RPBFT: A Scalable Consensus Mechanism for Large Blockchain Systems

Weizhe Wang[1,2,3], Daxin Tian[1,2,3,4(✉)], Xuting Duan[1,2,3,4], and Jianshan Zhou[1,2,3]

[1] School of Transportation Science and Engineering, Beihang University, Beijing, China
{weizhewang,dtian,duanxuting,jszhou}@buaa.edu.cn
[2] State Key Lab of Intelligent Transportation System, Beijing, China
[3] Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems and Safety Control, Beijing, China
[4] Zhongguancun Laboratory, Beijing, China

Abstract. As an emerging technology, blockchain has enabled trustworthy data sharing and efficient cooperation in many industries. It can also make data traceable and auditable, which perfectly meets the regulatory demands for data security and privacy. However, the employment of blockchain is limited in some fields due to the poor scalability of the consensus mechanism. Therefore, this paper introduces a scalable consensus mechanism called RPBFT to improve the performance of large-scale systems. To demonstrate the advantage of RPBFT over classic consensus mechanism, we find a way to implement it in blockchain systems and conduct performance tests to evaluate the throughput and latency of the established systems. The results show that RPBFT is superior and more suitable for large-scale blockchain systems.

Keywords: Blockchain · Consensus mechanism · Scalability

1 Introduction

With the rapid development of information technology and the digital transformation of many industries, a large amount of data is generated, collected and shared. As an important way to release the value of data, data sharing breaks the boundaries of cooperation between enterprises and organizations, which is significant for promoting the efficiency of operation. However, in the real business of data sharing, it's inevitable to face a series of data security issues and regulatory concerns. So it's important to take measures to ensure the security of data sharing.

In order to achieve secure data sharing that meets regulatory requirements, many researchers have explored related technologies, and one of the highly promising technologies is blockchain. Blockchain is a new kind of distributed database [1], which integrates various technologies such as peer-to-peer communication, consensus mechanism,

This research was supported by the National Key Research and Development Program of China under Grant No. 2022YFC3803700.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
V. C. M. Leung et al. (Eds.): Qshine 2023, LNICST 573, pp. 509–520, 2024.
https://doi.org/10.1007/978-3-031-65126-7_42

digital signature, and distributed application. Unlike traditional centralized architectures, blockchain decentralizes the network structure to avoid a single point of failure. It uses distributed consensus to achieve fault-tolerance against malicious behaviors, and uses a chain structure with cryptographic method to make data tamper-proof. These features make the data on a blockchain consistent, traceable and auditable, which is regarded to have the power of enabling secure data sharing and efficient cooperation [2].

However, the scalability of consensus mechanism limits the employment of blockchain in some industries that requires large scale deployment and quick response capability [3], like intelligent transportation system (ITS) industry. A typical way to integrate blockchain with transportation systems is to deploy nodes on roadside units to harness the hardware capability of roadside infrastructure [4], as depicted in Fig. 1. During high traffic volumes like traffic jams or rush hours, many vehicles may communicate with roadside units to update their condition on the blockchain, thus generating massive amount of transactions in a short time. In this scenario, a less-scalable consensus mechanism would slow down the processing of transactions, causing the pending transactions to gather or even lose.

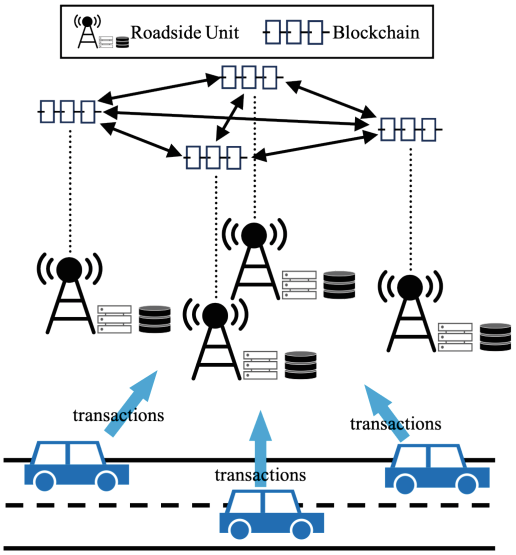

Fig. 1. A typical way to integrate blockchain with transportation systems.

