Optimization Scheme of Consensus Mechanism Based on Practical Byzantine Fault Tolerance Algorithm

Zhipeng Gao and Lulin Yang

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.
yanglulin@bupt.edu.cn

Abstract: Blockchain was first proposed in 2009, it is a kind of distributed ledger system with peer-to-peer network, which has drawn wide spread attention because of its characteristics such as decentralization ,security and credibility . The consensus algorithm of the blockchain is a mechanism for achieving agreement among the nodes in the blockchain. How to reach consensus quickly and effectively is the core issue of the blockchain. Byzantine nodes are invalid or malicious nodes in the blockchain. This paper considers the actual situation of Byzantine nodes in the blockchain. For the problem that the classical PBFT algorithm has too much communication spending and cannot dynamically follow the change of consensus nodes, an improved PBFT algorithm in this paper is proposed. In the improved Practical Byzantine consensus algorithm(IMP-PBFT), the convergence speed of the consensus process is effectively improved under the condition of the fault tolerance rate. The experiment proves the accuracy and effectiveness of the improved PBFT algorithm.

Keywords: blockchain consensus PBFT communication spending

1 Introduction

In 2009, Nakamoto[1] was first proposed the concept of blockchain in the paper "Bitcoin: A Peer-to-Peer E-Cash System". As the most typical application of blockchain. Bitcoin uses numbers techniques such as timestamps, signatures, asymmetric cryptographic algorithms, and consensus mechanisms based on workload proofing implement a truly decentralized trustworthy trading system. Since the block chain technology cannot be tampered with, safe and reliable characteristics, it has quickly received the attention of the financial and Internet industries, and various applications based on blockchain technology have also been developed. Such as the crowd funding platform based on Ethereum[2], IBM's hyperledger fabric project[3], EOS[4] public chain and so on. With the development of blockchain technology, the blockchain has been transitioned from the 1.0 era represented by Bitcoin to the 2.0 era represented by hyperledger and Ethereum. The blockchain's application scenarios are also developed from public chains to alliance chains suitable for commercial applications.

The core issue of the blockchain is how to achieve an effective consensus while meeting consistency and availability. Bitcoin uses a consensus mechanism for Proof

of work, in this case each node wins the billing right of the block by quizzing a random number. However, the POW [5] has obvious problems. It requires huge power costs, the consensus efficiency is very low, and the transaction delay is very high. In the case of Bitcoin, the confirmation of a transaction is about one hour after the generation of six new blocks. Therefore, it does not apply to business scenarios represented by the alliance chain. POS [6] (proof of stake) is a stake-based consensus mechanism that obtains the billing rights from the node with the highest property instead of the most powerful one, but this may cause the rights of some nodes to be too large, which also deviates from the original intention of decentralization of the blockchain. In the hyperledger project led by IBM, the PBFT [7] (practical Byzantine fault tolerance) algorithm based on Byzantine problem was used. The PBFT algorithm was first proposed by Castro and used to solve the problem of how the asynchronous distributed system agrees. The nodes reach agreement through negotiation. The algorithm can ensure that the blockchain can still operate normally when no more than one third of the nodes fail. However, in the PBFT algorithm, the client request needs to go through five stages of request, preparation, preparation, confirmation, and response, which will greatly delay the consensus process, and the number of nodes must be fixed and cannot be dynamically changed. Based on the above problems, this paper proposes an improved PBFT algorithm, which can effectively shorten the consensus process and dynamically add or delete nodes while having a high fault tolerance.

2 Related works

2.1 Practical Byzantine Fault Tolerant Algorithm

PBFT (Practical Byzantine Fault Tolerant Algorithm) [8] is a fault-tolerant algorithm based on the Byzantine general problem, which is divided into five phases to achieve consistency. PBFT solves the problem that the original Byzantine algorithm is not efficient, and makes the algorithm feasible in practical applications. In PBFT, a blockchain net of N nodes can accept f byzantine nodes. Where $N \geq 3\mathrm{f}+1$ .The PBFT algorithm has the following definitions:

All consensus nodes operate in a single view. Each view has a unique primary node, and the remaining nodes are called replica nodes. The corresponding number of the node is 0.1...n-1, The rule corresponding to the view of the primary node is:

$$p = v \bmod N \tag{1}$$

In this formula p is the primary node number, v is the view number, and N is the total number of nodes. PBFT's communication process [9] is as follows(see Fig. 1).

