Journal of Computers Vol. 33 No. 2, April 2022, pp. 11- 23
doi: 10.53106/199115992022043302002

Research on Practical Byzantine Fault Tolerant Algorithm Based on Trust Mechanism

Hui Pang, Yanan Li*, Xiumei Wen, Yingxue Mu

Department of Computer Science and Technology, Hebei University of Architecture,
Big Data Technology Innovation Center of Zhangjiakou, Zhangjiakou, Hebei, China
{pyanxuan, liyanan4912, xiumeiwen}@163.com.cn; 526110454@qq.com

Received 21 April 2021; Revised 19 July 2021; Accepted 20 October 2021

Abstract. The consensus algorithm is one of the core technologies of the blockchain, which determines how the nodes in the blockchain network reach a consensus and enable them to jointly maintain a piece of data. This paper focuses on the Practical Byzantine Fault Tolerant Algorithm (PBFT Algorithm), and proposes a Practical Byzantine Fault Tolerant Algorithm based on Trust Mechanism (TM-PBFT Algorithm) to solve the problems of its low fault-tolerant rate and high communication cost. This algorithm supports voting for trusted nodes before the consensus cycle starts. In order to make nodes actively vote and vote for reliable nodes, this paper models the trust degree of nodes. In addition, the time factor is added to the Shapley value to design a new reward and punishment scheme, which makes the distribution of income of each node more reasonable and further encourages nodes to vote for reliable nodes. As the system runs for a long time, the probability of malicious nodes becoming master nodes decreases. Finally, a comparison experiment between the proposed algorithm and the PBFT algorithm shows that the fault tolerance and throughput of the TM-PBFT algorithm are higher than that of the PBFT algorithm, and the communication bandwidth overhead is lower than that of the PBFT algorithm.

Keywords: blockchain, consensus algorithm, TM-PBFT algorithm, shapley value

1 Introduction

The term Bitcoin was proposed by Satoshi Nakamoto in 2008 [1], and the blockchain was first known to everyone as an accompanying technology of Bitcoin. In recent years, blockchain technology has been separated from Bitcoin and has been studied by scholars at home and abroad as a separate technology. With the development of blockchain, consensus algorithms have also become the key research objects of scholars, and various consensus algorithms have emerged in an endless stream. The most commonly used mainly include Proof of Work (PoW) [2], Proof of Stake (PoS) [3], Delegated Proof of Stake (DPoS) [4], and Practical Byzantine Fault Tolerant (PBFT) algorithms [5].

Based on the de-trust characteristics of blockchain technology, Byzantine error is the first problem to be solved by the consensus algorithm in the blockchain. Byzantine error signifies how to ensure the availability, reliability and data integrity, and make correct decisions of the system in the presence of malicious nodes in the system.

The PBFT algorithm was first proposed to solve the Byzantine General Problem. In recent years, it has been applied to the blockchain alliance chain. The fault tolerance rate of this algorithm is 1/3, that is, when the faulty node in the system does not exceed 1/3 , it can also ensure the flexibility and security of the consensus process. Although the consensus efficiency of the alliance chain has been improved after the PBFT algorithm is adopted, the communication cost of the PBFT algorithm is still very high. As the number of nodes in the system increases, the amount of communication that required for its consensus shows a quadratic increase. The more serious problem is because of the limitation of network bandwidth. If the number of nodes reaches a certain value, it will also cause communication bottlenecks, so the performance of the PBFT algorithm will also decrease.

In response to the problems of PBFT, a large number of research results have emerged at home and abroad. Literature [6] designed a random walk algorithm to compress the communication bandwidth of PBFT, but the data quality cannot be guaranteed because the selection is at random. Literature [7] proposed a cross-fault-tolerant scheme to increase the system's fault-tolerant rate to 2f+1. There are advantages when the number of system nodes is small. However, as the number of nodes increases, the consensus efficiency drops sharply, and it is not suitable for systems with many nodes. Literature [8] proposed a validated asynchronous Byzantine agreement (VABA) algorithm based on random oracle model verification. It supports the use of multiple parallel nodes to make proposals in each consensus process. And it randomly selects one of them as the result, but this randomly

* Corresponding Author

selected parallel scheme will lead to security vulnerabilities. Literature [9] proposed an attack scheme to detect the path, which can improve the security of the system. However, the node that fails is defined as the node that launches the attack, which is too expensive in terms of security.

The above algorithms have improved the PBFT algorithm to a certain extent. However, the trust degree of the nodes is not evaluated and statistics, as well as the security of the master node selection and the stability of the consensus process cannot be guaranteed. This paper starts with the trust degree of nodes, and proposes a Practical Byzantine Fault Tolerance Algorithm based on Trust Mechanism (TM-PBFT). The algorithm joins the voting process ahead of the consensus process. According to the voting results, the top H nodes are selected to form a list of trusted nodes. And in order to make nodes actively vote for reliable nodes, the trust of nodes is modeled, and rewards and punishments are designed. With the long-term operation of the algorithm, the trust of malicious nodes gradually decreases and cannot be elected as the master node. This reduces the probability of malicious nodes becoming the master node. Through experimental analysis and comparison, the algorithm can quickly identify malicious nodes in the blockchain network. The security and throughput of the system are improved, and the communication bandwidth of the system is reduced. It also improves the consensus efficiency of the blockchain, and further enables it to be applied to more fields.