To meet regulatory requirements, the identity of each node in a blockchain network must be known to all, so it's needed to adopt permissioned blockchain, a kind of blockchain that uses classic consensus mechanism to achieve distributed consistency. Such consensus mechanism requires all nodes in the network to communicate with each other frequently. While this messaging method plays an important role in maintaining the safety and liveness of a decentralized system, it incurs a huge amount of communication overhead when the scale of the system is large. Due to this, such consensus mechanism is difficult to support large-scale deployment of blockchain systems. Therefore, it's highly

needed to design a suitable consensus mechanism to improve scalability. With such goal, we introduce a scalable consensus mechanism called RPBFT, and verify its performance on real blockchain systems.

The contributions of this paper are summarized as follows. We introduce a scalable consensus mechanism to improve the performance of large-scale blockchain systems. The design of the consensus mechanism is based on the idea of selecting a part of the nodes in the system to run consensus protocol. We also introduce the method to select and replace the nodes that participate in consensus procedures to ensure security. To verify the effectiveness of the consensus mechanism, we evaluate its performance in real blockchain systems.

The remainder of this paper is organized as follows. Section 2 discusses related studies. Section 3 presents RPBFT consensus mechanism. Section 4 describes how we implement a blockchain system that runs the consensus mechanism, and how to carry out performance tests. Section 5 provides the test results and the performance analysis of RPBFT. Finally, Sect. 6 concludes the paper.

2 Related Work

In 1999, Miguel Castro and Barbara Liskov proposed Practical Byzantine Fault Tolerance (PBFT) [5]. PBFT runs in an environment where the identity of every node is known to all in advance, making it a natural fit for blockchain systems that aim to be secure and controllable. It can achieve Byzantine fault tolerance, which means it resists malicious behaviors from participants. Besides, it's relatively mature in terms of implementation. Due to these, it is one of the most widely-used blockchain consensus mechanisms.

However, PBFT has the problem of high communication overhead. In each round of consensus, all nodes need to broadcast their consensus messages to the whole network, making the communication complexity of the system $O(n^2)$ (n is the number of nodes participating in the consensus procedures). As a result, with the expansion of node scale, the performance deteriorates rapidly, and therefore the system would be unable to support large-scale blockchain applications.

Many researchers have proposed improved consensus mechanisms based on PBFT. Cowling et al. [6] propose Hybrid-Quorum (HQ), a hybrid Byzantine-fault-tolerant consensus mechanism which has a lightweight Byzantine quorum protocol. In HQ, all replicas have no need to interact with each other to keep consistent. Therefore the communication complexity is reduced. Kotla et al. [7] proposed Zyzzyva that uses speculation to reduce the cost of replication process. In Zyzzyva, replicas reply to the request of a client without first running an expensive three-phase commit protocol. Such design improves the system performance when there is no Byzantine faulty replicas. Yin et al. [8] proposed HotStuff, a consensus mechanism that combines with aggregate signature technique. Compared with PBFT, it simplifies the process of leader replacement and reaches a lower communication complexity, thus has better scalability. Some researchers try to introduce Trusted Execution Environment (TEE) into the design of BFT consensus mechanism, and one of the representative results is FastBFT [9]. By using TEE, the security model is simplified, thus achieving higher security threshold. Besides, the authors also design a tree broadcast strategy to reduce the communication complexity of the mechanism.

3 RPBFT Consensus Mechanism

3.1 Basic Idea

As for the relation between node scale and performance, there is a simple idea that, if no matter how large the node scale is, there are only a fixed number of nodes participating in the consensus algorithm, then the performance of the system will not decline rapidly as the total number of nodes increases. By this way, more nodes could be supported in a blockchain system. According to this idea, we introduce RPBFT, a more scalable consensus mechanism for blockchain [10]. As shown in Fig. 2, RPBFT selects a fixed number of nodes randomly in the whole network as consensus nodes. Consensus nodes participate in each round of consensus algorithm jointly. When a new block is linked to the blockchain, the consensus nodes will synchronize it to other nodes (called verification nodes) using tree broadcast strategy [11]. By eliminating the impact of node scale on communication complexity, RPBFT has better scalability able to reduce latency and improve throughput in large-scale blockchain applications. Moreover, RPBFT periodically replaces the consensus nodes to ensure safety and prevent conspiracy of the consensus nodes.

