Peer-to-Peer Networking and Applications (2021) 14:2826–2839
https://doi.org/10.1007/s12083-021-01103-8

An Optimized Byzantine Fault Tolerance Algorithm for Consortium Blockchain

Yuxi Li[1] · Liang Qiao[1] · Zhihan Lv[1]

Received: 30 November 2020 / Accepted: 11 February 2021 / Published online: 16 March 2021

Abstract
According to different application scenarios of blockchain system, it is generally divided into public chain, private chain and consortium chain. Consortium chain is a typical multi-center blockchain, because it has better landing, it is supported by more and more enterprises and governments. This paper analyzes the advantages and problems of Practical Byzantine Fault Tolerance (PBFT) algorithm for the application scenarios of the consortium chain. In order to be more suitable for consortium chains, this paper proposes a new optimized consensus algorithm based on PBFT. Aiming at the shortcomings of PBFT, such as the inability to dynamically join nodes, low multi-node consensus efficiency, and primary master node selection, our optimized algorithm has designed a hierarchical structure to increase scalability and improve consensus efficiency. The simulation results show that compared with PBFT and RAFT, our new consensus algorithm increases the data throughput while supporting more nodes, and effectively reducing the consensus delay and the number of communication times between nodes.

Keywords Byzantine fault tolerance · PBFT · Consensus algorithm · Consortium blockchain

1 Introduction

Blockchain is an underlying technology that supports Bitcoin operation proposed by Satoshi Nakamoto in 2008 [1]. In essence, it is a decentralized and multi-point maintenance distributed database system by comprehensively using cryptography, consensus algorithm and point-to-point communication [2]. It has the characteristics of unforgeability, information traceability, anonymous protection, openness and transparency [3]. According to different application scenarios of blockchain system, such as Artificial Intelligence and Internet of Things [4, 5],it is generally divided into public chain, private chain and consortium chain [6]. The public chains, such as Bitcoin [1] and Ethereum [7], are completely decentralized and open to all nodes that can access the Internet. All nodes can join, exit, access, and submit transactions without restrictions [8]. Private chains are only used by individuals and private organizations, generally not open to the outside, and there is a certain degree of centralized control [9]. The consortium chain is a multi-centered structure of the blockchain, which is jointly constructed and maintained by agreed organizations [10]. Each node usually has a corresponding entity organization. After the node is authenticated and authorized, it can join, access and submit transactions. Each consortium member also has different data permissions. Consortium chain is the current mainstream direction, which allows multiple companies and institutions to achieve substantial and powerful collaboration, and can promote the healthy and orderly development of the blockchain industry.

The consensus mechanism is a necessary element and core part of the blockchain, and it is the key to ensuring the efficient cooperation of the blockchain system [11]. In the

Yuxi Li and Liang Qiao contributes equally to this work, both are the first authors.

This article is part of the Topical Collection: *Special Issue on Blockchain for Peer-to-Peer Computing*
Guest Editors: Keping Yu, Chunming Rong, Yang Cao, and Wenjuan Li

✉ Zhihan Lv
lvzhihan@gmail.com

Yuxi Li
7117139@163.com

Liang Qiao
leonqiaoove@gmail.com

[1] School of Data Science and Software Engineering, Qingdao University, Qingdao 266071, China

blockchain system, how to make each node keep their data consistent through a rule is a core issue. The solution to this problem is to develop a consensus algorithm to achieve the consistency and correctness of the ledger data on different nodes. This requires learning from existing algorithms for achieving state consensus in distributed systems [12], determining the mechanism for selecting accounting nodes in the network, and how to ensure that the ledger data forms a correct consensus in the entire network. The existing blockchain consensus algorithms can be roughly divided into Proof-of-X (PoX) consensus algorithms and Byzantine Agreement algorithms [13].

The PoX series of algorithms mainly include Proof-of-work (PoW) [14], Proof-of-stake (PoS) [15] and Delegated-proof-of-stake (DPoS) [16]. The PoX algorithm is mainly applicable to public chains, which reach consensus by investing tokens on the chain, increasing the cost of service request proposals, and relaxing the need for final consistency confirmation [17, 18]. The private chain mainly uses the distributed consensus algorithm RAFT to complete the consensus [19]. RAFT mainly faces node downtime, without considering malicious nodes to tamper with data, and cannot solve the byzantine fault tolerance problem. In the consortium chain environment, it is not necessary to add a token incentive mechanism on the chain. Most of the consortium chains currently use the Byzantine Agreement algorithms. The Byzantine Generals problem refers to "it is impossible to try to achieve consistency through message passing on an unreliable channel with message loss." In order to solve the Byzantine generals problem, the method adopted by the Byzantine agreement is to ensure that a consensus can be reached through a distributed method. Even if a Byzantine failure occurs, it will not be affected. The "Byzantine failure" refers to the execution of the algorithm in the distributed system. Any mistake, including irrational behavior. PBFT is a Byzantine consensus algorithm and a general solution to the consistency problem of distributed systems with Byzantine error nodes. In the PBFT algorithm, the service is deterministic, that is, the results obtained by executing the operations in the same state under each consensus node are the same, and each consensus node must have the same when starting to perform the operation. status. The brief operating steps of the PBFT algorithm are: First, the client sends a request to the leader node. Second, the leader broadcasts the request to all backup nodes. Then, all nodes perform the requested service and then send the reply back to the client. Finally, when the client receives f+1 replies from different nodes in the network and obtains the same result, the request will be successfully processed, where m is the maximum allowed number of failed nodes. PBFT [20] is a Byzantine Agreement algorithm, which can reduce the operating complexity of the byzantine agreement and completely separate the blockchain from the token reward mechanism on the chain. PBFT has a small number of startup nodes, a relatively high consensus efficiency, and a fault tolerance rate close to 1/3, which does not require a lot of computing power to maintain. But PBFT also has some shortcomings. For example, the PBFT algorithm uses a C/S architecture [21], cannot adapt to P2P networks, cannot dynamically sense the number of nodes, and its performance drops sharply as the number of nodes increases [22, 23]. There is also the problem of low scalability, and the random selection of primary node leads to the problem of selecting malicious node as primary node [24].

The contribution of this paper is mainly to propose Scalable Hierarchical Byzantine Fault Tolerance (SHBFT), which is more suitable for multi-node participation in the consortium chain. The SHBFT algorithm optimizes the PBFT algorithm in many aspects. By improving the consensus node partition structure and making it hierarchical, it reduces the algorithm consensus delay and improves data throughput. The selection and impeachment mechanism of the master node is set in the SHBFT algorithm to optimize and improve the consensus efficiency. And design a dynamic joining mechanism of nodes to improve the scalability and dynamics of the algorithm. Moreover, in the SHBFT algorithm, all nodes participate in the consensus while ensuring the consensus efficiency, and local consensus can reach a global consensus.

2 Current status of BFT-type consensus mechanism

Consortium chain has better landing, so it is supported by a lot of enterprises and governments. At present, there are few consensus algorithms that are really proposed for the consortium chain. In 1999, Castro and Liskov proposed Practical Byzantine Fault Tolerance [20], or PBFT for short. This is the first time that the complexity of the Byzantine agreement has been reduced from exponential level to polynomial level, which is of great significance. PBFT is not proposed for the consortium chain, but PBFT can solve the byzantine problem and be applied without token incentives [25]. However, its application is limited due to its inability to dynamically join nodes, low multi-node consensus efficiency, and primary master node selection. Therefore, in terms of blockchain, many researchers have made various improvements and innovations to the BFT algorithm.