2 PBFT Algorithm

In 1999, Castro and Liskov proposed the Practical Byzantine Fault Tolerant Algorithm (PBFT) in order to solve the Byzantine Generals Problem, which is also considered to be the most classic algorithm for solving this problem. In a distributed system, if there are Byzantine faulty nodes, an agreement cannot be reached and the general consensus algorithm cannot solve the problem but PBFT can. Assuming that there are a lot of malicious nodes in the system, the PBFT algorithm requires that the number of summary points N is greater than or equal to 3f+1. In addition, we will introduce the consistency protocol, view-change protocol, and checkpoint protocol of the PBFT algorithm.

2.1 Consistency Protocol

The core of the PBFT algorithm is the consensus protocol, which is mainly used to ensure that the data in the blockchain remains consistent and correct. There are three roles in this algorithm: client, master nodes and replica nodes.

As shown in Fig. 1, the algorithm flow of the PBFT algorithm includes: request process, pre-prepare, prepare, commit and reply process. Among them, Replica3 indicates that the node is a faulty node and cannot communicate.

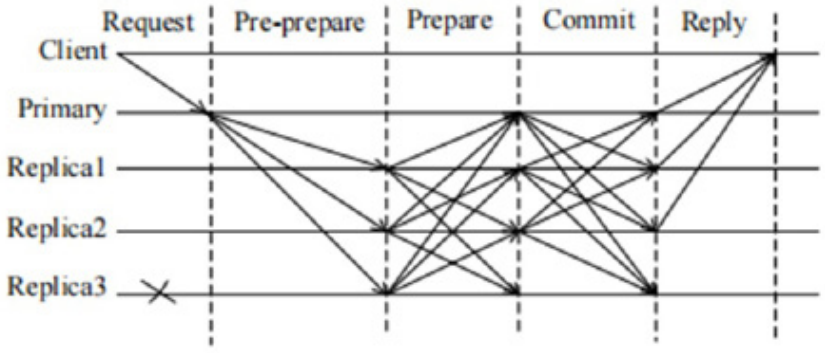

Fig. 1. PBFT algorithm execution process

Request process. The request process includes two steps: selecting the master node and sending the request. First, the master node must be selected through a random algorithm or a rotation. If it is a rotation, it needs to be determined by the collection of the number of nodes and the view number, that is, "m = v mod |N|". Among them, "m" is the main node number, "v" is the view number, and "|N|" is the total number of nodes. After that, the client will send a request message to the master node, the message format is <Request. operation, timestamp, client>. The Request contains the message content and message digest of the request. Operation is the specific operation

of the request, and timestamp is the time when the request is sent.

Pre-prepare stage. If the client sends an illegal request, it will be discarded. If it is a correct request, firstly the master node needs to assign a number "n" to the request sent by the client, and then send a pre-prepare message to all replica nodes in the system. The message format is <<Pre-prepare, view, n, digest>, message>. The "view" is the view number, "n" is the proposal number, and "digest" is the digital digest of the request message. Sign the master node of <Pre-prepare, view, n, digest >.

Prepare stage. After the replica node has received at least 2f verified pre-preparation messages, it can enter the preparation phase. The pre-preparation message sent by the master node is first verified, and if it is an illegal request, it is discarded. If it is a correct request, the replica node sends a preparation message to other nodes including the master node. The message format is <Prepare, view, n, digest, i >. The "view", "n", and "digest" are consistent with the total information of the prepared message, and "i" is the number of the replica node that sent the prepared message. For this preparation message, the replica node "i" needs to sign. In addition, when a replica node sends a preparation message, other replica nodes will also send a preparation message to it. Both the pre-preparation message and the preparation message will be added to the log. If the request is not completed, it can be used to restore the request in subsequent view-change.

Commit stage. After all nodes in the system receive the preparation message, they still need to be verified, and if they are illegal, they will be discarded. Otherwise, the replica node will send the submission information to other nodes after receiving 2f+1 preparation messages that have passed the verification. The message format is <Commit, view, n, digest, i >. The "view", "n", "digest", "i" are the same as the content in the preparation message. The commit message needs to be signed by each replica node, and then stored in the log, which can be used to restore unfinished requests later.

The three stages of pre-prepare, prepare, and commit belong to the broadcast process of the algorithm, and it is also known as the three-stage protocol. In general, the master node collects the request information sent by the client, and after the check passes, broadcasts the request to all other replica nodes in the system.

Reply process. After the nodes in the system receive the commit message, they still perform verification. If the replica node receives 2f+1 submission messages that have passed the verification, it means that most nodes in the system have reached a consensus on proposal "n" and are running the client's request operation operation. After all nodes process the request, they return the processed result message to the client. The format of the message is <Reply, view, timestamp, client, i, response>. Response is the processing result after operating on the client request. After the client receives f+1 identical response messages, it means that the system has reached a consensus on the request initiated by the client. On the contrary, the client can consider whether it needs to resend the request to the master node.

2.2 View-Change Protocol

In the consensus process, if the master node fails, the view switching protocol needs to be used to maintain the availability of the system. A view contains a master node, if the master node fails, you need to switch the view, and its number will change from "v" to "v+1", and replace the new master node. The view switching protocol is triggered by the replica node. The trigger conditions include: (1) The latest completed block shall prevail, and within the limited time "T1". No pre-preparation message from the master node has been received; (2) Within the limited time "T2", no new blocks are produced, where "T2> T1". If one of the conditions is met, and the view switching protocol is triggered.

