An Improved Blockchain Consensus Mechanism Based on Open Business Environment

Wenmiao Wu and Zhipeng Gao
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
Email: gaozhipeng@bupt.edu.cn

Abstract. With the rise of the blockchain, the underlying technology has also received more and more attention. The blockchain is a point-to-point distributed system, and the consensus algorithm is a mechanism to resolve the consensus of each node. Byzantine Fault Tolerance (BFT) is a fault-tolerant technology in the field of distributed computing. The blockchain network environment conforms to the Byzantine general problem model. In order to make it suitable for open business environment, we propose an improved blockchain consensus mechanism based on Practical Byzantine Fault Tolerance Algorithm (PBFT) that consider different business. The consensus algorithm proposed in this paper (OBE-PBFT) introduces the historical behavior credit rating of nodes to optimize the view change protocol in PBFT, so as to solve the problem of low consensus success rate and large traffic when encountering Byzantine nodes. The historical credit rating of the node can make the consensus process more robust, in order to solve the problem of different business in the alliance chain consensus mechanism in the open business environment.

1. Introduction
The blockchain first appeared in the paper "Bitcoin: a peer-to-peer electronic cash system"[1] published by Nakamoto in 2009. Its achievability has been proven by Bitcoin running to this day. Its outstanding advantage lies in decentralized design of distributed database, using time-stamp, Merkle tree structure, asymmetric key encryption algorithm, consensus algorithm and reward mechanism. The blockchain uses peer-to-peer network to achieve the transaction on decentralized credit, it proposes a new calculation paradigm for solving the problems of poor reliability and low efficiency in the centralized mode.

Among the internal mechanisms of the blockchain, the consensus algorithm is a key part of it, and has been the focus of researchers at home and abroad. The connotation of the consensus algorithm is: how each node distributed in the network agrees when each node is untrustworthy, and the trusted node is elected to obtain the accounting right. At present, the mainstream consensus algorithm algorithms include PoW (proof of work) [2], PoS (Proof of stake) [3], DPoS (delegated proof of stake) [4], Pool verification pool, paxos[5], Raft[6], PBFT(practical Byzantine fault tolerance algorithm)[9] and so on. However, there are many shortcomings in the current consensus algorithm of blockchain. As a result, the blockchain is far from meeting the performance requirements of many application scenarios in terms of efficiency and energy consumption. Consensus algorithms have become the bottleneck restricting the development of blockchain. Therefore, the study of high-performance consensus algorithms is of great significance for the development of blockchain.

This paper discusses various problems in existing consensus algorithms, especially those in the PBFT algorithm. We propose an improved consensus mechanism for the PBFT algorithm in an open business environment. The contribution of this paper lies in three aspects:
We propose an improved blockchain consensus mechanism based on Practical Byzantine Fault Tolerance Algorithm (PBFT), which has good scalability and increased the success rate of the consensus process.

We introduce the historical behavior credit rating of nodes to optimize the view change protocol in consensus algorithm, so as to solve the problem of low consensus success rate and large traffic when encountering Byzantine nodes.

We consider the optimization of consensus mechanisms in different business scenarios for the first time. The historical behavior of a node has different effects on the credit value, which could make the consensus process more robust. The consensus mechanism we propose can achieve the problem of focusing on different businesses in the context of open business, which makes it suitable for open business environment.

The rest of the paper is organized as follows. Section 2 introduces the existing exploration in the field of consensus algorithms and discusses the problems in PBFT. Section 3 describes the design of the improved consensus mechanism in detail. Section 4 shows the experiment results and analysis. Finally, Section 5 is the conclusion of this paper.

2. Related Work
This paper studies several consensus algorithms applied to the blockchain.

2.1. POW
PoW is an algorithm used in Bitcoin, which is known as mining. Through the OR operation, a random number satisfying the rule is calculated, that is, the current accounting right is obtained, and the data to be recorded in the current round is issued, and the other nodes of the whole network are verified and stored together. This is completely decentralized, and the nodes are free to enter and exit. However, Bitcoin has already attracted most of the world's computing power, and mining has caused a lot of waste of resources.