In terms of consensus strategy, [26] proposed Egalitarian Practical Byzantine Fault Tolerance (ePBFT), which optimizes the process of selecting the master node in PBFT so that each node in the chain is equal and efficient. This mechanism performs omission operations

without affecting the overall operation, improves the efficiency of data backup and verification, optimizes the consensus process of the blockchain, and accelerates the consensus process [26]. Wang et al. [24] applied the voting algorithm and proposed vBFT, which divides the nodes in the network into three categories. Nodes with different identities have different responsibilities, which weakens the centrality to a certain extent, and the algorithm is dynamic. When the number of nodes in the vBFT algorithm changes, there is no need to restart the system, nor waste or occupy resources [27]. Based on the combination of POV mechanism and PBFT algorithm, Zhu et al. (2019) introduced random parameters into the node recommendation algorithm, and the reputation value is no longer the only criterion for node recommendation. The ISODATA algorithm is used to solve the shortcomings of PBFT message complexity and poor scalability. And it simplifies the consensus processing process of the PBFT algorithm to improve efficiency [28].

In terms of consensus agreements, the HoneyBadgerBFT was proposed by [29], and its process consists of two parts: atomic broadcast and asynchronous common subset. HoneyBadgerBFT can carry out consensus under an asynchronous network and does not rely on any time assumptions about the network environment [29]. Gueta et al. (2019) proposed a scalable decentralized trust infrastructure for blockchains, which solves the two major challenges of scalability and decentralization, and can work on hundreds of backup nodes while supporting the execution of Ethereum smart contracts [30]. Li et al. [31] designed EPBFT for the problem that PBFT is not suitable for dynamic networks, and different steps can be taken to reach a consensus according to the network environment of the system. This algorithm uses verifiable random function (VRF) to elect consensus nodes. And it simplifies the checkpoint protocol and view change protocol to reduce communication overhead and time required [31]. The DBFT proposed by [32] establishes the double-response mechanism. Moreover, the establishment of a self-conflict checking mechanism solves the related problems of view changes, and realizes the graceful performance degradation under normal conditions [32].

In terms of method innovation, [33] implemented the Musch BFT algorithm, and improved the scalability and stability of the PBFT algorithm. By using windows, the algorithm adapts to the actual number of faulty nodes f, thereby avoiding unnecessary messages. The algorithm can tolerate more failures, ensure reliability, and increase the complexity of the message without sacrificing delay [33]. Gao et al. [34] proposed T-PBFT, which uses the characteristic trust model to construct a credible consensus group to reduce the number of consensus nodes and communication complexity [34]. The G-PBFT designed by Lao et al. (2019) is a location-based and scalable consensus protocol. It is mainly aimed at IoT-blockchain, so the problem of malicious nodes is basically not considered. Through the location-based endorser election and era switch mechanism, the network overhead is reduced, and the consensus efficiency and scalability are improved [35].

It can be seen that how to design a consensus mechanism with low latency, high efficiency, and high throughput is a problem that researchers all hope to solve.

3 Scalable hierarchical Byzantine Fault Tolerance

3.1 SHBFT consensus mechanism model

SHBFT hierarchizes all consensus nodes and divides the layers into the primary net layer and several secondary net layers. Such a hierarchical structure can significantly improve consensus efficiency. SHBFT can achieve consensus of multi-layer nodes. For the convenience of description, this paper mainly takes the situation of two layers as an example. As shown in Fig. 1, it briefly shows the structure of SHBFT consensus mechanism in the two layers situation.

3.1.1 Roles for nodes

In the SHBFT mechanism, nodes are divided into three main roles, namely primary node, secondary node and tertiary node, which participate in and complete the consensus process together. Among them, the primary node and secondary nodes are collectively referred to as leaders, and the nodes that are not leader of this layer are collectively referred to as backups.

1. Tertiary node: The tertiary nodes are responsible for generating and sending messages to their secondary node, which mainly include request information and time stamp. And hash the message into a digest message and send it to other tertiary nodes in the secondary net layer. Tertiary nodes also have the ability to impeach their secondary node.
2. Secondary node: The secondary node is responsible for receiving messages from tertiary nodes, and summarizes all request messages and its own request messages. After the secondary node obtains the consent of the tertiary nodes, it needs to further send the message set in the primary net layer, where consensus is then made.
3. Primary node: The primary node needs to collect the message sets from all the secondary nodes for the final summary and sort, and after reaching a consensus with

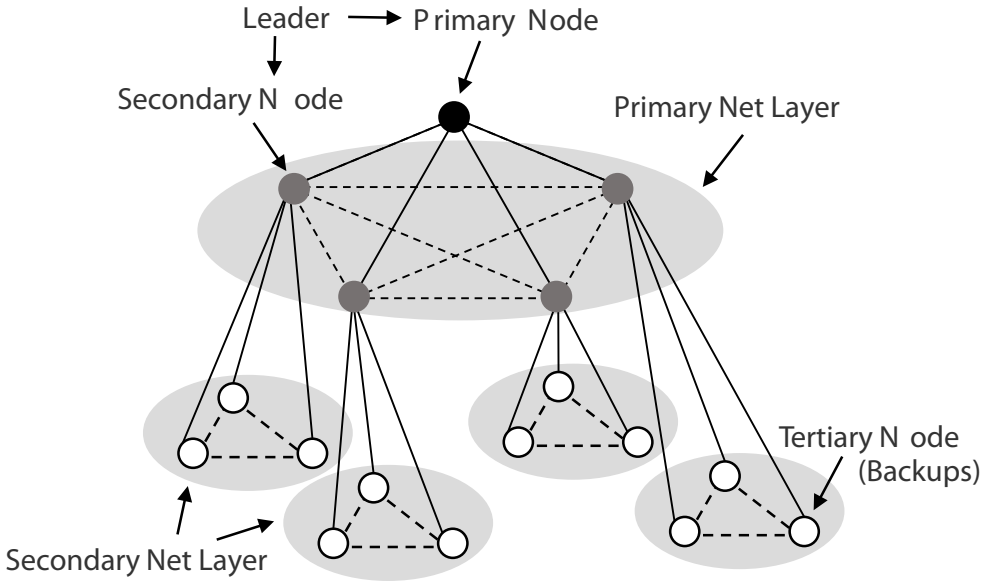

Fig. 1 Structure of consensus mechanism

the nodes of the primary net layer, the summarized set is packaged into blocks.

3.1.2 Running framework

The main process of the SHBFT consensus mechanism consists of three stages, and the description of each stage is as follows.

1. Secondary net layer submission stage: As shown in Fig. 2, each tertiary node in the secondary net layer submits the operation request and timestamp to the secondary node to which it belongs, and sends a digest message to other tertiary nodes of the secondary net layer. After the secondary node receives the messages submitted by each tertiary node within a period of time, the secondary node sorts the messages in time and packs them into a set, and sends a response containing the set information to each tertiary node. After verification by the tertiary nodes, the secondary node and each tertiary node reach a similar PBFT consensus, and each tertiary node feeds back "commit" message to the secondary node to agree to the secondary node to submit and consensus in the primary net layer. When the secondary node receives "commit" messages from f+1 (f is the maximum allowed number of failed nodes) tertiary nodes, the secondary node submits the request set to the primary node of the primary net layer.
2. Primary net layer consensus stage: As shown in Fig. 3, each secondary node broadcasts the message set to other secondary nodes and the primary node in the primary net layer. After the primary node of the primary net layer receives the message set for a period of time, it merges and sorts its own request and the message set submitted by all the secondary nodes according to time, and consensus with the secondary nodes. The primary node then packs the final set into a block. And digitally

