Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

CG-PBFT: an efficient PBFT algorithm based on credit grouping

Juan Liu[1], Xiaohong Deng[2,3*], Wangchun Li[1] and Kangting Li[2]

Abstract
Because of its excellent properties of fault tolerance, efficiency and availability, the practical Byzantine fault tolerance (PBFT) algorithm has become the mainstream consensus algorithm in blockchain. However, current PBFT algorithms have problems such as inadequate security of primary node selection, high communication overhead and network delay in the process of consensus. To address these problems, we design a novel efficient Byzantine fault tolerance algorithm based on credit grouping, called CG-PBFT. First, we propose a new credit evaluation model to obtain nodes' credit values and introduce an optimized three-way quick sorting algorithm to divide nodes into the master-node group, the consensus-node group and the observation-node group, which have different privileges. The nodes in the observation-node group are restricted from participating in consensus, which reduces the communication overhead and improves consensus efficiency. Second, we propose an optimized selection method for the primary node based on a voting mechanism whereby the consensus-node group and observation-node group vote to produce the primary node, which reduces the probability of malicious nodes acting as the primary node and improves the security of primary node selection. Finally, the identity conversion mechanism between node groups is designed, and the actual behavior of nodes within different groups is given credit rewards or punishment, so as to keep an incentive for nodes to participate in appropriate system behavior and improve the working enthusiasm of nodes. The experimental simulation results show that compared with existing PBFT algorithms, the CG-PBFT algorithm improves the average throughput by 51.3% and reduces the average delay by 64.5%; it greatly improves the operating efficiency of the system and can be more suitable for application in the consortium blockchain scenarios.

Keywords Blockchain, Consensus algorithm, Byzantine fault tolerance, Credit grouping, Quick sorting

Introduction

Blockchain is a decentralized, tamper-resistant, traceable, and highly transparent distributed ledger technology that uses a variety of techniques, such as timestamps, Merkle trees, asymmetric key encryption algorithms, P2P transmission [1], consensus algorithms [2], and incentive mechanisms [3]. In particular, transactions do not rely on third-party organizations, and each node in the network has the right to produce or verify blocks [4]. At present, blockchain can effectively integrate IoT, artificial intelligence, 5G, big data and other technologies, and it makes important contributions in the fields of health care [5], IoT [6], energy [7], and digital copyright authentication [8]. Generally, according to the scale and coverage of nodes in the blockchain, blockchain can be categorized into public blockchain, consortium blockchain and private blockchain [9]. Among them, public blockchain allows all nodes to freely access the system and verify transactions in the chain. In contrast, private blockchain is generally used within an organization and is not disclosed to the public. However, consortium blockchain

*Correspondence:
Xiaohong Deng
deng_xh@jxust.edu.cn
[1] School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
[2] School of Electronics and Information Engineering, Gannan University of Science and Technology, Ganzhou 341000, China
[3] Key Laboratory of Cloud Computing and Big Data, Ganzhou 341000, China

is between the above two categories of blockchain, and only the internal personnel of organizations with a cooperative relationship can be authorized to join the consortium blockchain. Compared with the public blockchain, the number and status of nodes in the consortium blockchain are more controllable, so it has higher security. On the other hand, consortium blockchain has higher decentralization than private blockchain. Therefore, consortium blockchain has become the preferred framework for blockchain application systems.

As the core technology of blockchain, the consensus algorithm is crucial to ensure data consistency and correctness. In particular, it determines the throughput, delay, fault tolerance and applicable scenarios of the whole blockchain system [10]. For this reason, in the past 15 years, many researchers have studied consensus algorithms and have made good progress. For instance, Deng et al. [11] proposed a CCAC consensus algorithm applied to consortium blockchain, they designed a credit mechanism to evaluate nodes' consensus behavior as the basis for selecting miner nodes, which solves the problem of low efficiency and insufficient incentive mechanism of current consensus algorithms. Liu et al. [12] proposed a new PoLe consensus mechanism, which used the meaningless computing power consumption of PoW for neural networks' training, and it solves the problem of high computational demand of neural network models applied in blockchain. Li et al. [13] proposed a PBT-BFT consensus algorithm, they designed a perfect binary tree communication topology to reduce the performance complexity of the consensus algorithm to linear and improve the efficiency of the consensus algorithm.

Currently, the PBFT algorithm [14] can tolerate faulty or malicious nodes, whose number is no more than 1/3. As a result, it has become a more popular consensus algorithm, which is widely used in consortium blockchain. First, in each round of consensus, the PBFT algorithm randomly selects a primary node based on the current view and the number of nodes. Then, the other nodes communicate with the primary node as slave nodes to complete the three-stage consensus process of pre-prepare, prepare and commit. Many studies have proven that the PBFT algorithm has high performance and practicality. For example, Qi et al. [15] proposed a PBFT algorithm for IoT multi-scenarios, which grouped large-scale IoT nodes according to their geographical locations, so as to improve the applicability of the PBFT algorithm for IoT scenarios. Wang et al. [16] proposed an improved PBFT algorithm for community governance, which set nodes with different roles in community activities to perform different consensus tasks with different permissions, so as to meet the community governance scenarios's requirements. Zhong et al. [17] proposed an improved PBFT algorithm for intellectual property transaction, which divided nodes into groups according to the nodes' IP transaction type, and each group processed requests in parallel to reduce the consensus communication's complexity. Liu et al. [18] proposed an improved PBFT algorithm which grouped nodes to perform consensus according to nodes' response speed, so as to improve the availability for the drug tracking system.

However, the PBFT algorithm also suffers from the following two shortcomings: (1) The primary node's selection is too random, so it is easy to select malicious nodes as the primary nodes, which wastes network resources. Although the malicious primary nodes can be replaced by view switching, continual view switching will reduce system's consensus efficiency. (2) There is a lack of an eviction mechanism for malicious nodes, and malicious nodes may always exist in the system and participate in consensus, which increases security risks and communication overhead in the network.

To address the PBFT algorithm's shortcomings, many researchers have introduced credit mechanisms to improve the system's security. On the one hand, these mechanisms preferentially select the high credit values' nodes as the primary nodes; on the other hand, they divide nodes according to their credit values and select high credit values' groups to participate in consensus, which significantly improves the consensus efficiency. For example, Zheng et al. [19] used the C4.5 algorithm to conduct credit evaluations and group nodes; the high credit values' nodes were selected to form a master consensus group, and an integral voting mechanism was used to determine the primary nodes to optimize the primary nodes' selection. Wang et al. [20] proposed a feature credit grouping model to divide nodes into multiple groups; afterward, the primary node only needed to send messages to each group of agent nodes. In this way, the communication overhead was reduced by intragroup consensus through each independent group. Ren et al. [21] divided nodes into a high-credit group and a low-credit group by voting among nodes. In the high-credit group, the nodes were divided into consensus nodes and supervisory nodes. Conversely, nodes in the low-credit group were backup nodes. Thus, the complexity of the system communication was reduced by restricting the participation of the supervisory nodes and the backup nodes in consensus. Liu et al. [22] designed a credit and voting mechanism. More precisely, first, the nodes voted to generate the leaders of the groups based on the nodes' credit values; next, this method grouped the remaining nodes according to the response speed of each group leader. Finally, the group leaders used the in-group's consensus result to determine participation in the out-of-group's consensus, which reduced the frequency of internode communication and improves the reliability of nodes. Xu et al. [23] divided nodes into a consensus group and a candidate group according to a credit-scoring mechanism to restrict nodes from participating in consensus. Meanwhile, it optimized the consensus process, which yielded higher efficiency. Wang et al. [24] randomly divided nodes into multiple node groups and introduced a credit mechanism to elect supervision nodes from each group to monitor the consensus nodes, which improved consensus security. Tang et al. [25] divided the nodes into consensus node group and ordinary node group by introducing the credit scoring mechanism, reduced consensus's scale, and simplified the pre-prepare phase of PBFT algorithm to improve the consensus's efficiency.

As mentioned previously, the above algorithms all use the method of node grouping to optimize the PBFT consensus mechanism, which benefits the system's security and operation efficiency to a certain extent. However, few studies have been conducted on the following two points. First, when the credit mechanism is used to select the high credit values' nodes as the primary nodes, the dynamic nature of the credit evaluation model is not taken into account, and a primary node with high credit value is prone to becoming an "oligarchy" node. In addition, the nodes with low credit values lack credit incentives, which can easily cause these nodes to lose consensus enthusiasm. Second, in the node credit grouping process, on the one hand, overly complex grouping methods are used, which generates new overhead problems for the system; for instance, if grouping is implemented by mutual voting among nodes, this affects the algorithm's efficiency. In contrast, the grouping method may be too simple; for example, if grouping is implemented according to the number of nodes, this neglects grouping methods that may make the grouping more reliable.

Consequently, an efficient PBFT algorithm based on credit grouping (CG-PBFT) is proposed in this paper. The main contributions of this paper are as follows.

(1) We design a new credit evaluation model that gives credit rewards to good nodes and credit punishments to malicious nodes. According to the nodes' behavior, the credit evaluation model evaluates and dynamically updates the node credit values in each round. Afterward, the nodes enter into groups with different privileges through the credit value rewards and punishments, which not only realizes the dynamic transformation of the nodes among the different groups but also improves the reliability and enthusiasm of the nodes participating in consensus.

(2) We propose a new credit grouping mechanism that uses an optimized three-way quick sorting algorithm to divide the nodes into the master-node group, the consensus-node group and the observation-node group according to the nodes' credit values. Furthermore, consensus is only carried out in the master-node group and the consensus-node group, which improves the grouping efficiency and reduces communication overhead.

(3) We propose a selection method for the primary node based on a voting mechanism. The consensus-node group and the observation-node group both vote for a primary node from the master-node group with a higher credit rating. Therefore, the selected primary node has a high credit value instead of being directly assigned by the highest credit value's node. The randomness of primary node selection clearly improves the security of the system and avoids the generation of "oligarch" nodes.

Related works

PBFT consensus

As previously discussed, the PBFT algorithm can solve the problem of Byzantine generals well. In particular, assuming that the number of current view is v, the total number of nodes is n, and the malicious nodes' number is f, then n and f need to satisfy $3f+1<=n$. Moreover, the identifier of the selected primary node is $p=v \bmod n$. For instance, assuming four nodes are numbered 0, 1, 2, and 3, where node 3 is the malicious node, which intentionally does not respond to requests from other nodes. During the initial consensus, the view's number is 0. According to the primary node selection rule, the number of the primary node in the current view is 0. Then, the system begins the consensus process. This process is shown in Fig. 1.

The consensus process is divided into five stages, three of which require the nodes to communicate with each other.

