Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 20(6): 10200–10222.
DOI: 10.3934/mbe.2023447
Received: 29 December 2022
Revised: 05 March 2023
Accepted: 10 March 2023
Published: 30 March 2023

Research article

PBFT optimization algorithm based on community contributions

Pengpeng Wang, Xu Wang, Yumin Shen, Jinlong Wang* and Xiaoyun Xiong

School of Information and Control Engineering, Qingdao University of Technology, Qingdao 266520, China

* **Correspondence:** Email: qdwangjinlong@163.com; Tel: +8613605324107.

Abstract: Community governance is the basic unit of social governance, and it is also an important direction for building a social governance pattern of co-construction, co-governance and sharing. Previous studies have solved the problems of data security, information traceability and participant enthusiasm in the process of community digital governance by building a community governance system based on blockchain technology and incentive mechanisms. The application of blockchain technology can solve the problems of low data security, difficulty in sharing and tracing and low enthusiasm on the part of multiple subjects regarding participation in community governance. The process of community governance involves the cooperation of multiple government departments and multiple social subjects. Under the blockchain architecture, the number of alliance chain nodes will reach 1000 with the expansion of community governance. The existing consensus algorithms for coalition chains are difficult to meet the high concurrent processing requirements under such large-scale nodes. An optimization algorithm has improved the consensus performance to a certain extent, but the existing systems still cannot meet the data needs of the community and are not suitable for community governance scenarios. Since the community governance process only involves the participation of relevant departments in users, all nodes in the network are not required to participate in the consensus under the blockchain architecture. Therefore, a practical Byzantine fault tolerance (PBFT) optimization algorithm based on community contribution (CSPBFT) is proposed here. First, consensus nodes are set according to different roles of participants in community activities, and participants are given different consensus permissions. Second, the consensus process is divided into different stages, and the amount of data processed by each consensus step is reduced. Finally, a two-level consensus network is designed to perform different consensus tasks, and reduce unnecessary communication between nodes to reduce the communication complexity of consensus among nodes. Compared with the PBFT algorithm, CSPBFT reduces the communication complexity from O(N2) to O(N2/C3). Finally, the simulation results show that, through rights management, network level setting and consensus phase division, when the number of nodes in the CSPBFT network is 100–400, the consensus throughput can reach 2000 TPS. When the node in the network is 1000, the instantaneous concurrency is guaranteed to be above 1000 TPS, which can meet the concurrent needs of the community governance scenario.

Keywords: community digital governance; blockchain; consensus algorithm; high concurrency; throughput

1. Introduction

With the rapid development of modern information technology and the virtual society, using big data, the Internet and other technologies to build community digital governance models, provide accurate personalized services and reshape the community governance structure has become a mainstream activity [1]. However, there are many problems and challenges in the process of community digital governance, including low data security, difficulty in sharing and tracing and low enthusiasm on the part of multiple subjects regarding participation in community governance [2,3].

The emergence of blockchain technology [4] provides new technical ideas for building digital governance models. Some studies [3,5] have applied blockchain technology to community governance and solved the problems existing in traditional digital governance by utilizing the characteristics of blockchain (e.g., multi-party data storage, data tamper-resistance, data traceability). Applying blockchain to community governance scenarios can enable secure and credible data management, promote data sharing and build an incentive system [6,7] based on blockchain to improve the enthusiasm of multiple subjects to participate in governance. The process of community governance involves the cooperation of multiple government departments and multiple social subjects, which belongs to the alliance chain scenario. With the expansion of the scale of community governance, its number of nodes can reach 1000 nodes. The large-scale node environment puts forward higher requirements for the efficiency of the blockchain consensus algorithm.

Moreover, the performance of the systems built in these studies is greatly limited when they are applied to large user groups involving high population density and wide administrative areas. Owing to the large number of user roles in the community, there are problems of malicious identity nodes and node disconnection. When applying blockchain technology to community digital governance, it is necessary to use the Byzantine fault-tolerant consensus algorithm. The performance of existing Byzantine fault-tolerant consensus algorithms, such as the practical Byzantine fault tolerance (PBFT) algorithm, can no longer meet the needs of community digital governance scenarios. When the number of nodes exceeds 100, the performance of this algorithm drops rapidly.