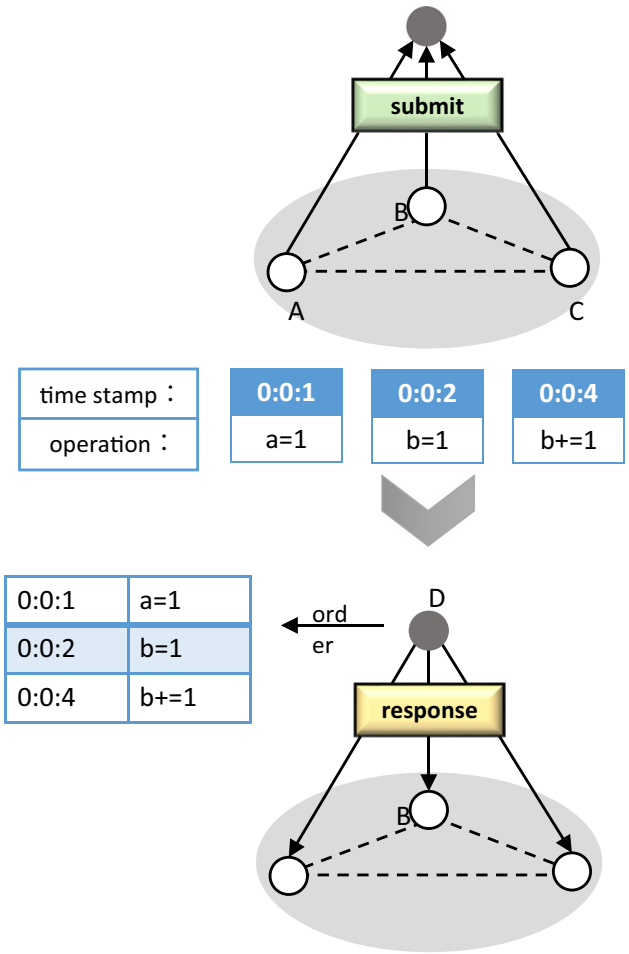

Fig. 2 Secondary net layer submission stage

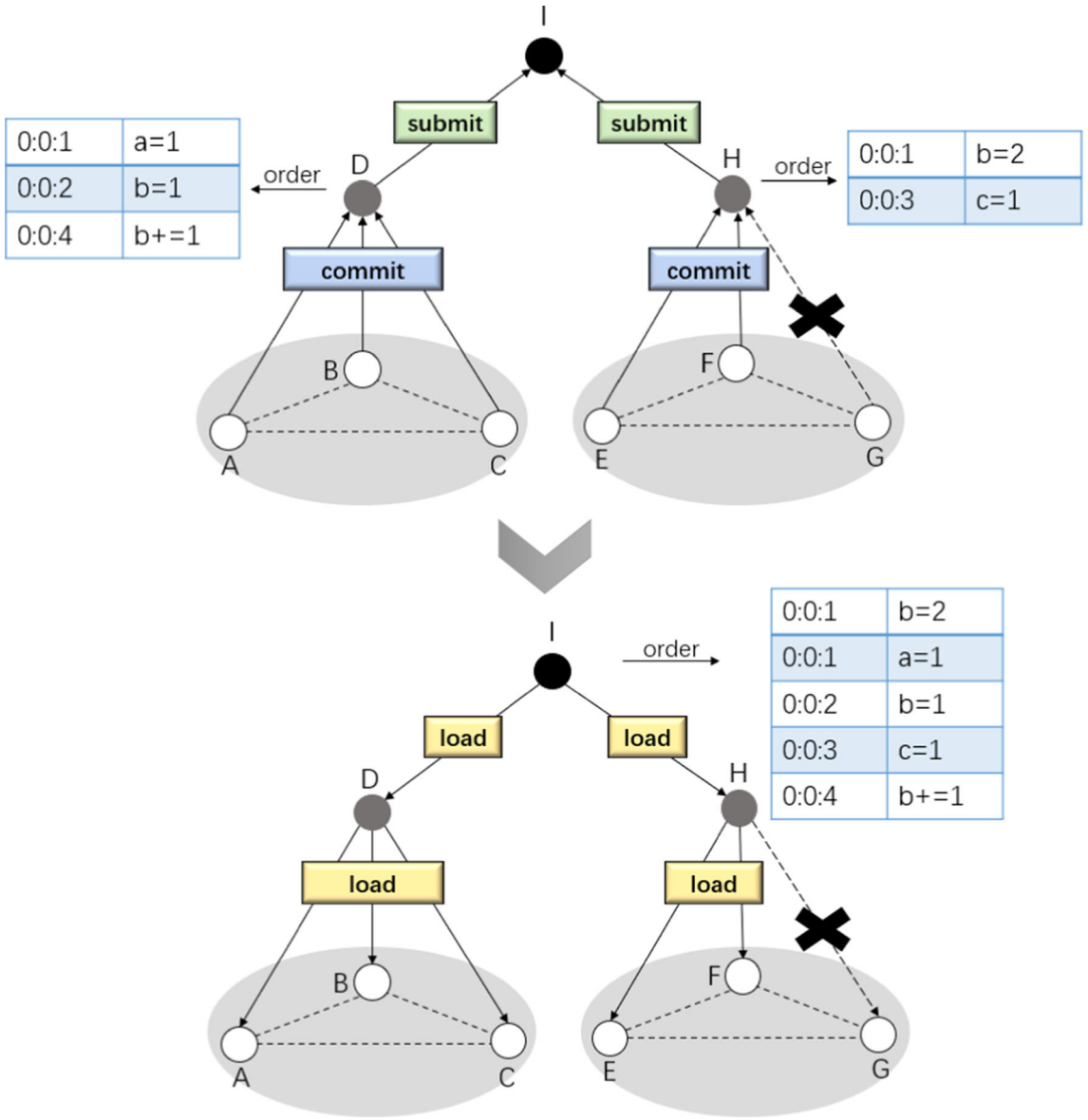

Fig. 3 Primary net layer consensus stage

sign the block, and then broadcast the "load" message containing the signed block and the block itself to the secondary node for verification.

3. Verification and execution stage: The secondary node verifies the message. If the verification fails, the impeachment mechanism against the primary node is triggered. If the verification is successful, the requested content of this block can be executed and the block recorded on the chain. And after the verification is successful, the secondary node needs to broadcast the "load" message to its subordinate tertiary nodes, and the tertiary nodes verify this message. If the verification fails, the impeachment mechanism against the secondary node is triggered.

The data transmission process of the SHBFT mechanism consensus process is shown in Fig. 4. In the period in which the leaders receive the submission information of each backup, unlike other consensus mechanisms with a fixed consensus period, SHBFT realizes the dynamic adjustment of the consensus window period. The leader calculates the next period time according to the number of transactions received in the current period and sends it to each backup with the returned information. When the leader receives more transactions in a unit period, the time of the next period is shortened, so that the transactions submitted by the node can be quickly responded. On the contrary, when there are fewer transactions received, the time of the next period is extended to save network resources.

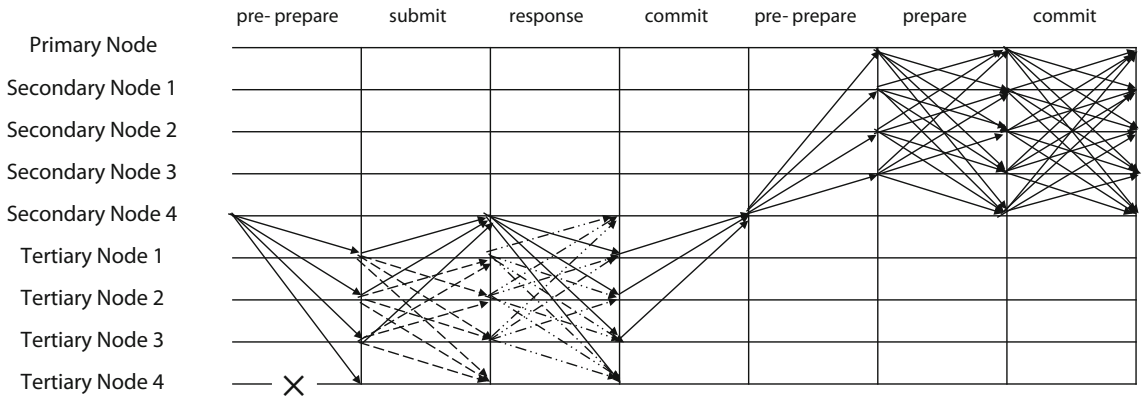

Fig. 4 SHBFT consensus process.