2.2. POS
The PoS consensus is an alternative to address the waste of resources and security flaws in the PoW consensus mechanism. The PoS consensus essentially uses equity proof to replace the hash-based workload proof in PoW, which is the highest in the system. Equity, rather than the highest computing power, obtains block accounting rights. Equity is expressed as the node's ownership of a specific amount of currency, called the coin age or coin days. The currency age is the specific amount of currency and its last transaction. The product of the length of time, each transaction will consume a certain amount of currency.

2.3. DPoS
The basic idea of the DPoS consensus is similar to the “board decision”, that is, each node in the system can grant its shareholding rights as a ballot to a representative, and the first N nodes that receive the most votes and are willing to become representatives will enter the “board”. The transaction is packaged and settled in turn according to the established schedule, and the new block is signed (ie produced). Although DPoS can significantly reduce the number of participating verification and accounting nodes, it can achieve second-level consensus verification, but the entire consensus mechanism relies on tokens.

2.4. Paxos
The Paxos algorithm is the most important algorithm for solving consistency problems in a distributed system. The application scenario of the algorithm in a distributed system is a scenario in which there is a failure but no malicious nodes exist. Paxos is the first proven consensus algorithm involving three different roles for proposer, acceptor, and learner. The principle is based on two-phase commit and extension.
2.5. PBFT

PBFT is a state machine replica replication algorithm, in which the service is modeled as a state machine, and the state machine performs replica replication at different nodes of the distributed system. If the total number of replica nodes is N, the system can withstand up to (N-1)/3 invalid replicas. The implementation process of the PBFT algorithm is as follows:

a) The master node generates a Pre-Prepare certificate, which includes the new zone fast, the certificate timestamp, and the master node signature. The primary node sends the Pre-Prepare certificate to the slave node, after which the primary enters the Prepare state.

b) After receiving the Pre-Prepare certificate from the node, if it is the first time to receive the certificate, the node enters the Prepare state and forwards the certificate to the other slave nodes.

c) The node receives the certificate sent by other nodes, and will verify the information in the certificate, including the correctness of the transaction within the block, the correctness of the block header information and the block height. If the block is approved, the approval feedback is sent back to the node that sent the certificate. A node receives its own 2f+1 approval feedback, indicating that the block is added to the end of the blockchain, and the certificate enters the Commit state.

To sum up, the traditional distributed consistency algorithm represented by Paxos and Raft does not consider Byzantine fault tolerance, that is, it only supports fault-tolerant fault nodes and does not support fault-tolerant nodes. Consensus algorithms represented by POW and POS are facing performance bottlenecks such as unavoidable centralization, long block confirmation periods, and low transaction frequency.

Therefore, the algorithm proposed in this paper will optimize the performance efficiency of the existing consortium blockchain consensus algorithm, and improve the overall consensus efficiency and increase the robustness.

3. Algorithm Design

3.1. Protocols in PBFT

Consistency protocol: At regular intervals in the blockchain system, a certain number of transactions or account state changes are packaged into blocks. The nodes in the system ensure that the recorded block information is correct and identical through the consistency protocol. There are two roles in the protocol: primary and secondary. There is only one primary node in a consensus, which is responsible for verifying the transactions received within a certain period of time, and the verified transactions will be packaged into blocks.

View Change protocol: All replicas operate in a rotation process called a View. In a view, one replica is used as the primary node and the other duplicate nodes are used as backups. Views are consecutively numbered integers. The master node is calculated by the formula \( p = v \mod |R| \), \( v \) is the view number, \( p \) is the replica number, and \( |R| \) is the number of replicas. The view rotation process needs to be initiated when the primary node fails.

Checkpoints protocol: The main purpose of the protocol is to maintain the size of the information stored by the node and to resolve the recovery of the certificate information, thereby reducing the memory overhead of the node. The checkpoint protocol defined in the PBFT mechanism is cleared after the inter-node timing negotiation. This is to prevent the individual nodes from being out of synchronization and need to collect the previous certificate.