The working process of View-Change is as follows: (1) Start the view-change protocol. It means the new master node enters view "v+1" from view "v", and broadcasts View-Change messages to other nodes; (2) After the replica node receives 2f+1 View-Change messages, it sends a View-Change-Ack message to the new master node in view "v+1", and the new master node enters the New-View stage after receiving the message; (3) New-View requests that the new master node selects the checkpoint as the starting state, and then executes the consistency protocol.

2.3 Checkpoint Protocol

During the long-term consensus process, the nodes in the system will generate a large number of logs, and the checkpoint protocol is used to reduce the storage scale of node log information. In order to reduce memory overhead, all log messages that have passed consensus authentication are released. In addition, if some nodes are not synchronized with other nodes due to problems such as network or failure, the normal operation of the system

will be affected. The checkpoint protocol performs periodic work and can clear the verified certificates after confirming the data consistency of the system nodes.

2.4 Problems with PBFT

Although the consensus performance of the blockchain is greatly improved after the PBFT algorithm is used, it still has the following shortcomings:

The fault tolerance rate is low. The faulty nodes in the entire system cannot exceed 1/3 of the total number of nodes in the system, and the fault tolerance rate cannot be improved.

Communication costs are high. Since the existence of faulty nodes in the system cannot be avoided, the communication cost and time-consuming increase. Assuming that the total number of nodes in the system is "N", the time complexity of communication is $O(N^2)$, after the three-phase protocol, the number of message transmissions is as high as "2N(N-1)". When the number of nodes exceeds a certain number, the performance of the PBFT algorithm will drop sharply, and even a bottleneck will appear.

The master node has a high probability of doing evil. Because the PBFT algorithm has no measures for the selection of the master node, it is likely that malicious nodes will be selected during the selection process, which will affect the operating efficiency of the system.

3 TM-PBFT Algorithm

Through the research of the PBFT algorithm, and aiming at these shortcomings, this paper proposes the TM-PBFT algorithm. In the algorithm of this paper, a voting cycle is added, during which each node can vote to select a list of trusted nodes. Secondly, for making nodes actively vote, the trust degree of nodes is modeled, and design a new reward and punishment scheme after adding the time factor to the Shapley value.

3.1 Establishment of Trusted Node List

The TM-PBFT algorithm supports that in the voting period "V", all nodes "i" in the system must vote. After the voting period "V" ends, the number of votes of each node "i" is counted as "n_i", and is sorted according to the number of votes. Combining the top "H" nodes of the votes into a list of trusted nodes, as shown in Fig. 2.

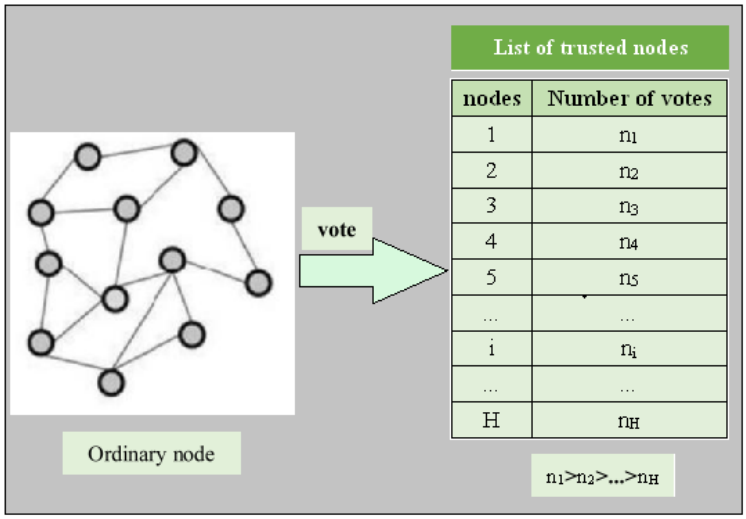

Fig. 2. List of trusted nodes

Among them, each node can only vote once, that is, the total number of nodes is equal to the total number of votes recorded as "N", so "n_i" needs to satisfy formula (1).

$$N = \sum_{i=1}^{\pi} n_i \quad (1)$$

The nodes in the TM-PBFT algorithm can be divided into ordinary nodes, trusted nodes and master nodes. As shown in Fig. 3, it is the network model of the TM-PBFT algorithm.

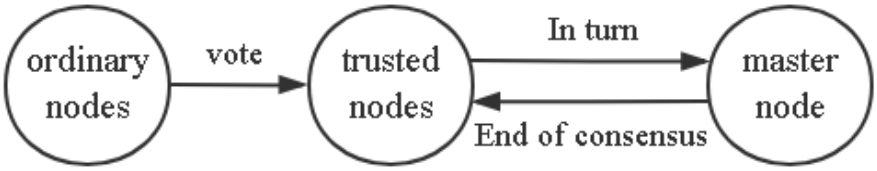

Fig. 3. Network model of TM-PBFT algorithm

As shown in Fig. 4, it is the relationship diagram of the number of nodes in the entire network.

