Journal of Computer and Communications, 2023, 11, 20-38
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Improved PBFT Consensus Algorithm Based on Node Role Division

Xiyu Ren, Xiangrong Tong, Wei Zhang

School of Computer and Control Engineering, Yantai University, Yantai, China
Email: rxy_ytu@163.com, xr_tong@163.com

How to cite this paper: Ren, X.Y., Tong, X.R. and Zhang, W. (2023) Improved PBFT Consensus Algorithm Based on Node Role Division. *Journal of Computer and Communications*, **11**, 20-38.
https://doi.org/10.4236/jcc.2023.112003

Received: January 16, 2023
Accepted: February 21, 2023
Published: February 24, 2023

Abstract

The PBFT (Practical Byzantine Fault Tolerance, PBFT) consensus algorithm, which addressed the issue of malicious nodes sending error messages to disrupt the system operation in distributed systems, was challenging to support massive network nodes, the common participation over all nodes in the consensus mechanism would lead to increased communication complexity, and the arbitrary selection of master nodes would also lead to inefficient consensus. This paper offered a PBFT consensus method (Role Division-based Practical Byzantine Fault Tolerance, RD-PBFT) to address the above problems based on node role division. First, the nodes in the system voted with each other to divide the high reputation group and low reputation group, and determined the starting reputation value of the nodes. Then, the mobile node in the group was divided into roles according to the high reputation value, and a total of three roles were divided into consensus node, backup node, and supervisory node to reduce the number of nodes involved in the consensus process and reduced the complexity of communication. In addition, an adaptive method was used to select the master nodes in the consensus process, and an integer value was introduced to ensure the unpredictability and equality of the master node selection. Experimentally, it was verified that the algorithm has lower communication complexity and better decentralization characteristics compared with the PBFT consensus algorithm, which improved the efficiency of consensus.

Keywords

Blockchain, Consensus Algorithm, Practical Byzantine Fault Tolerance, Node Role Division

1. Introduction

The distributed ledger technology known as blockchain makes it possible to store data consistently. Blocks are used to store data, and they are all connected to create an immutable chain. All information in this chain can be tracked and verified but is difficult to fiddle with. It offers many benefits like decentralization, lack of tampering, lack of falsification, and traceability [1]. Realistic applications like crucial forgery-proof data storage and data protection are particularly well suited to blockchain [2] [3]. Blockchain technology can solve the security problems such as tampering data and low transaction processing efficiency of traditional centralized backing institutions [4] [5] [6] in the fields of finance [7] [8] [9], healthcare [10], Internet of Things [11], property rights protection [12], and privacy protection [13], etc. When Satoshi Nakamoto published "Bitcoin: A Peer-to-Peer Electronic Cash System" in 2008, he first revealed the idea of bitcoin. This also marked the beginning of blockchain technology.

One of the key components of the blockchain system is the consensus algorithm, which is essential to the effectiveness, security, and stability of blockchain data processing [14]. Blockchain consensus algorithms constantly pursue high performance, low energy consumption, excellent security, and scalability on the basis of ensuring algorithm security, scalability, resource utilization, and process efficiency sex. One of the major blockchain technology research axes has been consensus algorithms. The common consensus algorithms are PBFT (Practical Byzantine Fault Tolerance, PBFT) [15], XFT [16], PoW (Proof of Work, PoW) [17], Paxos [18], and Zyzzyva [19]. In the federated chain, the PBFT consensus algorithm is a popular consensus method that can reduce algorithm complexity from exponential to polynomial levels [20]. Only when less than one-third of the total nodes are malicious can the PBFT consensus method successfully reach consensus. However, malicious nodes can slow down the block generation speed and thus reduce the throughput of the system. Currently, the PBFT consensus algorithm has done a lot of research on the selection of consensus nodes and the reduction of the communication complexity of the system.

For the selection of consensus nodes of the PBFT consensus algorithm, avoiding the master node as a malicious node and reducing the participation of anomalous nodes in consensus are key issues. WANG [21] proposed a novel consensus protocol called CDBFT (Credit-delegated Byzantine fault tolerance, CDBFT), which sets up a voting reward and punishment scheme. In the consensus process, motivate the reliable nodes and reduce the participation probability of abnormal nodes to 5%. ZHENG [22] combined PBFT with the improved C4.5 decision tree algorithm to classify nodes using the improved C4.5, introduced weighted average information gain to improve the classification accuracy, and introduced a point voting mechanism to determine the master node. CHANG [23] proposed a blockchain two-layer architecture model consisting of two parts, the bottom cluster and the top cluster. This model enhances consensus by using bandwidth-reserved multi-entry point operations, and eliminates the reliance on a single primary leader characteristic of PBFT. The above documents solve the problem of insufficient scalability of the original PBFT consensus algorithm, but ignore the evaluation of the trust of the master node.

To address the problem of excessive communication complexity in the consensus process: GAO [24] proposed T-PBFT, the Eigen Trust model is combined with the PBFT consensus algorithm, the credit of the node is evaluated by the transaction between nodes, and the node with a high reputation is selected as its trust group. TONG [25] proposed a practical Byzantine consensus algorithm based on Peer Trust, introducing Peer Trust to evaluate the credibility of nodes eligible to participate in consensus, which can improve the fault tolerance of the system, and devised a three-chain structure to retain assets, transactions, and feedback in three blockchain systems, separately. According to the network environment of the system, LI [26] designed an EPBFT consensus algorithm that takes different steps to reach consensus. He also added verifiable random functions (VRF) to choose consensus nodes, streamlined checkpointing protocols and cut down on the time and communication overhead required for view change protocols. However, in the process of consensus, the main node receiving the client request message may be the Byzantine node, which leads to the reduction of the efficiency of the system consensus.

To address the problems of the PBFT consensus algorithm, reducing the complexity of communication, the nodes in the system play different roles, and the system's nodes are separated into high reputation value groups and low reputation value groups according to the results of node mutual voting. The nodes in the high reputation value group are divided into consensus nodes and supervisory nodes, and the nodes in the low reputation value group are backup nodes. For the randomness of master node selection and to avoid centralization of power, an adaptive method is used to select the master node with a higher reputation value as the master node in the consensus process to improve the reliability of the system.

