(b), i\rangle$, where i is the number of the sending node to broadcast this S-interaction message to all consensus nodes. Next, the consensus node will verify the received S-interaction messages. The next stage can be performed when there are $g + 1$

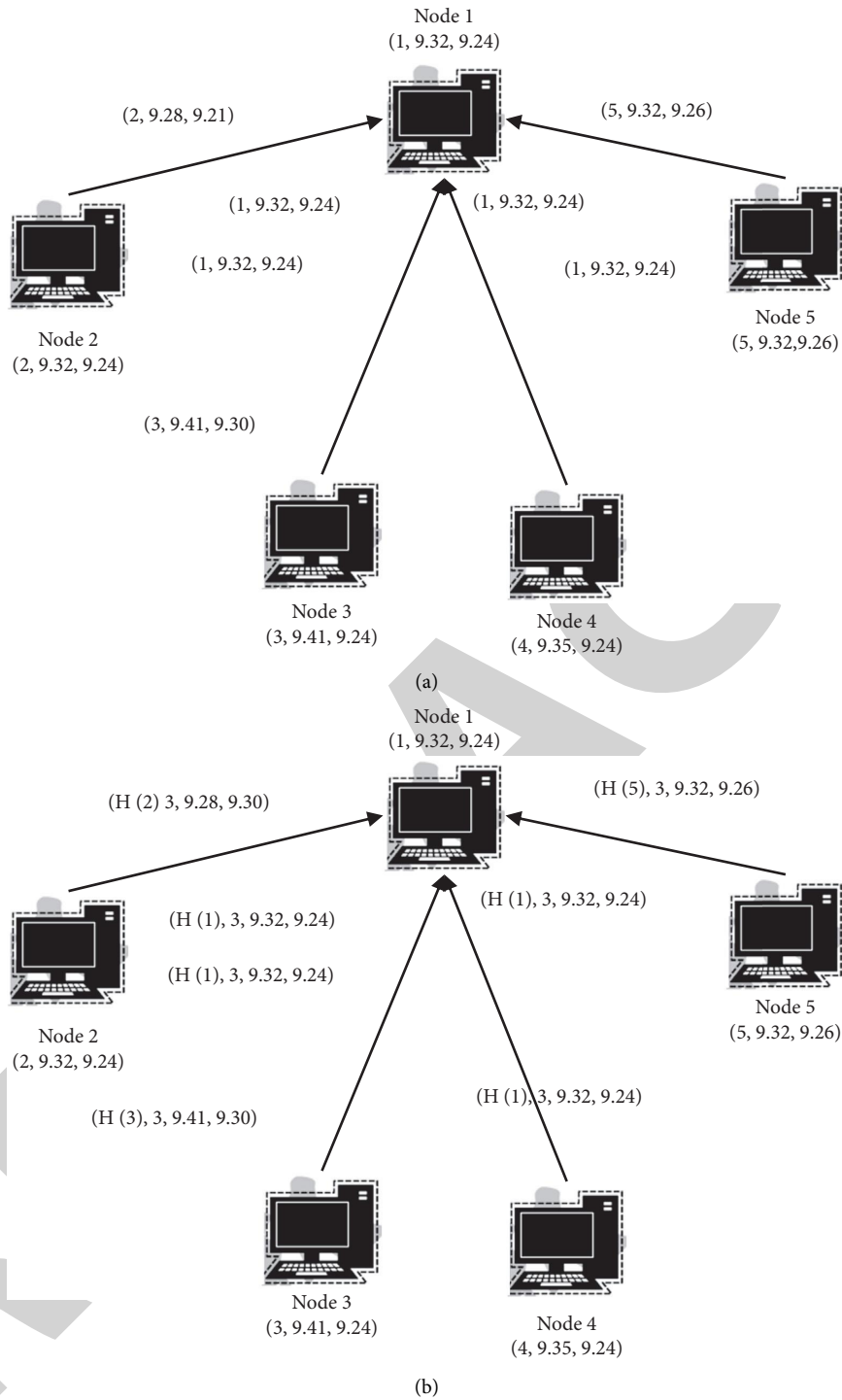


FIGURE 1: Process of master node election: (a) vote interaction process and (b) interaction process of election results.

consistent S-interaction messages from different consensus nodes. Otherwise, the consensus process will be aborted, the problem node will be replaced, and the restart will be performed—the consensus process.