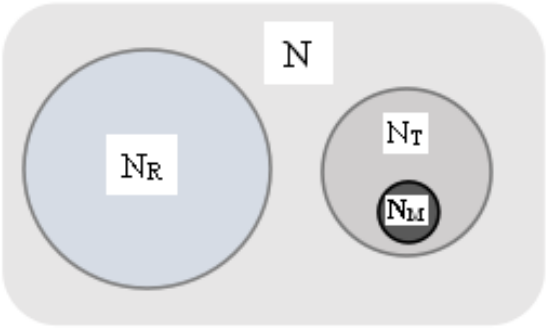

Fig. 4. Graph of the number of nodes

"N" is the number of ordinary nodes. Initially, all nodes in the system are ordinary nodes, and the main responsibility of them is to vote. Trusted nodes are voted by ordinary nodes, and the list of trusted nodes is also the candidate set of master nodes, which number is recorded as "N_T". Only one master node is required for each round of voting, which is assumed by the nodes in the trusted nodes list in turn, mainly responsible for block production and consensus, etc. The number is recorded as "N_M". In addition, there are replica nodes in the system, which are assumed by ordinary nodes whose trust degree exceeds a certain value. They are mainly used in the consensus phase, and the number is recorded as "N_R". Futhermore, in the entire network, except for "N_M" which is fixed at 1, the number of other types of nodes is not fixed. The number of "N" can increase with the increase of network nodes. "N_R" and "N_T" are determined by the values required by the system.

The system will initiate a voting proposal at regular intervals. The voting master node will first broadcast the voting preparation message to the ordinary nodes, and then the ordinary nodes will choose the nodes they want to vote and return the final result to the master node. After the master node counts the results of the votes, it will be broadcast to all nodes. When voting, each node has only one vote, and it can choose other nodes or choose itself. The obtained trusted node list will be used as the candidate set of the master node, which can be selected by the master node for the next voting and consensus.

3.2 Model for Evaluating Node Trust Degree

In order to select more secure and reliable trusted nodes, this section designs a nodes' trust evaluation model, which adds trust attributing to the nodes in the system, and the nodes' trust degree is modeled as follows.

In the TM-PBFT algorithm, a voting period "V" contains multiple consensus periods "T". The number of consensus periods is not fixed and can be adjusted flexibly. A consensus period "T" can be divided into multiple production intervals "t", and the packaging process of a block is completed in one production interval "t".

The trust degree of node "i" in the current production interval "t" is denoted as "$Cr(i)_{pro}$", which can be divided into three parts: the voting trust of node "i" in the current voting cycle is denoted as "$Cr(i)_{vote}$", the performance trust of node "i" in the current voting period "V" is denoted as "$Cr(i)_{beh}$" and the cumulative trust of node "i" at the end of the last voting period "V-1" is denoted as "$Cr(i)_{acc}$". "$Cr(i)_{pro}$" can be defined as:

$$Cr(i)_{pro}=\lambda_1 \times Cr(i)_{vote} + \lambda_2 \times Cr(i)_{beh} + \lambda_3 + Cr(i)_{acc} \quad (2)$$

"λ_1、λ_2、λ_3" are the weight of each trust degree, and need to satisfy formula (3). In addition, "λ_1" is a voting parameter, which can be used to control the degree of dependence on node voting; "λ_2" is the performance parameter, which can be used to control the degree of punishment for doing evil on the node. If the user attaches importance to the performance of the node, the weight can be increased. "λ_3" is the growth parameter, which can be used to control the growth rate of the trust degree of the node, so as not to increase the trust degree too fast. The larger the "λ_1", "λ_2", and "λ_3", the greater the degree of dependence on the corresponding trust degree, and the smaller the value, the smaller the degree of dependence.

$$\lambda_1 + \lambda_2 + \lambda_3 = 1, \quad \lambda_1,\ \lambda_2,\ \lambda_3 \in (0,1) \quad (3)$$

"$Cr(i)_{vote}$" is the voting confidence of node "i" in the current voting period "V", which can be defined as formula (4). Where "n_i" is the number of votes obtained by node "i" in the current voting period, and "N" is the total number of votes.

$$Cr(i)_{vote} = n_i/N \quad (4)$$

"$Cr(i)_{beh}$" is the performance credit of node "i" in the current voting cycle, which mainly depends on whether node "i" correctly verifies the block; generates the block; tampered with information, etc. In the current "V". It is used to evaluate whether node "i" is trustworthy. These factors all boil down to its performance in the current voting period, the mathematical expression is formula (5).

$$Cr(i)_{beh} = \sum_{x=1}^{n} f(x) - \alpha \times \sum_{x=1}^{n} g(x) \quad (5)$$

Among them, "n" represents the number of consensus processes that node "i" has participated in in the current voting period "V". "f(x)" indicates whether node "i" normally verifies and generates blocks in the x-th production interval "t", and has no malicious behavior. If the behavior is all normal, it is 1, otherwise, it is 0. g(x) indicates whether "i" did evil behaviors in the production interval "t", if yes, then =1, otherwise =0. "α" represents the weight that the system punishes node "i" if it has malicious behaviors. When there are more normal behaviors than malicious behaviors, "α" is only 1. If there are more malicious behaviors, "α" is greater than 1, and is a positive integer.

"$Cr(i)_{acc}$" is the accmulate trust degree of node "i" as of the end of the last voting period "V-1", and is also the trust degree of the first production interval at the beginning of the current voting period "V", that is, it satisfies formula (6).

$$Cr(i)_{acc_v^1} = Cr(i)_{pro_{v-1}^{last}} \quad (6)$$

3.3 Design of Reward and Punishment Plan

In the TM-PBFT algorithm, whether it is the gains obtained after the master node is successfully packaged or the penalty for failures, the ratio between itself and the nodes that vote for it is 1:1. In this way, the revenue can be used to make each node actively vote, and it can also identify malicious nodes in the system in a timely manner. Nodes will vote for reliable nodes when voting, so as to prevent the master node from failing or doing evil and causing itself to be punished.