In order to improve the performance of the PBFT algorithm, researchers have proposed various schemes, including WBFT [8] and mPBFT [9], and optimized them in different ways relating to the complexity of network communication, token settings and consensus message content. When the number of nodes exceeds 100, existing methods can effectively suppress the problems of reduced blockchain throughput and rapid increase in consensus time delay. However, in the case of community scenarios with more than 50,000 inhabitants, the concurrent demand of blockchain will exceed 1000 TPS Although the existing research has optimized the coalition chain consensus algorithm, it still cannot meet the high concurrency processing requirements of large-scale nodes in the community governance scenario. In addition, the community governance process only involves the participation of relevant departments and users, and it does not require all nodes in the network to participate in the consensus under the blockchain architecture. In response to the above problems, this paper proposes a PBFT optimization algorithm based on community contribution. Specifically, this work makes the following two contributions:

1) According to the difference between community user identity and work contribution, and by setting different consensus permissions for user nodes, the consensus work is divided into three stages, where each consensus stage only agrees on relevant data in that stage, thereby reducing the amount of consensus data and communication pressure.

2) A two-level consensus network is designed. This reduces unnecessary communication between nodes in different stages to reduce communication complexity, so as to optimize the throughput and consensus delay of the blockchain and provide a higher concurrency and response speed for community digital governance.

The rest of this paper is organized as follows. Section 2 summarizes existing related work; Section 3 elaborates on the scheme of this paper; and Section 4 verifies and evaluates the performance of the consensus algorithm designed in this paper, as well as analyzes the security of the scheme. Finally, Section 5 concludes the paper.

2. Related work

2.1. Blockchain community governance

With the advancement of economic development and urbanization, the scale of communities continues to expand, leading to a rapid increase in livelihood problems. A more efficient community governance model is needed to improve the work efficiency of the government and communities.

A digital platform in this context is a community digital governance model built with the help of big data, the Internet and other information technologies. It enables community governance affairs to be networked, thereby facilitating coordinated mobilization among and within the government, society and citizens, as well as improving the effective utilization of government and social resources [10]. Community digital governance has the benefits of quickly solving governance problems, improving governance efficiency, optimizing governance networks, etc., but it still faces many challenges in practical application [2], such as low data security, difficulty in sharing and tracing and low enthusiasm on the part of multiple subjects regarding participation in community governance.

The concept of the blockchain was first proposed by Satoshi Nakamoto [4]. It is a trustless and decentralized distributed ledger technology jointly maintained by multiple nodes. Once the data have been verified by the blockchain nodes and uploaded to the block after the chain, they are permanently stored and cannot be subjected to tampering. Owing to the characteristics of multi-party data storage, the difficulties of data tampering and data traceability, the applicability of blockchain technology in community governance has been studied from various different perspectives. Garcia-Garcia et al. [11] introduced five scenarios involving the use of blockchain technology in collaborative processing with community issues and made comparative references to the dimensions of the blockchain platform architecture, modeling language, intelligence and execution engine. Han [3] analyzed the application prospects of blockchain in community governance and designed a community governance mechanism based on blockchain, which could help to promote scientific community governance decision-making, service precision and management refinement. Elisa et al. [5] proposed an e-government system framework based on blockchain technology, which could ensure data security in the process of dealing with community issues and improve the privacy of data and the credibility of governmental departments.

Previous studies have integrated blockchain technology into community digital governance platforms, which effectively solved the problems of low data security, difficulty in sharing and tracing and low enthusiasm on the part of multiple subjects regarding participation in community governance faced by traditional community governance. However, in the case of a large administrative area and a large number of people, the performance limitations of the blockchain will not be able to meet the operational demands of community digital governance services. Therefore, the data management efficiency of the blockchain needs to be optimized.