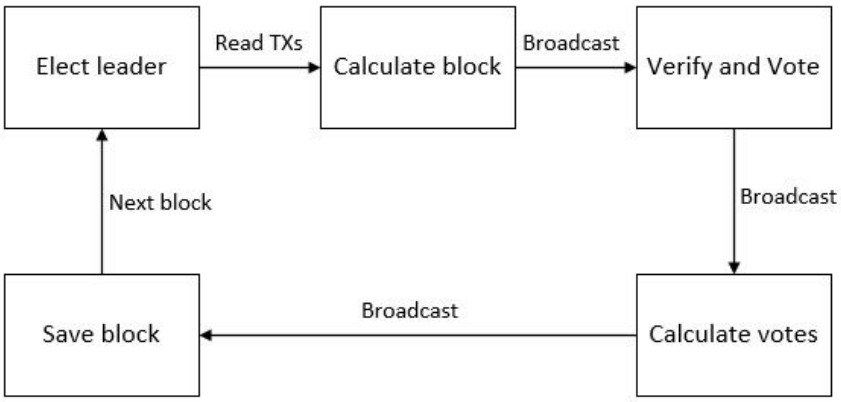

Fig. 1. the communication process of Practical Byzantine Fault Tolerant Algorithm

Request: The request is sent by the client to start the consensus process.

Pre-prepare: After receiving the request from the client, the primary forwards the prepared message to all replica nodes. The format of the message is *<pre-p,v,n,d>*, where v is the view number and n is the requested number, d is the content of the request;

Prepare: After the replica node accepts the preparation message, it enters the preparation phase. The replica node will broadcast the preparation message to all the remaining replica nodes. The message format is *<p,v,n,d,i>*, *i* is the replica node number. When at least *2f+1* preparation messages consistent with the prepared message, the node enters the confirmation phase and writes a *<d, v, n, i>* message to log.

Commit: Each node forwards an acknowledgment message to all remaining nodes. When a node receives *f+1* acknowledgment messages, it begins to execute the request and sends feedback to the client.

Confirm: The block will be written to the blockchain after verified. End of consensus process.

If the client does not reply within a valid time, the request will be broadcast to all replica nodes. If the request is not processed at the replica node, the replica nodes will forward the request to the primary node. If the primary node does not broadcast the request, then the primary node is considered invalid. If there are enough duplicate nodes to consider the primary node to be invalid, the view will be change.

3 Algorithm design

3.1 The overall idea of the algorithm

Suppose there are 3 nodes in the blockchain. $N_{1,}$ N_2 are honest nodes and N_{er1} is a byzantine node. Let N_1 receive the transaction vector of N_2 as TX_{21} and receive the N_{er1} transaction vector as TX_{er1}. If the correct consistency is to be achieved, the algorithm f [10] must satisfy

$$\begin{aligned} TX_{21} &= f(TX_{21}, TX_{er1}) \\ Store\ (TX_{er1}) &= false \end{aligned} \tag{2}$$

The classic PBFT algorithm uses a three-stage protocol to ensure consistency, but it pays a great deal of communication spending, and in PBFT the number of nodes is fixed. In the scenario of the alliance chain, the consensus node has no subjective malicious motives. Only the network has a downtime, or the communication is disconnected, the Byzantine node will appear, in general, the probability of such a thing occurring is extremely low, so there is no need to conduct a three-stage broadcast every time to reach a consensus.

This paper considers the first method of using two-step broadcast communication to reach a consensus, and switches to the classic PBFT algorithm if the consensus is not reached. In terms of the selection of the primary node, this paper adopts a voting mechanism to select the primary node. When a node joins/exits, the voting rights are added or deleted for the node, and the current primary node remains unchanged, after the view switching protocol is triggered, all voting nodes re-select the primary node. This saves system overhead due to node changes to change views.