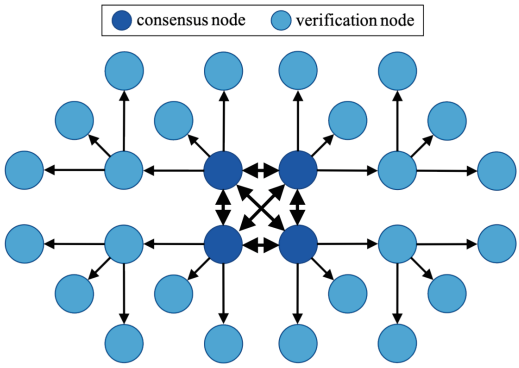

Fig. 2. RPBFT consensus mechanism.

3.2 Selection and Replacement of Consensus Nodes

Suppose the total number of nodes in the system is m. During initialization, the number of consensus nodes n and the period k are needed to be set. The meaning of k is that the replacement of the consensus node takes place every time a collection of k blocks is issued. Meanwhile, the nodeIDs of all nodes (a binary string of fixed length, unique to every node) are sorted. The sort order of each node is called node number (1, 2,…, m). When the system runs for the first time, the first n nodes become consensus nodes automatically.

In order to ensure the safety of the system, every time k blocks are produced by current consensus nodes, the system will remove a node from the consensus node group

and turn it into a verification node, then select one node from previous verification nodes and add it to the consensus node group. Nodes come in and out according to their node numbers, which is illustrated in Algorithm 1 and Fig. 3.

Algorithm 1: Selection and replacement of consensus nodes of RPBFT

```
1  Initialization
2  initialize m, n and k
3  initialize the nodeID of every node
4  initialize curHeight (current block height) to 0
5  sort the nodes by nodeID using 1, 2, 3,..., m
6  select n nodes whose nodeID ranges from 1 to n as consensus nodes, into con-
   sensus group
7  while the blockchain system is working do
8     wait for a new block to be added to the blockchain through PBFT consensus
   algorithm run by nodes in consensus group
9     curHeight += 1
10    if curHeight % k == 0 then
11       remove the oldest node in consensus group
12       add a new one whose nodeID is right after the latest node in the list
13    end if
14 end while
```

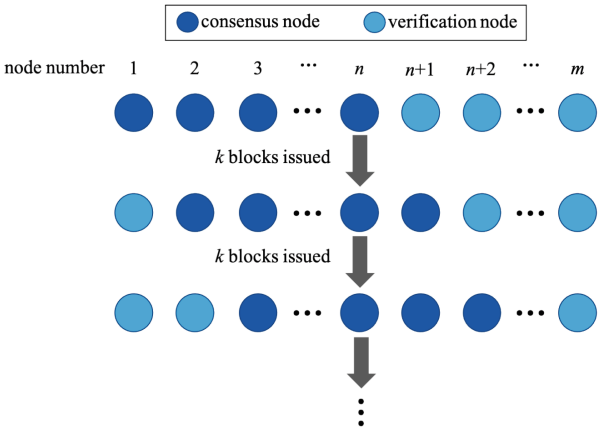

Fig. 3. The replacement of consensus nodes.

3.3 Algorithm for RPBFT Consensus Nodes

In RPBFT, consensus nodes run PBFT algorithm to reach consensus about new blocks. Assuming that the number of malicious nodes is f, according to the Byzantine fault tolerant model, the system can tolerate Byzantine errors when n is bigger than $3f$. Let's say n is equal to $3f + 1$.