- (c) S-confirmation stage (S-commit): after completing the verification of the S-interaction message, enter the S-confirmation stage. The S-confirmation stage is

mainly used to verify the correctness of the block data saved by each node. Therefore, all nodes, including candidate and reserve nodes, need to send S-confirmation messages to the controller node to ensure that the final block on the chain is correct. The content of the S-confirmation message is $\langle S\text{-COMMIT}, w, m, I(b), H(b)_i \rangle$, where $H(b)_i$ is the signature of node i to block b . When the controller

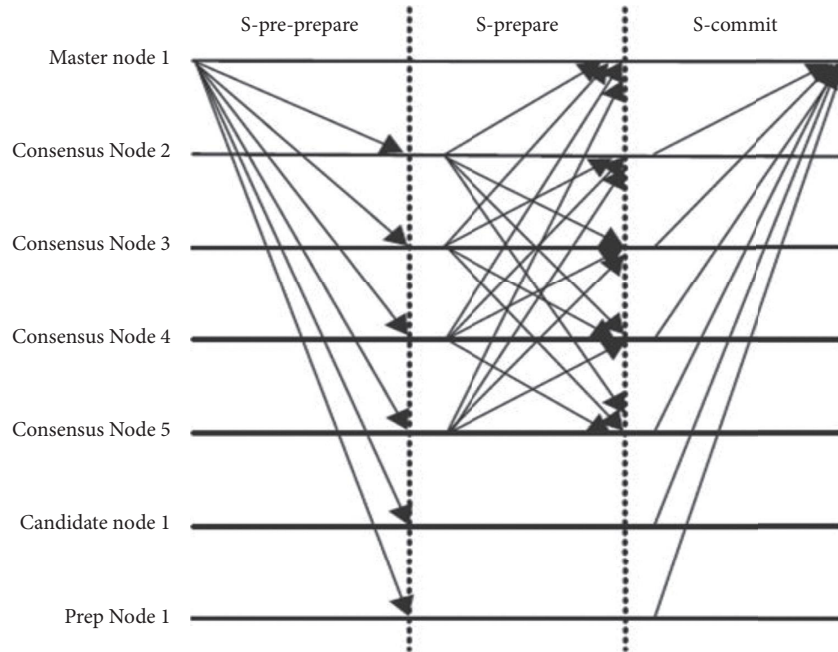


FIGURE 2: Execution flow of the S-PBFT algorithm.

node receives $2g + 1$ S-confirmation messages from different nodes, the consensus is completed, and block b can be saved in the alliance chain. The flow of the S-PBFT algorithm is shown in Figure 2.

3. Experimental Analyses

Through experiments, the S-PBFT algorithm and the PBFT algorithm are tested and analyzed from three aspects of consensus delay, communication overhead, and throughput. In addition, the performance of the two algorithms is compared. In this experiment, a consortium chain environment is built in the internal LAN of the laboratory, and several virtual nodes with the same configuration are set as system nodes through the Docker container. The specific configuration information of the experiment is shown in Table 1.

3.1. Consensus Latency. The consensus delay in a blockchain system refers to the time necessary for a node to submit a transaction request to the system in order to complete the consensus. It is a critical metric for measuring the consensus algorithm's performance. As a result, minimizing the consensus latency can increase the system's operational efficiency and practicability. The total number of system nodes is used as the experimental variable in this experiment. The number of nodes is raised from ten to forty, and the step size is increased from six to six. Multiple transactions are made with varying numbers of nodes, and the average of the other states is used to determine the final value of the state's consensus latency. Figure 3 illustrates the experimental findings in conjunction with Table 2.

TABLE 1: Experiment configuration information.

Object	Configuration information
CPU	Intel i7-7900X
Operating system	Windows 10
RAM	16 GB DDR4
Hard disk	512 GB SSD
Docker container version	Docker Engine 18.05

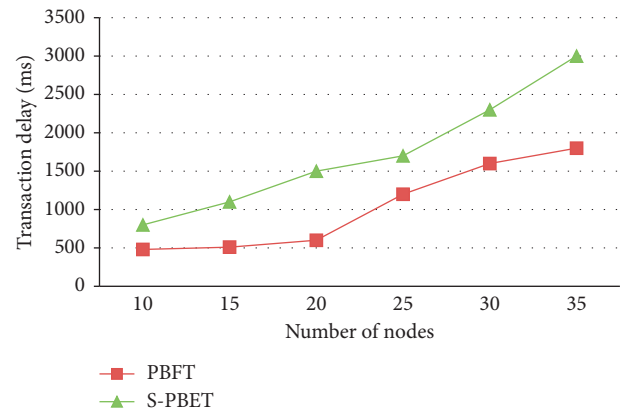


FIGURE 3: Results of consensus delay comparison.