3.2 Primary node selection strategy

In this paper, voting mechanism is adopted in the selection of the primary node. When the system is stable, selecting the most recognized master node of the whole network can effectively reduce the frequency of changing view; When the node changes, the system does not immediately switch the view. Instead, it waits for the primary node to make an error and then re-votes to determine the primary node. This prevents wasted time by frequently switching views. The voting strategy is as follows:

(1) Each consensus node can vote for one node, the voting is in the form of broadcast, and the node with more than half of the votes becomes the master node.

(2) If the primary node is not determined, select the three nodes with the highest number of votes and vote again, the node with the highest number of votes is elected as the primary node.

(3). the primary node determines the switching order of the views. If no nodes join/exit in the process of consensus, the views will be switched, in the determined order.

(4) If a node joins or exits, assigns a number to the new node and deletes the exit node number.

(5). When the view switching protocol is triggered, if there is a new node at this time, vote again and determine the view switching order.

3.3 Block synchronization strategy

When all the nodes in blockchain are honest nodes, this paper uses a two-stage agreement to reach a consensus. When the Byzantine node appears, it switches to the PBFT algorithm, the IMP-PBFT algorithm uses the following strategy to reach a consensus:

(1) Select the primary node, and the primary node initiates the consistency protocol.

(2) Primary node package transaction records and send a broadcast message to the replica nodes. The message's format is *<pre,v,n,d>*, where *v* is the view number of the primary and *n* is the requested number, *d* is the content of the request.

(3) After receiving the primary's message, the replica nodes verify the message and send a confirmation message to the primary node. The confirmation's format is *<com, v, n, d,i>*,and *i* is the number of replica node .

(4) If all replica nodes are verified. the primary node packages the transaction records into block and add it to the blockchain.

(5) The replica nodes verified the block and synchronizes the block to the blockchain.

In the fourth step, when the primary node finds that there is a conflict in the response from the replica nodes or does not receive all the verified messages within one timestamp, it will consider that there is a byzantine node in the replica nodes, and triggers the PBFT algorithm. In the fifth step, if the replica node finds that the block packed by the primary node is inconsistent with the block verified by itself, the primary node will be considered to be faulty, and the view switching protocol is triggered.

3.4 Algorithm overall process

The overall process design of the IMP-PBFT algorithm is designed as follows (See Fig 2).

