T-PBFT: an EigenTrust-based Practical Byzantine Consensus Algorithm

Tianyu Yu[1], Sheng Gao[1] and Jianming Zhu[1]

[1] School of Information, Central University of Finance of Economics, Beijing, China
cufeyty@163.com

Abstract. Blockchain with these characteristics of decentralized structure, transparent and credible, time-series and tamper-resistant, has been considering as a promising technology. Consensus algorithm as one of the main blockchain techniques directly affects the scalability of blockchain system. Existing probability-based consensus such as PoW, PoS, DPoS suffers from power consumption and low scalability; while determinacy-based consensus algorithm such as PBFT, SBFT could not meet large scale network nodes. In this paper, we propose a novel consensus algorithm, namely T-PBFT, based on EigenTrust model. T-PBFT is a multi-stage consensus algorithm. It evaluates node trust by the transaction between nodes so that the high quality nodes in the network environment will be selected as a consensus group which the quantity can be strictly restricted. Because of this, the efficiency of consensus cannot be affected by the number of nodes. We analyze T-PBFT theoretically and compare it with other similar algorithms. Our algorithm get a higher fault tolerance and scalability on the basis of maintaining the communication complexity.

Keywords: Blockchain, Consensus protocol, Byzantine, Trust model

1 Introduction

Blockchain, as the key support of Bitcoin proposed by Satoshi Nakamoto[1], is a kind of data structure with time series which is combined by the computer technology such as encryption algorithm, time stamp, distributed system and consensus mechanism. Blockchain has the characteristics of decentralization, transparent account, non-tamperable and node anonymity, which can solve the disadvantages of high risk, high cost and low efficiency inherent in the centralized architecture system[2]. In recent years, it has aroused widespread concern from all walks of life.

Consensus algorithm is one of the key technologies for blockchain. Its purpose is to formulate and use strict and unambiguous rules, so that the nodes in the system which do not trust each other can keep the state of the system data consist. The current application scenarios of blockchain technology are concentrated among enterprise alliances. Compared with the common chain, the number of nodes involved in the alliance chain is small, but the system throughput and fault tolerance have higher requirements. At present, several major consensus algorithms have serious drawbacks when applied to

the alliance chain. The consensus method based on the calculation rules represented by Proof of Work (POW)[1,3] relies on the computational power or other resources which is in a large number of distributed implementations. But due to the small number of nodes in the consortium blockchain, it is easy to achieve centralized resources, forming a centralization trend, vulnerable to malicious attacks, so the consensus algorithm for Byzantine fault-tolerant classes in the consortium chain is more extensive. The practical Byzantine Fault Tolerant (PBFT)[4] consensus mechanism can meet the needs of small-scale consortium chains, but due to the low scalability, the performance of the consortium chain will decline sharply with the increase of the number of nodes. To solve the problem in current consensus algorithms, many algorithms based on PBFT or other agreement have emerged[5].

Therefore, this paper proposes a new byzantine fault-tolerant consensus, named T-PBFT, which consist of two stage of establishing the trust system and reaching a consensus. In the first stage of T-PBFT, we introduces the trust model to evaluate the trust value of every node in the system, which is used to be the basis for consensus grouping. After grouping, we can have controllable number of nodes to complete the process of the improved PBFT. Overall, the main contributions of this paper are as follows.

- It introduces a trust model to make T-PBFT have a flexible and complete mechanism which is used to set up a consensus group. Grouping can prevent the nodes with low credit from participating in consensus and limit the number of nodes in the second stage which can improve the scalability and fault-tolerant rate of the system.
- It details the whole process of T-PBFT from three parts which contain the use of the trust model, grouping process and improved PBFT by Pseudo code.
- It compares T-PBFT with other related Byzantine fault-tolerant algorithm , and analyzes the advantages and disadvantages of our T-PBFT theoretically.

The remainder of this paper is organized as follows. Section 2 introduces related works. Section 3 introduces the preliminaries of T-PBFT. Section 4 describes system model and the design of T-PBFT. Section 5 compares T-PBFT with related BFT algorithms. Section 6 includes the conclusion.