3.2. The Experimental Results. The S-PBFT algorithm outperforms the PBFT algorithm in terms of consensus delay due to the simplification of the confirmation nodes in the consensus protocol when the number of nodes is varied, and that as the number of nodes increases, the S-PBFT algorithm can outperform the PBFT algorithm. The growing rate of the consensus delay is slower, and the stability is better.

TABLE 2: Consensus delay.

Serial	PBFT	S-PBFT
10	480	800
15	510	1100
20	600	1500
25	1200	1700
30	1600	2300
35	1800	3000

3.3. *Communication Overhead.* The S-PBFT algorithm's fundamental concept is to minimize the complexity of the algorithm and increase its efficiency by simplifying the PBFT method's consensus mechanism. As a result, a communication overhead comparison experiment is constructed to evaluate and examine the two algorithms' performance in terms of communication overhead. The communication overhead is defined in this study as the average number of communications required between nodes to accomplish a consensus procedure. Assume the likelihood of changing the system perspective is p .

3.3.1. *The PBFT Algorithm's Communication Overhead.* According to the study of the PBFT algorithm in Section 2.2, the consensus protocol requires $6g(3g+1)$ communications between nodes, and f is the maximum number of Byzantine nodes that the system can accept.

During the transition accept phase, the agent nodes must convey view transition information via pairwise interaction, with a total of $9g^2$ communications. Finally, after successfully changing the view, the controller node must send a view confirmation message to the agent node, with a maximum of $3f$ communications at this time. When the view change probability p is included, the average total communication time for the PBFT algorithm is as follow:

$$D_p = 18g^2 + 6g + p(9g^2 + 3g). \quad (2)$$

3.3.2. *Communication Overhead of S-PBFT Algorithm.* In the consensus protocol, the S-PBFT algorithm only performs three-phase interactions. In the S-preparation phase and the S-confirmation phase, the number of communications is $3f$. The consensus nodes other than the controller node in the S-interaction phase need to broadcast block information to other consensus nodes. The number of transmissions is $4g^2$. Therefore, in the consensus protocol phase, the total number of contacts is $4g^2 + 6g$.

In the controller node election stage, the consensus node first needs to broadcast the election credentials to consensus nodes other than itself, and the number of communications is $2g(2g+1)$. Then, after each consensus node elects the pending controller node, it broadcasts the election result to each consensus node for confirmation, and $2g(2g+1)$ interactions are also performed in this process. Finally, the new controller node sends confirmation information to all consensus nodes, and the number of communication is $2f$.

Therefore, the average total number of communications for the S-PBFT algorithm is as follows:

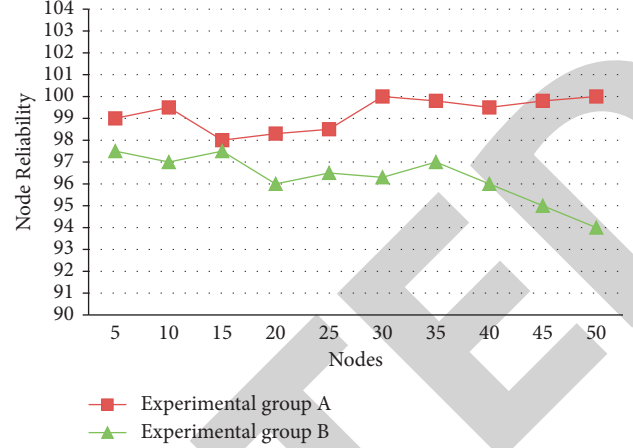


FIGURE 4: Results of the node reliability experiment.

$$D_{T-p} = 4g^2 + 6g + p(12g^2 + 8g). \quad (3)$$

It can be obtained that the ratio Q of the times of communication between the S-PBFT algorithm and the PBFT algorithm is as follows:

$$S = \frac{D_{T-p}4g^2 + 6g + p(12g^2 + 8g)}{D_p18g^2 + 6g + p(9g^2 + 3g)}. \quad (4)$$

The visualization of this formula is obtained through MATLAB, as shown in Figure 4, where p takes values from 0 to 1 with a step size of 0.1 and f takes matters from 3 to 23 with a step size of 2.

As can be seen, regardless of how the values of p and g change, the S value is always less than 1; that is, the S-PBFT algorithm's communication times are always shorter than those of the PBFT algorithm. Additionally, when the number of nodes increases, the S value steadily declines, demonstrating that the S-PBFT method's communication cost is still lower than that of the PBFT algorithm in a multinode scenario. Additionally, the S-PBFT algorithm has a scoring system that screens nodes many times, reducing the likelihood of view transitions. As a result, the S-PBFT algorithm will have a lower communication overhead throughout the actual operation.