The remainder of the essay is structured as follows. Work related to the PBFT consensus algorithm is reviewed (Section 2). The PBFT consensus algorithm based on role division is proposed to divide the nodes in the network. It is described in detail in four aspects: calculating the initial reputation value of the nodes, the preparation phase, the consensus phase, and the ending phase (Section 3), simulation experiments are analyzed and validated, compared with the initial PBFT (Section 4), the full paper is summarized (Section 5).

2. Related Work

This section gives a quick overview of the PBFT consensus algorithm's body of knowledge.

2.1. The Problem of Byzantine Generals

At present, the consensus problems studied by most experts and scholars mainly include two kinds, algorithmic consensus and decision consensus. These two kinds of consensus problems originally originated from the Byzantine general problem, and the consensus of the blockchain system is also based on the Byzantine general problem.

The Byzantine general's problem is that generals in the Byzantine Empire lead their respective armies in a siege of an enemy city. Because the armies are far apart, each general can only control his own army, and the only way to negotiate a reasonable battle plan and reach a consensus is to transmit messages to the other generals via messengers. However, there could be traitors among the generals who tried to deceive the other loyal generals by spreading false plans of action during the transmission of messages. The Byzantine general problem is to find a workable algorithm that allows the loyal generals to reach a consensus despite the presence of a few traitors.

2.2. PBFT Consensus Algorithm

The practical Byzantine fault-tolerant consensus algorithm can be applied in an asynchronous network [27] environment, and PBFT is the most adopted consensus algorithm in federated chains, which can guarantee replication across different nodes in a distributed system. In the PBFT consensus algorithm, the existence of Byzantine nodes is allowed, but Byzantine nodes cannot be identified, and the consensus is reached as long as the system receives more than half of the confirmation messages.

The Practical Byzantine Fault Tolerance algorithm is a state-as-replica-based fault tolerance algorithm [28], which can provide fault tolerance guarantees for no more than $(n - 1)/3$ failed nodes. To protect the distributed system from malicious users, the roles of nodes in the consensus process are divided into the client, master, and replica nodes. The core of the PBFT consensus algorithm is composed of consistency protocol, checkpointing protocol, and view change protocol. The network system executes only the consistency protocol and checkpointing protocol under normal operation. When the master node is in error or the system is running slowly, the view replacement protocol is performed to maintain the system able to continue responding to client requests.

2.3. PBFT Consensus Algorithm Consensus Process

The workflow of the PBFT consensus algorithm when no master node fails is shown in **Figure 1** [29]. A completely unanimous opinion needs to be requested, pre-prepared, and prepared; among them, pre-preparation, preparation, and submission are the key links to consensus.

Request phase: The client sends a request message to the master node with information such as a timestamp. The timestamp ensures that the request will not be executed repeatedly, and as the timestamp t value increases, the order of execution of the operation can be identified.

Pre-preparation phase: The host responds to the request sent by the client, tests the request and specifies the number, and then broadcasts to other replica nodes.

Preparation phase: The node receives the pre-preparation information sent by the master node, confirms its correctness, and broadcasts the pre-preparation information to other nodes after confirmation.

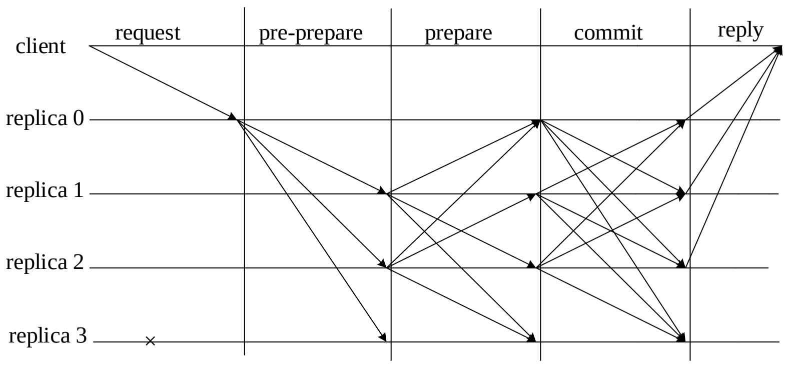


Figure 1. PBFT algorithm consensus process.

Confirmation phase: When the node receives at least $2f$ pieces of information (f is the number of Byzantine nodes in the system) and performs verification, it broadcasts confirmation information to the entire network. Replica nodes send acknowledgments to other replica nodes. When a node in a system receives a $2f+1$ valid reply message, it can send a reply message to the client.

Reply phase: After the replica nodes execute the above operations, the replica nodes send reply messages to the clients, and when the clients receive $2f+1$ independent return results, they are considered to have reached a consensus.

3. PBFT Consensus Algorithm with Role Division

On this basis, we designed a PBFT consensus algorithm based on node role division to guarantee the security of the protocol, decrease the communication complexity of such consensus process and enhance consensus efficiency of the consensus process by voting among nodes, dividing high-reputation value and low reputation value groups, setting supervisory nodes for node role division and selecting trusted master nodes.

3.1. The Flow of the Consensus Algorithm

For node group consensus, although the efficiency is improved compared to the initial consensus. As the number of groups reaches the maximum, the number in the consensus group will also increase, because carrying out the group consensus at the same time still also leads to an increase in the number of exchanges. In response to the above problems, the algorithm is improved to divide the nodes in the system into roles, and the roles played by the nodes determine their authority in the system, which are divided into a total of three categories consensus nodes, backup nodes, and supervisory nodes, only consensus nodes are involved in consensus, which reduces the number of nodes participating in the consensus process, backup nodes and supervisory nodes can provide a guarantee for the honesty of the nodes in the system, and each of the three types of nodes has its own role so that consensus can be reached quickly in a reliable environment.

The flowchart of the consensus algorithm is shown in **Figure 2**. The method proceeds round by round, and each phase is divided into three phases: the preparation phase, the consensus phase, and the final phase. Before the first consensus, all nodes in the system need to vote.

3.2. Node Initial Credit

The initial reputation value of a node is determined by voting among the nodes. Most of the existing approaches will determine the number of votes through mutual communication between nodes to obtain their reputation value by calculating locally. Setting an initial bookkeeping node before nodes vote each other can centralize all voting records in one node, avoiding complicity among nodes and ensuring the authenticity of voting results. The credit will change with the performance of the nodes in the system, and the size of the credit will affect the position and authority of the node in the blockchain. The following are the specific steps.