3.2. Algorithm Details

The improved algorithm introduces the historical behavior credit rating of the node, and comprehensively considers the different influences of the historical behavior of the node on the credit value in the case of different business scenarios. Behavioral influence factors and time-sensitive functions are introduced to calculate the trust value of the node. A node with a high credit rating will have a greater probability of becoming a new master node in the view replacement protocol.

Method of calculating trust: The goal of the trust system is to estimate the trust of each node, which is determined by the behavior of each node during voting. Trust can be performed as part of a
consensus protocol and executed on each participating node. Trust is calculated independently and can be synchronized with the data information at any time.

**Behavioral Impact Factor:** During the operation of the blockchain, the impact of each transaction on the system will be completely different with the importance of the transaction. Therefore, it is necessary to analyze and evaluate the transactions between nodes, analyze the behavior of nodes when evaluating node trust, and introduce behavioral influence factors to solve the impact of historical behavior of nodes on trust between nodes, thus effectively curbing maliciousness. An attempt by a node to perform various malicious acts.

**Time-sensitive function:** The time axis is divided into equal time segments of length \( t_0 \). In the classic model, the integrated trust is recorded as \( \text{Trust}(i) \), which is specified to decay once every \( t_0 \), and \( t_{in} \) and \( t_{end} \) are time nodes entering and leaving the network. Then the node trust evaluation function with time-sensitive function is

\[
H(\text{Trust}(i)) = \text{Trust}(i)_{end} \left( 1 + \frac{\sum_{i=1}^{k} \text{Trust}(i)_{end} - \text{Trust}(i)_{in}}{k} \right)
\]

\[
k = \left\lfloor \frac{t_{end} - t_{in}}{t_0} \right\rfloor
\]

In the formula, \( \text{Trust}(i)_{in} \) and \( \text{Trust}(i)_{end} \) respectively represent the initial and final evaluation of the node during the evaluation period. The node trust evaluation function \( H(\text{Trust}(i)) \) with time-sensitive function can better reflect the influence of behavior on trust degree decaying with time in a certain period of time, which makes the trust value of the node more reasonable.

The flow of the improved view conversion protocol is as follows:

**Algorithm 1 Improved view change protocol**

1: system initialization
2: if the current consensus process fails, begin view change, do
3: for num in business scenarios, do
4: for node in current business scenario nodes, do
5: introduce behavioral impact factors to calculate node trust
6: integrate time-sensitive functions to update node trust
7: select a node with the highest trust value, vote for it and broadcast it
8: end for
9: end for
10: the node with the highest credit rating becomes the new primary node
11: end if
12: begin the next consensus process

Specifically, the process is as follows:

1) If the current consensus process fails, start the view conversion.
2) Each node calculates the trust value of each relevant node according to the business scenario it is in, combined with the behavior influence factor and the time-sensitive function, and then selects the highest node, votes and broadcasts it.
3) Each node aggregates all the votes received, with the highest number of votes as its approved new primary node.
4) If it is itself, the view conversion protocol is triggered, and other nodes are only recognized when receiving the message from the approved primary node triggering the view conversion protocol.
5) The node with the highest credit rating becomes the new primary node in the view change protocol.
6) Begin the next consensus process.
4. Experiment and Analysis

In this section, we used Ethereum to build a blockchain operating environment for experiment. We set the nodes to 5, 6, 7, 8, 9, 10. Each node is configured as i5-6100 CPU, Ubuntu 16.04 Operating System, 1GB RAM, 20GB ROM.

In several aspects, we compare the differences between OBE-PBFT, the algorithm we propose, and the classic PBFT, and analyze the experimental results.