2 Related work

2.1 Introduction to Consensus Algorithm

The consensus algorithm in the blockchain is to achieve consistent records between nodes that do not trust each other. At present, common consensus algorithms are mainly divided into two categories according to different implementation mechanisms: consensus algorithms based on calculation and consensus algorithms based on voting. The most classic consensus algorithm based on calculation is proof of work (PoW)[1,3] first used in Bitcoin. The proof of work algorithm utilizes the characteristics of the hash function, so that the node must do a lot of calculations to obtain the region. But because the repeated calculation has no meaning, the proof of work algorithm has serious resource waste problem. In order to solve the defects of the proof of work, Peercoin[6]

proposed the proof of stake algorithm. The proof of stake means that the node's stake is used to replace the computing power in the proof of work, so as to reduce the waste of resources such as computing power and electricity. In Peercoin system, the account with an older coin age is more likely to get the right to generate a new block. It is just one of the POS classes. Proof of Stake (DPoS)[7] allows users to grant other nodes their own stake, and the nodes which are selected by the other nodes will complete the consensus process on behalf of the whole network nodes, which reduces the burden on other nodes. Proof of Luck[8] is based on trusted execution environments.

The other class of consensus algorithms is based on voting and passing messages. Its characteristic is that it can still make a correct judgment when some nodes in the system fail. Many different algorithms have been proposed for Byzantine problems[9]. The practical Byzantine protocol is proposed in 1999[4]. It reduces the complexity of Byzantine algorithm so that PBFT can be applied to realistic scenes. But the number of the messages in the agreement is huge which limit the network seriously. To improve the efficiency of PBFT, Tendermint[10] introduces voting with weight instead of the equal vote. So to reach the consensus, it only needs to gather $\frac{2}{3}$ weight. Comparing with waiting $\frac{2}{3}$ nodes response, this method can be more effective when the stake is more concentrated. The Honey Badger of BFT[11]improves the weak synchronization hypothesis of PBFT, and proposes a consensus algorithm that can be applied in asynchronous network environment which also improves security. Zyzzyva[12] assumes that in most cases there are none failures in the process, so it simplifies complex interaction process. But if some failures occurs, the system will still start a traditional PBFT to achieve consistency. SBFT[13] introduces threshold signatures, modern cryptographic advances and other factor to reach a pretty performance in the network of hundreds of nodes. In HQ[14], all nodes only interact with the client. But when there is a disagreement, the process of PBFT will be executed. And XFT[15] assumes the number of malicious nodes are less than the classic Byzantine so it can improve the performance by optimizing the process of messages transmission and also increase the fault-tolerance rate to $\frac{n-1}{2}$ where n is the number of nodes in the system. Parallel BFT[16] use different multicore machines to processing requests simultaneously to get greater throughput. Optimistic BFT[17] uses a weaker replica consistency model in which the replica will execute a request without a total order of the message to achieve more powerful performance.

3 Preliminaries

Before introducing T-PBFT, we would some basic knowledge related to our T-PBFT algorithm which consists PBFT model and EigenTrust model.

3.1 PBFT

PBFT (Practical Byzantine Fault Tolerance) is a state machine replica copy algorithm that can be applied to synchronous network environments[4]. It reduces the complexity

of the Byzantine fault-tolerant algorithm and enables it to be applied to the actual environment. .

There are three important components in PBFT, namely view, primary, and replica. The master node is the initiator of the voting process. The replica node is responsible for verifying the authenticity of the voting content and the correctness of the voting source, and feeding back to the network by means of voting (message verification). When the primary node has a work barrier, the system uses the view rotation function to change the current primary node to ensure a correct consensus[3].

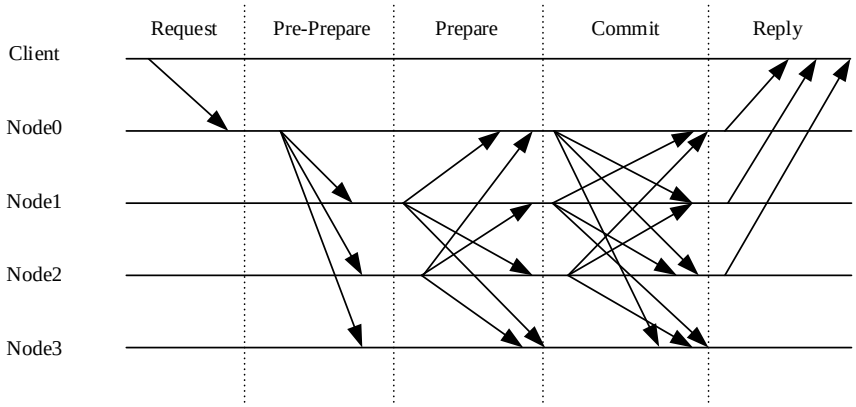

Fig. 1. The process of Practical Byzantine Fault Tolerance.