Algorithm 1 Dynamically adjust the consensus window period.

Input: transNum, nodesNum
Output: curWinTime

$preTransRate \leftarrow curTransRate$
$preWinTime \leftarrow curWinTime$
$curTransRate \leftarrow transNum/nodesNum$
if $curTransRate > 1$ **then**
    $curTransRate \leftarrow 1$
end if
$curWinTime \leftarrow preWinTime*(1+curTransRate-preTransRate)$
return curWinTime

3.2 SHBFT node management

3.2.1 Leader election

SHBFT adopts a leader election mechanism similar to RAFT [36], adding the concept of term. The term is represented by a continuous integer number and is monotonically increasing, representing a period of time of any length. All nodes record the term number of the leader of this layer and the leader of the next layer. Normally, the term of the leader of this layer stored by all nodes in this layer is the same.

When each node receives an election request in this layer, the term stored by the nodes will automatically increase by one, and then vote for election. Each node can only cast one vote during a term, and vote for the first request to vote it receives. Nodes will only vote for nodes whose term number is greater than or equal to their stored current term number.

Leaders keep their heartbeat with their backups. When a node finds that it has not received a heartbeat, that is, when it loses connection with the leader and does not have a leader, or thinks that the leader is negligent and needs impeachment, it will send election requests to other nodes to apply to become a new leader. Of course, other nodes may also send election requests. Other nodes judge and vote on the election request, and nodes that do not want to change the leader can directly abstain. When the node that initiates the election request counts that it has more than half of the votes within a limited time, it will default to becoming the new leader. The new leader will then periodically send heartbeats to the backups. The heartbeat contains the current term number of the node of the successfully elected new leader. Figure 5 is an example of the election process of the leader, in which the node A did not receive the heartbeat of the leader E for a period of time, and the node B discovers that the leader E is negligent and triggers the impeachment mechanism. Node A and node B initiate election applications at the same time. Then node B is successfully elected as the new leader. Leader B immediately broadcasts heartbeats, and exchanges positions with the old leader E.

After receiving the heartbeat, the node checks whether the term number in the heartbeat is greater than or equal to the term number stored by itself. If this condition is met, the node recognizes the new leader and submits subsequent request messages to the new leader. If the term number is less than the term number stored by the node itself, the node does not agree with the new leader.

When a new leader is elected, the new leader exchanges theoretical positions with the old leader, the old leader moves down one layer, and both sides need to exchange neighbor tables. The neighbor table includes the leader of this node, other nodes in the same layer, the backups that belong to it, and two term numbers. Attributes in the neighbor table without parameters are represented as 0. Figure 6 illustrates the replacement process of the old and new leaders including the exchange of neighbor tables, where node H is the old leader and node E is the new leader.

The election mechanism can ensure the effectiveness and efficiency of the leader, and also enable the backups to get high-quality consensus services, thereby ensuring the consensus efficiency of this layer.

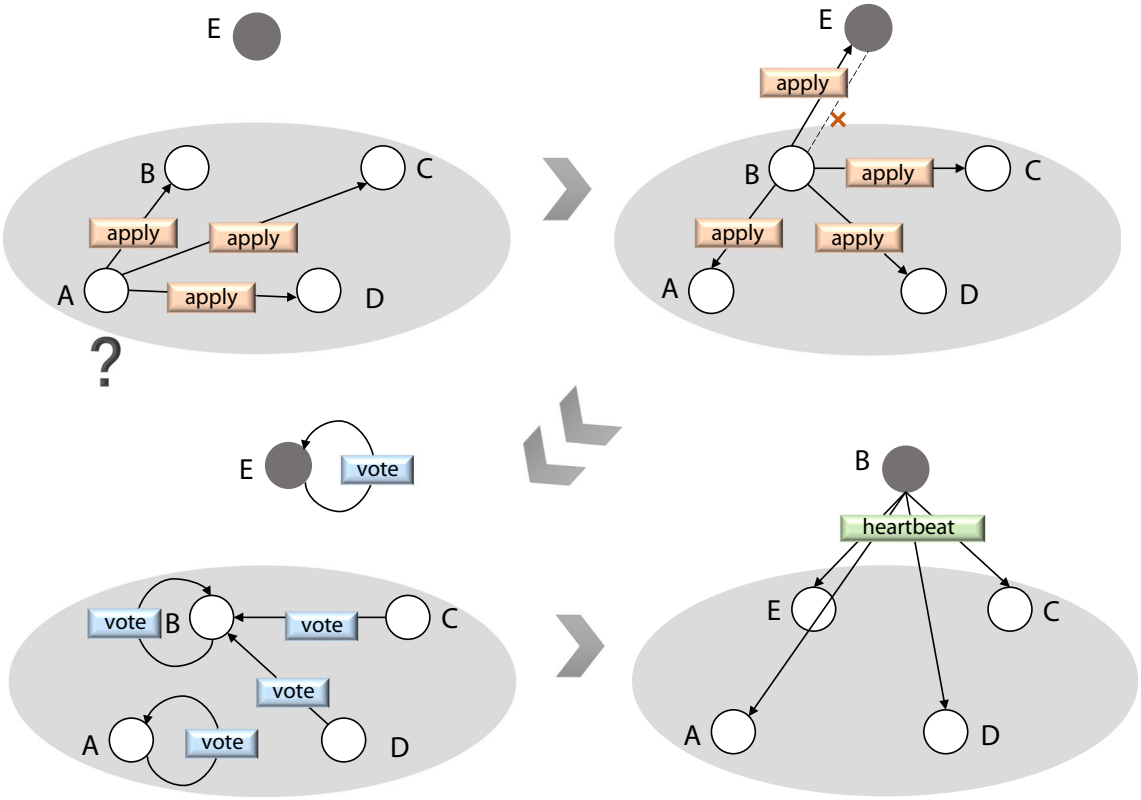

Fig. 5 Leader election process

3.2.2 Impeachment mechanism

The impeachment mechanism is designed in SHBFT. When a node believes that its leader has lost contact or neglected his duty, it can trigger the impeachment mechanism. The impeachment mechanism and the election mechanism can replace the leaders with problems in time to improve fault tolerance and efficiency.