3.4. *Throughput.* Throughput refers to the number of events that the system can process per unit of time. In this experiment, the number of events per second (TPS) represents the system throughput. The following formula can express the system TPS:

$$\text{TPS} = \frac{\text{Trade}_{\Delta q}}{\Delta q}. \quad (5)$$

Among them, $\text{Trade}_{\Delta q}$ is the number of events processed by the system within the block generation interval and Δt is the block generation interval.

The total number of nodes was used as a variable in the experiment. The number of nodes was increased from 10 to

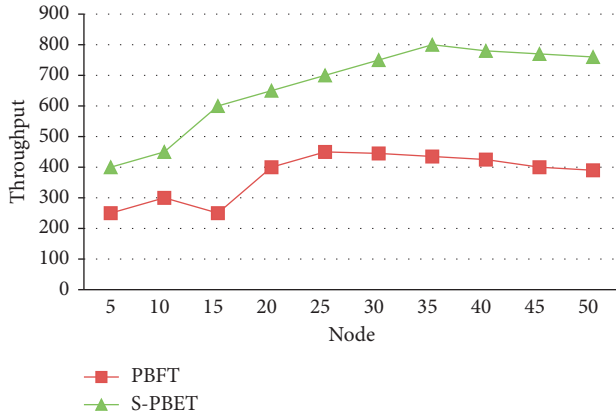


FIGURE 5: Throughput comparison between S-PBFT algorithm and PBFT algorithm.

70, the step size was 6, and 20 repeated experiments were carried out under different nodes. The experimental results of the throughput values under a different number of nodes are shown in Figure 5.

It can be seen from the experimental results that, when the number of nodes is small, and as the number of nodes increases, the throughput of both the algorithms show an upward trend. Therefore, the S-PBFT algorithm has better throughput performance, and data of throughput comparison are shown in Table 3.

3.5. Consensus Node Reliability. The consensus node is the execution node for the S-PBFT algorithm's consensus phase. The dependability of the consensus node directly affects the system's capacity to attain consensus. As a result, the dependability of the consensus node is critical in the S-PBFT method. As seen in the preceding study, the S-PBFT algorithm chooses consensus nodes using both the starting and actual points of nodes.

The initial score of a node embodies the node's comprehensive strength, but the actual score is a complete evaluation of the node's performance in the consensus process over time, which can represent the node's consensus implementation's stability to a degree. As a result, the consensus node found using the preceding technique may have a higher degree of dependability and stability than random node selection.

The dependability of consensus nodes is evaluated in the following comparison tests. The inquiry employs the S-PBFT method and establishes two control groups, A and B: the consensus nodes in group A are picked using node integration guidelines, whereas the consensus nodes in group B are chosen at random from all nodes. The experiment is studied by comparing the algorithm's consensus success rate when the same numbers of nodes are used.

The total number of nodes in the experiment was raised from 10 to 70, with a step size of 6. Numerous tests were conducted with varying numbers of nodes, and the

TABLE 3: Throughput comparison.

Serial	PBFT	S-PBFT
5	250	400
10	300	450
15	250	600
20	400	650
25	450	700
30	445	750
35	435	800
40	425	780
45	400	770
50	390	760

TABLE 4: Node reliability.

Serial	Experimental group A	Experimental group B
5	99	97.5
10	99.5	97
15	98	97.5
20	98.3	96
25	98.5	96.5
30	100	96.3
35	99.8	97
40	99.5	96
45	99.8	95
50	100	94

algorithm's consensus success rate was determined for each node count. In both the groups of studies, processing power, bandwidth, and other node resources are given randomly. The experimental findings are summarized in Figure 4 with the accompanying Table 4.

As can be observed from the experimental data, the consensus success rate of experimental group A has always stayed around 98.5 percent as the number of nodes increases, but the consensus success rate of practical group B shows a general negative trend as the number of nodes increases. The consensus nodes in experimental group A are chosen based on the node points and beginning points of the nodes. As a result, the consensus nodes can always be assured to have greater computational power, greater bandwidth, and greater stability. The performance disparity between consensus nodes in experimental group B, on the other hand, is rather considerable. Due to the system's limited resources, as the number of nodes increases, the nodes are assigned more minor and less resources, leading in a drop in the consensus success rate. Nonetheless, it is clear that by dynamically adjusting nodes, the trustworthiness of consensus nodes may be increased.