PBFT achieves distributed consistency through voting. As shown in Algorithm 2 and Fig. 4, in each round of consensus, the leader broadcasts a pre-prepare message about a selected transaction to all consensus nodes. After receiving the pre-prepare message, each node will verify whether the transaction is valid. If valid, a prepare message about that transaction is broadcast to all nodes to notify that they have received a pre-prepare message about that transaction. After that, if a node receives up to $2f + 1$ prepare messages from other nodes, it broadcasts a commit message to all nodes. Subsequently, if the node receives up to $2f + 1$ commit messages from other nodes, it confirms the transaction locally. Through the above process, the transaction will finally reach consensus in the whole network.

Algorithm 2: PBFT protocol

```
1  //pre-prepare phase
2  as a leader
3     choose a transaction tx for consensus
4     broadcast pre-prepare(tx) to all nodes (including itself)
5  for every node
6     wait for pre-prepare(tx) message from leader
7     if pre-prepare(tx) is validated then
8        broadcast prepare(tx) to all nodes
9     end if
10 //prepare phase
11 for every node
12    wait for 2f+1 prepare(tx) messages
13    broadcast commit(tx) to all nodes
14 //commit phase
15 for every node
16    wait for 2f+1 commit(tx) messages
17    commit tx
```

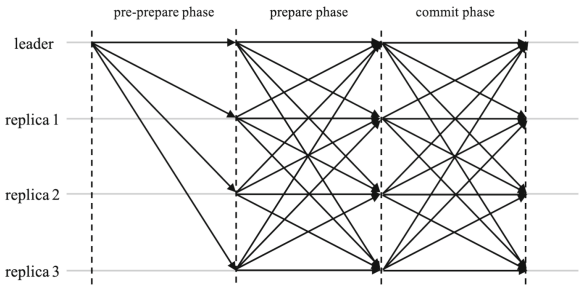

Fig. 4. Consensus process between consensus nodes in RPBFT.

Through multiple rounds of message broadcast and collection, the capacity of malicious nodes is strictly limited. As long as the number of malicious nodes is less than one-third of the consensus nodes, the consensus algorithm can run normally [12]. When the leader is abnormal, the algorithm can also replace it. At this point, the messages

collected in the aforementioned process can serve as a proof to restore the consensus process.

3.4 Communication Overhead Analysis

In pre-prepare phase, the leader broadcasts to all nodes, which means $n-1$ messages are sent. Here we exclude the message that is sent to itself. In both prepare and commit phases, each node sends messages to all nodes, meaning $n(n-1)$ messages across the network in each phase. After a round of consensus finishes, the consensus nodes should send new block to verification nodes, meaning additional $m-n$ messages. Therefore, the total number of messages in each round of RPBFT can be expressed as

$$S_{\text{RPBFT}} = (n-1) + n(n-1) + n(n-1) + (m-n) = 2n^2 - 2n + m - 1. \quad (1)$$

For a PBFT system with m nodes, the total number of messages in each round is

$$S_{\text{PBFT}} = (m-1) + m(m-1) + m(m-1) = 2m^2 - m - 1. \quad (2)$$

So the difference of communication overhead caused by RPBFT can be expressed as

$$S_{\text{PBFT}} - S_{\text{RPBFT}} = 2(m-n)(m+n-1). \quad (3)$$

As $n \geq 1$ and $m > n$, we have

$$S_{\text{PBFT}} - S_{\text{RPBFT}} > 0. \quad (4)$$

This shows that the communication overhead of RPBFT is smaller than that of PBFT for the same node scale. From the equation we know that, as the node scale expands, i.e. the value of m increases, the difference of communication overhead will also increases in the case that n is constant. Moreover, for a fixed m, the difference will get smaller if n increases.

4 Implementation

This paper uses FISCO BCOS to build blockchain system. FISCO BCOS is an enterprise-level financial blockchain platform open-sourced by Chinese enterprises [13]. It provides developers with many handy tools to build and connect with blockchain systems. After building and starting a chain locally, we use the built-in console to interact with the blockchain node to verify that the chain we build functions normally.