The process of PBFT implementation is shown in the figure 1, where node 3 is the faulty node. The implementation process is mainly divided into three stages: pre-preparation, preparation, and confirmation:

1. In the pre-preparation phase, the client first sends a request to the master node, and the master node processes the user request and organizes the message containing the view number, the request number, and the abstract to broadcast to the replica nodes. Then the replica nodes receive and verify the legitimacy of the message. Once the verification is passed, replica nodes will accept the request and enter the preparation phase.
2. In the preparation phase, the replica nodes accepted request broadcast the preparation information on the network while continuously obtaining preparation information from other nodes in the network. When the node gets 2f (f is the number of faulty nodes in the system) preparation messages from different replica nodes, the preparation phase ends.
3. During the confirmation phase, the node itself is in a prepared state, but in order to prevent the number of nodes in the network from reaching the prepared state is less than $2f + 1$. Therefore, the request cannot be executed immediately, and the confirmation information needs to be broadcasted. Then the nodes including the master node receive the confirmation information from the network. When the node gets 2f

confirmation message from different nodes, the user's request will be executed and a receipt will be sent.

PBFT can have $\frac{n-1}{3}$ fault tolerance rate (n is the number of nodes in the system) under the condition that guarantees the system is not intruded and the messaging is successful. As a consensus algorithm, the practical Byzantine fault-tolerant algorithm can achieve consensus without relying on tokens, and the consensus efficiency is high and the speed is fast.

3.2 EigenTrust trust model

The EigenTrust trust model is one of the most authoritative trust models[18,19] that can be applied in the P2P environment proposed by Kamvar et al. of Stanford University. The EigenTrust model can effectively assess the trust value of a node and provide security by supervising the node's dishonest implementation. It obtains a unique global trust value for every node in the system by recording the transaction history between nodes. The global trust value t_i^{k+1}in our model can compute by taking

$$t_i^{k+1} = c_{1i}t_1^{(k)} + \cdots + c_{ni}t_n^{(k)}$$

where t_i is the global trust value of node i and c_{ij} is the local trust value of node i to node j.

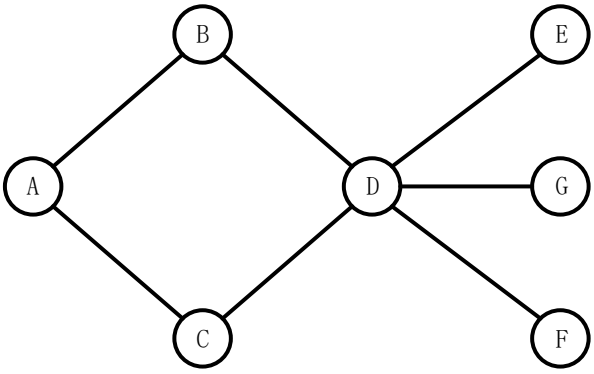

Fig. 2. A relation graph between nodes.

The figure 2 shows a relation graph between nodes. Nodes connected by lines mean that they have already complete transaction. So we can easily divide the relationship between nodes into two types: the nodes with transaction and the nodes without transaction. Take node A as an example, there will be three different kinds of trust value in the EigenTrust model:

1. Local trust value C_{AB}: The local trust value is the result of evaluation between nodes that have undergone direct interaction like the trust value of node A to node B. The value $s_{AB}$is based on the number of satisfactory or unsatisfactory transactions that

node A has had with node B. We define $s_{AB} = \mathrm{sat}(\mathrm{A},\mathrm{B}) - \mathrm{unsat}(\mathrm{A},\mathrm{B})$ and $C_{AB} = \frac{\max(s_{AB},0)}{\sum_x(s_{Ax},0)}$.

2. Recommended trust value C_{AD}: The recommended trust value is the evaluation result between nodes that have not interacted like node A and node D. It is built on the basis of transitive trust and its value is related to the local trust value. The C_{AD} can be taken by $C_{AD} = \sum_k C_{Ak}C_{kD}$. (Notice that C_{Ak} and C_{kD} must be local trust value).
3. Global trust value t_A^{k+1}: The global trust value is the quantifiable degree of trust that the system evaluates for node A. The global trust value of the node A integrates the trust value evaluation information of all nodes in the network and combines the current global trust value of each node, so it can be used as an evaluation index for the trust degree of node A.