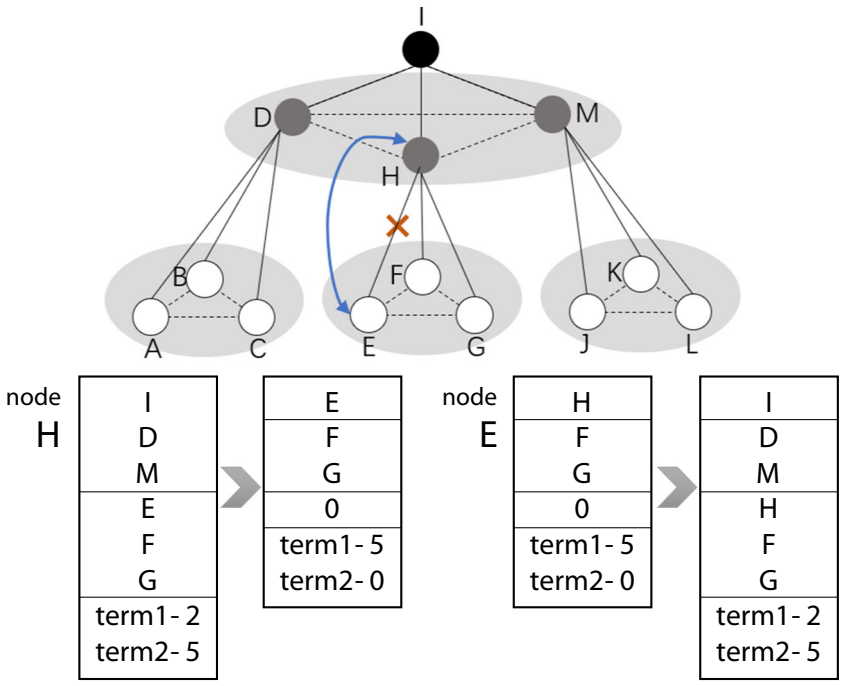

Fig. 6 Replacement between old and new leader

In the secondary and lower net layer submission stage, after receiving the response message, the backups verifies whether there is a problem with this message. If there is a problem, it initiates impeachment and submits an election request, and broadcast their last submitted complete message to other backups. The other child nodes hash this message and compare it with the digest message received last time. If it proves that the message is a malicious request, the impeachment mechanism is ignored. If it is proved that the leader is negligent, a vote is taken to select a new leader.

In the primary net layer consensus stage, when the secondary node discovers that the primary node is not acting or has problems, that is, after the primary node is offline for a period of time or the consensus content is incorrect, the secondary node can trigger the impeachment mechanism. The secondary node that finds the problem sends an election request to impeach the existing leader and fight for a new term leader.

And in SHBFT, in order to prevent nodes from initiating malicious impeachment to submit election requests, each node can only initiate impeachment once every 3 terms. Although the impeached leader only moves down one layer, if it continues to make malicious actions, it will be impeached until the lowest layer, thus ensuring that inefficient and malicious nodes do not affect the overall consensus efficiency.

3.2.3 Node joining

In response to the problem that nodes cannot dynamically join in PBFT, SHBFT has established a node joining mechanism. The decision maker who approves or rejects the node to join is actually the consortium participant to which the node belongs, and the participant will decide whether to approve the new node to join according to their own interests and consortium rules. The node joining mechanism can enable authorized nodes to dynamically join the consensus network without affecting the overall architecture.

The node joining process is shown in Fig. 7. When a new node (the blue node in this figure) wants to join the network, a join request is sent to the current primary node. In order to ensure the consensus efficiency, when the number of nodes in one layer reaches 8, the newly added nodes need to be placed in the next layer. Therefore, the primary node calculates the location of the next node based on the current number of nodes, and forwards a certain secondary node information to the new node, and the new node asks the secondary node where the next layer should be. Repeat this process until it becomes a leaf backup. The leader of this layer needs to give the information of other nodes in this layer to the new node, and the new node adds the node information to its own neighbor table. After the new node joins, it immediately maintains heartbeat with the leader to obtain the current term information, and gets in touch with other nodes in the same layer.

Algorithm 2 Transaction submission verification and impeachment mechanism.

Input: transList[]
Output: bool

```
Package Signature, timeStamp, transList to Trans object t1
Calculate Hash value h1 according to t1
for each ninneighborNode[] do
    remind(n, h1)
end for
submitTrans(curLeader, Trans)
if response() ≠ h1 then
    for each ninneighborNode[] do
        recordW ← recordW + warning(n, h1)
    end for
    if recordW > count(neighborNode)/2 then
        for each ninneighborNode[] do
            recordI ← recordI + impeach(n)
        end for
        if recordI > count(neighborNode)/2 then
            for each ninneighborNode[] do
                newLeader(n)
            end for
        end if
    end if
end if
```

When a new node is allowed to join, if the number of nodes in the layer becomes even, the new node does not have the right to become the leader and vote on the leader until the next node joins.

It should be noted that SHBFT currently does not have a node exit mechanism, and inactive nodes will automatically be impeached to the bottom. This ensures that the network structure is always a complete n-ary tree, and also ensures that active nodes are distributed in the upper layer of the network to obtain better consensus resources.

3.2.4 Network construction

According to the changes in the consensus efficiency of the consensus algorithm, in order to ensure a low consensus delay, each layer has a maximum of 8 nodes. The network construction process is to repeatedly operate the joining mechanism of new nodes, and then start the consensus process.

In the initial network and the new net layer, when there are three or more odd-numbered nodes, the election

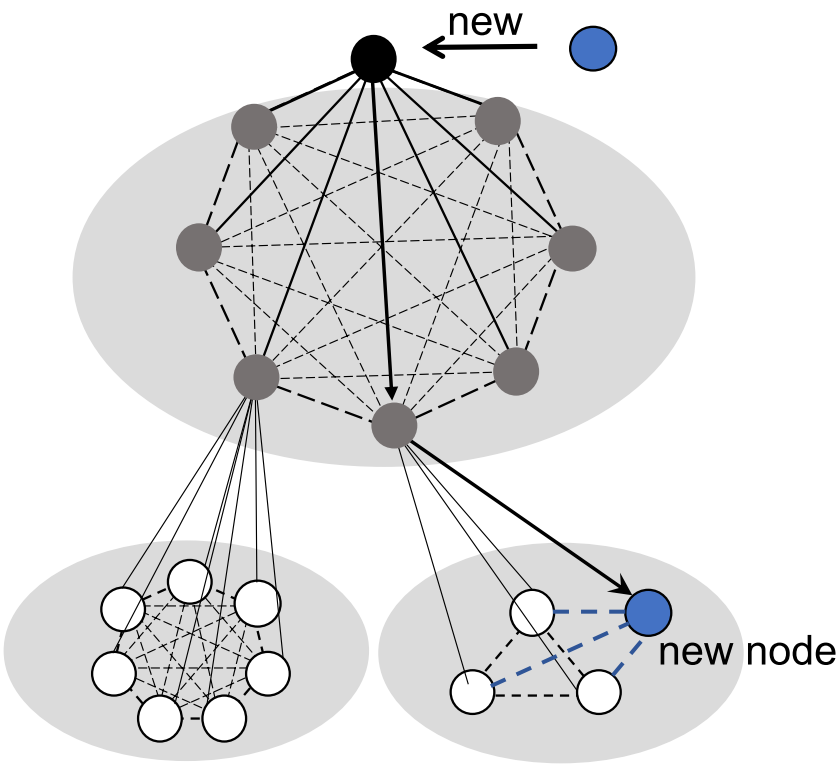

Fig. 7 Node joining process

begins. When a node is successfully elected as a leader, it immediately starts to send heartbeats to each backup to notify them that the leader has been elected, preventing the nodes from initiating a new election to tamper with power.

Adopting this kind of network construction method will also be helpful for the malicious situation of nodes. When a malicious node attempts to forge the upper layer, the newly joined node will be immediately identified.

Assuming that there are currently n nodes in the network, and m nodes in each layer, the calculation formula for the node communication times in the network is

$$Ta = (n-1) \times 2 \tag{1}$$

$$Tb = \left\lfloor \frac{n-1}{m} \right\rfloor \times m \times (m-1) \tag{2}$$

$$Tc = (n-1)\ mod\ m \times [(n-1)\ mod\ m - 1] \tag{3}$$

$$NCT = Ta + Tb + Tc \tag{4}$$

The relationship between the number of layers k and the number of nodes is

$$k = \lfloor \log_m [(m-1 \times n]] \rfloor + 1 \tag{5}$$

3.2.5 Malicious nodes

Even in a consortium chain that can only join with authorization, it is unavoidable that a very small number of nodes have malicious behaviors. The following is an analysis of some situations where some nodes may do evil.