To conduct performance test on the blockchain system, we install Hyperledger Caliper, a blockchain performance benchmark framework. It supports standardized test results and is compatible with multiple blockchain platforms [14], including FISCO BCOS. As shown in Fig. 5, Caliper plays a role of a client in the performance test [15]. It sends transactions to the system under test at a certain rate, collects performance indicators such as transaction confirmation latency and throughput, and generates a graphic report. By configuring the network condition and smart contract correctly, we connected Caliper to a blockchain system and output a report in HTML format [16], as shown in Fig. 6.

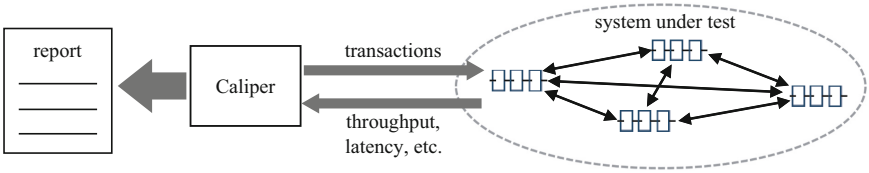

Fig. 5. The form of performance test.

Caliper Report

Basic information

DLT: fisco-bcos
Benchmark: Hello World
Description: This is a helloworld benchmark of FISCO BCOS for caliper
Test Rounds: 2
Details

Benchmark results

Summary
get
set

System Under Test

Version: 2.0.0
Size: 20 Nodes
Distribution: Single Host
Details

Summary

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
get	10000	0	1000.2	0.02	0.00	0.00	1000.1
set	10000	0	368.8	1.41	0.09	0.49	361.3

get

Test performance of getting name

round 0

Performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
get	10000	0	1000.2	0.02	0.00	0.00	1000.1

Resource consumption

Type	Name	Memory(max)	Memory(avg)	CPU% (max)	CPU% (avg)
Process	node0(avg)	27.6MB	27.4MB	4.80	3.12

Fig. 6. Performance report generated by Hyperledger Caliper.

5 Performance Evaluation

The performance tests were done on a macOS machine with 2.3 GHz 8-Core Intel Core i9 processor and 16 GB memory. Throughput and latency are the main indicators of blockchain system performance. In this paper, we run tests over systems of different numbers of nodes to see how the throughput and latency change with the expansion of the number of nodes, which implies the scalability of consensus mechanism.

Figures 7 and 8 depict the throughput and latency of blockchain systems using different consensus mechanisms in different node scales. In this experiment, the number of RPBFT consensus nodes is set to 4, and we take PBFT for comparison. As illustrated in Fig. 7, the increase in node number leads to a quadratic growth of consensus communication overhead in PBFT system, thus resulting in a rapid decline in throughput. In comparison, the decline for RPBFT system is gentler as the node scale expands due to the much slower increase in communication overhead. For the same system scale, it can also be seen that RPBFT reaches a much higher throughput than PBFT. From Fig. 8, we can observe that the latencies of the two mechanisms are comparable in 10 and 20 nodes systems. However, due to the quadratic communication complexity across the network, the latency of PBFT systems is not as stable as that of RPBFT systems when

the node scale expands. Therefore, the design of splitting nodes into different functional parts in RPBFT can improve the scalability of consensus mechanism and realize higher performance in large scale blockchain systems.

We also test the performance of RPBFT systems with different number of consensus nodes while the total number of nodes remains constant. In this experiments we set the total number of nodes to 40, and take 40-nodes PBFT system for comparison. Figures 9 and 10 display the variation of throughput and latency respectively. In Fig. 9 we can see that the throughput keeps decreasing as the number of consensus nodes increases. This is because the consensus nodes inside the system also run PBFT algorithm, thus the increase in the number of consensus nodes also leads to a quadratic increase in communication overhead. Similarly, while the latency keeps stable between 4 and 28 consensus nodes, it starts growing steadily too as the number of consensus nodes increases, which is shown in Fig. 10.