4 System overview

4.1 Framework of T-PBFT

As a typical P2P network architecture application, blockchain has received extensive attention due to its anonymity and decentralization characteristics. However, due to its own characteristics, blockchain is vulnerable to attacks by malicious nodes. How to reduce the impact of malicious nodes on the system has always been an important research content in the consensus field. Therefore, this paper proposes a two-stage consensus protocol based on the trust model. And the main process of T-PBFT is shown by figure 3. Our consensus algorithm based on the trust model includes two main phases: building consensus groups and completing consensus. In the first stage, we will use the EigenTrust trust model to obtain the global trust value of the nodes which will be the basis to select the nodes. (Notice that to get the global trust value, system should get the local trust value between nodes first.) And then, the nodes with high trust value will be used as representatives to form a consensus group. In the process of consensus, because of the consensus group, the number of nodes participating in the consensus has decreased, making it feasible to apply the improved PBFT consensus algorithm. After that, there will be a new block connected to the blockchain and new transactions for nodes, so the global trust value can dynamically change with the block and new process of T-PBFT can start a new round.

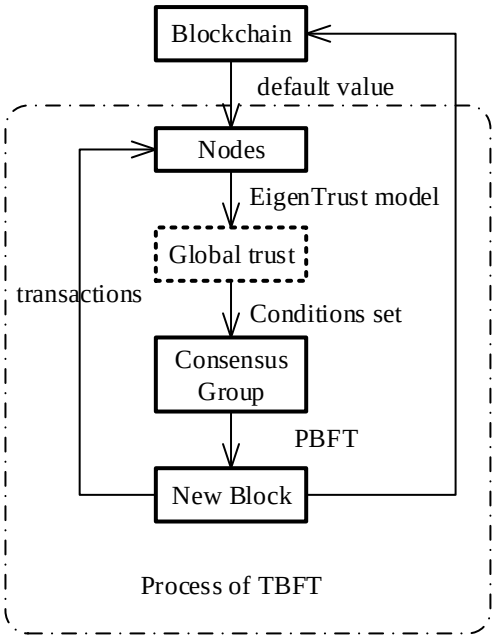

Fig. 3. The main process of T-PBFT.

For the EigenTrust trust model, since it does not have a penalty mechanism for malicious nodes, a malicious node can perform malicious behavior without any loss in the system, harming the interests of other nodes. But after using the global trust value to evaluate a node, it is possible to marginalize malicious nodes and reduce or even eliminate the effects of malicious node behavior. In addition, although the PBFT algorithm has strong consensus ability and has been applied to some applications, it cannot be applied to the environment with a large number of nodes. Through the EigenTrust model's trust evaluation results and setting conditions, we can reduce the number of nodes participating in the consensus. After the formation of the consensus group, there will be an applicable environment for PBFT which can keep the high efficiency.

4.2 Assumptions

Behavioral consistency. In the T-PBFT, we define the nodes with high global trust value are more worthy of trust. It means that these nodes have higher probability of honey behavior. However, if the nodes obtained high trust in the first phase act maliciously in the second phase, the model cannot ensure the correctness of consensus. So, there must be an assumption that the behavior of nodes is consistent.
Limited time. Time is an important element in the Eigentrust model. As time goes on, the number of transactions will grow to a huge quantity, which can leads to the fact that the global trust value of nodes can not be affected by a small amount of transactions. And at that time, the model will fail because the node with high trust will still be trusted

even if it cheats in some transaction. So in order to ensure T-PBFT effective, time cannot be infinite should be an assumption.

4.3 Node Trust Evaluation

At the beginning of T-PBFT, we initialize the global trust values of all nodes to $\frac{1}{n}$. To use the EigenTrust model for getting the global trust value[7], the satisfaction and the direct local trust value between nodes will be calculated first. And the other group of nodes which have none transactions with each other should obtain the recommended trust value by asking their neighbor nodes to report the direct local trust value they knows. When the trust between all nodes is broadcast, the global trust value of every node can be calculated.