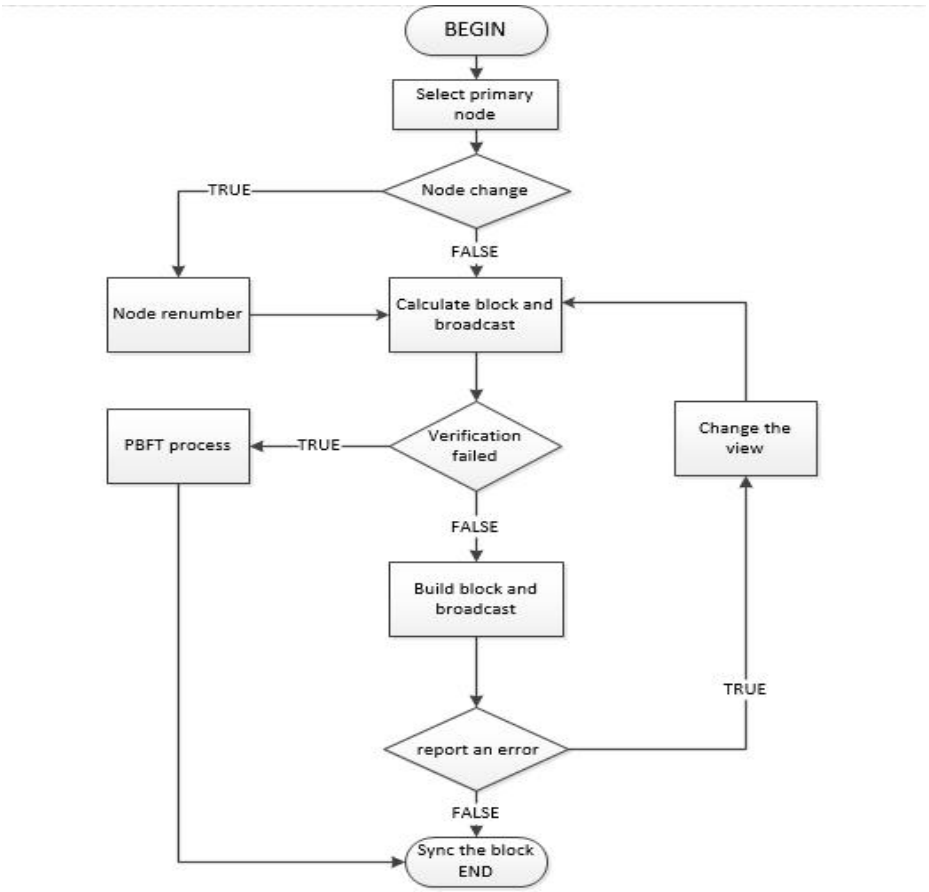

Fig. 2 The improved Practical Byzantine Fault Tolerant Algorithm

4 Experimentation and analysis

4.1 Computing overhead verification

In the classic PBFT algorithm, the communication overhead in the network is

$$TS = N * N * (headsize) + N * (blocks) \tag{3}$$

TS is the total spending, *N* is the number of nodes, and in the IMP-PBFT, the total spending is

$$TS = N * (headsize) + N * (blocks) \tag{4}$$

As can be seen from above formulas, the improved algorithm reduces the cost of a broadcast compared to PBFT.We can verify this inference from the blockchain's TPS.

In this paper we used Ethereum to build a blockchain operating environment and set the nodes to 4,5,6,7,8. Each node is configured as follows table 1:

Table 1. the configuration of experimental environment

CPU	Intel(R) Core(TM) i5-6100 CPU @3.70Ghz
OS	Ubuntu 16.04
RAM	1GB
ROM	20GB

Then we observe the system throughput under different nodes, the result is shown as fig3.

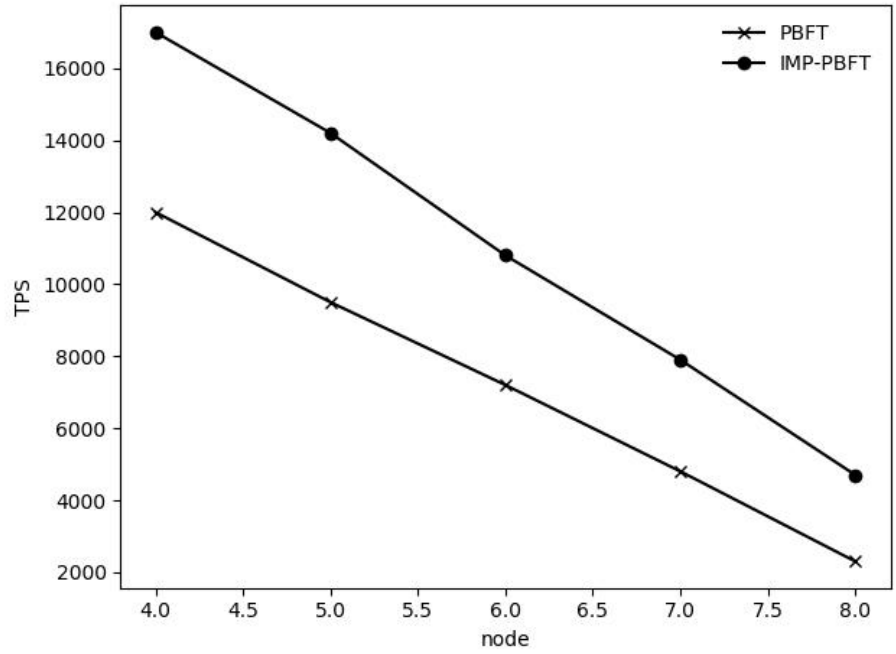

Fig. 3 Comparison of TPS under different nodes

As can be seen from the figure, due to the reduction of communication costs, the IMP-PBFT‘s TPS has been significantly improved compared to PBFT.