Stage 1: Request stage. The client sends a request message to the primary node 0. After the message is successfully validated, the system enters the pre-preparation stage.

Stage 2: Pre-preparation stage. Primary node 0 broadcasts the client request to nodes 1, 2, and 3 and sends a pre-preparation message. After these three nodes receive the pre-preparation message, they verify whether this message is correct, and then the system enters the preparation stage.

Stage 3: Preparation stage. After nodes 1, 2, and 3 receive and verify the pre-preparation message, they

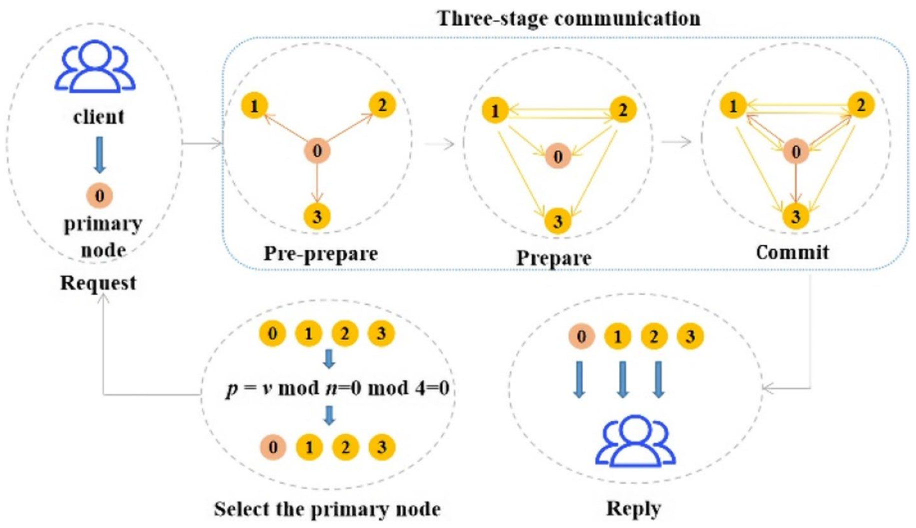

Fig. 1 The PBFT algorithm consensus process

enter the preparation stage to broadcast the preparation message. However, node 3 is malicious and does not send messages, so node 1 only receives messages from node 2, and node 2 only receives messages from node 1. After the nodes receive at least $2f$ preparation messages (including from themselves) that are identical to the pre-preparation messages, the system enters the next stage.

Stage 4: Commit stage. Nodes 0, 1, 2, and 3 broadcast the confirmation message to each other. In particular, because node 3 does not send the message, node 0 only receives the confirmation messages from nodes 1 and 2, node 1 only receives the confirmation messages from nodes 0 and 2, and node 2 only receives the confirmation messages from nodes 0 and 1. After the nodes receive at least $2f+1$ confirmation messages (including from themselves), the system enters the next stage.

Stage 5: Reply stage. Nodes 0, 1, and 2 update their logs and reply with a confirmation message to the client. After the client receives $f+1$ reply messages, it concludes that the current view consensus is complete.

According to the PBFT algorithm's consensus process, the primary node role is taken by nodes in turn. In other words, the primary node selected by this selection method in the next round is predictable, which poses a threat to system's security. Although the primary node can be reselected through view switching, this will consume many network resources, which is the main reason for the low throughput and high delay of the PBFT algorithm. In addition, communication is needed in the stages 2, 3 and 4, which is closely related to the number of nodes participating in consensus. However, the algorithm lacks an exit mechanism for malicious nodes, so they always remain in the system, which not only increases the communication overhead but also weakens the security of the system.

Three-way quick sorting algorithm

Generally, introducing a credit scoring mechanism into the PBFT algorithm can effectively enhance the reliability of nodes. For example, Zheng et al. [19] proposed that each node's credit value should be assigned as 50 points during initialization. In the subsequent consensus process, if a node can successfully produce a block, the system will reward it with 10 credit points. Conversely, if the node acts maliciously, the system will deduct 10 points from its credit value. Therefore, in the case of a large number of nodes and consensus rounds, there will inevitably be many nodes with the same credit values. When the nodes' credit values need to be sorted and grouped, the three-way quick sorting algorithm can be adopted, which has a greater advantage in time complexity and efficiency when the data are already ordered or there are a lot of repetitive data. Furthermore, the three-way quick sorting algorithm can not only ensure the efficiency of the system but also quickly screen out the nodes with high and low credit values. In particular, it can avoid the problems of system burden caused by complex grouping methods and unreliable grouping caused by simple grouping methods.

Generally, the three-way quick sorting algorithm adopts the idea of "partitioning". It selects a baseline element (x) and divides the set of data to be sorted into three intervals: less than *x*, equal to *x*, and greater than *x*. In the subsequent sorting process, only the intervals less than *x* and greater than *x* need to be sorted recursively, while the interval equal to *x* is not processed. The algorithm is presented in Fig. 2.

The partition method traverses the array once from left to right. In particular, before partition, it maintains a pointer *lt* such that the elements to the left of *lt* are all greater than *x*. Similarly, it maintains a pointer *gt* such that the elements to the right of *gt* are all less than *x*. It also maintains a pointer *i* such that the elements in a[*lt*..*i*-1] are equal to x and those in a[*i*..*gt*] are waiting to be scanned. When partitioning, it compares a[*i*] and *x* and treats the elements differently according to the comparison result. If a[*i*] is greater than *x*, it swaps a[*lt*] with a[*i*] and moves *lt* and *i* one place back. If a[*i*] is less than *x*, it swaps a[*gt*] with a[*i*] and moves *gt* one place forward. Finally, if a[*i*] is equal to *x*, it moves *i* one place back. It should be noted that the scanning operation should be performed until $i > gt$. After partitioning, recursion is performed on the intervals greater than *x* and less than *x*. The algorithm needs to compare the elements in these two intervals and the baseline element; then, it performs the same operation as in partitioning. As a result, the final ordered three-interval array is obtained.

As mentioned above, the data in the middle interval are only a single value *x*. That is in the case of a mass of data, many data will be divided into the intervals less than *x* and greater than *x*. However, the same operation needs to be completed for these two intervals as in partition. Therefore, the algorithm needs to perform a large number of recursions to complete the final partition. Furthermore, the algorithm is more restrictive in the range of the middle interval data, so it faces some limitations in dividing the data into three intervals.

Proposed CG-PBFT algorithm

Algorithm model

The proposed CG-PBFT algorithm divides nodes into three groups according to their credit values and votes the primary node from the master-node group, which avoids the predictability of the primary node selection. Additionally, it limits the privileges of the observation-node group to participate in consensus, which reduces the algorithm's communication overhead. For clarity, we take 10 nodes in the system as an example, and the overall algorithm model is described in Fig. 3.

As presented in Fig. 3, there are 4 steps in the proposed CG-PBFT algorithm.

Step 1: Evaluate the nodes' credit values. Before each round of consensus, 10 nodes' credit values are evaluated by using the credit evaluation model, which will be discussed in detail below.

Step 2: Group nodes by their credit values. Through the optimized three-way quick sort algorithm, the 10 nodes are first ranked according to their credit values. Then, the credit values of the nodes are divided into 3 intervals. As shown in Fig. 3, first, nodes 0 and 1 are grouped into the master-node group. Second,