Algorithm 1 CalcTxNodeTrust

Input: Node i & group Nodes (in which nodes are related to node i)
Output: Local trust *value* C_{ij}
For node j in Nodes:
$S_{ij} = \text{sat(i, j)} - \text{unsat(i, j)}$
$S_{\text{total}} = \sum \max(s_{ij}, 0)$
end for
if $S_{\text{total}} = 0$
set every $C_{ij} = \frac{1}{n}$ (n is the scale of Nodes)
else
For node j in Nodes:
$C_{ij} = \frac{\max(s_{ij}, 0)}{S_{\text{total}}}$
end for

Algorithm 2 CalcNonTxNodeTrust

Input: Node i & group Nodes (in which nodes arenot related to node i)
Output: Recommended trust *value* C_{ij}
For node j in Nodes:
$C_{ij} = \sum_k C_{ik} * C_{kj}$ (k belongs to the set of the nodes related to node i)
end for

In the algorithms 1, 2, the node i and the node set Nodes are taken as input parameters, and the trust value evaluation method of the node i to other nodes in the chain is respectively displayed. It can be clearly seen that the calculation of the local trust degree is divided into two cases: When the other node j has transactions with the node i, the node i will calculate the absolute satisfaction value S_{ij} by querying the historical transaction satisfaction record with the node j. And normalize it to get the direct local trust value C_{ij} to the node j. When the other node j and node i have not had a direct connection, the recommended trust value calculation method may be adopted. The recommendation here refers to the process in which the interaction node transmits its own trust status of the non-interactive node to the node i, that is, the node i asks the node group in which nodes are directly related to node i about their trust value to the node j. Then node i will combine with its own trust judgment of its neighbor nodes and obtains the recommended trust value to the non-interactive node j. The generation of recommended trust is an irreversible iterative process. The basic idea is to ask the target node's trust information from the neighbor node. When the neighbor node does not interact with the target node, the neighbor node will ask new neighbor node. A message is made back to the originating node until all the requested nodes and the target node have direct trust values or the node asked does not have any new neighbor nodes. Because of the huge amount of messages made by asking and replying, we had better let the nodes broadcast their trust value to a smart contract which is used to calculate the recommended trust value so that the message about the trust only need to be passed once.

Algorithm 3 CalcGlobalTrust

Input: node i
Output: global trust T
For the nodes j
$T_i^{k+1} = \sum c_{ji} T_j^k$
end for

After algorithm 1, 2, all nodes establish a local trust relationship and have an accurate local trust value. But the local trust value does not fully represent the trust level of a node, so we need the global trust value. This value of all nodes in the initial stage is $\frac{1}{n}$ (n is the number of nodes in the system), but when a new block is generated, the node needs to reevaluate its global trust value. Algorithm 3 reveals the calculation method of the global trust value: For node i, its global trust value should be the sum of the product of all other nodes and its current global trust value. The global trust value is a changeable value, and its evaluation result has a certain relationship with all other nodes. So it can more accurately describe the trustworthiness of the node and reduce the interference of some malicious nodes.

It can be known from the above three algorithms that the calculation of local trust between nodes has a great difference with whether there are direct relationship between nodes, so a classification algorithm should be designed.

Algorithm 4 DivideNodes

Input: Evaluation node i & other node groups Nodes
Output: TxnNodes & NonTxnNodes
For node j in Nodes:
If j and i have been traded:
TxnNodes ← j
Else:
NonTxnNodes ← j
end if
end for

Algorithm 4 is a method for classifying nodes according to whether a transaction relationship exists. It is mainly used for the system to clearly distinguish the relationship between nodes and select the correct method when calculating the local trust relationship.

4.4 Construction of Consensus Group

Through the application of the EigenTrust model, all nodes in the system have quantitative trust evaluation results at any point in time. Then, according to the quantitative global trust value, nodes with higher trust value can be selected into a consensus group as a representative to be responsible for the consensus process. The criteria for selecting nodes can be various, such as setting a variable threshold of trust values, selecting a fixed number of nodes with high trust, etc. In this paper, a fixed proportion of nodes are selected to form a consensus group, in order to reduce the system's burden while ensuring the security of the system.

Algorithm 5. getConsensusGroup

Input: All nodes Nodes & Global Trust T
Output: ConsensusGroup
Sort Nodes by T
For node with T in top N%
Add to ConsensusGroup
end for

In Algorithm 5, the process of generating consensus groups in the system is shown. First, the system needs to sort the nodes according to the size of the global trust value of the node, and then selects the nodes with higher trust values to form a consensus group. The number of nodes in the consensus group will change dynamically with the

total number of nodes in the system. That is to say, only the nodes with the top N% of the global trust value in the system can participate in the consensus process. Such a selection method can make the consensus group full of node with higher trust value. Since the trust degree is formed by the joint evaluation of all nodes in the system, the nodes in the consensus alliance can not only represent all the nodes in the system, but also have higher security

4.5 Consensus Process

After the consensus group is established, the new block is generated by voting within the group. In the blockchain environment, there is no traditional C/S structure, so the PBFT protocol can be slightly modified in the process. In the new voting protocol, the system will select a few nodes with the highest trust value to form the voting leading group, which is responsible for initiating the voting and confirming the correctness of the new block generation. Compared with the setting of the master node in the traditional PBFT, the leading group establishes the ability to prevent certain problems with a malicious master node.