4.2 Fault tolerance verification

We use 8 nodes to build a blockchain and gradually increase the number of Byzantine nodes, and then observe the TPS in the blockchain. The results are shown below：

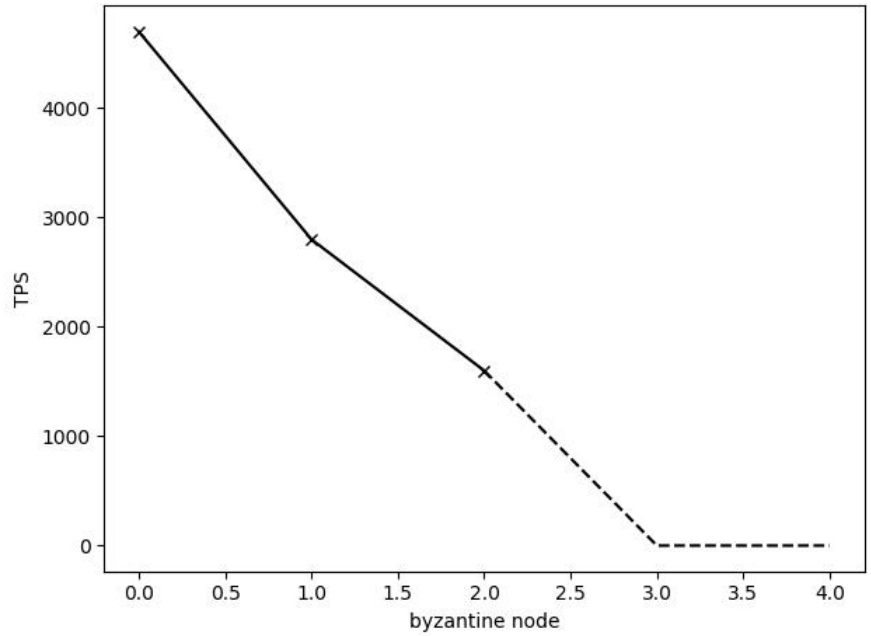

Fig. 4 Relationship between tps and byzantine nodes

It can be seen that when there are more than 3 Byzantine nodes, the TPS is 0, which means that nodes cannot reach a consensus.

5 Conclusion

Targeted at the problems encountered by PBFT in practical application scenarios, this paper proposes an improved Byzantine (IMP-PBFT) consensus algorithm for adaptive and dynamic monitoring nodes. Compared to the classic PBFT algorithm. The improved algorithm enables faster consensus and guarantees fault tolerance. The focus of the future is to further effectively enhance the stability of the algorithm in various client environments and apply it to actual production practices.

References

1. Nakamoto S. Bitcoin:A peer-to-peer electronic cash system[J/OL],2008.

2. Ethereum [EB/OL].2017.https://www.ethereum.org/
3. Hyperledger [EB/OL].2017 https://www.hyperledger.org/
4. EOS[EB/OL] .2018 https://eos.io/
5. Vasin P. Blockchain's proof-of-stake protocol v2[J/OL].2014
6. Larimer D.Delegated proof-of-stake white paper[EB/OL]
7. Lamport L,Shostak R,Pease M.The byzantine generals problem[J].ACM Trans on Programming Languages & Systems.1982.4(3):382-401
8. Castro M.Practical byzantine fault tolerance and proactive recovery [J].ACM Trans on Computer Systems(TOCS),2002,20(4):398-461
9. Jianjun Sun ,Jiaqi Yan ,kem Z.K.Zhang.Blcokchain-based sharing services :What blockchain technology can contribute to smart cities[J].2016.2
10. Happe A ,Krenn S,Lorunser T.PBFT and Secret-shring in Storage Settings[C]//Twenty-fourth International Workshop on Security Protocols .2016.