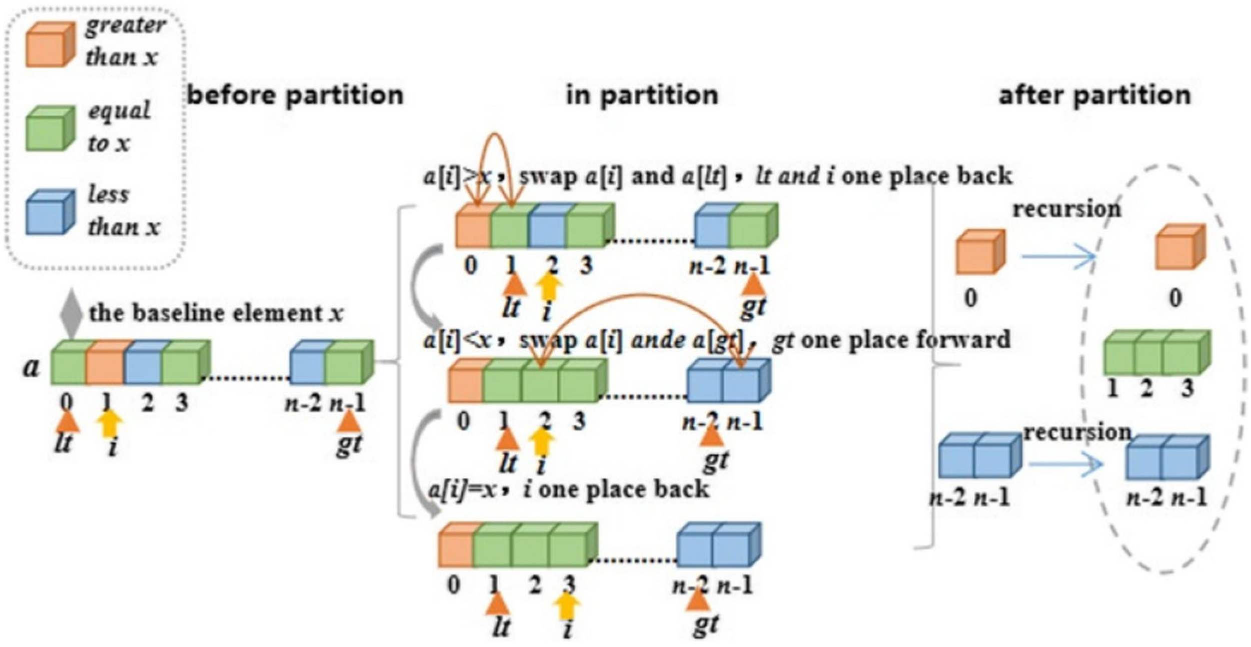

Fig. 2 The three-way quick sorting algorithm division

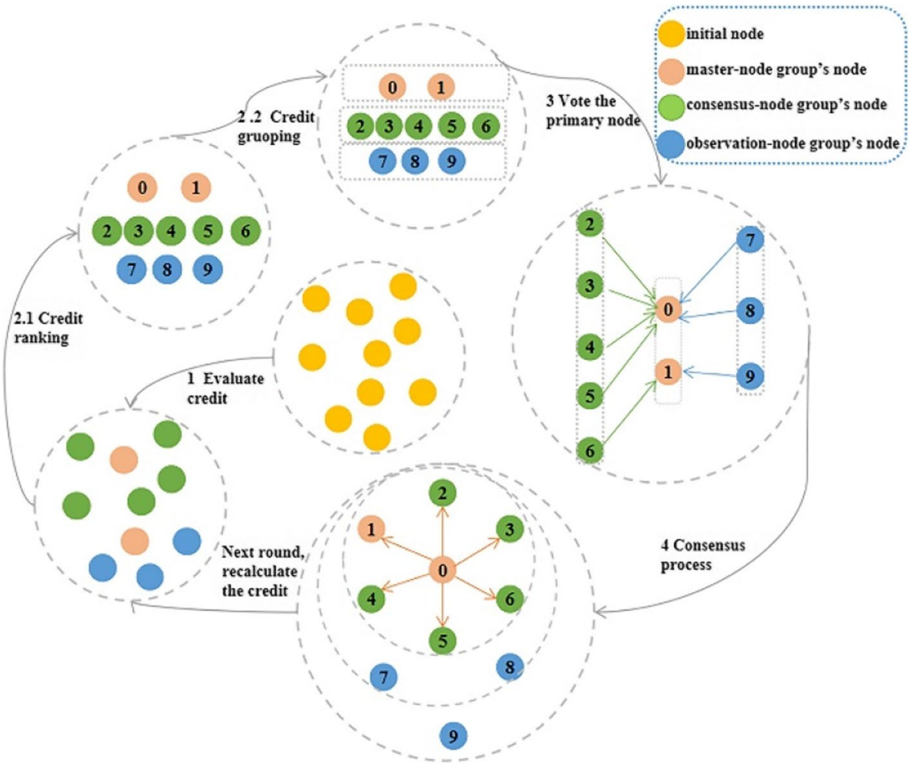

Fig. 3 The overall algorithm model

nodes 2 to 6 are grouped into the consensus-node group. Finally, nodes 7 to 9 are grouped into the observation-node group.

Step 3: Select the primary node. In CG-PBFT, the consensus-node group and the observation-node group vote for nodes in the master-node group. As shown in Fig. 3, in the consensus-node group, nodes 2 to 5 vote for node 0, and node 6 votes for node 1. In the observation-node group, nodes 7 and 8 vote for node 0, and node 9 votes for node 1. According to the designed vote counting rules, the votes obtained by the nodes in the master-node group are counted, and node 0, with the most votes, is elected as the primary node; this method ensures the unpredictability of primary node selection.

Step 4: Begin the consensus process. Nodes 0 and 1 in the master-node group and nodes 2 to 6 in the consensus-node group participate in consensus. After consensus, the numbers of times each of the 10 nodes has been in each group is counted. In this case, node 9 has been in the observation-node group for a long time, and its credit value is ranked at the bottom of the list, so it is recognized as an "inactive" node and is removed from the system. Finally, all nodes' credit values are recalculated to start the next round of consensus.

Credit evaluation model

The node's credit evaluation indicators designed in this paper are divided into three categories: direct credit value, indirect credit value and historical credit value. The direct credit value is related to a node's behavior of voting and consensus completion. The indirect credit value is related to the situation of a node in each group and its credit ranking. The historical credit value is related to the accumulated credit value of a node.

Definition 1 Consensus credit C_con. The consensus credit is used to evaluate the consensus completion degrees of nodes. The consensus credit of node i C_con_i is calculated by the following formula:

$$C\_con_i = \frac{Num\_c_i}{TNum\_c_i} \tag{1}$$

where Num_c_i represents the number of times node i has successfully completed consensus and $TNum\_c_i$ denotes the total number of times node i has participated in consensus. Whether a node can successfully complete consensus is the most important criterion by which to judge the quality of nodes. Therefore, the node's consensus credit holds the maximum weight in the credit value evaluation indicators, which has an obvious influence on the nodes' credit values. In brief, the more times nodes successfully complete consensus, the higher the consensus credit rewards they will receive. This makes it easier for the master-node group's nodes to remain in this group for a long time. Additionally, it encourages the consensus-node group's nodes to enter the master-node group to contest primary node election.

Definition 2 Voting credit *C_vot*. The voting credit is used to evaluate the voting completion degrees of nodes. The voting credit of node *i*, C_vot_i, is calculated by the following formula:

$$C\_vot_i = \frac{Num\_v_i}{TNum\_v_i} \tag{2}$$

where Num_v_i is the number of times node *i* has successfully completed voting and $TNum\_v_i$ is the total number of times node *i* has participated in voting. Notably, the more times nodes successfully complete voting, the higher the voting rewards they will receive. In particular, the voting credit encourages the consensus-node group's nodes to increase their credit rewards by actively voting in order to enter the master-node group. Similarly, observation-node group's members have the opportunity to enter the consensus-node group to obtain more privileges.

Definition 3 Direct credit value *C_dir*. The direct credit value evaluates nodes' behavior during the whole consensus process. In this process, the main behavior of a node is to complete consensus and voting. As a result, its direct credit value is the weighted sum of the consensus credit and voting credit. The node *i*'s direct credit value, C_dir_i, is calculated by the following formula:

$$C\_dir_i = \lambda * C\_con_i + \mu * C\_vot_i \tag{3}$$

where λ is the consensus credit weight, μ is the voting credit weight, and these two weights satisfy $\lambda + \mu < 1$.

Definition 4 Active credit *C_act*. The active credit estimates nodes' active status in each group. If a node exists in the master-node group or consensus-node group multiple times, then it is active and more likely to be a trusted node. The node *i*'s active credit, C_act_i, is calculated by the following formula:

$$C\_act_i = \frac{\gamma * Num\_GM_i + (1-\gamma) * Num\_GC_i}{TNum\_c_i} \tag{4}$$

where Num_GM_i is the number of times node *i* has been in the master-node group and Num_GC_i is the number of times node *i* has been in the consensus-node group. γ is the weight of Num_GM_i, and $\gamma > 0.5$, which means that the more times a node is in the master-node group, the more active it is, and the more active credit it will be awarded.

Definition 5 Incentive credit *C_inc*. The incentive credit estimates motivate nodes with lower credit values to work hard. For this reason, it is related to the credit ranking of nodes. The node *i*'s incentive credit value, C_inc_i, is calculated by the following formula:

$$C\_inc_i = e^{-\frac{n - C\_rank_i}{n}} \tag{5}$$

where C_rank_i represents the node *i*'s credit ranking and *n* represents nodes' total number in the system. More precisely, if a node with the highest credit value, then it is ranked 1, the incentive credit of this node is $e^{-\frac{n-1}{n}}$. In contrast, the node with the lowest credit value is ranked *n*, then its incentive credit is 1. Notably, the lower the node ranking is, the higher the incentive credit the node receives. For example, nodes in the consensus-node group will receive higher incentive credit than nodes in the master-node group. As seen from the formula, the incentive credit range is $[e^{-\frac{n-1}{n}}, 1]$, which is related to the number of nodes participating in the system.

Definition 6 Indirect credit value *C_idir*. The indirect credit value is the indirect evaluation of node consensus behavior, which is used to reward nodes with good long-term performance and incentivize low credit values' nodes in the system. The indirect credit value is the weighted sum of the active credit and incentive credit. The node *i*'s indirect credit value, C_idir_i, is calculated by the following formula:

$$C\_idir_i = \alpha * C\_act_i + \beta * C\_inc_i \tag{6}$$

where α is the active credit weight, β is the incentive credit weight, and these two weights satisfy $\alpha + \beta < 1$.

Definition 7 Comprehensive credit value *C*. The comprehensive credit value is a comprehensive evaluation of the direct, indirect and historical behavior of nodes used to give good nodes credit rewards and give malicious nodes a credit penalty. The comprehensive credit value is the weighted sum of the direct credit value, indirect credit value and historical credit value. The node *i*'s comprehensive credit value, C_i, is calculated by the following formula:

$$C_i = \begin{cases} C\_dir_i + C\_idir_i + \eta * C\_his_i, \textit{if } i \textit{ is a good node} \\ \frac{1}{2} * C\_his_i, \textit{if } i \textit{ is a malicious node} \end{cases} \tag{7}$$

where C_his_i is node *i*'s historical credit value and η is its weight.

Definition 8 Credit value normalization. After the comprehensive credit value is calculated, data normalization is used to make it fall in the range of [0,1]. The normalization result is

$$Credit_i = \frac{C_i - C_{\min\ i}}{C_{\max\ i} - C_{\min\ i}} \tag{8}$$

where C_{maxi} is the maximum comprehensive credit value of nodes and C_{mini} is the minimum comprehensive credit value of nodes in each round of consensus.

Generally, by dynamically adjusting the values of λ, μ, α, β and η, the direct credit value, indirect credit value and historical credit value are adjusted. Moreover, it is worth mentioning that the weights need to satisfy $\lambda+\mu+\alpha+\beta+\eta=1$ and $\lambda+\mu>\alpha+\beta$, where λ is the largest weight. The description of each credit evaluation indicator is shown in Table 1.

Credit grouping mechanism

After evaluating the node credit values according to the credit evaluation model, the nodes are divided into the master-node group, the consensus-node group and the observation-node group according to their credit values. The privileges for each node group are shown in Table 2 below.

Master-node group

The master-node group is the candidate group used to obtain the primary node. In other words, only nodes in the master-node group can serve as the primary nodes. Specifically, when the primary nodes need to be switched, nodes will be revoted from this group to replace them. In addition, nodes that are not selected as the primary nodes will participate in consensus together with the consensus-node group's nodes. In particular, to ensure the fairness of consensus, this group's nodes do not have the authority to vote for primary nodes.

Table 1 Credit evaluation indicators

Node type	First-level evaluation indicators	Symbolic representation	Weight	Second-level evaluation indicators	Symbolic representation	Weight	Third-level evaluation indicators	Symbolic representation	Weight
Good nodes	Direct credit value	C_dir_i	$\lambda+\mu$	Consensus credit	C_con_i	λ	Number of successful consensus completions	Num_c_i	-
							Total number of consensus participation rounds	$TNum\_c_i$	-
				Voting credit	C_vot_i	μ	Number of successful voting completions	Num_v_i	-
							Total number of voting participation rounds	$TNum\_v_i$	-
	Indirect credit value	C_idir_i	$\alpha+\beta$	Active credit	C_act_i	α	Number of times the node exists in the master-node group	Num_GM_i	γ
							Number of times the node exists in the consensus-node group	Num_GC_i	$1\text{-}\gamma$
							Total number of consensus participation rounds	$TNum\_c_i$	-
				Incentive credit	C_inc_i	β	Credit ranking	C_rank_i	-
							Number of nodes	n	-
	Historical credit value	C_his_i	η	-	-	-	-	-	-
Malicious nodes	Historical credit value	C_his_i	1/2	-	-	-	-	-	-

Table 2 Node privileges

Node Group	Privileges		
	Acting as the primary node	**Participating in consensus**	**Participating in voting**
Master-node group	✓	✓	
Consensus-node group		✓	✓
Observation-node group			✓

Consensus-node group

Nodes in this group participate in consensus and vote on the primary node selection. In particular, nodes in this group have high voting credibility.