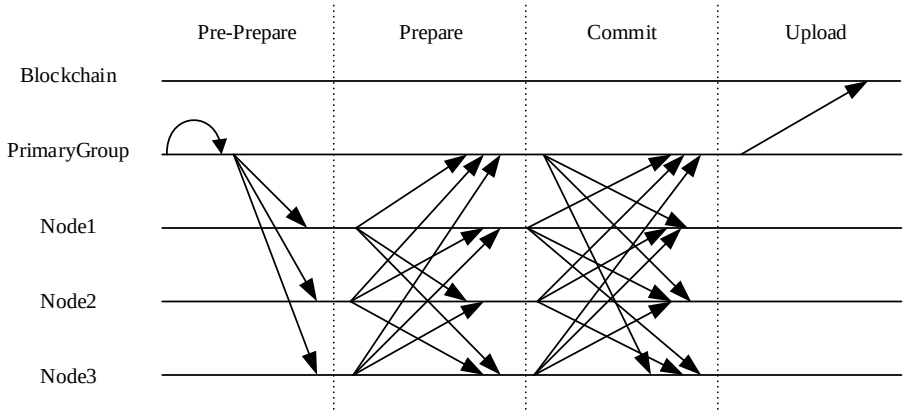

Fig. 4. the process of Simplified Practical Byzantine Fault Tolerance.

The simplified PBFT process is shown in figure 4, removing the client's request and receipt, and adding the blockchain as a main body to the process. In the first half of the pre-prepare phase, the nodes in the leadership team need to ensure preliminary verification of the pre-generated block information within the team. When most of the nodes in the leadership team confirm that the structure and information are correct, the leader team will launch a new round to start the consensus process for the new block. In the pre-prepare phase, the prepare phase, and the commit phase, the nodes in the consensus group need to verify the integrity, correctness, and reliability of the information source through a multi-stage voting protocol. In the traditional PBFT protocol, nodes have the same share of voting rights, but in order to achieve higher efficiency and better security, the voting in this paper will introduce a weighting mechanism according to the global trust value of the node. When the primary node group receives an acknowledgment

message with a certain weight in the commit phase, it will confirm the block and writes the block into the chain. In the commit phase, messages received by other nodes outside the primary node group are used to confirm that the primary node group has not changed due to its malicious behavior.

Algorithm 6 getPrimaryGroup

Input: ConsensusGroup
Output: PrimaryGroup
For node with trust value in top n%:
Add to PrimaryGroup
end for

Algorithm 7 getWeightforVote

Input: ConsensusGroup
Output: ConsensusGroup with weight
Calculate total T in Consensus
for nodes in ConsensusGroup
node. weight $= \frac{T_{node}}{T_{total}}$
end for

Algorithm 6 shows the selection process of the leading group in the consensus process based on the global trust value of the node, the nodes with the highest trust value in the consensus group are selected to form the leading node group, which is responsible for building and recording a new block. Algorithm 7 records the method of assigning node weights in the consensus group. It can be seen that the weight assignment is completely based on the global trust value. The sum of the global trust values in the consensus node group is used as the base, and the global trust of the individual nodes is taken as the voting weight. The sum of the weights of all the consensus nodes is 1. Since the weights are divided according to the method of Algorithm 7, the high-trust nodes will obtain higher weights. If the nodes are consistent in behavior, then weight voting instead of equal voting can achieve higher fault tolerance and efficiency.

The entire flow of a non-leader node in the voting verification block is shown in Algorithm 8. First, in the pre-prepare phase, the node needs to obtain the pre-generated block from the leader node group. After obtaining the pre-generated block, the node needs to verify the block information, including the authenticity of the source, the authenticity of the block content, the block structure and the legality of the information in the block. When the node completes the verification, the verification result with its own weight is broadcasted in the network. When the weight of the verification result received by a node exceeds the threshold N, the process enters the commit phase. In this

phase, the node needs to broadcast the acknowledgement result with the weight, and waits to receive the acknowledgement information from other nodes. When the weight of the acknowledgement information is greater than the threshold N, the pre-generated block will be confirmed and the state of the block is changed. At this point, the node has the ability to verify the correctness of the block. Even if the leader node group tampers with the block information at the end of the consensus process, as long as most nodes in the network are honest, the malicious behavior can be corrected.