2.2. Consensus algorithm

The Asa blockchain consists of many nodes, and malicious attacks are possible, which can lead to Byzantine problems. According to the different settings of node identities in the blockchain network, consensus algorithms can be divided into Byzantine algorithms [12] and non-Byzantine algorithms [13]. Consensus errors in non-Byzantine algorithms are generally related to system failures that occur in distributed systems, such as machine downtime and node reporting errors, but there are no malicious nodes in the system that interfere with the distributed system. When malicious nodes in the system carry out malicious activities such as tampering with data, non-Byzantine fault-tolerant algorithms cannot guarantee data security and system stability. Existing non-Byzantine algorithms, such as Paxos, VR and Raft, are difficult to apply to open networks with many nodes and have low reliability.

Byzantine fault-tolerant consensus algorithms can solve various limitations of non-Byzantine consensus algorithms, including their inability to deal with malicious nodes and the difficulty of applying them to the open Internet environment. To this end, researchers have designed a series of Byzantine fault-tolerant consensus algorithms that can solve any type of error in distributed systems to a certain extent and ensure the security and stability of the distributed system [14,15]. Byzantine fault-tolerant consensus algorithms can deal with the existence of malicious nodes in the environment and ensure the security and activity of the system. However, the complex consensus verification logic leads to a high network communication cost of existing Byzantine consensus algorithms. For example, when the PBFT consensus algorithm is applied to community governance, owing to the large number of nodes in the blockchain network, it cannot achieve efficient transactions.

To solve the problem of low consensus efficiency, it is necessary to achieve consensus and improve transaction efficiency through reasonable consensus master node selection methods and optimized consensus methods. WBFT [8] dynamically weights the nodes participating in the consensus, distinguishing malicious nodes and reducing the impact of malicious nodes, but it fails to solve the problem of consensus data redundancy and cannot effectively improve consensus efficiency. mPBFT [9] adjusts the probability of nodes participating in block production according to the reliability of nodes and reduces network operating costs, but it lacks a method for evaluating node reliability and cannot be applied in real scenarios. SG-PBFT [16] uses a fractional grouping mechanism in an Internet of Vehicles scenario, which reduces the communication complexity between nodes and optimizes consensus efficiency. The above consensus algorithms assign different weights to nodes in the network based on equity, reliability and ratings as reference certificates, etc., and differentiate each node by weight when selecting a consensus master node. However, the reference certificates set in the network do not have reasonable distribution rules, so effective incentives cannot be provided based on community user contributions. DPN-PBFT [17], SHBFT [18] and other consensus algorithms reconstruct the blockchain network communication method and divide the nodes in the network into node organizations according to factors such as geographical scope, network access time or user identity, reducing the number of consensus communications between nodes and improving the overall consensus efficiency. DPN-PBFT and SHBFT grant the same authority to nodes in the network, without distinguishing node roles, and are not suitable for community digital governance scenarios where user permissions are differentiated. tPBFT [19] compresses the consensus message by simplifying the content of information sent between nodes in each stage of the consensus process, thereby reducing the consensus communication time and the delay and success of block generation by nodes. Rate and other factors measure the credibility of a node in the network and serve as a reference for selecting the master node for subsequent block production work. However, the tPBFT algorithm lacks the step of ordinary nodes participating in consensus certification and does not fully realize decentralized management.

The existing Byzantine consensus optimization algorithms have improved on the performance of the PBFT algorithm to a certain extent, but they have no clear means of dividing the identity and role of the nodes in the network. When they are applied to a community governance scenario, they cannot provide sufficient throughput, resulting in a block. There are also problems such as poor concurrency and long delay time when accessing services on the chain.

3. CSPBFT consensus algorithm

In order to meet the high concurrent access requirements of community digital governance scenarios and solve the shortcomings of existing Byzantine optimization algorithms, we used the work contribution of users in community governance as a carrier to design a PBFT optimization algorithm based on community contributions (PBFT Optimization Algorithm Based on Community Contribution, referred to as CSPBFT). Since the community governance process only involves relevant departments and users, the relevant data do not require all nodes in the network to participate in the consensus. The CSPBFT algorithm organizes and divides the blockchain network and divides the consensus work into stages, where each stage runs periodically. This reduces the complexity of communication between nodes during consensus work. The verification department in community governance forms the monitoring node, and the departments responsible for the organization of each community cluster form the initial subnet master node. The initial subnet master node invites the work department to join the regional sub-organization as an ordinary node. When a node joins the network, the subnet master node conducts identity verification. The node settings and division in the network are shown in Figure 1.