Observation-node group

In contrast to the consensus-node group, nodes in this group can only vote on nodes in the master-node group to select the primary nodes in the next round. In particular, nodes in this group have low voting credibility.

Grouping based on optimized three-way quick sorting After the credit evaluation mechanism is introduced into the PBFT algorithm, since most nodes are good nodes, there will be more nodes' credit values in the middle interval and more nodes with the same credit values. Similarly, in the credit mechanism proposed by Zheng et al. [19], after multiple rounds of consensus, a very few excellent nodes' credit values will continue to rise to near the maximum threshold through the reward of 10 points per round, and a very few malicious nodes' credit values will continue to fall to near the minimum threshold through the punishment of 10 points per round. However, most normally behaving nodes' credit values will be in the maximum interval centered on the initial credit value through the credit value reward and punishment. Therefore, it is difficult to meet the requirements by using the ordinary three-way sorting method. For this reason, an optimized three-way quick sorting algorithm is considered in this paper. In particular, it divides nodes' credit values into three groups, which is more applicable to use a range instead of the intermediate threshold value. On the one hand, the optimized three-way quick sorting algorithm expands the range of the middle-of-the-road data. On the other hand, regardless of whether nodes are already in order or whether there are many nodes with the same credit values, the proposed algorithm can perform better than the ordinary three-way quick sorting algorithm. More precisely, the optimized algorithm reduces the number of recursions to improve the system operation efficiency, and it can divide the array more efficiently.

As mentioned previously, the range of node credit values is [0,1]. To ensure the system's security, we offset the ratio of the three groups to 2:5:3. More precisely, first, nodes with credit values in the interval (0.8,1] are divided into the master-node group, which ensures that the primary nodes have higher credit values and are more likely to be trusted nodes. Second, nodes with credit values in the interval [0.3,0.8) are divided into the consensus-node group, which ensures that enough nodes can participate in consensus. Finally, nodes with credit values in the interval [0,0.3) are classified into the observation-node group, which ensures that enough nodes can participate in voting. The specific grouping process of optimized three-way quick sorting is presented in Fig. 4 below.

As shown in Fig. 4, the optimized three-way quick sorting method traverses the credit array Cn once from left to right. More precisely, it determines the grouping interval [0.3, 0.8]. It maintains a pointer lt such that all elements to the left of lt are greater than 0.8. It maintains a pointer gt such that all elements to the right of gt are less than 0.3. Finally, it maintains a pointer i such that all elements in $Cn[lt..i\text{-}1]$ are less than or equal to 0.8 and greater than or equal to 0.3. In particular, the elements in $Cn[i..gt]$ are the elements to be scanned. When grouping, the array $Cn[i]$ is compared with the interval value. If $Cn[i]$ is greater than 0.8, then $Cn[lt]$ and $Cn[i]$ will be exchanged; afterward, lt and i will be shifted one place back. Conversely, if $Cn[i]$ is less than 0.3, then $Cn[gt]$ and $Cn[i]$ will be exchanged; afterward, gt will be shifted one place forward. Significantly, if $Cn[i]$ is in the interval, there is no need to process the element, and i is shifted forward one place until the end of scanning for $i > gt$. Once the grouping is completed, the three intervals are recursively sorted by the three ways of quick sorting, and the final credit-ordered three-interval grouping array is obtained. The credit-grouping algorithm for the optimized three-way method is described below.

According to the optimized grouping algorithm's process, the algorithm's time complexity is analyzed. In the worst case, when there is no node with the same credit values, each division requires comparing each element of the credit array and swapping its position, and the i^{th} split requires $n\text{-}i$ keyword comparisons to find the i^{th} record, which is the pivot's position. In this case, the time complexity is:

$$\begin{aligned} T(n) &= C_1 n + T(n-1) = C_1 n + C_2(n-1) + T(n-2) \\ &= \dots = \sum_{i=1}^{n-1} (n-i) = O(n^2) \end{aligned} \tag{9}$$

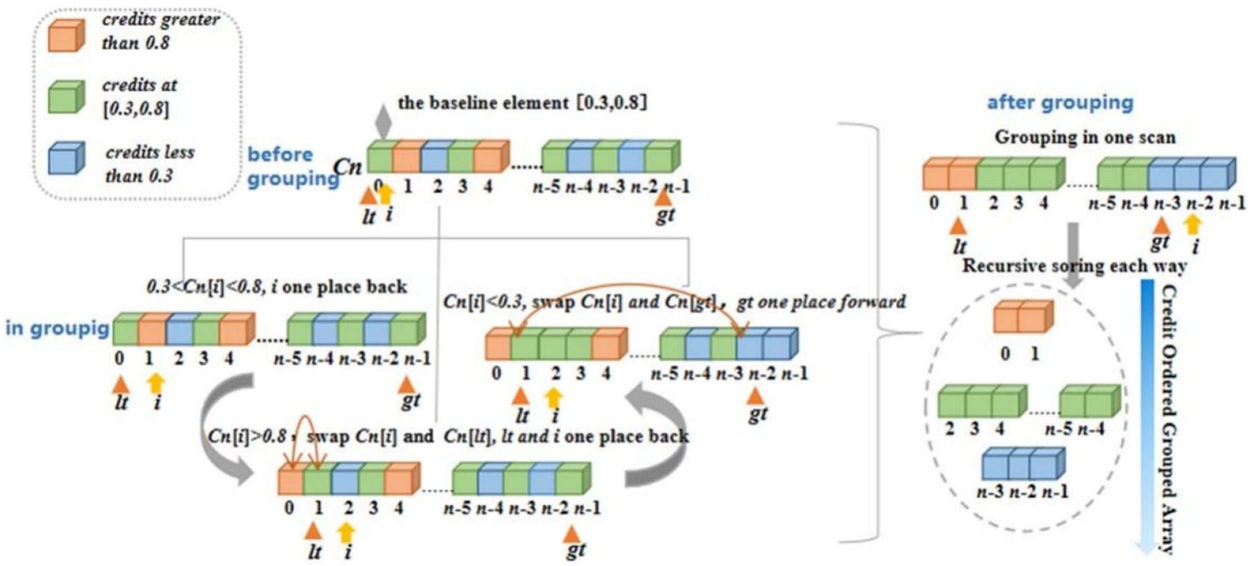
credits greater than 0.8
credits at [0.3,0.8]
credits less than 0.3
the baseline element [0.3,0.8]
before grouping
Cn
lt i gt
0.3<Cn[i]<0.8, i one place back
Cn[i]<0.3, swap Cn[i] and Cn[gt], gt one place forward
in groupig
Cn[i]>0.8, swap Cn[i] and Cn[lt], lt and i one place back
after grouping
Grouping in one scan
Recursive soring each way
Credit Ordered Grouped Array

Fig. 4 The process of optimized three-way quick grouping

In the general case, that is, each division exactly divides the nodes into three equal parts, and in each equal part, each interval recurs separately, it can be analyzed that the optimized algorithm's recursion tree is a ternary tree, and its tree height is $\log_3(n+1)$, then the average time complexity of the algorithm is:

Algorithm 1. Optimized three-way credit grouping algorithm

```
Input: $Cn$ (nodes' credit value array)
Output: $groupArray$ (nodes' credit grouping array)
1:  $OptThrQSort(Cn)$:  // Optimized three-way sorting function
2:     $piovt = partition(Cn, low: 0.3, high: 0.8)$  // Set the range of the intermediate interval to (0.3,0.8)
3:     $OptThrQSort(Cn[0: piovt])$  // Recursive sorting for values greater than intermediate interval
4:         $OptThrQSort(Cn[piovt+1: len(Cn)])$  // Recursive sorting for values less than intermediate interval
5:  $Partition(Cn, low, high)$:  // Credit grouping function
6:     $lt = 0; gt = len(Cn) - 1$ // Initialize pointers $lt$ and $gt$.
7:     $i = lt; pivot = Cn[i]$  // Initialize index $i$ and determine the baseline element $pivot$
8:     $for\ i \le gt$  //Traverse all nodes
9:        $if\ \ Cn[i] > pivot$ // Judge whether the node credit values are greater than the maximum value of the interval
10:          $Cn[lt], Cn[i] = Cn[i], Cn[lt]; lt++; i++$// If yes, swap $Cn[lt]$ and $Cn[i]$, then move $lt$ and $i$ one place back
11:       $else\ if\ \ Cn[i] < pivot$ // Judge whether the node credit values are less than the minimum value of the interval
12:          $Cn[gt], Cn[i] = Cn[i], Cn[gt]; gt--$ // If yes, swap $Cn[gt]$ and $Cn[i]$, then move $gt$ one place forward
13:       $else\ \ i++$ // Otherwise, move $i$ one place back
14:     $end\ if$
15:   $end\ for$
16: $groupArray = OptThrQSort(Cn)$  // Call the optimized three-way sorting function
```

$$\begin{aligned} T_{avg}(n) &= C_1 n + 3T_{avg}(n/3) \\ &= C_1 n + 3(C_2 * n/3 + 3T_{avg}(n/9)) \\ &= C_1 n + C_2 n + 9T_{avg}(n/9) = ... = O(n\log_3 n) \end{aligned} \tag{10}$$

Node group conversion After node credit grouping, the identity conversion between different node groups is achieved based on nodes' behavior in the whole consensus process and the proposed credit evaluation model. The process of node identity conversion between the three groups is described in Fig. 5.