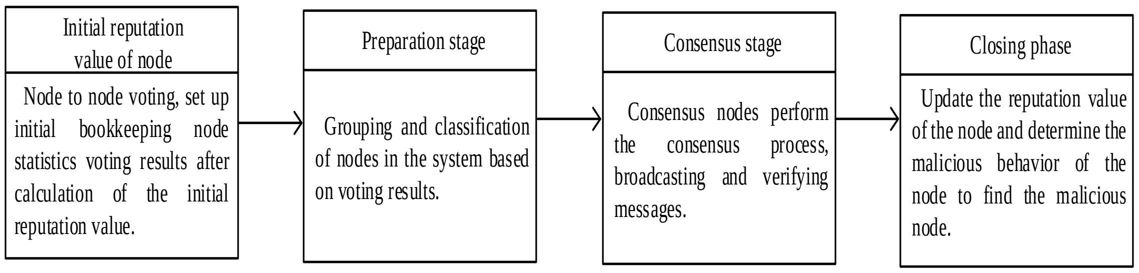
1) Select the initial bookkeeping node. Each node is assigned a number 1, 2, 3… N in the order of entering the system, and then the system generates a random number from 1 to N. The node whose number is consistent with this random number is the initial bookkeeping node.

2) Voting. A voting cycle is set in which a trust list node can be voted for. The nodes in the trust list are automatically classified in the high reputation value group. In the voting cycle "VT", all system nodes must participate in the voting. When the voting cycle "VT" ends, the votes of each node i are counted as C_i, and the voting results are copied twice, one is written to the local log, and the other is sent to the bookkeeping node, which records the initial reputation value. The top H nodes are sorted according to the number of node votes, and the top H nodes are composed into a list of nodes with high reputation value. As shown in Figure 3.

The initial bookkeeping node aggregates all the voting results and counts the pro votes $A_i\left(i=1,2,3,\cdots,N\right)$ and con votes O_i obtained by each node, and then calculates the final initial reputation value of each node according to Equation (2). Finally, the initial value of each node is sent to the corresponding node.

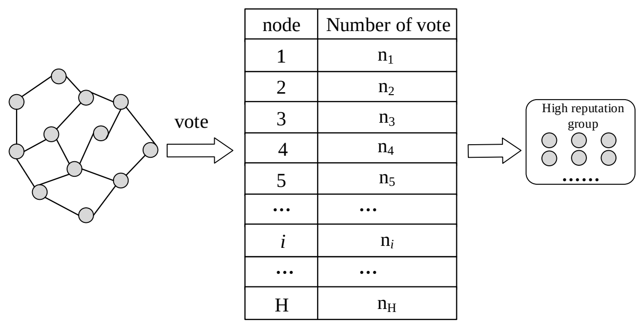
$$C_i = A_i + O_i \tag{1}$$

$$Trust_i = \frac{A_i - A_{\min}}{A_{\max} - A_{\min}} \tag{2}$$



Initial reputation value of node
Node to node voting, set up initial bookkeeping node statistics voting results after calculation of the initial reputation value.
Preparation stage
Grouping and classification of nodes in the system based on voting results.
Consensus stage
Consensus nodes perform the consensus process, broadcasting and verifying messages.
Closing phase
Update the reputation value of the node and determine the malicious behavior of the node to find the malicious node.

Figure 2. RD-PBFT algorithm flow.



node	Number of vote
1	n_1
2	n_2
3	n_3
4	n_4
5	n_5
…	…
i	n_i
…	…
H	n_H

Figure 3. Voted trust list node.

3.3. Preparatory Stage

In the preparation stage, according to the initial credit value of the node, it is divided into high credit level and low credit level. The nodes are divided into high-confidence groups, and nodes are divided into two types: consensus nodes and supervision nodes. The nodes in the low credit group become backup nodes. As shown in Figure 4.

3.3.1. Node Role Delineation

The nodes in the system are divided into three categories, consensus nodes, supervisory nodes, and backup nodes. The three types of nodes take different roles in the system, play different roles, supervise each other and form a whole. This ensures that the consensus is reached smoothly in a safe and reliable environment. The model diagram of node division in the system is shown in Figure 5.

Consensus nodes. Consensus nodes consist of two parts, primary and replica, which participate in the consensus process. The master node is responsible for confirming and receiving the request information sent by the client, while the copy node is responsible for confirming the information of the master node and other copy nodes, so as to ensure the consistency of the system.

Supervisory nodes. The main role of supervisory nodes is to provide a reliable guarantee for the consensus process of the system and ensure that the consensus can be reached in a safe and reliable environment. Supervisory nodes have the right to view the local logs of all other local nodes. Supervisory nodes supervise the behavior of the whole nodes in the system and update the reputation value of the nodes after each round of consensus.

Backup nodes. The nodes within the low reputation value group become backup nodes, which do not participate in the consensus process of the system and passively update their local information according to the information sent by the master node to reach synchronization. The backup nodes in the system are also able to supervise the supervisory nodes.

3.3.2. Selection of Master Node and Supervisory Node

Consensus nodes include a master and a copy of the consensus process participating in the system, only one of the consensuses can participate in the consensus and compete, and only one of the others can vote. By storing data, communication overhead can be reduced and system performance improved. The algorithm adopts an adaptive master node selection algorithm, by randomly selecting nodes with a high reputation as the master node, and reducing the concentration of authority by randomly selecting the master node randomness. Among the high-credit groups, monitoring nodes get the most votes. Assume that the credit value of the nodes in the group is *m*.

Step 1: The probability of selecting the node with a high reputation value to be elected as the master node is calculated among the consensus nodes, and the formula for calculating the probability of consensus node *i* to be the master node is as follows.

$$P(i,m)=\frac{Trust(i,m)^{\lambda}}{\sum_{j=1}^{C} Trust(j,m)^{\lambda}} \tag{3}$$

where C is the total number of consensus nodes, j denotes a replica node, and the index is calculated as follows.