Algorithm 8 CheckAndVote

Input: BlockChain (Pregenerated)
Output: BlockChain (confirmed)
#pre − prepare phase
get BlockChain(Pregenerated) $from$ PrimaryGroup
#prepare phase
If BlockChain(Pregenerated)is accepted
Broadcast prepare message with weight
end if
get prepare message from other nodes
if prepare message′weight $>$ N
Enter commit status
end if
#commit phase
Broadcast commit message
get commit message
if commit news′weight $>$ N
make BlockChain(Pregenerated) confirmed
end if

In the voting protocol of this paper, the rotation of the primary node group changes dynamically as the block generating. After the new block is generated, since the new transaction is confirmed to be on the chain, more evaluation information is added between the nodes, so the global trust value of the node needs to be recalculated. The global trust value is used as the basis for the selection of the consensus group and the main node group. The change of its value will inevitably lead to the change of the voting node, which will make the whole consensus process change dynamically with the outbound behavior and ensure the security of the system.

5 Evaluation

5.1 Number of messages delivered

In the Byzantine consensus algorithm based on the trust model proposed in this paper, the number of messages transmitted in the consensus process is improved. Assume that the total number of nodes in the blockchain system is N. We allow the nodes with top $d\%(0 < d \leq 100)$ trust value in the system participate in the consensus process, and the number of leading groups is determined as $x(1 \leq x \leq d\% * N)$. Then, the number of messages generated during a round of voting in the consensus process should be the sum of the number of messages between the leading groups and the number of messages between the consensus nodes, which is $x^2 + (d\% * N - x)^2$. The number of messages generated by the consensus must be less than the consensus message of the non-leading group which is $(d\% * N)^2$. When $x = \frac{1}{2} * d\%$, the number of messages reaches a minimum value which is$\frac{1}{2}(d\% * N)^2$. It can be seen that the minimum value of the communication message can reach half of the original number, so there will be a certain increase in the efficiency of the consensus.

For the whole, due to the introduction of the trust model, the overall number of messages may not drop significantly. However, it should be noted that the trust value of the node can be updated in real time when the transaction is completed, that is, the time of the consensus process is not occupied. Therefore, although the evaluation process of the trust value will generate some communication messages, it will not affect the consensus efficiency. On the contrary, due to the generation and establishment of trust values, it will allow more nodes join the system without significantly affecting the efficiency of the consensus, while ensuring the scope of the system and the efficiency of consensus.

5.2 Fault Tolerance Rate

As we all know, the traditional PBFT algorithm own$\frac{N-1}{3}$ fault tolerance rate. But in the group Byzantine consensus based on the trust model, the fault tolerance rate will be improved. Assume that the node behavior is consistent, that is, every node's behavior does not change significantly. Without considering the interference of the swing node and the voting weight, when the system selects the nodes with the trust value in top $d\%$, the nodes excluded from the consensus group will have no effect on the consensus. So it is easy to get the highest fault tolerance of the system which is$1 - \frac{2}{3}d\%$. When all nodes participate in the consensus, the fault tolerance rate is the lowest which is $\frac{N-1}{3}$. In the process of reaching consensus, since the system will assigns different voting weight values according to the trust value and the node with relatively high weight will have low probability of occurrence of downtime or wrong behavior, so the system's fault tolerance rate will increase after the introduction of weight voting.

5.3 Comparison with other Byzantine algorithms

The performance comparison of the Byzantine consensus algorithm based on the trust model (T-PBFT) with other Byzantine fault-tolerant algorithms is shown in table 1. In terms of communication complexity, T-PBFT still remains $O(N^2)$, there is no order of magnitude improvement. In terms of fault tolerance, the lower limit of T-PBFT theoretical fault tolerance can be equal to PBFT, CBFT and other algorithms, and its ratio of fault tolerance relates to trust value $d\%$. It also needs to pay attention that the lower the proportion $d\%$, the stronger the trend of system centralization. In terms of scalability, due to the complexity of traditional Byzantine algorithm message communication, when the number of participating nodes reaches a certain number, the system performance will drop sharply. Zyzzyva optimizes the message flow to Increase scalability. Although T-PBFT does not reduce the number of communications in the system, but by introducing the trust model and grouping, the scalability is improved. For the dependence of the master node, the judgment is mainly based on whether the system can work normally when master node gets out of order and the impact to the system when the master node generates malicious behavior. Since the master node in T-PBFT is composed of the leading node group. So the power of the master node is relatively low, and the system has weak dependence.