In this algorithm, the voting nodes that vote for the same node all eagers to generate revenue by investing in the node to become the master node. Therefore, the relationship between the voting nodes is not a competitive relationship but a cooperative relationship. Furthermore, these nodes form an alliance game, denoted as (N, V(S)) [10], where "N" represents the set of players in the game. In this algorithm, it refers to the voting nodes that vote for the same node. Assuming that there are "Y" nodes voting for node "X", then "N={1, 2, 3, ..., n }". "S" represents the cooperative alliance that these n voting nodes may form, so "S" is also a subset of the set "N", namely "S⊂N"; "V(S)" is the characteristic function of the cooperative "Alliance S". It can also be interpreted as the maximum benefit that "S Alliance" gets after taking advantage of its own advantages. The main problem that needs to be

solved in this alliance game is how to distribute the income of each voting node fairly and reasonably. "(N, V(S))" stipulates that the contribution made by each node in each "Alliance S" is called the marginal contribution, which is recorded as formula (7). According to the value of the marginal contribution in different alliances, the Shapley Value can be obtained, which is recorded as formula (8).

$$\delta_i(S) = V(S \cup \{i\}) - V(S) \tag{7}$$

$$\varphi_i = \sum_{i \in I} \delta_i \left(S_i(J)\right) / N! \tag{8}$$

Suppose "E" is the profit that the master node can obtain after the end of a consensus process, and the profit that the node i that voted for is:

$$E_i = E \times \varphi_i \tag{9}$$

In order to prevent nodes from voting around, the algorithm limits the number of votes a node can vote, and each node has only one vote. In addition, considering that every time a vote is initiated, the response level of each node is different. Some nodes are online all the time and actively vote, and some nodes hang up all the time, and they did not vote at the end of the voting. Therefore, in order to distribute more revenue to the nodes that are actively voting, the algorithm adds the time factor "T_f" to the Shapley value, which is based on the time when the system initiates the vote. The closer to the voting initiation time, the greater the revenue the node will receive. The farther away from the voting initiation time, the less profit the node will get. If the voting time is over and a node has not voted, it will be punished accordingly. Therefore, if a node wants to get more revenue, it needs to stay online as much as possible and actively and timely vote for other nodes.

3.4 Algorithm Flow

In the TM-PBFT algorithm, the trust degree is an important indicator used to measure whether the node is trustworthy, and the value will also change accordingly with its performance in the voting and consensus process. The algorithm flow is as follows.

Flow chart. The flow chart of the algorithm is shown in Fig. 5.

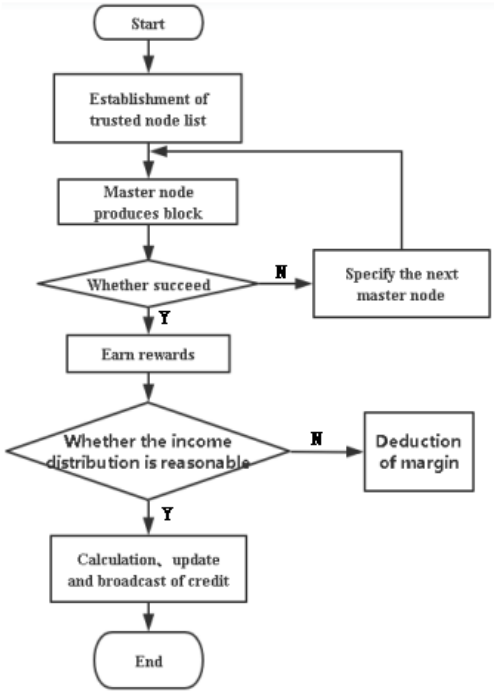

Fig. 5. Flowchart of TM-PBFT algorithm

Node number. Suppose the total number of nodes in the system is "N", the system will assign a sequence number i ($1 \leq i \leq N$) to each node according to the time when it joins the system, and each node needs to maintain its own trust level array C[i][Cr], used to record their own node trust.

Vote to establish a list of trusted nodes. When the system initiates the first vote, all nodes have the same degree of trust and can vote for any node in the system. If the system needs H trusted nodes, the top "H" nodes will be formed into a trusted node list according to the ranking of votes after the voting ends, and these "H" nodes will be used as master nodes for block packaging and consensus processes in turn.

Income Distribution. After the end of a round of voting cycle "V", the system will distribute tokens as revenue to the successfully packaged nodes. At the same time, the system will return the revenue distribution of each trusted node to the trusted node according to the revenue distribution algorithm, and finally the trusted node will distribute it.

Nodes' trust degree calculation. After the revenue distribution is completed, the first successfully packaged node initiates a trust calculation request to other nodes. After receiving the request, each node starts to calculate the trust degree of all nodes in the system and returns the calculation result to the array "C". Finally, add its own signature to the array "C" and return it to the master node.

Node trust degree is updated. After the master node receives the array message, it first verifies it. If 2N/3 identical calculation results are received for a certain node "i", the trust level of the node is updated and stored locally.

Node trust degree broadcast. After the master node updates the node trust level, it broadcasts the local trust level information to other nodes. After other nodes receive the broadcast message, they update the local trust level array. When the next voting request is initiated, it can also be used as a reference to reduce the probability of malicious nodes being selected as trusted nodes.

3.5 Algorithm Execution Flow

The execution process of the TM-PBFT algorithm is different from the PBFT algorithm, including the election process and the consensus process.