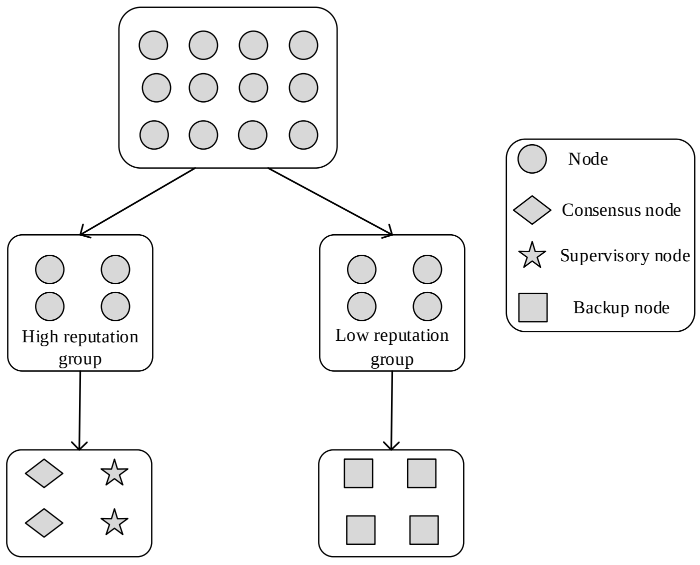


Figure 4. Diagram of node classification.

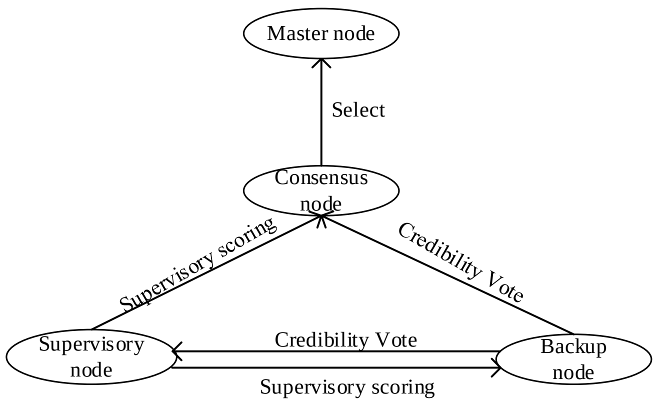


Figure 5. Improved consensus algorithm node model.

$$\lambda = \frac{c}{\frac{Trust_{\max}}{\overline{Trust}} - 1} \tag{4}$$

where $Trust_{\max}$ and $\overline{Trust}$ are the maximum and average of the reputation values of all consensus nodes, respectively, and a constant.

Step 2: Each consensus node is sorted from lowest to highest according to its probability value of being elected as the master node (if the nodes have the same calculated probability value, then the consensus nodes are sorted from smallest to largest), and the cumulative probability of each consensus node is calculated in this order with the following formula.

$$s_i = \sum\nolimits_{j-1}^{i} p(j, m) \tag{5}$$

Step 3: In order to guarantee the randomness of the master node selection, a random number RN in the range [0, 1] needs to be found, and the RN is calculated by hashing the block header of the current block using the SHA256 algorithm with the following formula.

$$RN = StrToInt\left(SHA256\left(block_{head}\right)\right) mol\ N \tag{6}$$

Subsequently, the cumulative probability of each consensus node is used to determine the master node for the next round by the following formula, and the cumulative probability of the master node needs to satisfy the following condition.

$$s_{i-1} < \min\left(1, \frac{RN}{C} + \tau\right) \le s_i \tag{7}$$

where τ is a constant between [0, 1], which can be adjusted to ensure that nodes with high cumulative probability values, *i.e.*, nodes with high reputation values, have a higher probability of being elected as master nodes.

3.4. Consensus Stage

The schematic diagram of the consensus phase of the improved algorithm is shown in **Figure 6**. The original algorithm of PBFT involves the whole network nodes in consensus. Among N nodes, about $2N^2$ communications are required to reach consensus. As the scale of network nodes increases, the number of network communications will increase sharply, which will cause network communication overhead and affect the overall performance of the network.

In order to reduce the communication overhead of the network, the supervisory nodes will complete the verification work in the consensus process. Since the group is divided at the beginning and the one with high trust is selected as the consensus node, the consensus nodes involved in the consensus process can be considered honest nodes with high probability. During the consensus process, each node can make its own judgment, and the master node aggregates the judgments made by other nodes and then submits the final conclusion to the client. The consensus protocol flow is shown in **Figure 7**, where c denotes the client, the monitor node denotes the supervisor node, the backup node denotes the backup node, P denotes the master node, and 1, 2, and 3 denote the replica nodes.

The consensus process is similar to the original PBFT consensus algorithm, except that the supervisory node and the backup node are added. The supervisory node supervises the behavior of the nodes in the consensus process, supervises the information update of the backup node, and is responsible for updating the node reputation value after each circular of consensus. The backup node can only passively receive information, and the master node transmits the consensus result message to the client while the backup node updates its own local information based on the information passed by the master node.

3.5. Closing Stage

The work in this phase is mainly to complete the updating of node reputation values, packaging of data, and malicious node logging.

3.5.1. Node Calculation of Node Reputation Value

The reputation value of a node is largely determined by the performance of the node in this consensus phase.

Definition 1 The degree of activity of a node in participating in consensus. The degree of activity is the number of times a node participates in consensus within a certain time frame, and the activity of a node can be expressed by the function $\rho(n)$, which is calculated as follows.

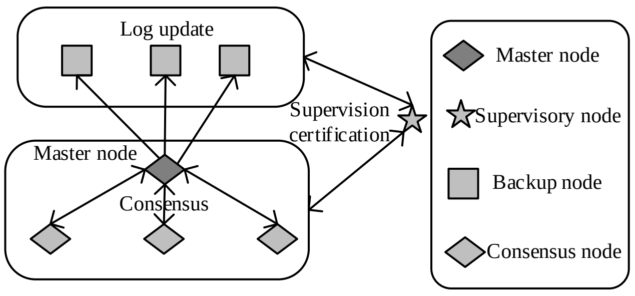


Figure 6. Consensus stage diagram.