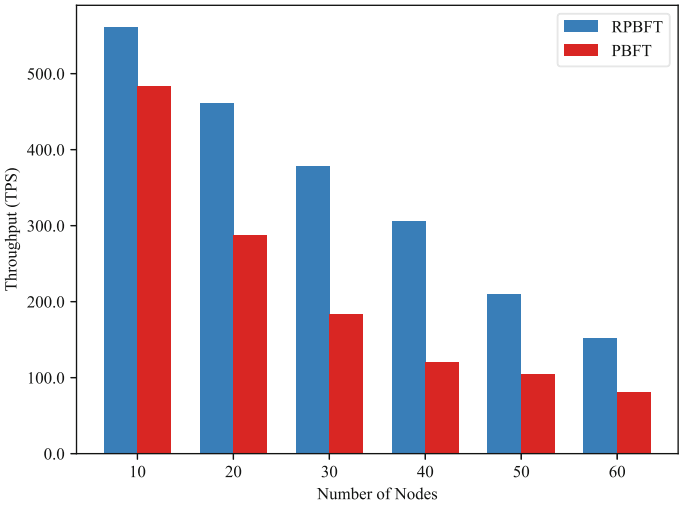

Fig. 7. Comparison of RPBFT and PBFT in terms of throughput.

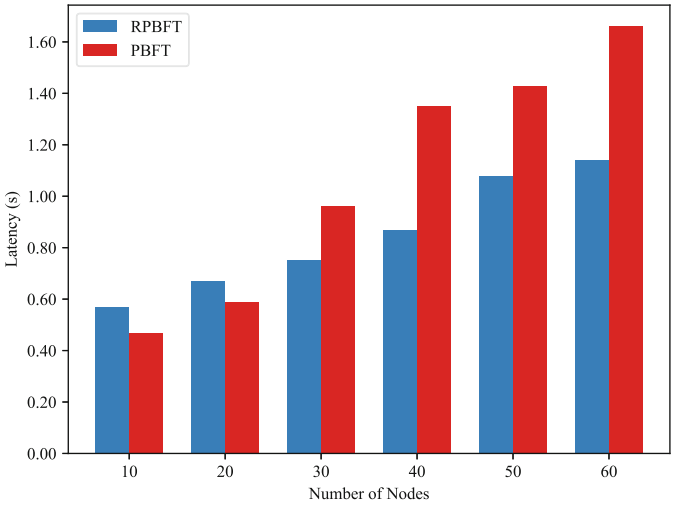

Fig. 8. Comparison of RPBFT and PBFT in terms of latency.

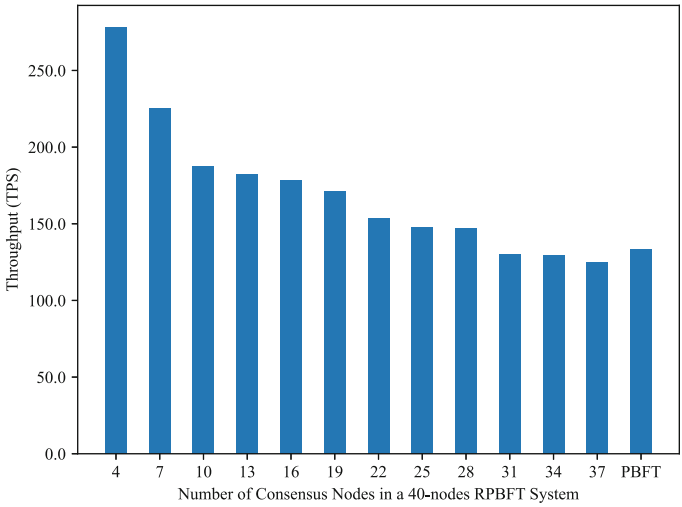

Fig. 9. Throughputs of RPBFT systems with different numbers of consensus nodes.