As mentioned previously, in the master-node group, the nodes' credit values are higher than 0.8. As long as this group's nodes complete consensus normally, they

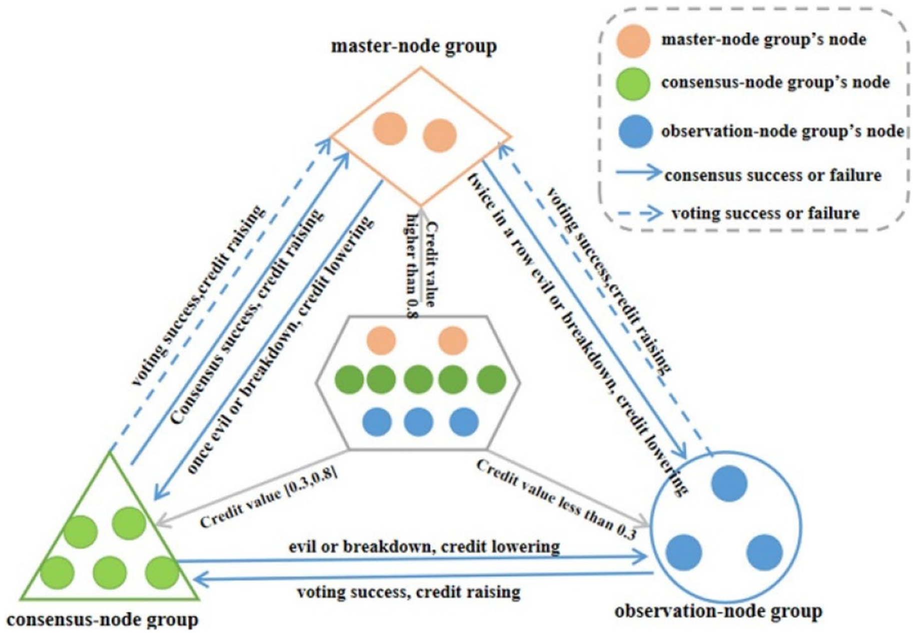
master-node group
master-node group's node
consensus-node group's node
observation-node group's node
consensus success or failure
voting success or failure
voting success,credit raising
Consensus success, credit raising
once evil or breakdown, credit lowering
Credit value higher than 0.8
twice in a row evil or breakdown, credit lowering
voting success,credit raising
Credit value [0.3,0.8]
Credit value less than 0.3
evil or breakdown, credit lowering
voting success, credit raising
consensus-node group
observation-node group

Fig. 5 Node's identity conversion between the three groups

can obtain consensus credit and active credit rewards, which can help them stay in this group for a long time. In addition, the more times they are in the master-node group and the more times they complete consensus, the higher the credit rewards they can obtain. For this reason, it is easier for them to be voted as the next primary node. However, if they fail to reach a consensus or behave maliciously once, their credit values will be halved. As a result, they are first moved to the consensus-node group, which gives them the opportunity to participate in consensus. In particular, it should be noted that if there are two consecutive instances of malicious behavior or failure, the node will be relegated to the observation-node group. Similarly, nodes in the consensus-node group can improve their credit values by actively completing consensus and voting to enter the master-node group. However, once nodes in the consensus-node group behave maliciously, their credit values will be halved; as a result, they will be very likely to move into the observation-node group. Notably, in the observation-node group, nodes' credit values are lower than 0.3, and they can only obtain credit rewards by actively voting for the primary node consistently to have the chance to enter the consensus-node group or even the master-node group. In particular, nodes that have been in the observation-node group for a long time and have lower credit rankings will be judged as "inactive" nodes, and the system will remove them to reduce overhead.

Primary node selection

As previously discussed, to guarantee the fairness and randomness of the primary node selection, a voting mechanism is introduced. Interestingly, it is guaranteed that the primary nodes will be nodes with high credit values but not necessarily the highest credit value's node. For this reason, the voting mechanism can avoids the emergence of "oligopoly" nodes. To ensure that the primary node selected by voting has higher credibility, we introduce voting weights. The number of votes for nodes is calculated by the following formula:

$$Sum\_v_i = P_c * Sumv\_GC_i + P_o * Sumv\_GO_i \quad (11)$$

where Sum_v_i represents the final count of votes received by the master-node group's nodes, $Sumv\_GC_i$ denotes the total number of votes received from the consensus-node group's nodes, and $Sumv\_GO_i$ represents the total number of votes received from the observation-node group's nodes. Furthermore, P_c indicates the consensus-node group's vote weight, P_o indicates the observation-node group's vote weight, and $P_c+P_o=1$. The consensus-node group should have more weight than the observation-node group. Therefore, in this paper, we set the two voting weights according to their credit interval ratio of 5:3; that is, we set P_c as 0.625 and P_o as 0.375.

Based on the voting rules for selecting the primary nodes, nodes that receive voting information also need to act as counting nodes for voting. In addition, the primary nodes must store not only transaction information but also voting information during their term. More precisely, the voting information format is < *vote, p, g, i* >, where i denotes the voted node's identifier, g denotes the voting node's group, and p denotes the voting node's identifier. When the voting message is broadcast to each node participating in voting, the nodes that receive the voting message calculate the final votes of the nodes according to Formula (11). Afterward, they broadcast to each other the node identifier with the most votes. In particular, it is worth mentioning that if there are nodes with the same number of votes, the node with the smaller identifier is broadcast. After the node identifier is broadcast, the nodes confirm whether the received voting results are consistent with each other. If more than half of the nodes receive the same node identifier information, then this node will become the primary node for the next round. Finally, the voting information will be saved in the local record.

The voting process is shown in Fig. 6. As mentioned previously, nodes store and send the voting information to the consensus-node group's nodes and the observation-node group's nodes. Afterward, nodes receiving the information perform vote counting and then broadcast and confirm the message to each other to select the primary node for the next round.

The main algorithm for the primary node's selection is described below. Through the primary node selection algorithm, when there is a node in the master node group with the highest credit value and serving as the primary node for many times, the nodes in the consensus node group and the observation node group can jointly decide the next round of primary nodes. Doubtlessly, most of them will not choose to vote for the node that serves as the primary node for many times, which avoids the generation of "oligopoly" nodes and improves the enthusiasm of other nodes.

CG-PBFT algorithm flow

The CG-PBFT algorithm's overall flow is described in Fig. 7 below. The initial credit values of all nodes are set to 0.7. In the first round, all nodes in the system are randomly divided into two equal groups: the master-node group and the consensus-node group. After grouping is completed, since there is no observation-node group at this point, only the consensus-node group needs to vote for the master-node group's nodes. Starting in the second round, when nodes already have different credit values, they are grouped by the optimized three-way quick sorting algorithm and reselect the primary node.

In the whole voting process, if any node acts maliciously, does not vote, votes incorrectly, or does not respond to the voting message, this node's credit value is halved. As a result, the group and credit ranking of the nodes will be changed. Therefore, voting will be restarted after updating the local voting records Num_v_i, $TNum\_v_i$, Num_GC_i and C_rank_i. After voting is completed, the CG-PBFT algorithm enters the consensus stage. To improve the consensus efficiency, the PBFT consensus process is optimized, and the commit stage is removed. Generally, when view switching occurs, the commit phase ensures the consistency of a node's status in the system. Specifically, we use the transaction serial number and block height to determine the block state. More precisely, when the primary node acts maliciously or becomes unavailable, resulting in the need for view change, the system will begin the process of revoting to select the primary node. After a transaction request is successfully executed, each node will correspondingly update the successful records by adding 1 to the records Num_v_i, $TNum\_v_i$, Num_c_i, $TNum\_c_i$, Num_GM_i, Num_GC_i and Num_GO_i of nodes in different groups. After updating all the records, the system saves them in local. Next, the node credit values are calculated based on the credit evaluation model. Finally, the system determines

Algorithm 2. Primary node selection algorithm

```
Input: $groupArray$ (nodes' credit grouping array)
Output: $i$ (primary node's identifier)
 1:  $Sum\_v = \{\}$  // Define the voting result variable
 2:  $for\ each\ cnode\ in\ CGroup\ and\ each\ onode\ in\ OGroup$: // Nodes in the consensus-node group and observation-node
 3:     $for\ each\ pnode\ in\ PGroup$:                          group vote for the nodes in the master-node group
 4:          $vote = \{"vote": p, "group": g, "voted": i\}$ // Create a voting message
 5:          $sendVote(pnode, vote)$  // Send the voting message
 6:     $end\ for$
 7:       $totalVotes = Pc * Sumv_{GC(pnode)} + Po * Sumv\_GO(pnode)$  // Count the total votes
 8:     $Sum\_v[i] = totalVotes$ // Store the number of votes in $Sum\_v$
 9:       $i = findLeader(Sum\_v)$// Select the primary node with the most votes and the smallest identifier
10:     $broadcastLeader(i)$ // Broadcast the primary node
11:  $end\ for$
```

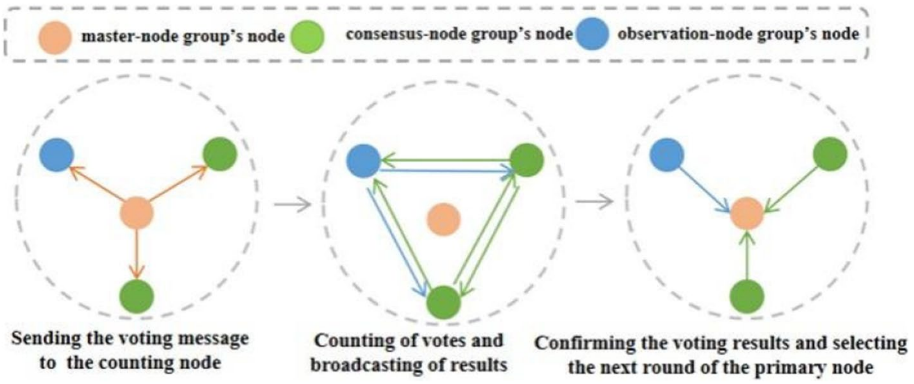
master-node group's node
consensus-node group's node
observation-node group's node
Sending the voting message to the counting node
Counting of votes and broadcasting of results
Confirming the voting results and selecting the next round of the primary node

Fig. 6 The voting process

whether nodes have been in the observation-node group many times and have lower credit rankings; if so, they are excluded from the system. Furthermore, the number and the credit rankings of nodes are updated again. Finally, the current round ends, and we determine whether to continue consensus.

Algorithm analysis

Security analysis

The proposed CG-PBFT algorithm is an improved version of the PBFT algorithm; Essentially, the fault tolerance is the same as PBFT algorithm, which tolerates no more than one-third of malicious nodes. Similar to the PBFT algorithm, its security relies on digital signature, time stamp and Merkle tree technologies to ensure the security of messages. In addition, the proposed CG-PBFT algorithm introduces a credit grouping model and a voting mechanism, which not only divide nodes into three groups but also assign different privileges to nodes in different groups. In particular, nodes with higher credit values are elected to act as the primary nodes. At the same time, this method prevents lower credit values' nodes from entering consensus stage. Furthermore, it eliminates "stubborn" nodes to ensure the system's reliability and security. According to Credit evaluation model section, as nodes' credit values are a comprehensive evaluation of their behavior and status in consensus, this evaluation lasts from the start of voting to the completion of consensus. The penalty mechanism for nodes' credit values effectively limits the malicious behavior of nodes. Through the above analysis method, we can conclude that the proposed CG-PBFT algorithm is able to carry out consensus while ensuring the security of the system.