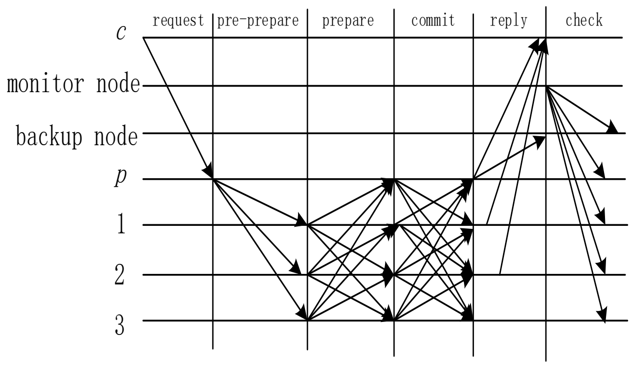


Figure 7. Improved consensus flow of the algorithm.

$$\rho(n) = \mathrm{e}^{-\frac{a}{n}} \tag{8}$$

where n is the number of nodes participating in the consensus, and the parameter a ($a \in \mathrm{Z}$, $a \geq 1$) is adapted to regulate the development rate. The value of $\rho(n)$ becomes larger as the value of n increases.

Definition 2 Success rate of node consensus. The success rate of node consensus is the frequency of nodes participating in consensus to reach consensus successfully. It is represented by the consensus success rate y. The larger the value of y, the better the performance, and it is calculated as follows.

$$y = \sqrt{\frac{s}{n}} \tag{9}$$

where s is the number of normal consensus completions, and n is the number of nodes participating in consensus.

Definition 3 Historical influence degree. Historical influence degree is the degree of influence of the historical trust value of a node on the trust value of the current node. It can be expressed by $\omega(\Delta t)$ and calculated as follows.

$$\omega(\Delta t) = \left(\frac{1}{2}\right)^{\frac{\Delta t}{\lambda}} \tag{10}$$

where Δt is denoted as the time interval between the current time and the last consensus time, the parameter λ ($\lambda \in \mathrm{Z}$, $\lambda > 0$) is able to adjust the degree of influence. The value of $\omega(\Delta t)$ decreases as Δt increases, and the smaller the value obtained from the calculation, the smaller the influence of the historical trust value.

Definition 4 Transaction Impact Factor. The transaction impact factor identifies the importance of a transaction. Let the transaction importance parameter be m. The transaction impact factor calculation function $F(m)$ is calculated as follows.

$$F(m) = \begin{cases} \dfrac{m}{M_0} & m < M_0 \\ 1 & m \geq M_0 \end{cases} \tag{11}$$

where M_0 is the threshold value of the transaction importance parameter, the value of $F(m)$ increases as the value of m increases, and a larger $F(m)$ indicates higher transaction importance.

Definition 5 Behavioral evaluation. Behavioral evaluation refers to the corresponding evaluation value given to a node based on its performance in the consensus process. The behavioral evaluation value E is calculated as follows.

$$E = \frac{1}{2^{(c \times t)}} \times \frac{g}{N} \tag{12}$$

where t is the time taken by nodes to complete the consensus process, c ($c \in \mathrm{Z}$, $c \geq 1$) is the evaluation value adjustment factor, g is the number of nodes agreeing to the message, and N is the total number of nodes. The shorter the time it takes for a node to a total agreement when most nodes agree on the message, the higher the evaluation.

Definition 6 The combined belief value of a node. It is calculated as follows.

$$Trust_{ij} = \rho(n)\gamma \frac{\sum_{l=1}^{n} E_l \omega(\Delta t_l) F_l}{\sum_{l=1}^{n} \omega(\Delta t_l) F_l} + (1-\rho(n)) Trust_i \quad (13)$$

where $Trust_i$ is the initial value of trust value, the trust value of a node is calculated precisely by combining the performance of nodes in the consensus process.

3.5.2. Dynamic Update of Credit Value

On this basis, a dynamic credit evaluation model based on node behavior is proposed. The model includes the increase or decrease of credit and the status of setting credit value. After each negotiation is reached, the credit value of the node will be updated according to the credit model, and then the role of the node will be determined according to the credit value $Trust_i$. The credit rating is the credit rating of a node, the higher the credit rating, the higher its credit rating. The initial reputation value of the node is set as $Trust_i \in [0,1]$.

1) The increment or diminish of reputation value. Different nodes have different behaviors in the consensus process, and the reputation value of nodes is set dynamically according to the behavior of nodes. Let $Trust_i(r)$ denote the reputation value of node *i* in the *r*th round of consensus, then in the *r* + 1th round of consensus, the reputation value of node *i* can be expressed by the following Equation (14).

$$Trust_i(r+1) = \begin{cases} Trust_i(r) + \alpha(1 - Trust_i(r)) & \text{honest node} \\ \beta Trust_i(r) & \text{anomaly node} \\ Trust_i(r) e^{-\lambda \Delta h} & \text{offline node} \\ 0 & \text{byzantine node} \end{cases} \quad (14)$$

If node *i* generates a block in *r*+1 rounds of consensus by packing and synchronizing the messages of nodes across the network and reaches consensus successfully, then $Trust_i(r+1) = Trust_i(r) + \alpha(1 - Trust_i(r)), \alpha \in (0,1)$. The coefficient α is used to regulate the rate at which the node reputation value grows, and its value of it can be chosen in accordance with the system's particular application requirements. When α is set to steady, the larger the reputation worth $Trust_i(r)$ is, the slower the development rate of $Trust_i(r+1)$ will be, and eventually it will converge to 1. After each circular of consensus, based on the performance of the nodes in the system at the current consensus stage and the historical behavior of the nodes, the supervising node will update the reputation value of the nodes, and the reputation value of the nodes will be updated according to Equation (14), which is sent to the nodes in the system by Equation (15). The other nodes receive the updated reputation value message from the supervisory node and verify it, and after the verification, the supervisory node updated its own reputation value according to the verification results of other nodes. After all, the reputation values of all nodes in the system are updated, the

Byzantine nodes that appear in the current consensus phase are numbered, and the Byzantine nodes that are recorded cannot participate in the consensus process; they play the role of backup nodes in the system and can only passively update their local information based on the information sent by the master node.

$$\langle Trust_i, Trust_i', \text{conduct}, \text{character} \rangle \tag{15}$$

When node i packs invalid blocks or node i is inconsistent with other nodes' messages across the network in $r + 1$ rounds of consensus, $Trust_i(r+1) = \beta Trust_i(r)$, where $\beta \in (0,1)$, as a penalty factor, controls the rate of reputation value decline. When node i is an abnormal node, its reputation value will drop linearly. **Algorithm 1** shows the updating of credit values.