4.1. Communication Cost Comparison

During the commit process in PBFT, it will be broadcast from the node, which will undoubtedly cause huge waste. If the amount of data transmitted each time is BlockSize, then there are N nodes, every time a view change occurs, the power consumption is:

\[ \text{Cost}=N(N-1)\times \text{BlockSize} \]  

(3)

So when the count of view change is \( V_a \), the total power consumption is:

\[ \text{Total}_a = \text{Cost} \times V_a \]  

(4)

For OBE-PBFT, due to the introduction of the credit rating mechanism, the new primary node selected in the view change protocol has a higher probability of making the consensus process successful, so the frequency of view change will be reduced. So \( V_b < V_a \)

\[ \text{Total}_b = \text{Cost} \times V_b \]  

(5)

So \( \text{Total}_b < \text{Total}_a \). That is, in the long-term operation of the system, the communication cost of OBE-PBFT is less than PBFT.

4.2. Operating Efficiency Comparison

In the experiment, we compared the two algorithms, the classical PBFT and the OBE-PBFT we proposed, from different aspects. We compared the system operation efficiency of the two algorithms at different numbers of nodes (see Figure 1), different numbers of Byzantine nodes (see Figure 2), and different numbers of business scenarios (see Figure 3).

![Figure 1. Result of different numbers of nodes](image1.png)  

![Figure 2. Result of different numbers of Byzantine nodes](image2.png)

The reliability of the system communication is enhanced due to the trust mechanism we introduced. As shown in Figure 1, the OBE-PBFT’s TPS has been significantly improved compared to PBFT.

The introduction of trust mechanism and behavioral influence factors will make the view change protocol more reliable. As shown in Figure 2, as the number of Byzantine nodes in the system increases, the advantages of the OBE-PBFT become more obvious.
As can be seen in Figure 3, the OBE-PBFT is significantly better than the PBFT algorithm in the case of different numbers of open business scenarios.

4.3. Fault tolerance Performance Comparison

In PBFT, if the total number of replica nodes is N, the system can withstand up to (N-1)/3 invalid replicas. In OBE-PBFT, since the number of nodes participating in the trust calculation is equal to the total number of replica nodes, the system can also withstand up to (N-1)/3 invalid replicas. So the fault tolerance performance of the two consensus algorithms is consistent.

5. Conclusion

In this paper, we propose an improved PBFT consensus mechanism OBE-PBFT based on the open service environment. Aim at the problems existing in the PBFT algorithm, we introduce the credit evaluation in different business environments, which to solve the problem of low consensus success rate and large traffic when encountering Byzantine nodes and achieve the effect of significantly improved the efficiency of the blockchain system. It is verified by experiments that OBE-PBFT is more efficient and robust than PBFT in improving the long-term operation of the alliance chain system. In the future work, we will continue to optimize the algorithm and further apply it to the actual consortium blockchain application system.

6. References

[1] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system[J]. Consulted (2009).
[2] Jakobsson M , Juels A . Proofs of Work and Bread Pudding Protocols[M]// Secure Information Networks. Springer US (1999).
[3] Vasin P. Blockchain’s proof-of-stake protocol v2[J/OL]. (2014).
[4] Larimer D.Delegated proof-of-stake white paper[EB/OL]. (2015).
[5] Lamport L. Paxos made simple. ACM Sigact News, 2001,32(4): 18¡25 (2001)
[6] Ongaro D, Ousterhout J K. In search of an understandable consensus algorithm. In: Proceedings of the USENIX Annual Technical Conference. Philadelphia, PA, USA:USENIX ATC, 2014. 305¡19(2014)
[7] Copeland C, Zhong H X. Tangaroa: a byzantine fault tolerant raft [Online], available: http://www.scs.stanford.edu/14au-cs244b/labs/projects/copeland zhong.pdf, April 10,2018.
[8] Lamport L. The part-time parliament[J]. Acm Transactions on Computer Systems, 1998, 16(2):133-169.(1998)
[9] Castro M, Liskov B. Practical Byzantine fault tolerance[C]// Symposium on Operating Systems Design & Implementation. (1999).
[10] Androulaki E, Barger A, et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains [EB]. https://arxiv.org/abs/1801.10228. (2018)