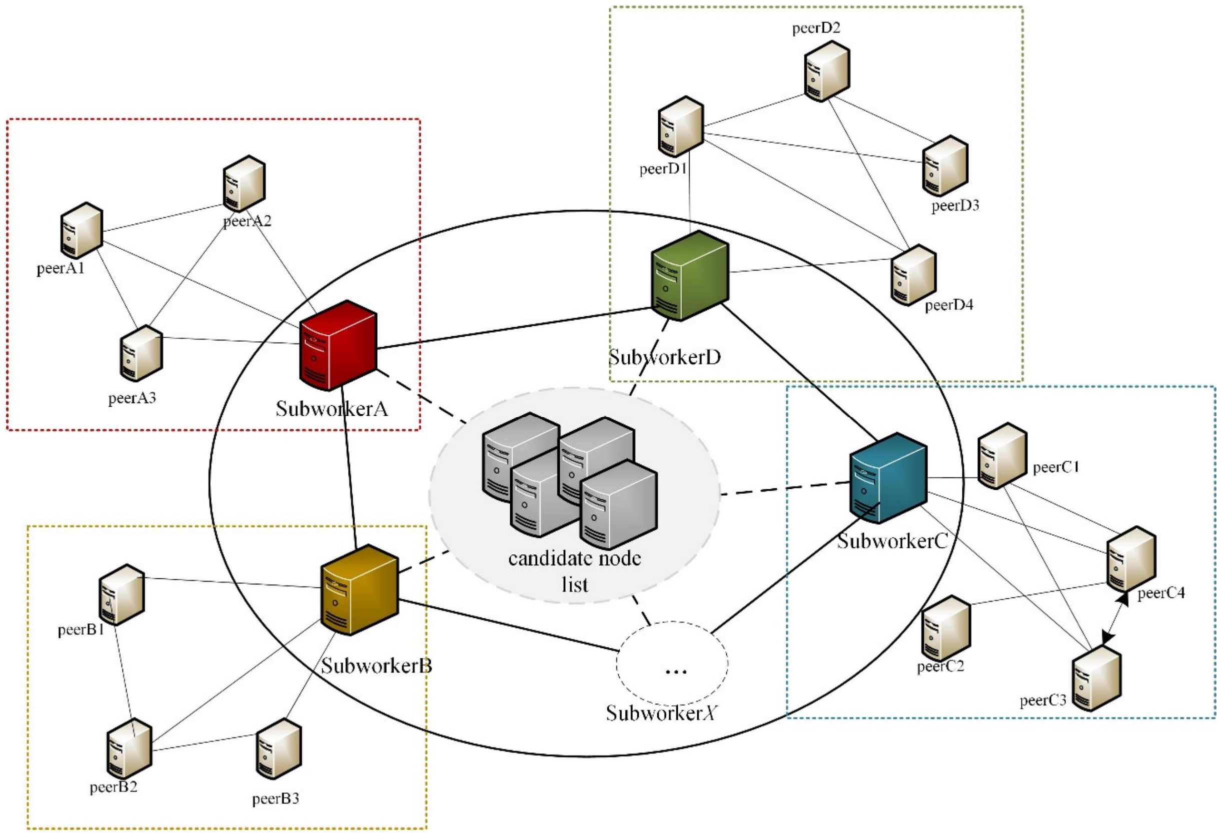

Figure 1. Consensus network design.

As shown in Figure 1, the consensus network consists of two levels. SubworkerA-X forms the consensus main network, and the consensus subnetwork is composed of subworker nodes and ordinary nodes: for example, SubworkerA and peerA1 to peerA3 form a subnetwork. The nodes in the subnetwork participate in the first stage of CSPBFT, and the subworker nodes in the subnetwork are elected. Subworker nodes in each sub-network participate in the second and third stages of CSPBFT for data consensus.