The credit value of a node in a system will gradually decline if it is offline for an extended period of time, which means it is not taking part in the consensus process. λ is the decay coefficient, and Δh is the variation between the height of the block and the height of the current block when node i last participated in the consensus process.

Algorithm 1. Update of credit value.

```
Input ( $Trust_i$, $N_i$, $\alpha$, $\beta$ )
Output ( $Trust_i$, $N_i$ )
1) for $i \in N$
2) if complete this round of consensus
3) $Trust_i \leftarrow Trust_i(r) + \alpha(1 - Trust_i(r))$
4) else if Malicious behavior
5) $Trust_i \leftarrow 0$
6) else $Trust_i \leftarrow \beta Trust_i(r)$
7) if $Trust_i(r) < n$
8) $STrust_i(r)$ ←error
9) $Trust_i(r) \leftarrow n$ recover in the next cycle
10) else if $Trust_i(r) < l$
11) $STrust_i(r)$ ←abnormal
12) else if $Trust_i(r) < m$
13) $STrust_i(r)$ ←normal
14) else
15) $STrust_i(r)$ ←excellent
16) if $Trust_i(r) > m$
17) $Trust_i(r) \leftarrow l$ reset at next cycle
18) return ( $Trust_i$, $N_i$ )
end for
```

If node i intentionally sends an error message to other nodes, it is considered a Byzantine node in the system, and the reputation value of node i is 0, and it cannot participate in the consensus process

2) Reputation value state setting is determined by the node's reputation status $Trust_i$, and the node's authority is set in accordance with the node's reputation status. Table 1 displays the four different types of node reputation states are given in the table, and m, l, and n are indicated as the thresholds for reputation state updates. The threshold values can be chosen in accordance with the network's distribution of node reputation values and the system's security settings.

4. Experimental Analysis

The algorithm is simulated using Java and compared with the original consensus algorithm in terms of fault tolerance, node reputation value, communication complexity, and throughput. The experimental environment is a Windows 10 operating system with 16 GB of system memory and two processors with Intel(R) Xeon(R) Gold 6148 CPU @ 2.40 GHz 2.39 GHz CPUs.

4.1. Algorithm Fault Tolerance Analysis

Let the entire number of nodes be N, the number of malicious nodes is m, the number of faulty nodes is s, the number of Byzantine nodes be $f = m + s$, and the number of nodes in the system that can function properly be $N - f$. Assuming that Q nodes agree on a message during the consensus process, then consensus is considered to be reached.

When f Byzantine nodes in the framework do not participate in the consensus process, then the consensus protocol should guarantee that the remaining N-f urban nodes reach consensus, as shown in the following equation.

$$Q \le N - f \tag{16}$$

When there are Byzantine nodes involved in the consensus process, it must be ensured that the number of honest nodes should exceed the number of Byzantine nodes, then there are:

$$N - f - f > f \tag{17}$$

Therefore, from (17), we can get $N > 3f$, so N must be at least greater than or equal to $3f + 1$ so as to ensure security and activity in the blockchain system.

Table 1. Reputation status of nodes.

$Trust_i$ range	$STrust_i$	Node Classification
$[m, 1]$	Excellent	Consensus node, Supervisory node
$[l, m)$	Normal	Consensus node
$[n, l)$	Abnormal	Backup nodes
$[0, n)$	Error	Prohibit participation in consensus

4.2. Reputation Value of the Node

A small blockchain system is built using Java language with 30 nodes set up, containing 9 Byzantine nodes. The experiment contains consensus stage 1, consensus stage 2, consensus stage 3, and consensus stage 4. Each phase contains 30 rounds of consensus, and at the end of each phase, the reputation values of the nodes are updated, as shown in Figure 8.

1) After the consensus stage 2, the honest nodes and Byzantine nodes in the system can be distinguished by the reputation value, which ensures the security and stability of the ensuing operation of the whole blockchain framework.

2) The reputation value of erroneous nodes with node numbers 7, 10, 17, and 26 exceeds the threshold value of 0.5 after consensus stage 1, mainly because these four nodes have lower initial reputation values compared with other erroneous nodes and are not detected by the system because they do not have the right to act as master nodes and supervisory nodes in consensus stage 1, but after consensus stage 1, they take up the corresponding roles and are finally recorded by the system. The consensus success rate of consensus stage 1 is low because the initial reputation esteem of nodes mainly relies on the voting decision among nodes, which has a certain chance, and there may be a situation in which some malicious nodes obtain a higher initial reputation, but as the consensus process continues, the system records all malicious nodes one by one, and the consensus success rate reaches 100%.

4.3. Communication Complexity

The PBFT consensus algorithm is a network-wide broadcast by nodes for message delivery, which will significantly consume communication resources in the system. Communication overhead is one of the indicators of algorithm efficiency. Improving the algorithm can improve consensus efficiency and reduce the waste

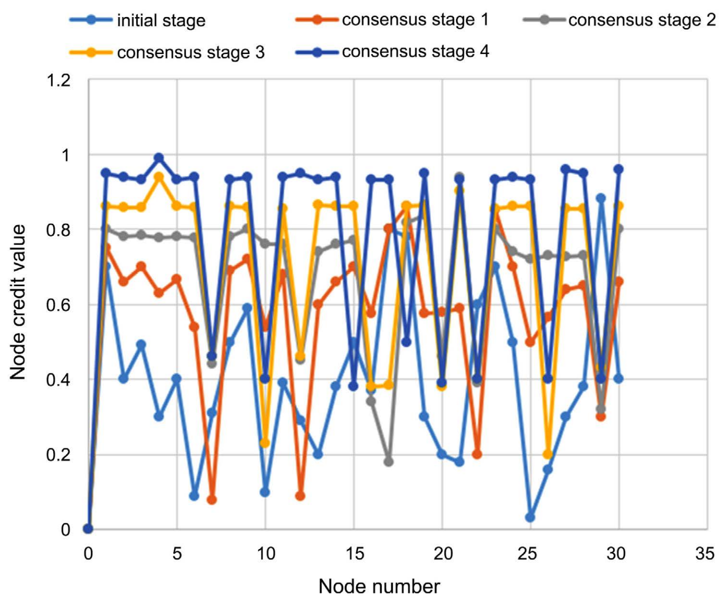


Figure 8. Node credit value distribution.

of communication resources. In the original PBFT consensus algorithm, network-wide broadcast communication is required in the preparation and confirmation stages, and each node needs to communicate $N-1$ times in this process, and the number of communications required to complete a consensus round is $2N(N-1)$, which will put a large burden on the communication network when the number of nodes is large.