Since the communication between the secondary net layer and the primary node is all carried out by a secondary node, this secondary node has higher power. The primary node has its own digital signature in the "load" message sent, so the secondary node cannot legally modify the data. This ensures that the tertiary nodes can correctly verify this message. However, when the secondary node is malicious, that is, when all the upstream and downstream data are intercepted by the secondary node, the tertiary nodes will not know this situation. Therefore, a timer is set in the node, and the impeachment mechanism against the secondary node will be triggered after the child node has not received a response message after the timeout.

When the old leader is impeached, it is possible for the old leader to create a fake upper layer leader, allowing the new leader to communicate with the fake upper layer leader. But when they are exchanging, the old leader also needs to exchange the neighbor table with the new leader, and the new leader will obtain the information of nodes in the same layer and can communicate with other nodes. The old leader needs to create many fake nodes to deceive the new leader, and the cost of fraud increases exponentially. Moreover, if the new leader forges information during his term, the previous leader will immediately notice it.

The situation of malicious nodes can be better handled in the SHBFT consensus mechanism, and the cost of malicious

nodes is high. The consensus algorithm can maintain good disaster tolerance and stability.

3.3 Simulation experiment

In order to analyze the performance of SHBFT in practical applications, we compare it with the classic algorithm RAFT of the private chain and the consensus algorithm PBFT of the consortium chain in simulation experiments. The simulation experiment mainly conducts performance testing from the aspects of data throughput, consensus latency, node communication times and number of nodes. We developed a prototype system based on the SHBFT consensus algorithm, and used 13 terminal devices to form a wired and wireless hybrid LAN for distributed consensus experiments to verify the feasibility and key performance indicators of the consensus algorithm. The system runs on the following two terminal devices in a distributed manner.

1. Ubuntu 18.04 (64-bit) virtual machine, Intel®Core™ i5-9400F CPU @ 2.90 GHz and 8 GB RAM
2. Android v10.0 mobile device, Qualcomm® Snapdragon™ 865 CPU @ 2.84GHz and 8GB RAM

The experiment successively tested the network construction phase, the election process, the impeachment process, and the consensus process including backup transaction submission, leader consensus sorting response, and verification execution. And set up monitoring points at each network node and network link to test the network performance. In addition, abnormal test points such as offline nodes, error transactions, mutually exclusive resource access, network attacks, etc. are set up to prove the fault tolerance of the consensus algorithm to faulty nodes and malicious nodes.

4 Results

We compared the SHBFT algorithm with the RAFT algorithm and the PBFT algorithm in three directions: data throughput, consensus latency, and number of node communications.

4.1 Data throughput

Data throughput is an important indicator to measure the performance of the consensus algorithm. This is expressed in the blockchain as the number of transactions packaged per unit time, expressed as TPS (transactions per second):

$$TPS = \frac{transactions}{\Delta t} \tag{6}$$

Figure 8 shows the relationship between the time of different algorithms and the number of packaged transactions. The slope of the tangent line at a certain point indicates the data throughput during that time. It can be seen that as time increases, the average data throughput gap of the three algorithms gradually increases.

Figure 9 shows how the data throughput changes with the average transaction commit rate of the node. Obviously, no matter how the request rate of the node changes, the data throughput of PBFT and RAFT is a steadily rising straight line and tends to be stable around 1500TPS. This is because the SHBFT consensus algorithm we proposed will automatically dynamically adjust the consensus window period according to the number of transactions, shorten the window period when the amount of data is large, and reduce the consensus confirmation time.

4.2 Consensus latency

Consensus latency is another important indicator to measure the consensus algorithm. In the blockchain, it represents the time difference between transaction submission and writing.

Figure 10 shows the relationship between the number of devices and the average consensus latency. When the number of nodes is the same, the average delay and performance of the SHBFT algorithm are close to the RAFT algorithm. The same result can be seen in Fig. 11. Among them, SHBFT performs better as the average transaction commit rate of nodes increases. At the same time, it can be found from these two figures that when the number of devices increases and the request rate increases, the average response time delay of these three algorithms will increase, and the influence of these two factors on the algorithm is basically the same.

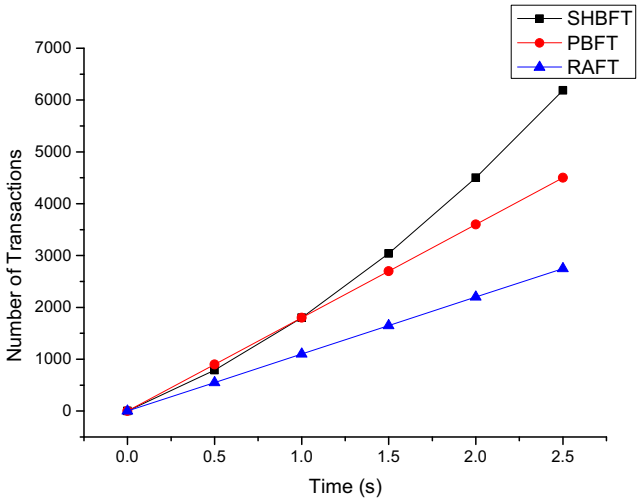

Fig. 8 Comparison of the relationship between time and the number of transactions

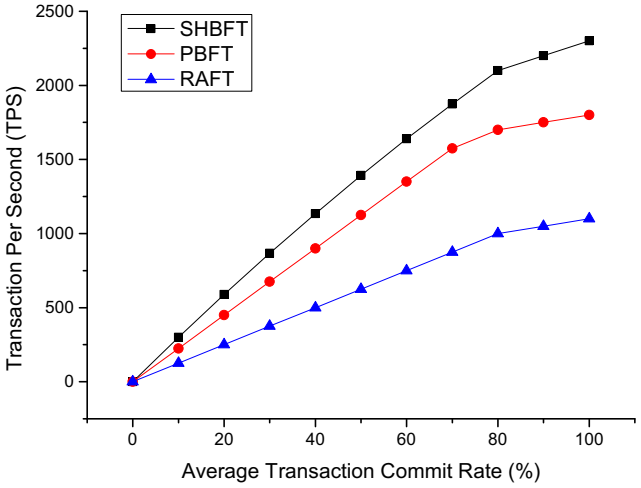

Fig. 9 Comparison of the relationship between average transaction commit rate and data throughput

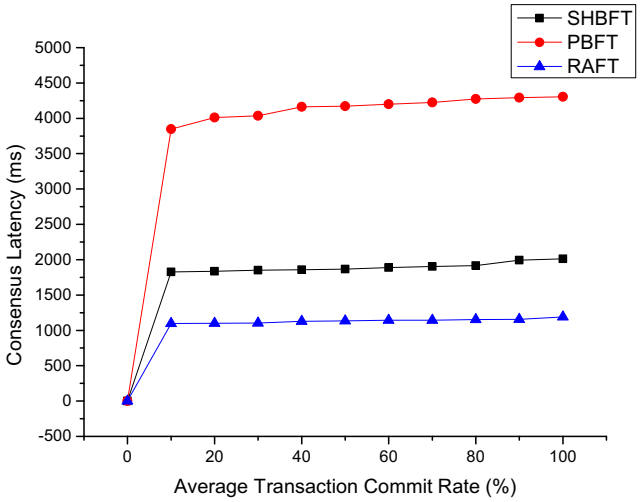

Fig. 11 Comparison of the relationship between average transaction commit rate and consensus latency

4.3 Node communication times

The node communication times represent the traffic generated by nodes in the process of consensus.

Figure 12 shows the relationship between the node communication times and the number of nodes in different algorithms. It can be seen that with the increase of the number of nodes, the communication times of PBFT algorithm increase geometrically, while the curve representing RAFT algorithm is linear. However, SHBFT's traffic growth curve is much smoother, and it rises slowly in a stepwise manner. This is because the hierarchical structure of SHBFT makes most nodes only need to reach a consensus with their own related nodes (the leader of this node, other nodes in the same layer and the backups that belong to it) without establishing communication with all nodes in the network, which greatly reduces the amount of communication between nodes and saves network resources.