3.1. Consensus global settings

In the online community governance scenario, the total number of nodes in the network is *N*, and the node types can be divided into the following five categories according to the functions of users participating in the consensus work: 1) consensus master node (primary node); 2) subnet master nodes (subworker node); 3) subnet common node (peer node); 4) monitoring node (check node); 5) candidate node (candidate node). CSPBFT sets up corresponding node accounts for each community governance user. Each user's account has three asset certificates: credit certificate (CTN), consensus asset (CAS) and temporary consensus asset (CAS_T). Users can convert part of the CTN into a temporary consensus asset CAS_T as part of the consensus contributions used. In CSPBFT, the consensus network consists of a two-layer network. The main network is composed of subnet master nodes responsible for generating blocks, and the subnet master node list is composed of C subworker nodes. Each subworker node is elected by peer nodes in the subnetwork. The CSPBFT network consensus communication process is shown in Figure 2.

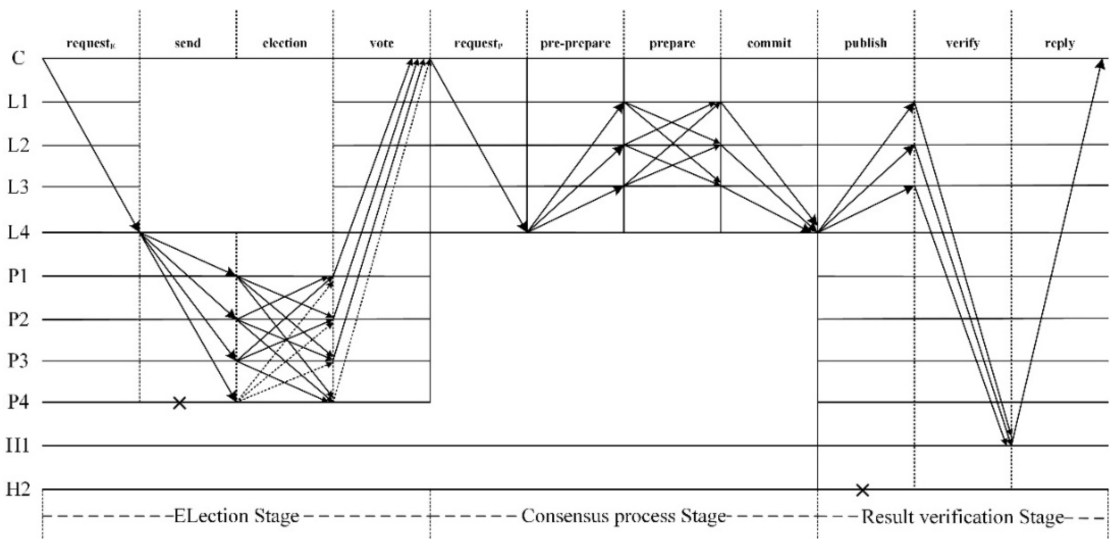

Figure 2. CSPBFT consensus communication process.

The consensus work of various nodes in the network is as follows:

1) Consensus master node: belongs to the subnet master node and is responsible for the current round of consensus block production.

2) Subnet master nodes (L1–L4): the nodes in the subnetwork that participate in the information consensus work are elected and rotated by the community manager of each area in the community. During the term of office, they will perform identity verification and network access for new access nodes in the organization to which they belong.

3) Subnet common node (P1–P4): composed of ordinary users in the community and community manager, they communicate with the subnet master node and participate in the election and voting for the new subnet master node after the end of the term of the subnet master node (election to become the subnet master node).

4) Monitoring node (H1 and H2): responsible for various regulatory departments of the government; the final determination is made during the consensus process, and the consensus results of the consensus master nodes are verified.

5) Candidate node (L4): serves as the rotation community manager for the next term. Ordinary nodes in the subnetwork become master nodes of the subnetwork through the voting mechanism. During the voting process, peer nodes temporarily become candidate nodes until the voting is completed.

3.2. Consensus node selection

In the election stage of the CSPBFT consensus principle, the candidate nodes in the subnetwork are selected according to consensus by voting. The verification work in the consensus process and the result verification stage are carried out by the consensus master node and the monitoring node responsible for block generation. This section explains the methods used to select candidate nodes, consensus master nodes and monitoring nodes.