The improved algorithm is to decrease the number of consensus nodes involved in the consensus process by dividing the nodes in the system into roles. Nodes in the network are divided into three categories: consensus nodes, supervisory nodes, and backup nodes, which serve different functions in the system. Supervisory nodes and backup nodes do not participate in the consensus process but only supervise and vote on the reputation value of the nodes. This greatly reduces the number of consensus nodes compared to the original algorithm, thus reducing the communication overhead of the system.

4.4. Throughput Capacity

The throughput of a framework is the number of exchanges prepared per unit of time within the framework, and the capacity of the framework to handle exchanges depends on the estimate of the throughput. The higher the throughput, the higher the capacity to handle exchanges, and bad habit versa. The equation for calculating throughput is as takes after.

$$TPS = \frac{transactions_{\Delta t}}{time} \tag{18}$$

where $transactions_{\Delta t}$ is the number of exchanges processed by the system amid the agreement handle. It is the time required by the system to process the transactions, the throughputs of the improved algorithm and the original PBFT consensus algorithm are compared at the same time with the same number of nodes. This is shown in Figure 9.

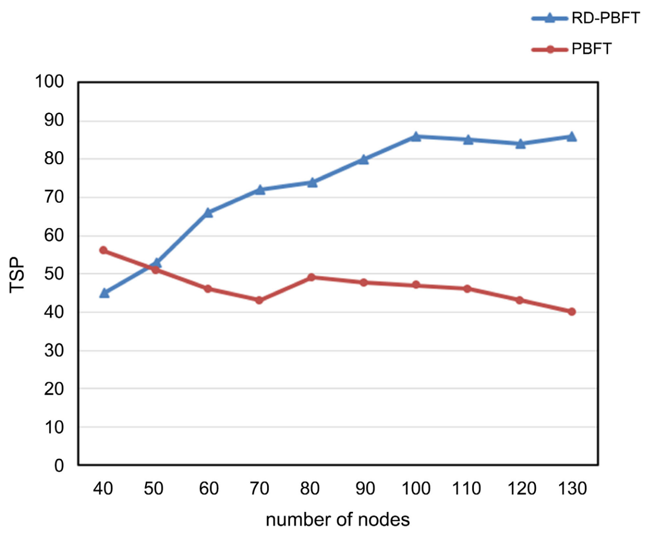


Figure 9. Throughput of node.

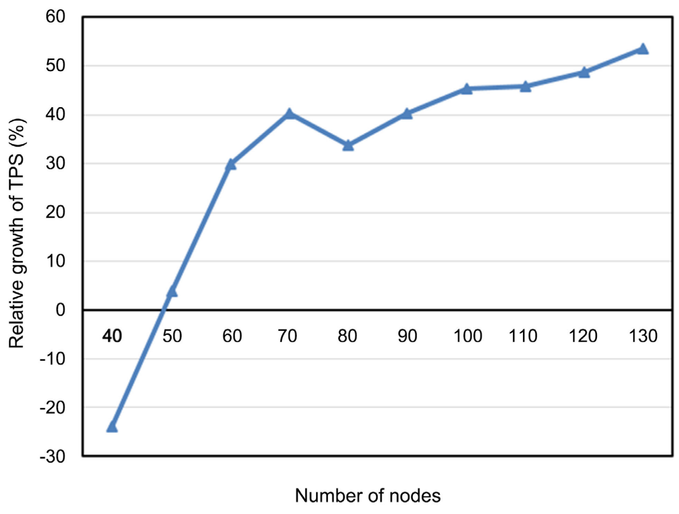


Figure 10. Node throughput ratio.

To superior compare the throughput of the RD-PBFT algorithm and the PBFT algorithm, the relative growth rate of their throughputs is calculated using Equation (19).

$$E = \frac{TPS_{PBFT} - TPS_{RD\text{-}PBFT}}{TPS_{RD\text{-}PBFT}} \times 100\% \tag{19}$$

where TPS_{PBFT} is the throughput of PBFT and $TPS_{RD\text{-}PBFT}$ is the throughput of RD-PBFT algorithm, the results are shown in Figure 10.

5. Conclusions

The consensus algorithm, the foundational technology of blockchain, has been extensively studied by numerous academics. In recent decades, blockchain technology has been utilized in a variety of sectors. Different consensus algorithms are required to enable blockchain systems under various application scenarios. The PBFT consensus algorithm, which effectively lowers communication overhead and resists the trend of centralization, is frequently employed in coalition chains, however, it performs poorly in large-scale network nodes.

The modified PBFT consensus algorithm based on role division is suggested in this study as a solution to the issue of choosing a trustworthy master node when there are many nodes. The preparation phase, the consensus phase, and the end phase make up this process. In the planning stage, the nodes in the system are classified into three types: consensus nodes, backup nodes, and supervisory nodes. Roles are assigned based on the node trust value. The three different types of nodes each play a specific part in keeping the system operational. To increase the security of the system, the adaptive technique is utilized to choose the master node during the consensus phase. Only consensus nodes are allowed to participate in the consensus process, which reduces the total number of consensus nodes involved and significantly lowers the system's communication complexity. In the closing phase, nodes' reputation values are changed in accordance with how they behaved during the consensus process, and nodes' responsibilities are redistributed, which increases the system's stability.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Yuan, Y. and Wang, F.Y. (2016) Blockchain: The State of the Art and Future Trends. *Acta Automatica Sinica*, **42**, 481.

[2] Db, A., So, B., Ns, A., Dp, A. and Yj, A. (2021) A Survey on Blockchain for Information Systems Management and Security. *Information Processing & Management*, **58**, Article ID: 102397. https://doi.org/10.1016/j.ipm.2020.102397

[3] Cai, X.Q., Deng, Y., Zhang, L., *et al.* (2021) Blockchain Principles and Its Core Technologies. *Chinese Journal of Computers*, **44**, 84-131.