Election process. As shown in Fig. 6, the election process of the TM-PBFT algorithm.

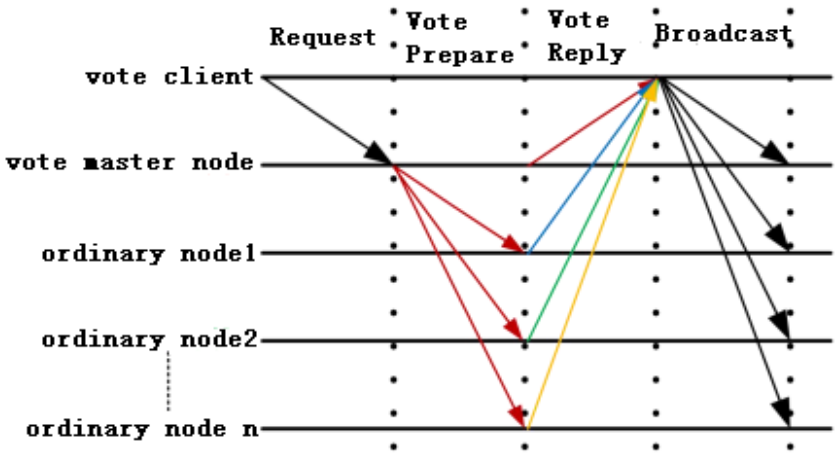

Fig. 6. Election process

An election process is conducted in one round of voting cycle, which can support multiple rounds of consensus. The election process can be divided into four processes: voting request, prepare, reply and broadcast. The algorithm design is as follows.

```
  Start
  vote client → vote master node: <Request, operation, timestamp,client>
//Vote Request
  vote master node → ordinary node: <<Prepare, vote,digest1>, message1>
```

```
//Vote Prepare
  If Hash(message1) = digest1        //check
  ordinary node → vote client: <<Reply, vote, digest, id>, message2>
      //Vote Reply
  If Hash(message2) = digest2        //check
  Vote client → all nodes:<<Result, vote,digest3, client>,message3>
      //Broadcast
  End
```

Consensus process. As shown in Fig. 7, the consensus process of the TM-PBFT algorithm.

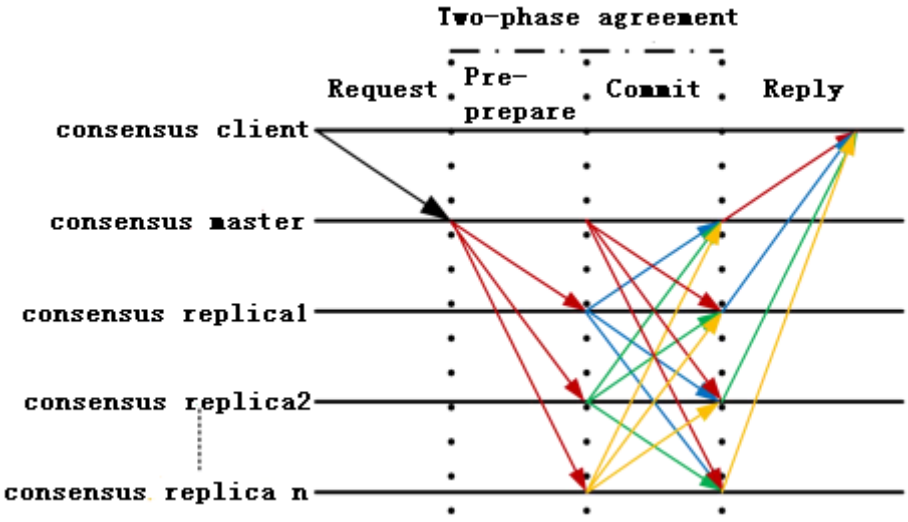

Fig. 7. Consensus process

The broadcast process in the consensus process of the TM-PBFT algorithm has changed from the previous three-phase protocol to a two-phase protocol. The algorithm design is as follows.

```
  Start
  consensus client → consensus master: <Resquest, operation, timestamp, cli-
ent>        //Request
  consensus master → consensus replica: <<Pre-prepare, view, n, digest>,
mmessage>         //Prepare
  If Hash(message) = digest          //check
  consensus replica → other nodes: <Commit, view, n, digest>//Commit
  all nodes → consensus client:<Reply, view, timestamp, client,id, response>
      //Reply
  End
```

4 Algorithm Analysis and Discussion

4.1 Algorithm Rationality Analysis

Firstly, for the node trust evaluation model, the trust of nodes that perform well will increase, and the trust of malicious or failed nodes will decrease, and parameters are used to control the degree of punishment for malicious nodes, if the user wants to increase the punishment and increase the value.

Secondly, for the design of the reward and punishment scheme, on the basis of the Shapley value, the time factor "T_f" is added, which encourages nodes to actively and earnestly vote to obtain greater benefits. If there are nodes "N_1" and "N_2" in the system and their calculated Shapley value are all 1/4, according to the original income distribution, both nodes will get 25% of the income. But after adding the time factor, if Node "N_1" has an earlier voting time than "N_2", the system will eventually calculate the revenue of node "N_1" than "N_2". Therefore, if the

nodes in the system want more revenue, they must vote in a timely manner. In addition, nodes must carefully choose reliable nodes when voting, and they cannot vote at will. If malicious nodes are selected, the system will also impose corresponding penalties, which also reduces the possibility of malicious nodes being selected as master nodes.