3.2.1. Candidate node selection

When the CSPBFT network is constructed, subworker nodes are selected from each subnetwork in the secondary network to form the initial subnetwork master node list. In the subsequent consensus process, peer nodes in the subnetwork vote to update the subworker nodes in their respective subnetworks. Regarding this article's settings, the subworker node in the subnetwork performs one block production or a term is re-elected. The subworker node voting in the subnetwork is divided into three stages: 1) becoming a candidate node; 2) voting by peer nodes in the subnetwork; 3) candidate nodes becoming subworker nodes. The client selects candidate nodes following the PBFT selection rules; the calculation method is shown in Eq (1):

$$l = kmodT \tag{1}$$

where k is the current term number of the subnetwork. When the tenure is updated, the lth community manager node in the subnetwork is converted from a normal node to a candidate node, and the client sends election information to the candidate node. If the current candidate node fails and the client response times out, the term number k will be automatically incremented by 1, the next term will be entered and the election work will be sent to the new candidate node again.

3.2.2. Consensus and verification node selection

In the consensus process, this method divides nodes into consensus nodes and monitoring nodes according to the business scenarios of community governance. Nodes follow the PBFT election rules for selection rules.

The consensus node selection process is as follows. The subworker node list and the block node list in the CSPBFT network are set to record the subworker nodes of C organizations in the network and the block nodes of the last 2C/3 blocks. The calculation method used for the client to select the current block consensus master node is shown in Eq (2):

$$l = vmodC \tag{2}$$

where v is the current consensus round, and the consensus master node of this round is the lth node in the subworker node list. When selecting the lth node as the consensus master node, it is necessary to verify the list of block-producing nodes. The same consensus master node is allowed to continuously perform the blocking of two blocks. In order to avoid the problem that there are malicious nodes in the subworker node and a large number of illegal blocks being continuously generated, it is necessary to ensure that the number of blocks produced by a consensus master node in the last 2C/3 blocks does not exceed two.

The monitoring node selection process is as follows. When selecting a monitoring node for secondary verification of the consensus result, the selection principle is the same as that for the consensus master node. The CSPBFT network sets the verification master node list and the verification node list to respectively record the S verification master nodes in the network and the verification master nodes of the last 2S/3 blocks. The calculation method used by the client to select a monitoring node is shown in Eq (3):

$$l = vmodS \tag{3}$$

The monitoring node for this round of verification is the *l*th node in the list. According to the CSPBFT settings, when selecting the *l*th node as the verification master node, it is necessary to ensure that the number of verifications performed by a verification master node in the latest 2S/3 blocks does not exceed three.

3.3. CSPBFT consensus principle

In order to improve the efficiency of consensus among nodes and ensure the accuracy of consensus, while resisting the attack of malicious nodes and speeding up the process of reaching consensus, the CSPBFT presented here is designed as a consensus algorithm consisting of three stages: election stage, consensus process and result verification. This section explains the functions and principles of the three stages.

3.3.1. Election stage

The first stage of the CSPBFT consensus algorithm is the election stage, which is used to elect the subnet master nodes in each subnetwork. The election phase consists of four steps: *requestE*, *send*, *election* and *vote*. The election process of the subnet master node is shown in Figure 3.

requestE: the client sends an election message to the candidate node in the election subnet, and the candidate node processes the received client requestE message and performs the next operation.

send: in an election cycle, the peer node is converted to a candidate node and sends election notifications to other peer nodes in the subnetwork.

election: after the peer node receives the election message from the candidate node, it verifies the content of the message, votes for the candidate node after the message is verified and sends a verification message to other nodes in the subnetwork (including the candidate node).

vote: after each node in the subnetwork receives the verify message, it counts the weights of the received nodes whose verification is true or false and sends the majority selection result to the client.