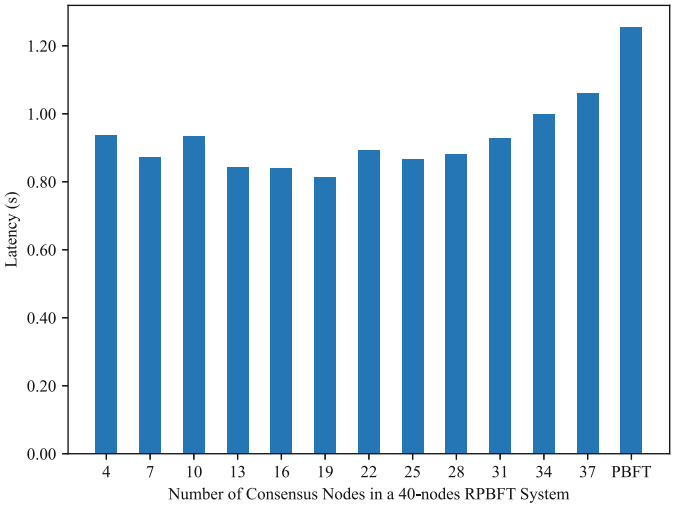

Fig. 10. Latencies of RPBFT systems with different numbers of consensus nodes.

6 Conclusion

In this paper, we introduced RPBFT, a consensus mechanism for blockchain in regulatory scenario. A separate architecture, which divides all nodes into consensus nodes and verification nodes, is designed to improve the scalability of the mechanism and thus facilitate large scale use of blockchain system. Furthermore, RPBFT has a mechanism for consensus node replacement, which strengthens its fault-tolerant capability by avoiding conspiracy. The performance evaluation reveals the advantage of RPBFT over classic PBFT in terms of throughput and latency. Systems running RPBFT gain higher throughput and lower latency when the node scale gets larger, and such advantage remains when the system contains more consensus nodes. The outperformance shows that RPBFT is more suitable for regulatory blockchain use in industries requiring large scale deployment and quick response capability.

References

1. Shrestha, R., Bajracharya, R., Shrestha, A.P., Nam, S.Y.: A new type of blockchain for secure message exchange in VANET. Digit. Commun. Netw. **6**(2), 177–186 (2020)
2. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: architecture, consensus, and future trends. In: IEEE International Congress on Big Data, pp. 557–564. IEEE (2017)
3. RPBFT design analysis. https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/articles/3_features/32_consensus/rpbft_design_analysis.html. Accessed 20 Jul 2023
4. Kang, J., Yu, R., Huang, X., et al.: Blockchain for secure and efficient data sharing in vehicular edge computing and networks. IEEE Internet Things J. **6**(3), 4660–4670 (2019)
5. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI), pp. 173–186. USENIX Association (1999)

6. Cowling, J., Myers, D., Liskov, B., Rodrigues, R., Shrira, L.: HQ replication: a hybrid quorum protocol for byzantine fault tolerance. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI), pp. 177–190. USENIX Association (2006)
7. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative byzantine fault tolerance. ACM Trans. Comput. Syst. **27**(4), 45–58 (2009)
8. Yin, M., Malkhi, D., Reiter, M, K., Gueta, G.G., Abraham, I.: HotStuff: BFT consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC), pp. 347–356. Association for Computing Machinery (2019)
9. Liu, J., Li, W., Karame, G.O., Asokan, N.: Scalable byzantine consensus via hardware-assisted secret sharing. IEEE Trans. Comput. **68**(1), 139–151 (2019). https://doi.org/10.1109/TC.2018.2860009
10. RPBFT. https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/design/consensus/rpbft.html. Accessed 20 Jul 2023
11. FISCO BCOS synchronization optimization. https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/articles/3_features/31_performance/sync_optimization.html. Accessed 28 Jul 2023
12. PBFT in FISCO BCOS. https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/en/docs/design/consensus/pbft.html. Accessed 28 Jul 2023
13. FISCO BCOS introduction. https://fisco-bcos-documentation.readthedocs.io/en/latest/docs/introduction.html. Accessed 24 Jul 2023
14. Hyperledger Caliper introduction. https://hyperledger.github.io/caliper/v0.2/getting-started. Accessed 26 Jul 2023
15. Caliper stress test practice on FISCO BCOS platform. https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/articles/4_tools/46_stresstest/caliper_stress_test_practice.html. Accessed 28 Jul 2023
16. FISCO BCOS adapter. https://hyperledger.github.io/caliper/v0.2/fisco-config. Accessed 28 Jul 2023