Communication overhead analysis

As the description in PBFT consensus section shows, messages need to be sent in three phases of the PBFT algorithm, and the communication overhead T_1 for the whole consensus phase is the sum of these three phases:

$$\begin{aligned} T_1 &= (n-1) + (n-1)^2 + n*(n-1) \\ &= 2n*(n-1) \end{aligned} \tag{12}$$

In contrast, the communication overhead of the proposed CG-PBFT algorithm mainly lies in the voting process and consensus process. More precisely, assume that there are n nodes in the system, the number of nodes in the master-node group is n_1, the number of nodes in the consensus-node group is n_2, and the number of nodes in the observation-node group is n_3, where $a = \frac{n_1}{n}$, $b = \frac{n_2}{n}$, and $c = \frac{n_3}{n}$. On the one hand, as described in Fig. 6, three stages of the voting process need to send messages. Therefore, when the voting information is sent to the counting nodes, the number of times of communications among nodes is (n_2+n_3). When the voting results are broadcast after calculating the number of votes, the number of times of communications among nodes is $(n_2+n_3)*(n_2+n_3\text{-}1)$. When the voting results are confirmed, the number of times of communications among nodes is (n_2+n_3). As a result, the communication overhead of the voting process is:

$$\begin{aligned} T_v &= (n_2+n_3) + (n_2+n_3)*(n_2+n_3-1) + (n_2+n_3) \\ &= (n_2+n_3)*(n_2+n_3+1) \end{aligned} \tag{13}$$

On the other hand, during the consensus process, messages need to be sent in the pre-preparation and preparation phases. Similarly, the number of times of communications among nodes is $(n_1+n_2\text{-}1)$ and $(n_1+n_2\text{-}1)^2$. As a result, the communication overhead in the consensus process is:

$$\begin{aligned} T_c &= (n_1+n_2-1) + (n_1+n_2-1)^2 \\ &= (n_1+n_2)*(n_1+n_2-1) \end{aligned} \tag{14}$$

In summary, the whole algorithm's communication overhead T_2 is:

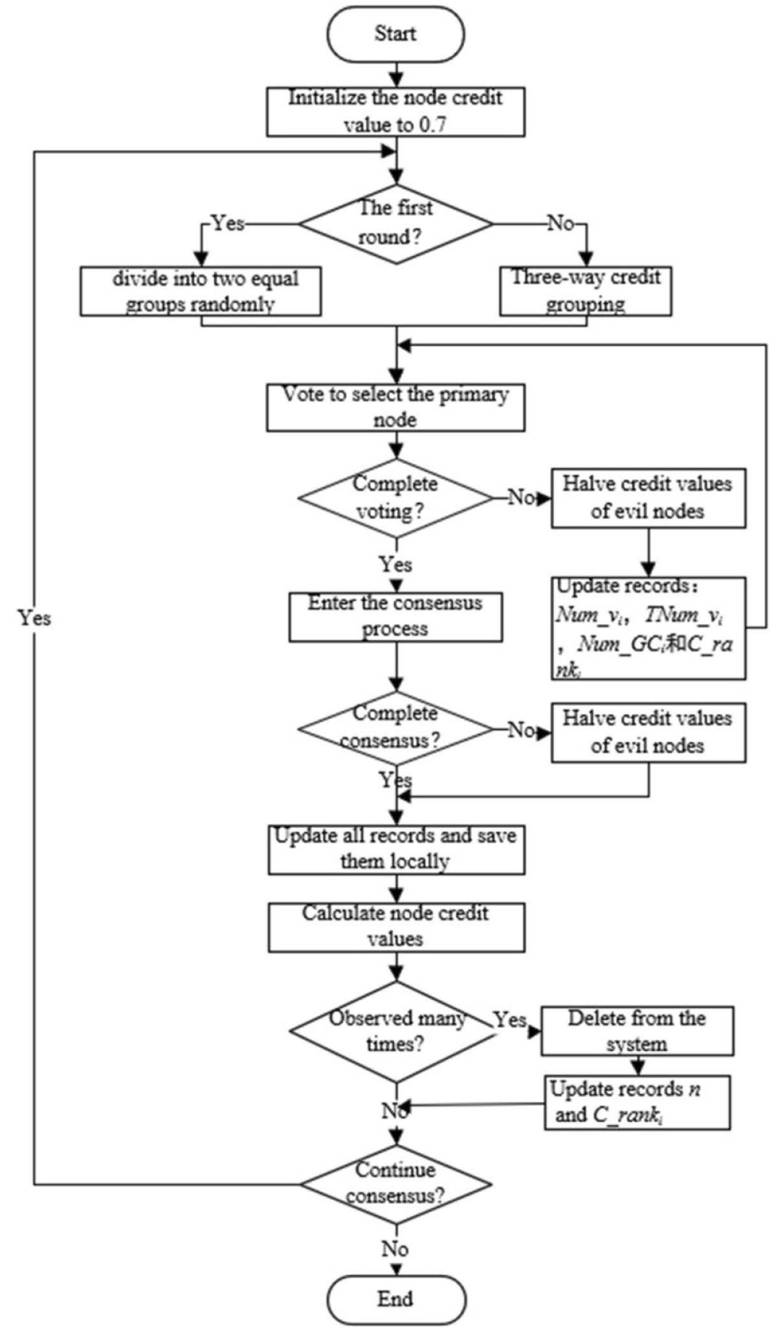
Start
Initialize the node credit value to 0.7
The first round?
Yes
No
divide into two equal groups randomly
Three-way credit grouping
Vote to select the primary node
Complete voting?
No
Halve credit values of evil nodes
Yes
Update records: Num_v_i, TNum_v_i, Num_GC_i和C_rank_i
Enter the consensus process
Complete consensus?
No
Halve credit values of evil nodes
Yes
Update all records and save them locally
Calculate node credit values
Observed many times?
Yes
Delete from the system
Update records n and C_rank_i
No
Continue consensus?
Yes
No
End

Fig. 7 CG-PBFT algorithm flow

$$\begin{aligned} T_2 &= T_v + T_c \\ &= (n_2+n_3)*(n_2+n_3+1)+(n_1+n_2)*(n_1+n_2-1) \\ &= (a^2+b^2+2b)*n^2+(c-a)*n \end{aligned} \quad (15)$$

According to Eqs. 12 and 15, the communication overhead ratio T between the proposed CG-PBFT algorithm and PBFT algorithm is obtained:

$$T = \frac{T_2}{T_1} = \frac{(a^2+b^2+2b)*n^2+(c-a)*n}{2n*(n-1)} \quad (16)$$

$$\leq \frac{(a+b)^2*n^2+c*n}{2n*(n-1)} = \frac{(1-c)^2*n^2+c*n}{2n*(n-1)} \quad (17)$$

where the value of c is in the range [0,1). We set the range of n from 4 to 28, and the three-dimensional surface diagram of the communication overhead ratio is obtained, as shown in Fig. 8, the x-axis is c, the y-axis is n, and the z-axis is the T.

In the given range, regardless of how c and n change, the value of T' is always less than 1, and the maximum value is no more than 0.7. In other words, the communication overhead of the proposed CG-PBFT algorithm is always less than that of the PBFT algorithm. In particular, as the number of nodes increases, the communication overhead ratio decreases. In addition, due to the introduction of a voting mechanism in the proposed CG-PBFT algorithm, the primary node selection method is more secure and reliable, which greatly reduces the view switching's probability; thus, the proposed CG-PBFT algorithm is more efficient in practical applications.

Simulation experiment results and analysis

The proposed CG-PBFT algorithm's effectiveness is verified through simulation experiments, which include a performance analysis and comparison of the proposed credit evaluation model, the node identity conversion between node groups, and the algorithm's throughput and delay. Specifically, the simulation experiments use an Intel(R) Core(TM) i7-7500U processor with a 2.70 GHz clock rate, an 8 GB memory environment, the Windows 10 operating system and the Golang language to implement the simulation test. According to the the credit evaluation model's requirements, direct credit value is the main evaluation index of a node's credit value, which should be greater than indirect credit value and historical credit value. In the direct credit value, the consensus credit's weight is the largest. In the indirect credit value, the weight of nodes in the master node group should be higher than that of nodes in the consensus node group. Different weight values only change the proportion of direct credit value, indirect credit value and historical credit value in the comprehensive credit value, and have little effect on the node's identity conversion. According to this, Under the condition that $\lambda+\mu>\alpha+\beta$ and λ is the largest, three different values of direct credit value's weight $\lambda+\mu$, indirect credit value weight's $\alpha+\beta$ and historical credit value's weight η are tested, which are (0.4, 0.35, 0.25), (0.5, 0.3, 0.2) and (0.6, 0.25, 0.15) respectively. When the three values change, the change of node's credit value for 5 rounds of testing is shown in Fig. 9.

With the increase of consensus rounds, the node's credit value shows an upward trend under different parameter values. Compared with the other two curves, when the node's weight is set to (0.6, 0.25, 0.15), the credit value

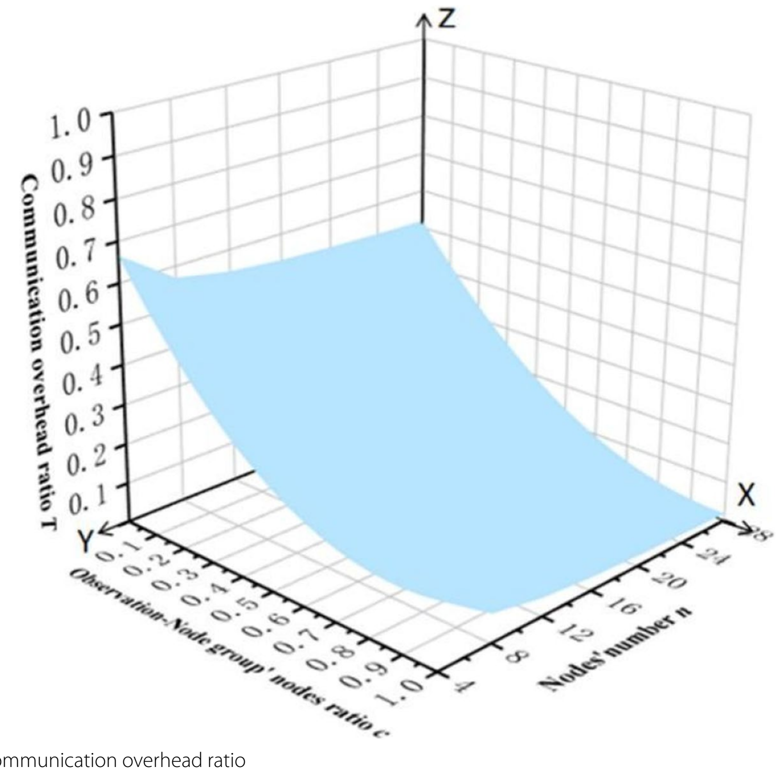

Fig. 8 3D surface of the communication overhead ratio

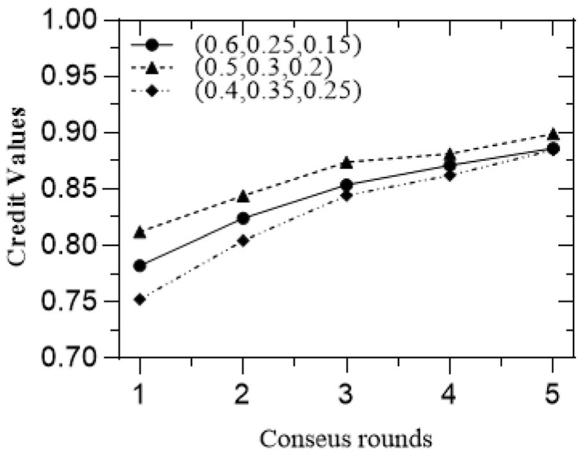

Fig. 9 Weight change test

changes the smoothest, indicating that its credit value evaluation is the most stable. Therefore, this paper sets the experimental parameters: direct credit value's weight is 0.6, indirect credit value's weight is 0.25, and historical credit value's weight is 0.15. The individual values of λ, μ, α, β have little influence on the overall credit value, more precisely, the detailed parameters can be seen in Table 3 below.