Table 1. Comparison of some Byzantine Consensus

	PBFT	CBFT	Zyzzyva	T-PBFT	Honey-Badger BFT
Communication complexity	$O(N^2)$	$O(N^2)$	$O(N)$	$O(N^2)$	$O(N^3 \log N)$
Fault tolerance	$\frac{N-1}{3}$	$\frac{N-1}{3}$	$\frac{N-1}{3}$	$\left(1-\frac{2}{3}d\%\right)N$	$\frac{N-1}{2}$
Scalability	weak	weak	strong	strong	weak
Dependency on the primary node	middle	middle	strong	weak	weak

6 Conclusions

This paper proposes a new consensus algorithm based on trust model, which is intended to improve the scalability and security of the blockchain system. At present, there is still much room for improvement. The main direction can focus on the application of the trust model and the setting of corresponding parameters. However, from the overall point of view, the algorithm have desirable performance. It improves the application scenario and scope of the Byzantine fault-tolerant algorithm, and at the same time improve the security, which is a consensus mechanism to be examined.

Acknowledgement

This work was supported by Nature Key Research and Development Program of China (No.2017YFB1400700), the National Natural Science Foundation of China (Nos.61602537, U1509214) and the Central University of Finance and Economics program of the Youth Talent Support Plan(No. QYP1808).

References

1. Bitcoin:A peer-to-peer electronic cash system, https://bitcoin.org/en/bitcoin-paper.last accessed 2018/09/20.
2. YUAN Yong,WANG Fei-Yue. Blockchain:The State of the Art and Future Trends[J]. Acta Automatica Sinica,2016,42(04):481-494(in Chinese).
3. Eyal I, Gencer A E, Renesse R V. Bitcoin-NG: a scalable blockchain protocol[C]// Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2016:45-59.
4. Castro M, Liskov B. Practical Byzantine fault tolerance[C]// ACM, 1999:173-186.
5. Anh D T T, Zhang M, Ooi B C, et al. Untangling Blockchain: A Data Processing View of Blockchain Systems[J]. IEEE Transactions on Knowledge & Data Engineering, 2017, PP(99):1-1.
6. King S, Nadal S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake[J]. 2012.
7. Delegated Proof-of-Stake Consensus, https://bitshares.org/technology/delegated-proof-of-stake-consensus,last accessed 2018/09/30.
8. Milutinovic M, He W, Wu H, et al. Proof of Luck: an Efficient Blockchain Consensus Protocol[J]. 2017.
9. Fan J, Yi LT, Shu JW. Research on the technologies of Byzantine system. Ruan Jian Xue Bao/Journal of Software, 2013,24(6):1346−1360 (in Chinese).
10. TenderMint: Consensus without Mining, https://tendermint.com/docs/tendermint.pdf. last accessed 2018/09/30.
11. Miller A, Xia Y, Croman K, et al. The Honey Badger of BFT Protocols[C]// ACM Sigsac Conference on Computer and Communications Security. ACM, 2016:31-42.
12. Kotla R, Alvisi L, Dahlin M, et al. Zyzzyva: Speculative Byzantine fault tolerance[J]. Acm Transactions on Computer Systems, 2009, 27(4):7.
13. Gueta G G, Abraham I, Grossman S, et al. SBFT: a Scalable Decentralized Trust Infrastructure for Blockchains[J]. 2018.
14. Cowling J, Myers D, Liskov B, Rodrigues R, Shrira L. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In:Proc. of the 7th Symp. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006. 177−190.
15. Liu S, Viotti P, Cachin C, et al. XFT: Practical Fault Tolerance Beyond Crashes[J]. Computer Science, 2015.
16. Zbierski M. Parallel Byzantine Fault Tolerance[M]// Soft Computing in Computer and Information Science. Springer International Publishing, 2015:321-333.
17. Zhao W. Optimistic Byzantine fault tolerance[J]. Parallel Algorithms & Applications, 2015, 31(3):254-267.
18. Kamvar S D, Schlosser M T, Garcia-Molina H. The Eigentrust algorithm for reputation management in P2P networks[C]// International Conference on World Wide Web. ACM, 2003:640-651.

19. Heba A. Kurdi. HonestPeer: An enhanced EigenTrust algorithm for reputation management in P2P systems[J]. Journal of King Saud University - Computer and Information Sciences,2015,27(3).