Figure 13 is a comparison of the relationship between the number of nodes and the node communication times under different network structures. It can be found that the greater the number of nodes in each subnet, the more node communications are required to reach a consensus, but this does not mean that three nodes in each subnet are the best. Because the smaller the number of subnet nodes, the greater the number of network layers, and the transaction requests of most nodes need to be forwarded a lot, which increases the difficulty of verification, increases the risk, and also causes network congestion.

Figure 14 shows the relationship between the number of nodes and the average waiting time under different consensus window periods. Through the dynamic consensus

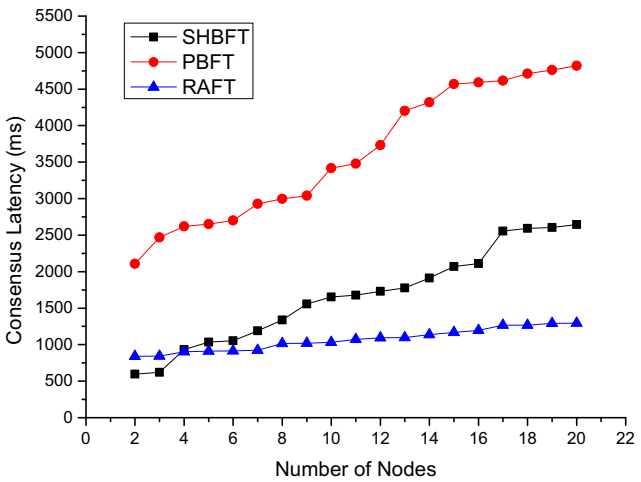

Fig. 10 Comparison of the relationship between the number of devices and the average consensus latency

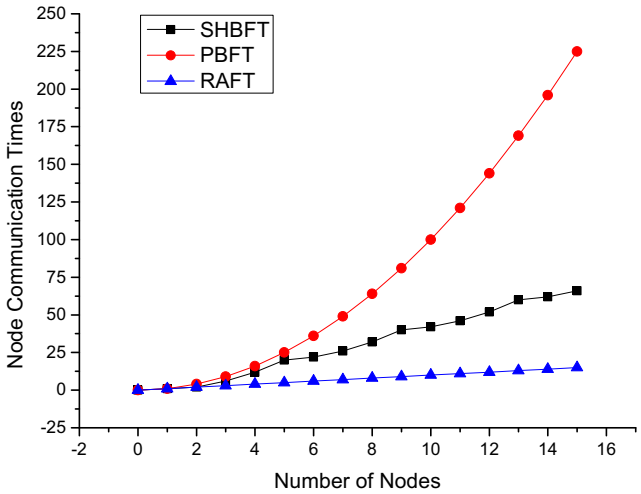

Fig. 12 Comparison of the number of nodes and the node communication times

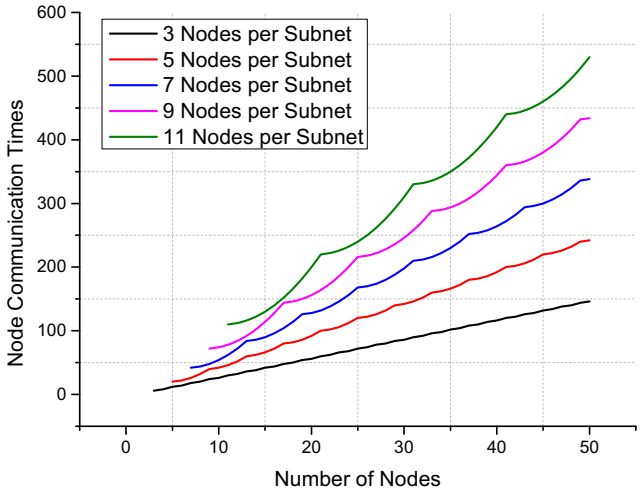

Fig. 13 The relationship between the number of nodes and the node communication times under different network structures

window period adjustment mechanism, the average waiting time of nodes can be ensured to the minimum.

5 Discussion

In order to optimize the management of a large number of nodes and the dynamic joining of nodes in the application scenario of the consortium chain. Based on PBFT, a hierarchical structure solution was proposed, the consensus process was optimized, and the network construction and node joining mechanism were redefined. The SHBFT consensus algorithm is proposed and simulated, and the experiment is compared with PBFT and RAFT. Under different conditions, the consensus latency of SHBFT is lower than that of PBFT and RAFT. SHBFT exhibits extremely high performance in data throughput. As the number of transactions and node request rates increase per unit time, SHBFT's data throughput increases faster than PBFT and RAFT algorithms. Especially when the bottleneck of 1000TPS is reached, SHBFT can continue to rise when PBFT and RAFT are both stable. Due to the advantages of SHBFT's hierarchical structure, nodes do not need to communicate with all other nodes in the network during the consensus process, which greatly reduces the node communication times in the network. The overall simulation result has reached the expected goal of the research. SHBFT realizes the dynamic joining of nodes, and its multi-layer structure ensures that after impeachment and election, active nodes closer to the top can obtain a shorter consensus delay. The hierarchical consensus structure not only achieves a global consensus through local consensus, but also enhances disaster tolerance, that is, some faulty nodes will not affect the communication of most nodes. This mechanism changes the information on the chain from transaction to database access transaction, which has achieved a leap in capacity expansion, thereby breaking the limitation of insufficient storage capacity of blockchain technology and broadening the application scenarios. The results show that the use of SHBFT mechanism can optimize a large number of node management and consensus speed in the consortium chain scene.

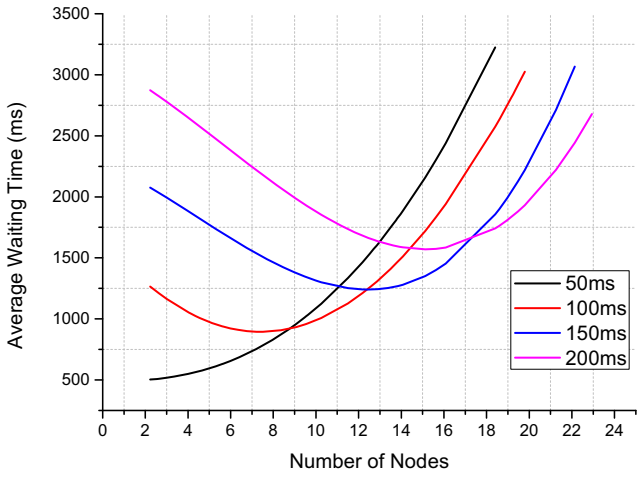

Fig. 14 The relationship between the number of nodes and the average waiting time under different window periods

6 Conclusion

In this paper, we designed a brand-new and scalable hierarchical Byzantine fault tolerant mechanism for the consortium chain of many nodes. SHBFT optimizes many aspects based on PBFT, designs a hierarchical structure to speed up the consensus process, and achieves a global consensus through local consensus. In addition, a leader election and impeachment mechanism has been established in this mechanism to promptly replace inactive and problematic leaders to improve byzantine fault tolerance. We have also implemented a dynamic node joining mechanism in SHBFT, which solves the cumbersome problem of joining PBFT consensus nodes. We also conducted a theoretical analysis of the malicious nodes. Simulation experiments show that our SHBFT has different degrees of improvement in increasing data throughput, reducing consensus latency, and decreasing the node communication times. For future work, we will continue to improve the construction of network node initialization and increase the node exit mechanism to further improve the SHBFT consensus mechanism.

Author Contributions Prof. Zhihan Lv conceived the work and suggested the outline of the paper. Mr.Yuxi Li and Mr. Liang Qiao carried out investigations and wrote the paper.