4. Conclusion

With the further growth of the blockchain, academics have focused their efforts on improving the consensus algorithm. The quality of the consensus algorithm has a direct impact

on the blockchain's performance and security. This article develops an improved S-PBFT method based on the PBFT technique. Nodes are classified into three groups by the S-PBFT algorithm: consensus nodes, candidate nodes, and reserve nodes. The scoring system ensures the highest degree of trustworthiness for consensus nodes. Additionally, the S-PBFT algorithm improves the consensus protocol and its execution process, decreases the method's complexity while maintaining its functionality, and increases operational efficiency. However, the S-PBFT algorithm is not without flaws. The following work will develop the node scoring mechanism in conjunction with specific application situations, create a dynamic network structure, increase the system's flexibility, and improve the system's practicability.

Data Availability

The data shall be made available on request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University for funding this work through Research Group no. RG-21-07-09.

References

- [1] S. Gao, T. Yu, J. Zhu, and W. Cai, "T-PBFT: an Eigen Trust-based practical Byzantine fault tolerance consensus algorithm," *China Communications*, vol. 16, no. 12, pp. 111–123, 2019.
- [2] L. Chen and W. Zhou, "Byzantine Fault tolerance with window mechanism for replicated services," in *Proceedings of the 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, pp. 1255–1258, Qinhuangdao, China, September 2015.
- [3] K. Lei, Q. Zhang, L. Xu, and Z. Qi, "Reputation-based Byzantine Fault-tolerance for consortium blockchain," in *Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 604–611, Singapore, December 2018.
- [4] W. Zhao, "Application-aware Byzantine Fault tolerance," in *Proceedings of the 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, pp. 45–50, Dalian, China, August 2014.
- [5] H. Wang and K. Guo, "Byzantine Fault tolerant algorithm based on vote," in *Proceedings of the 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 190–196, Guilin, China, October 2019.
- [6] S. Kim, S. Lee, C. Jeong, and S. Cho, "Byzantine Fault tolerance based multi-block consensus algorithm for throughput scalability," in *Proceedings of the 2020 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–3, Barcelona, Spain, January 2020.
- [7] L. He and Z. Hou, "An improvement of consensus fault tolerant algorithm applied to alliance chain," in *Proceedings of the 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 1–4, Beijing, China, July 2019.
- [8] W. Zhou and L. Chen, "Formal specification for Byzantine Fault tolerant algorithm suiting web services," in *Proceedings of the 2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing*, pp. 35–38, Wuhan, China, October 2011.
- [9] Z. Zhang, D. Zhu, and W. Fan, "QPBFT: practical Byzantine Fault tolerance consensus algorithm based on quantified-role," in *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 991–997, Guangzhou, China, January 2020.
- [10] W. Jiang, L. Chen, Y. Wang, and S. Qian, "An efficient Byzantine fault-tolerant consensus mechanism based on threshold signature," in *Proceedings of the 2020 International Conference on Internet of Things and Intelligent Applications (ITIA)*, pp. 1–5, Zhenjiang, China, November 2020.
- [11] A. Song, J. Wang, W. Yu, Y. Dai, and H. Zhu, "Fast, dynamic and robust Byzantine Fault tolerance protocol for consortium blockchain," in *Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 419–426, Xiamen, China, December 2019.
- [12] S. Sakho, J. Zhang, F. Essaf, K. Badiss, T. Abide, and J. K. Kiprof, "Research on an improved practical byzantine fault tolerance algorithm," in *Proceedings of the 2020 2nd International Conference on Advances in Computer Technology, Information Science and Communications (CTISC)*, pp. 176–181, Suzhou, China, March 2020.
- [13] R. Kashyap, K. Arora, M. Sharma, and A. Aazam, "Security-aware ga based practical Byzantine Fault tolerance for permissioned blockchain," in *Proceedings of the 2019 4th International Conference on Control, Robotics and Cybernetics (CRC)*, pp. 162–168, Tokyo, Japan, September 2019.
- [14] L. Zhang and Q. Li, "Research on consensus efficiency based on practical Byzantine Fault tolerance," in *Proceedings of the 2018 10th International Conference on Modelling, Identification and Control (ICMIC)*, pp. 1–6, Guiyang, China, July 2018.
- [15] S. Tian, Y. Liu, Y. Zhang, and Y. Zhao, "A Byzantine Fault-tolerant Raft algorithm combined with schnorr signature," in *Proceedings of the 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pp. 1–5, Seoul, Korea (South), January 2021.
- [16] S. Bonomi, A. Del Pozzo, M. Potop-Butucaru, and S. Tixeuil, "Approximate agreement under mobile byzantine faults," in *Proceedings of the 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 727–728, Nara, Japan, June 2016.