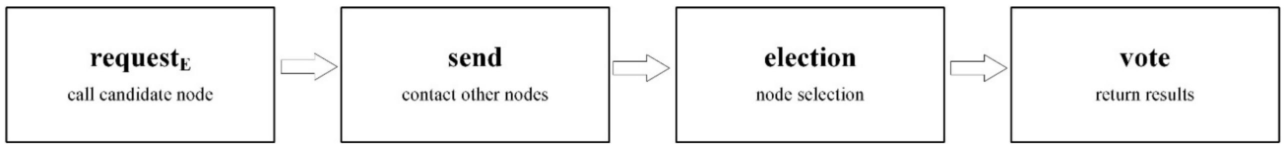

Figure 3. Election process.

After the execution of the four steps in the election phase (the pseudo-code is shown in Algorithm 1), the elected subnet master node participates in the information consensus work during this term.

Algorithm 1: Election Phase Start

```
Step1: Request_E
        r := new(Requeste)// Create a RequestE object for parsing messages from clients
        tenureIDAdd()//， enter a new term
        signInfo := RsaSignBySha256(r , node.rsaPrivKey)// Encrypt message content
        s := Send(*r , d , tenureID , signInfo)// Build the Send object
        broadcast(kSend , s , P)// Send election information to P (candidate node), Communication
        times:
Step2: Send
        s := new(Send) // Create a Send object for parsing the election information from the
client, Communication times:1
        thisresult := RsaVerifySignBySha256(s , { d , tenureID , getPubKey(nodeId) , …})//
Verify the message body, message digest, term number, signature ciphertext and other
information in the received message
        if thisresult is true:
        sign := RsaSignBySha256(r , node.rsaPrivKey)// sign the message
        e := Election(d , tenureID, nodeId , t , sign)// Build the Election
        object
        broadcast(kElection , e , signlistsub)// Publish voting information to other nodes in the
subnetwork to which it belongs, Communication times:
Step3: Election
        e := new(Election)// Build an Election object for parsing voting messages from candidate
        nodes
        result := RsaVerifySignBySha256(e , { d , tenureID , getPubKey(nodeId) , …})// Verify the
message body, message digest and other information in the received voting message
for electionget(){}
        thisresult := Complete(len(nodelist)/2 , sumele)// Verify whether the user votes in the
 subnet are more than half
       if result && thisresult is
        true
        v := Vote(d , tenureID, nodeId , sign , result , …)// Build the Vote object
        broadcast(kVote , e , P)// Send election results to candidate nodes, Communication times:
        Step4: Vote
        v := new(Vote)// get election news
broadcast(krequest ,    thisresult  , clientAdder)// Return  the election  result  to  the client,
Communication times: end
```

In the election phase, after the client receives a voting message from each node in the subnetwork, it counts the weight of the node whose message is selected as true or false. If Rtrue (pass votes) > Rfalse (negative votes), then it sends the master node's election success message and the transaction content that needs consensus to the candidate node, as well as sends the subworker node replacement message to the peer node in the subnetwork.

3.3.2. Consensus process

After the election of the subnet master nodes of each subnetwork in the CSPBFT consensus network has been completed, the nodes participate in the information consensus work in the current term. The consensus process consists of four steps: requestP, pre-prepare, prepare and commit. The process of information consensus is shown in Figure 4.

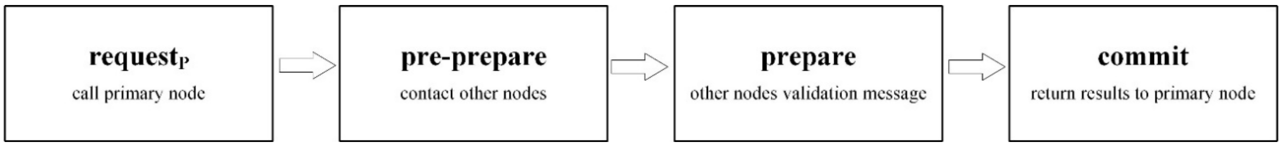

Figure 4. Information consensus process.

request: the client sends the consensus information to the master node of this round of consensus, and the consensus master node carries out follow-up work in the consensus network after processing the information.

pre-prepare: after the consensus master node receives the information from the client, it preprocesses the message, simplifies the message and publishes it to other subnet master nodes.

prepare: after each subnet master node in the network receives the *pre-prepare* message sent by the consensus master node, it analyzes and checks the credibility of the message and publishes the check results to other subnet master nodes.

commit: after the master node of the subnet receives the *prepare* message indicating that more than half of the master nodes of the subnet have been successfully verified, it sends a commit message confirming successful verification to the consensus master node.