[4] He, P., Yu, G., Zhang, Y.F. and Bao, Y.B. (2017) A Prospective Review of Blockchain Technologies and Applications. *Computer Science*, **44**, 1-7.

[5] Yuan, Y., Ni, X.-C., Shuai, Z. and Wang, F.-Y. (2018) Current Status and Outlook of Blockchain Consensus Algorithm Development. *Acta Automatica Sinica*, **44**, 2011-2022.

[6] Belchior, R., Vasconcelos, A., Guerreiro, S. and Correia, M. (2020) A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *ACM Computing Surveys*, **54**, Article No. 168. https://doi.org/10.1145/3471140

[7] Foss, B. and Stone, M. (2001) Business to Business: Lessons on Segmentation and Other Issues from Financial Services Markets. *Journal of Financial Services Marketing*, **5**, 308-313. https://doi.org/10.1057/palgrave.fsm.4770028

[8] Zhang, L., Xie, Y., Zheng, Y., Xue, W., Zheng, X. and Xu, X. (2020) The Challenges and Counter-Measures of Blockchain in Finance and Economics. *Systems Research and Behavioral Science*, **37**, 691-698. https://doi.org/10.1002/sres.2710

[9] Chen, L., Cong, L.W. and Xiao, Y. (2020) A Brief Introduction to Blockchain Economics. In: *Information for Efficient Decision Making*, World Scientific, Singapore, 1-40. https://doi.org/10.1142/9789811220470_0001

[10] Moriizumi, S., Chu, B., Cao, H. and Matsukawa, H. (2011) Supply Chain Risk Driver Extraction Using Text Mining Technique. *International Journal on Information*, **14**, 1935-1945.

[11] Atzori, M. (2018) Blockchain Governance and the Role of Trust Service Providers: The Trusted-Chain Network. *The Journal of British Blockchain Association*, **1**, 1-17. https://doi.org/10.31585/jbba-1-1-(3)2018

[12] Jing, N., Liu, Q. and Sugumaran, V. (2021) A Blockchain-Based Code Copyright Management System. *Information Processing & Management*, **58**, Article ID: 102518. https://doi.org/10.1016/j.ipm.2021.102518

[13] Caporale, G. M., Gil-Alana, L. and Plastun, A. (2017) Persistence in the Cryptocurrency Market. *Research in International Business and Finance*, **46**, 141-148. https://doi.org/10.1016/j.ribaf.2018.01.002

[14] Liu, A.D., Du, X.E., Wang, N. and Li, S.Z. (2018) Blockchain Technology and Its Research Progress in the Field of Information Security. *Journal of Software*, **29**, 24.

[15] Miguel, C. and Barbara, L. (2002) Practical Byzantine Fault Tolerance and Proactive

Recovery. *ACM Transactions on Computer Systems*, **20**, 398-461. https://doi.org/10.1145/571637.571640

[16] Liu, S., Cachin, C., Quéma, V. and Vukolić, M. (2015) Xft: Practical Fault Tolerance beyond Crashes. Computer Science.

[17] Yusoff, J., Mohamad, Z. and Anuar, M. (2022) A Review: Consensus Algorithms on Blockchain. *Journal of Computer and Communications*, **10**, 37-50. https://doi.org/10.4236/jcc.2022.109003

[18] Lamport, L. (1998) The Part-Time Parliament. *ACM Transactions on Computer Systems*, **16**, 133-169. https://doi.org/10.1145/279227.279229

[19] Kotla, R., Alvisi, L., Dahlin, M., Clement, A. and Wong, E. (2007) Zyzzyva: Speculative Byzantine Fault Tolerance. *ACM Sigops Operating Systems Review*, **41**, 45-58. https://doi.org/10.1145/1323293.1294267

[20] Chen, Y.N., *et al.* (2022) An Improved Algorithm for Practical Byzantine Fault Tolerance to Large-Scale Consortium Chain. *Information Processing & Management*, **59**, Article ID: 102884. https://doi.org/10.1016/j.ipm.2022.102884

[21] Wang, Y.H., *et al.* (2019) Study of Blockchainss Consensus Mechanism Based on Credit. *IEEE Access*, **7**, 10224-10231. https://doi.org/10.1109/ACCESS.2019.2891065

[22] Zheng, X., Feng, W., Huang, M. and Feng, S. (2021) Optimization of PBFT Algorithm Based on Improved C4.5. *Mathematical Problems in Engineering*, **2021**, Article ID: 5542078. https://doi.org/10.1155/2021/5542078

[23] Qushtom, H., Mii, J., Chang, X. and Mii, V.B. (2021) A Scalable Two-Tier PBFT Consensus for Blockchain-Based IoT Data Recording. *IEEE ICC* 2021, Montreal, 14-23 June 2021, 1-6. https://doi.org/10.1109/ICC42927.2021.9500260

[24] Gao, S. (2019) T-PBFT: An Eigentrust-Based Practical Byzantine Fault Tolerance Consensus Algorithm. *China Communication*: *English Version*, **16**, 13. https://doi.org/10.23919/JCC.2019.12.008

[25] Tong, W., Dong, X. and Zheng, J. (2019) Trust-PBFT: A PeerTrust-Based Practical Byzantine Consensus Algorithm. 2019 *International Conference on Networking and Network Applications* (*NaNA*) *IEEE*, Daegu, 10-13 October 2019, 344-349. https://doi.org/10.1109/NaNA.2019.00066

[26] Li, Y., Wang, Z., Fan, J., Zheng, Y. and Ding, J. (2019) An Extensible Consensus Algorithm Based on PBFT. 2019 *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (*CyberC*), Guilin, 17-19 October 2019, 17-23. https://doi.org/10.1109/CyberC.2019.00013

[27] Qu, Y. and Xiong, N. (2012) RFH: A Resilient, Fault-Tolerant and High-Efficient Replication Algorithm for Distributed Cloud Storage. *International Conference on Parallel Processing IEEE*, Pittsburgh, 10-13 September 2012, 520-529. https://doi.org/10.1109/ICPP.2012.3

[28] Miguel, C. and Barbara, L. (2002) Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, **20**, 398-461. https://doi.org/10.1145/571637.571640

[29] Castro, M. and Liskov, B. (1999) Practical Byzantine Fault Tolerance. *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, New Orleans, February 1999, 1-14.