Table 3 Detailed experimental parameters

Parameters	Value
C_con weight λ	0.35
C_vot weight μ	0.25
C_act weight α	0.15
C_inc weight β	0.1
Num_GM weight γ	0.6
C_his weight η	0.15

Changes in the node credit value

To verify the validity of the credit evaluation model clearly, we set up four nodes with different attributes, namely, good nodes, malicious voting nodes, non-malicious consensus failure nodes and malicious consensus nodes. We test a total of 10 rounds of the consensus process and track the changes in credit values and the dynamic conversion between the four types of nodes.

More precisely, node 1 is a "good" node, which normally reaches consensus and votes in each consensus round. In contrast, node 2 is set to not vote in the second and seventh rounds but always normally participates in consensus. Similarly, node 3 is set as a non-malicious consensus failure node, which becomes inactive in the fourth round but returns to normal in the later consensus rounds. Node 4 is set as a Byzantine node, which continues to act maliciously from the fifth round onwards. The results of the credit value changes and intergroup dynamic conversions of the four types of nodes are shown in Fig. 10a and b, respectively.

As exhibited in Fig. 10a, the initial credit values of the four nodes are set to 0.7. Next, there is a significant change in the nodes' credit values after the first round of consensus. Notably, the credit values of node 4 and node 1 rise by the same amount from the initial value. In contrast, the credit values of node 2 and node 3 decline from their initial values. This is because node 1 and node 4 are randomly assigned to the consensus-node group, while node 2 and node 3 are randomly assigned to the master-node group; compared to node 1 and node 4, they lack voting credit rewards. For this reason, these results provide substantial evidence that the initial nodes randomly assigned to the master-node group will not always be in the master-node group; they may appear in the consensus-node group in the second round. Generally, they need to complete consensus through their own

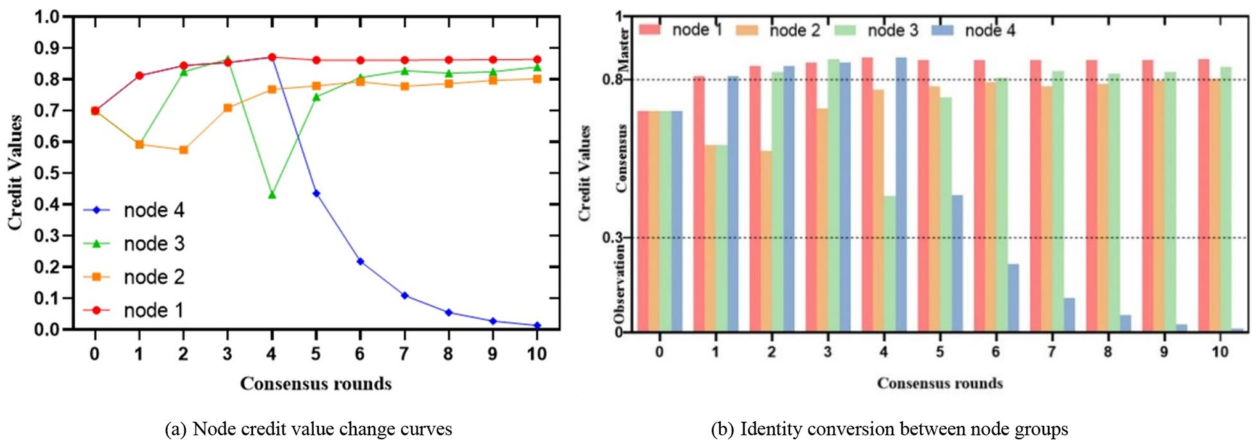

Fig. 10 Node credit value changes

efforts to move into the master-node group. This reduces the chance that nodes in the master-node group will act maliciously; while increases the node consensus motivation and dynamicity.

In each round of consensus, node 1 has no voting mistakes or malicious actions, and it performs well, so its credit value continues to rise steadily. Specifically, at the beginning, its credit value increases significantly and then slows down after the node moves into the master-node group, but the node always remains in the master-node group and its credit value is the highest. Node 2 fails to vote in the second and seventh rounds, so its credit value decreases in these rounds. In addition, due to the failure of node 3 in the fourth round, its credit value quickly falls to the lowest point; however, it does not fail in subsequent rounds, so its credit value rises sharply at first and then slowly after rising to a certain height. Last, node 4 persistently acts maliciously after the fifth round, and its credit value drops rapidly, halving several times to approach zero.

In Fig. 10a, the changes in four different nodes confirm that the credit evaluation model can reliably assess the behavior of nodes. It can not only prevent some nodes' credit values from being too high so that other nodes lose enthusiasm but also avoid over penalization of non-malicious consensus nodes. Moreover, it gives the non-malicious consensus nodes the opportunity to quickly recover their credit values to regain the right to participate in consensus and be elected as the primary node.

The Fig. 10b shows that nodes are all in the consensus-node group at the beginning. Afterward, node 1 enters the master-node group by performing well, and it remains in the master-node group for a long time. Conversely, the behavior of node 2 largely remains good; however, there are two voting failures, so it still stays in the consensus-node group and is unable to enter the master-node group. It should be noted that node 3 initially performs well, and it enters the master-node group from the consensus-node group. However, due to one consensus failure, it is moved to the consensus-node group. Interestingly, through continuous effort, it enters the master-node group again. However, node 4 continues to act maliciously; as a result, it is moved from the master-node group to the consensus-node group and then to the observation-node group; finally, it remains in the observation-node group.

The node identity conversions between the three node groups are shown in Fig. 10b. This verifies that dynamic identity conversion between node groups can be effectively realized by giving credit rewards and punishments to nodes through a credit evaluation mechanism.

The numbers of times the four different-attribute nodes appear in the different node groups are shown in Fig. 11.

Node 1 performs well; undoubtedly, it is most often in the master-node group. Node 3 breaks down once, so it has the second-highest frequency in the master-node group. Node 2 has the least frequency in the master-node group because of two nonvoting occurrences. Finally, node 4 continues to act maliciously after the 5th round, so it has the highest frequency in the observation-node group. The results in Fig. 11 provide substantial evidence that the proposed credit evaluation model can provide a good foundation for the credit grouping mechanism, which can strengthen the credibility of the primary node's selection and continuously weaken the influence of malicious nodes.

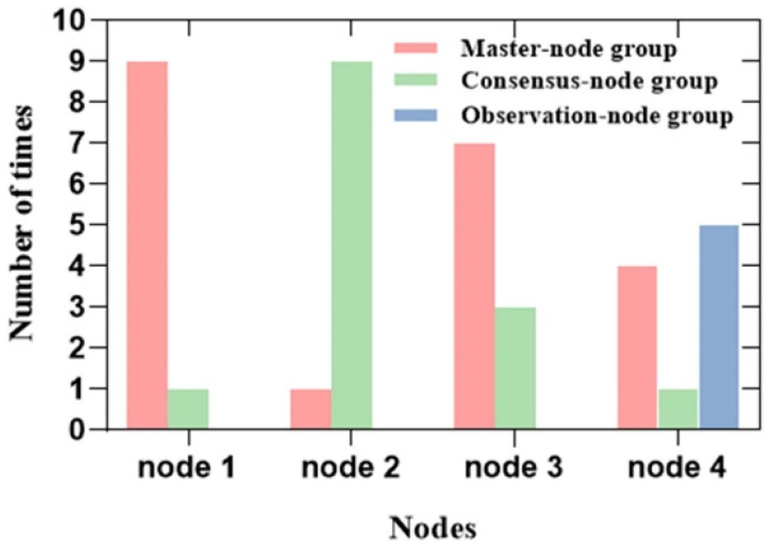

Fig. 11 Numbers of times the 4 nodes are in each group

Throughput

To verify the proposed CG-PBFT algorithm's efficiency, we compared the throughput of several typical algorithms, including the PBFT algorithm, the Paxos algorithm [26], the HotStuff algorithm [27], and the GPBFT algorithm proposed in [20]. More precisely, the experiments test the throughput of the five algorithms for 4, 8, 12, 16, 20 and 24 nodes, and each group of experimental tests is conducted 50 times. The average results are presented in Fig. 12.

At the beginning, the throughput of the PBFT algorithm decreases with the maximum degree, and that of the proposed CG-PBFT algorithm decreases with the minimum degree. Afterward, when the number of nodes varies from 4 to 24, the throughput of the five algorithms decreases continuously and remains stable eventually. In particular, the proposed CG-PBFT algorithm maintains 35 TPS, with a 51.3% increase in average throughput over the PBFT algorithm. In addition, due to the need of pairwise communication between nodes and the lack of an exit mechanism for malicious nodes, the PBFT algorithm has the largest throughput decrease. Contrastively, HotStuff algorithm uses star topology communication to resist malicious nodes, which is greatly limited by hardware resources and devices, and it also lacks an exit mechanism for malicious nodes, which leads to low throughput and an obvious downward trend in the experiment. In contrast, Paxos algorithm relies on a single primary node to collect and distribute messages, with the increase of the number of system nodes, the load pressure of the primary node increases, resulting in a continuous decline in throughput. the PBFT 、the Paxos and the HotStuff algorithm all exhibit an obvious decreasing trend of throughput. However, the GPBFT and the proposed CG-PBFT algorithm have a slow decrease in throughput, and the throughput eventually remains almost flat. It is worth mentioning that the GPBFT algorithm also uses credit grouping to improve the PBFT algorithm, but it has low throughput due to the lack of exit mechanism for nodes with low credit values. In particular, the experimental results show that the CG-PBFT algorithm has considerably high efficiency; it can not only efficiently carry out fast grouping but also adjust the consensus scale to improve the system's operation efficiency.