Finally, before the master node performs packaging, the system will charge a deposit. If it fails to distribute the proceeds to the voting node according to the system requirements after the master node is packaged, the system will confiscate the deposit for the node. After the confiscation, it is first distributed to the voting node that is the node, and the remaining part is used as the system revenue. The margin here is much greater than the revenue obtained from the production block, and the system can control it by itself. Therefore, if the node wants to monopolize the gains obtained, it will not be worth the loss.

4.2 Safety Analysis

The trustworthiness evaluation model designed in this paper adds the attribute of trustworthiness to the nodes in the system. Nodes with higher trustworthiness are normally the nodes that have participated in multiple consensuses and successfully produced blocks without committing evil behaviors, which enables the system to be timely identify malicious nodes. The new reward and punishment scheme designed also encourages voters to vote for more trustworthy nodes, so that it can also obtain high returns on its own. If the master node fails or commits malicious acts during the packaging process, it will lose the deposit paid and the penalty of trust. In other words, if an ordinary node wants to become a master node to participate in consensus, a certain threshold is required. Only when the trust level reaches a certain value can it be packaged. If a node wants to attack the system, it needs to accumulate trust for a long time, which greatly improves the security of the system to a certain extent.

4.3 Compare with other Consensus Algorithms

As shown in Table 1, it is a partial comparison between the proposed TM-PBFT algorithm and other consensus algorithms.

Table 1. Comparison of consensus algorithms

Consensus algorithms	Consumption	Effectiveness	Threshold	Safety	Flexibility	Punishments
PoW	high	-	-	high	weak	no
PoS	general	-	-	general	weak	no
DPoS	low	high	no	general	weak	no
PBFT	high	high	no	general	weak	no
TM-PBFT	general	high	yes	high	strong	yes

4.4 Analysis and Comparison of Throughput

Throughput is a standard for testing the operational efficiency of a blockchain system. It refers to the average number of transactions packed into a block within a unit time. The calculation formula is:

$$TPS = N_{transaction} / \Delta t \tag{10}$$

Among them, " $\triangle$ t" is the time to pack the block, and "$N_{transaction}$" is the number of transactions packed into the block. As shown in Fig. 8, the throughput of this algorithm is higher than that of the PBFT algorithm. As the number of nodes increases, the throughput advantage of this algorithm becomes more obvious. The main reasons are: a) The communication time in the consensus phase is reduced; b) The probability of a reliable node as the elected master node is increased. The probability of system errors is reduced, and the consensus time is reduced.

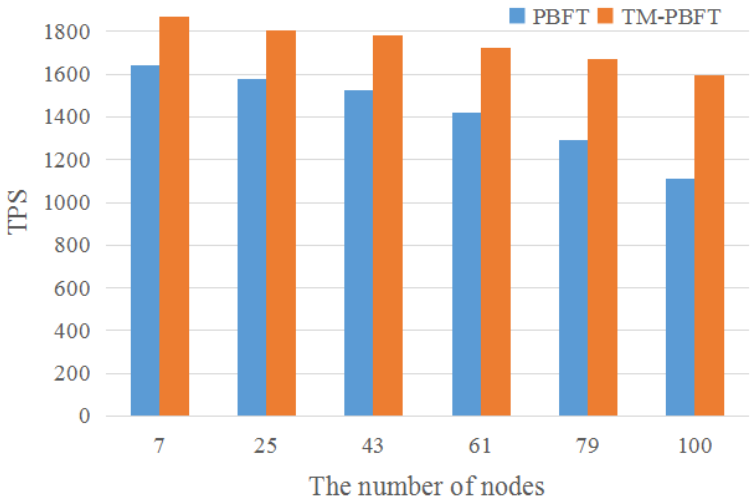

Fig. 8. Comparison of system throughput

4.5 Analysis and Comparison of Communication Bandwidth

This algorithm is divided into two processes: voting and consensus, both of which generate communication bandwidth overhead. The voting process is divided into four stages: voting request, preparation, reply and broadcast. The consensus process is divided into 4 stages: request, pre-prepare, commit and reply. Its communication bandwidth is expressed as:

$$\begin{aligned} \text{Bandwith}_{TM-PBFT} &= k \times [1 + (N-1) + N \times (N-1) + N] \times SizeBlock \\ &+[1 + (N_V - 1) + N_v + N_v] \times Vote = k \times (N^2 + N) \times SizeBlock + 3N \times Vote \end{aligned} \tag{11}$$

Among them, “Bandwith” is the communication bandwidth, “N” is the number of consensus nodes, and “Nv” is the number of voting nodes. In this algorithm, “N = Nv”, SizeBlock is block data, “Vote” is voting data, and “k” is the number of consensus in a round of voting cycle.

In the same way, the PBFT algorithm only contains the consensus process. Specifically, it is divided into 5 stages: request, pre-prepare, prepare, commit and reply. The communication bandwidth is expressed as:

$$\begin{aligned} \text{Bandwith}_{PBFT} &= k \times [1 + (N-1) + (N-1) \times (N-1) + N \times (N-1) + N] \times SizeBlock \\ &= k \times (2N^2 - N + 1) \times SizeBlock \end{aligned} \tag{12}$$

As shown in Table 2, it is the simulation parameter table of communication bandwidth:

Table 2. Comparison of consensus algorithms

Parameter	Value
SizeBlock	1024KB
Vote	0.2KB
k	10
N	10~100
Nv	10~100

According to the parameters in Table 2, the TM-PBFT and PBFT algorithms are simulated by Matlab 2017a, and the results are shown in Fig. 9.