Funding This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61902203, Key Research and Development Plan - Major Scientific and Technological Innovation Projects of ShanDong Province (2019JZZY020101).

Declarations

Competing interests The authors declare no competing interests.

References

1. Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system
2. Peck ME (2017) Blockchain world-Do you need a blockchain? This chart will tell you if the technology can solve your problem. IEEE Spectrum 54(10):38–60
3. Zheng Z, Xie S, Dai H, Chen X, Wang H (2017) An overview of blockchain technology: Architecture, consensus, and future trends. In: 2017 IEEE international congress on big data (BigData congress). IEEE, pp 557–564
4. Yang S, Deng B, Wang J, Li H, Lu M, Che Y, Wei X, Loparo KA (2019) Scalable digital neuromorphic architecture for large-scale biophysically meaningful neural network with multi-compartment neurons. IEEE Trans Neural Netw Learn Sys 31(1):148–162
5. Yang S, Wang J, Deng B, Liu C, Li H, Fietkiewicz C, Loparo KA (2018) Real-time neuromorphic system for large-scale conductance-based spiking neural networks. IEEE Trans Cybern 49(7):2490–2503
6. Morabito V (2017) Business innovation through blockchain. Springer International Publishing, Cham
7. Buterin V (2014) A next-generation smart contract and decentralized application platform. White Paper 3(37)
8. Crosby M, Pattanayak P, Verma S, Kalyanaraman V (2016) Blockchain technology: Beyond bitcoin. Applied Innovation 2(6-10):71
9. Mohan C (2019) State of public and private blockchains: Myths and reality. In: Proceedings of the 2019 international conference on management of data, pp 404–411
10. Guo Y, Liang C (2016) Blockchain application and outlook in the banking industry. Financial Innovation 2(1):24
11. Gramoli V (2020) From blockchain consensus back to byzantine consensus. Future Gener Comput Syst 107:760–769
12. Xiao Y, Zhang N, Lou W, Hou YT (2020) A survey of distributed consensus protocols for blockchain networks. IEEE Commun Surv Tutor 22(2):1432–1465
13. Pahlajani S, Kshirsagar A, Pachghare V (2019) Survey on private blockchain consensus algorithms. In: 2019 1st International conference on innovations in information and communication technology (ICIICT). IEEE, pp 1–6
14. Jakobsson M, Juels A (1999) Proofs of work and bread pudding protocols. In: Secure information networks. Springer, Boston, pp 258–272
15. King S, Nadal S (2012) Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Self-published paper, August, 19, 1
16. Larimer D (2017) Delegated proof-of-stake consensus. bitshares.org. https://bitshares.org/technology/delegating-proof-of-stake-consensus. Accessed March 28th, 2017
17. Sankar LS, Sindhu M, Sethumadhavan M (2017) Survey of consensus protocols on blockchain applications. In: 2017 4th international conference on advanced computing and communication systems (ICACCS). IEEE, pp 1–5
18. Mingxiao D, Xiaofeng M, Zhe Z, Xiangwei W, Qijun C (2017) A review on consensus algorithm of blockchain. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, pp 2567–2572
19. Huang D, Ma X, Zhang S (2019) Performance analysis of the raft consensus algorithm for private blockchains. IEEE Trans Sys Man Cybern Sys 50(1):172–181
20. Castro M, Liskov B (2002) Practical Byzantine fault tolerance and proactive recovery. ACM Trans Comput Sys (TOCS) 20(4):398–461
21. Khosravi A, Kavian YS (2016) Broadcast gossip ratio consensus: Asynchronous distributed averaging in strongly connected networks. IEEE Trans Signal Process 65(1):119–129
22. Sukhwani H, Martínez JM, Chang X, Trivedi KS, Rindos A (2017) Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric). In: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS). IEEE, pp 253–255
23. Zhang L, Li Q (2018) Research on consensus efficiency based on practical byzantine fault tolerance. In: 2018 10th International conference on modelling, identification and control (ICMIC). IEEE, pp 1–6
24. Wang S (2019) Performance evaluation of hyperledger fabric with malicious behavior. In: International conference on blockchain. Springer, Cham, pp 211–219
25. Wang X, WeiLi J, Chai J (2018) The research on the incentive method of consortium blockchain based on practical byzantine fault tolerant. In: 2018 11th international symposium on computational intelligence and design (ISCID), vol 2. IEEE, pp 154–156
26. He L, Hou Z (2019) An improvement of consensus fault tolerant algorithm applied to alliance chain. In: 2019 IEEE 9th international conference on electronics information and emergency communication (ICEIEC). IEEE, pp 1–4
27. Wang H, Guo K (2019) Byzantine fault tolerant algorithm based on vote. In: 2019 international conference on cyber-enabled distributed computing and knowledge discovery (CyberC). IEEE, pp 190–196
28. Zhu S, Zhang Z, Chen L, Chen H, Wang Y (2020) A PBFT consensus scheme with reputation value voting based on dynamic clustering. In: International conference on security and privacy in digital economy. Springer, Singapore, pp 336–354
29. Miller A, Xia Y, Croman K, Shi E, Song D (2016) The honey badger of BFT protocols. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 31–42
30. Gueta GG, Abraham I, Grossman S, Malkhi D, Pinkas B, Reiter M, Seredinschi D, Tamir O, Tomescu A (2018) Sbft: a scalable decentralized trust infrastructure for blockchains (1804)
31. Li Y, Wang Z, Fan J, Zheng Y, Luo Y, Deng C, Ding J (2019) An extensible consensus algorithm based on PBFT. In: 2019 international conference on cyber-enabled distributed computing and knowledge discovery (CyberC). IEEE, pp 17–23
32. Zhang J, Rong Y, Cao J, Rong C, Bian J, Wu W (2019) DBFT: A byzantine fault tolerant protocol with graceful performance degradation. In: 2019 38th symposium on reliable distributed systems (SRDS). IEEE, pp 123–12309
33. Jalalzai MM, Busch C (2018) Window based BFT blockchain consensus. In: 2018 IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE Cyber, physical and social computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, pp 971–979
34. Gao S, Yu T, Zhu J, Cai W (2019) T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm. China Commun 16(12):111–123
35. Lao L, Dai X, Xiao B, Guo S (2020) G-PBFT: a location-based and scalable consensus protocol for IOT-Blockchain applications.

In: 2020 IEEE International parallel and distributed processing symposium (IPDPS). IEEE, pp 664–673
36. Okusanya O (2019) Consensus in Distributed Systems: RAFT vs CRDTs. https://repository.stcloudstate.edu/csitetds/29

Yuxi Li is a graduate student of Software Engineering in the School of Data Science and Software Engineering, Qingdao University. His research direction is Blockchain. He received a bachelor's degree from Qingdao University in 2020. And he has extensive experience in software development and algorithm design.

Liang Qiao is currently a graduate student in the School of Data Science and Software Engineering at Qingdao University. His research interests include Machine Learning, Blockchain and Virtual Reality. In 2019, he obtained a bachelor's degree from Qingdao University. In 2018, he won the second prize of the National Software and Information Technology Competition in China. In 2020, he won the the third prize of the China Collegiate Computing Contest. He has rich experience in algorithm design.

Zhihan Lv is currently an Associate Professor of Qingdao University, China. He has been an assistant professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences from 2012 to 2016. He received his Ph.D. from Ocean University of China and Paris7 University in 2012. He worked in CNRS (France) as Research Engineer, Umea University (Sweden) as Postdoc Research Fellow, Fundacion FIVAN (Spain) as Experienced Researcher, University College London (UK) as Research Associate, University of Barcelona(Spain) as Postdoctor. He was a Marie Curie Fellow in European Union's Seventh Framework Program LANPERCEPT.