Delay

Under the same experimental conditions, we test the delay of the five algorithms and conduct a comparative analysis. The results are obtained by taking the average of 50 tests, as shown in Fig. 13.

The five algorithms have little difference in delay at the beginning. However, when the number of nodes reaches 16, due to the increase in internode communication, the delays of the PBFT, Paxos, and HotStuff algorithms start to increase rapidly and are much greater than those of the other two algorithms. Among them, the Paxos algorithm's communication complexity is lower than PBFT algorithm's communication complexity, so its delay is lower than that of PBFT algorithm, and due to the chain consensus method and multi-master node consensus, the delay of HotStuff algorithm is lower than that of Paxos algorithm. However, due to the limitation of hardware equipment, its delay enters the bottleneck period. Because the GPBFT and CG-PBFT algorithm use grouping consensus to optimizes the amount of communication between nodes, the system's delay rises slowly and

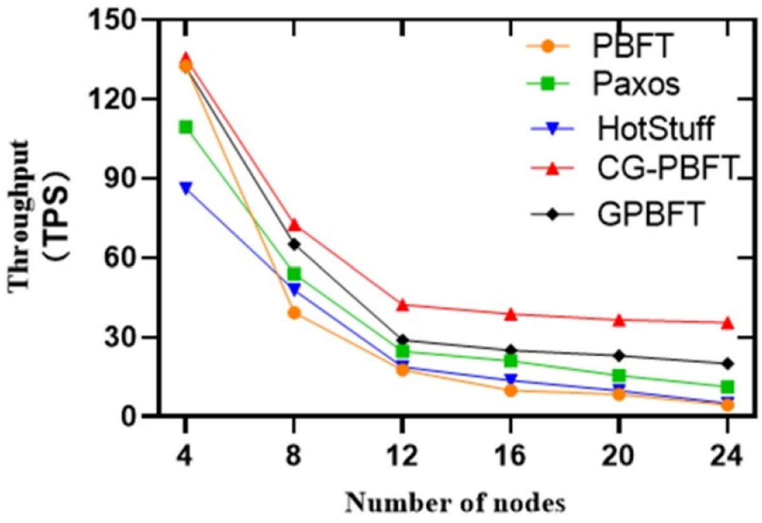

Fig. 12 Throughput test comparison

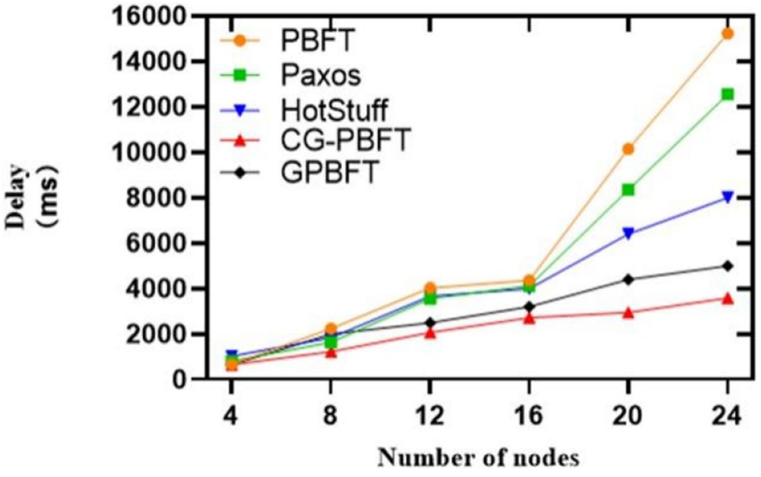

Fig. 13 Delay test comparison

eventually levels off gradually. Compared to the PBFT algorithm, the proposed CG-PBFT algorithm reduces the average latency by 64.5%. However, the GPBFT algorithm is unable to evict malicious nodes from the system, and the number of groups increases with the number of nodes. Therefore, its delay is higher than that of the proposed CG-PBFT algorithm. In particular, the experimental results show that the CG-PBFT algorithm can greatly improve system's operation efficiency, which can enable the system's transactions to be quickly completed.

Conclusion

In summary, we propose an efficient CG-PBFT algorithm based on credit grouping. According to the node's behavior in the system, we design a credit mechanism to accurately measure the quality of nodes. In addition, we use an optimized quick grouping mechanism to restrict node's privileges and introduce a voting mechanism to increase the security of the selection of the primary nodes. Furthermore, we adopt a credit reward and punishment mechanism for nodes to encourage them to participate in appropriate system behavior. The experimental results show that the proposed method has higher security, higher throughput and lower delay, so it is able to improve the system's operation efficiency. Moreover, it improves the system's consensus positivity. In the face of application scenarios with a large number of nodes and the need to quickly complete the consensus, the CG-PBFT algorithm is used for credit grouping, which can screen out high-quality consensus nodes, reduce the consensus's scale, and ensure system's security and efficiency.

However, the proposed CG-PBFT algorithm still has deficiencies in terms of credit evaluation indicators and communication complexity and needs further investigation. In future research, we will continue to optimize the credit evaluation indicators and the communication topology to reduce the algorithm's communication complexity to the linear level. In addition, a direct proof of the usability of the proposed CG-PBFT algorithm in practical application scenarios was not obtained, and the detailed deployment process could be clarified and further investigated.

Authors' contributions

Juan Liu and Xiaohong Deng contributed the main ideas and wrote the main manuscript tests. Wangchun Li and Kangting Li established the evaluation and provided the simulation experimental ideas. All authors reviewed the manuscript.

Funding

This work was supported by the National Natural Science Foundation of China (No. 61762046, No.62166019), the National Natural Science Foundation of Jiangxi Province (No. 20224BAB202019) and the Science and Technology Research Project of the Education Department of Jiangxi Province (No. GJJ218506).

Availability of data and materials

The coding data that support the findings of this study are available from the corresponding author upon request.

Declarations

Competing interests

The authors declare no competing interests.

Received: 6 November 2023 Accepted: 20 March 2024
Published online: 25 March 2024

References

1. Seven S, Gang Y, Soran A, Onen A, Muyeen S (2020) Peer-to-peer energy trading in virtual power plant based on blockchain smart contracts. IEEE Access 8:175713–175726
2. Xia Q, Dou W, Guo K, Liang G, Zuo C, Zhang F (2021) Survey on blockchain consensus protocol. J Softw 32(2):277–299

3. Liu Y, Fang Z, Cheung MH, Cai W, Huang J (2022) An incentive mechanism for sustainable blockchain storage. IEEE Trans Netw 30(5):2131–2144
4. Shao Q, Jin C, Zhang Z, Qian W, Zhou A (2018) Blockchain: architeture and research progress. Chin J Comput 41(5):969–988
5. Rehman M, Javed IT, Qureshi KN, Margaria T, Jeon G (2023) A cyber secure medical management system by using blockchain. IEEE Trans Comput Soc Syst 10(4):2123–2136
6. Yang Y, Zou Y, Xu M, Xu Y, Yu D, Cheng X (2022) Distributed consensus for blockchains in internet-of-things networks. Tsinghua Sci Technol 27(5):817–831
7. Woo J, Fatima RF, Kibert CJ, Newman RE, Tian Y, Srinivasan RS (2021) Applying blockchain technology for building energy performance measurement, reporting, and verification (MRV) and the carbon credit market: a review of the literature. Build Environ 205(9):108199–108209
8. Zhu C, Xu D, Ren N, Cui H, Zhao Y (2021) Model and implementation of geographic data transaction certificate and copyright protection based on blockchain and digital watermarking. Recent Adv Comput Sci Commun 50(12):1694–1704
9. Deng X, Wang Z, Li J, Wang J, Li K (2022) Comparative research on mainstream blockchain consensus algorithms. Appl Res Comput 39(1):1–8
10. Zhang S, Lee J (2020) Analysis of the main consensus protocols of blockchain. ICT Express 6(2):93–97
11. Li S, Huang L, Deng X, Wang Z, Liu H (2021) Consortium chain consensus algorithm based on credit. Appl Res Comput 38(08):2284–2287
12. Liu Y, Lan Y, Li B, Miao C, Tian Z (2021) Proof of Learning (PoLe): empowering neural network training with consensus building on blockchains. Comput Netw 201:108594
13. Li S, Xiong W, Deng X (2023) Byzantine fault-tolerance consensus algorithm based on perfect binary tree communication. J Electron Inf Technol 45(07):2484–2493
14. Castro M, Liskov B (2002) Practical byzantine fault tolerance and proactive recovery. ACM Trans Comput Syst 20(4):398–461
15. Qi W, Si P, Gu C (2023) Improved PBFT consensus algorithm for multi-scenario Internet of Things. Appl Res Comput 41(3):9–20
16. Wang P, Wang X, Shen Y, Wang J, Xiong X (2023) PBFT optimization algorithm based on community contribution. Math Biosci Eng 20(6):10200–10222
17. Zong W, Feng W, Huang M, Feng S (2023) ST-PBFT: an optimized PBFT consensus algorithm for intellectual property transaction scenario. Electronics 12(325):325
18. Liu S, Zhang R (2023) P-PBFT: an improved blockchain algorithm to support large-scale pharmaceutical traceability. Comput Biol Med 154:106590
19. Zheng X, Feng W, Huang M, Feng S (2021) Optimization of PBFT algorithm based on improved C4.5. Math Probl Eng 4(1):1–7
20. Wang Y, Zhong M, Cheng T (2022) Research on PBFT consensus algorithm for grouping based on feature trust. Sci Rep 12(1):1–12
21. Ren X, Tong X, Zhang W (2023) Improved PBFT consensus algorithm based on node role division. J Comput Commun 9(2):20–38
22. Liu S, Zhang R, Liu C, Xu C, Zhou J, Wang J (2022) Improvement of the PBFT algorithm based on grouping and reputation value voting. Int J Digit Crime Forensics 14(3):1–15
23. Xu G, Bai H, Xing J, Luo T, Xiong N, Cheng X, Liu S, Zheng X (2022) SG-PBFT: a secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent Internet of vehicles. J Parallel Distrib Comput 164(6):1–11
24. Wang R, Xing C, Xu Q, Yuan S (2021) Efficient byzantine fault tolerant algorithm with supervsion mechanism. Comput Eng Appl 57(18):142–148
25. Tang S, Wang Z, Jiang J (2022) Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain. Sci Rep 12(1):4426
26. Lamport L (1998) The part-time parliament. ACM Trans Comput Syst 16(2):133–169
27. Yin M, Malkhi D, Reiter M, Gueta, G, Abraham I (2019) HotStuff: BFT consensus with linearity and responsiveness. In: Proceedings of the annual ACM symposium on principles of distributed computing. p 347–356