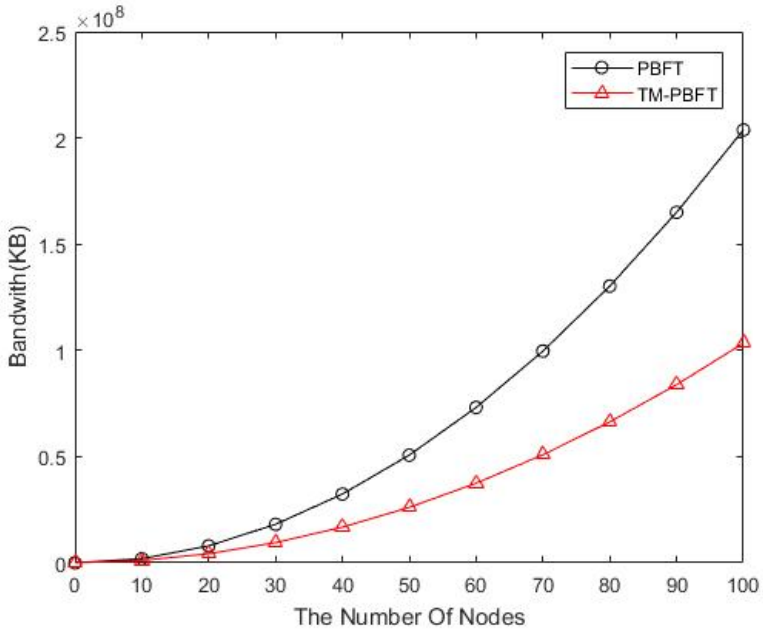

Fig. 9. Comparison of bandwith

Under the simulation conditions of this article, the communication bandwidth of TM-PBFT is always smaller than that of PBFT algorithm. When the number of nodes is 100, the communication bandwidth of the PBFT algorithm is 2.038×10^8KB. The communication bandwidth of the TM-PBFT algorithm is 1.034×10^8KB, which is about 50% lower than that of the PBFT algorithm.

5 Conclusion

This article first introduces the PBFT algorithm and through analysis and comparison, it is found that there are still problems such as high communication resource consumption, inflexible number of nodes, low fault tolerance, and inability to identify malicious nodes in time. In response to the above problems, this paper proposes the TM-PBFT algorithm, which introduces a trust mechanism, models the trust degree of nodes, and designs a new reward and punishment scheme. This enables nodes to actively and earnestly participate in voting, identify malicious nodes in a timely manner, and improves the fault tolerance rate. The broadcast process is also simplified from three stages to two stages, reducing communication costs. The number of nodes can also be adjusted according to system requirements, which is more flexible. However, this algorithm still has some shortcomings. For example, if nodes only invest in a node with a high degree of trust each time, or there are nodes make bribes, the gap between the rich and the poor will become more and more obvious. There have been preliminary ideas for solving this problem, and further research will be carried out in the future.

6 Acknowledgement

Fund Project: Research Project of Fundamental Scientific Research Business Expenses of Provincial Colleges and Universities in Hebei Province 2021QNJS04.

I would like to thank the sponsor of this research-the “Research Project of Fundamental Scientific Research Business Expenses of Provincial Colleges and Universities in Hebei Province”, which Grant is 2021QNJS04. In addition, thanks to Zhangjiakou Big Data Technology Innovation Center for providing experimental platforms and equipment.

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Journal for General Philosophy of Science 39(1)(2008) 53-67.

[2] C. Pirrong, Securities Market Macrostructure: Property Rights and the Efficiency of Securities Trading, Journal of Law,

Economics & Organization 18(2)(2002) 385-410.
[3] S. King, S. Nadal, Ppcoin: Peer-To-Peer Crypto-Currency with Proof-of-Stake. <https://decred. org/research/king2012. pdf>, 2012 (accessed 21.05.19).
[4] D. Larimer, Delegated Proof-of-Stake (Dpos). <https://www.x- network.io/whitepaper/XCASH_ Yellowpaper_DPoPS. pdf>, 2018 (accessed 30.06.19).
[5] S. Biffl, M. Eckhart, A. Luder, E. Weippl (Eds.), Security and Quality in Cyber-Physical Systems Engineering, pp. 471-495, Springer, Cham, 2019.
[6] H. Zheng, W. Guo, N. Xiong, A kernel-based compressive sensing approach for mobile data gathering in wireless sensor network systems, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48(12)(2018) 2315-2327.
[7] S. Liu, P. Viotti, C. Cachin, V. Quéma, M. Vukolić, XFT: practical fault tolerance beyond crashes, in: Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), 2016.
[8] I. Abraham, D. Malkhi, A. Spiegelman, Validated asynchronous Byzantine agreement with optimal resilience and asymptotically optimal time and word communication. <http://arxiv.org/abs/1811.01332.pdf>, 2018 (accessed 10.11.20).
[9] Y. Liu, M. Ma, X. Liu, N. N. Xiong, A. Liu, Y. Zhu, Design and analysis of probing route to defense sink-hole attacks for Internet of things security, IEEE Transactions on Network Science and Engineering 7(1)(2020) 356-372.
[10] K. Leyton-Brown, Y. Shoham, Essentials of game theory: A concise, multidisciplinary introduction, Synthesis lectures on artificial intelligence and machine learning 2(1)(2008) 1-88.