After the consensus phase of the CSPBFT algorithm (its pseudo-code is shown in Algorithm 2), the consensus master node analyzes the commit information and enters the result verification phase.

Algorithm 2: Consensus Process Start

```
Step1: Requestp
    r := new(Requestp)// Create a RequestP message for parsing consensus messages
    sequenceIDAdd()//,Enter a new round of consensus work
    signInfo := RsaSignBySha256(r , node.rsaPrivKey)// Encrypt the consensus message Requestp
    pp := PrePrapre(*r , d , sequenceID , signInfo)// Create PrePrepare for sending messages
    broadcast(kPrePrepare , s , L)// Send a consensus message to L (consensus master node address), Communication times:
Step2: Pre-Prepare
    pp := new(PrePrepare)// Create a PrePrepare object for receiving consensus messages
    thisresult := RsaVerifySignBySha256(pp , { d , sequenceID , getPubKey(nodeId) , …})// Verify the message body, consensus round, encrypted ciphertext and other information data in the consensus message
    if thisresult is true
    sign := RsaSignBySha256(r , node.rsaPrivKey)// Encrypt the verification result of this step
    pre := Prepare(d , sequenceID, nodeId , t , sign)//Create Prepare for sending messages
    broadcast(kPrepare , pre , pamlistsub)// Send consensus messages to other subnet master nodes, Communication times:
Step3: Prepare
    pre := new(Prepare)// Create a Prepare object for parsing messages from the consensus master node
    c := Commit(d , sequenceID , sign , …)// Build the Commit message body after performing the same consensus message verification in Step2
    broadcast(kCommit , c , pamlistsub)// Send the verification result of this step to other subnet master nodes in the network, Communication times:
Step4: Commit
    c := new(Commit)// Build a Commit object to receive messages from each subnet master node
    for preget(){}// Count the number of messages received and passed the verification
    thisresult := Complete(len(pamlist)*2/3 , sumpre) && Check(nodeinplist) < 2// Execute consensus master node identity verification
    if thisresult is true
    next->broadcast:publish// Proceed to the next stage of broadcasting: result verification, Communication times:
end
```

3.3.3. Result verification

After the subnetwork master nodes in the CSPBFT consensus network have completed the consensus process on the message, in order to prevent the consensus master node from performing malicious activities when publishing blocks, the monitoring nodes in the network will perform verification. The verification steps are shown in Figure 5.

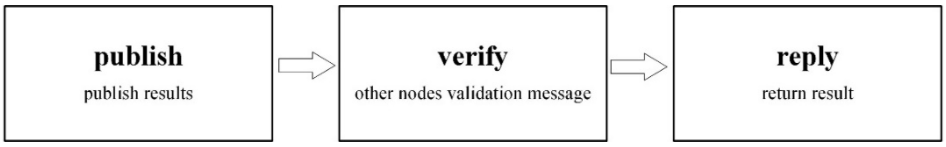

Figure 5. Verification process.

The result verification phase of CSPBFT consists of three steps: *publish*, *verify* and *reply*. The consensus master node sends the final consensus result publish message to the subnet master nodes of each subnetwork, and the subnet master node analyzes the results and sends the verify message of the analysis results to the current round of monitoring nodes. The monitoring node makes statistics on the analysis results of the master nodes of each subnet and returns the final result to the client.

4. Experimental analysis

This section presents the analysis and testing of the consensus characteristics and performance of the CSPBFT algorithm proposed in this paper, as well as a comparison with the existing Byzantine algorithm. First, in Section 4.1, the communication cost of the CSPBFT algorithm is analyzed and compared and the security of the CSPBFT algorithm is analyzed and explained; second, in Section 4.2, the time delay and throughput of the CSPBFT algorithm are tested; finally, in Section 4.3, the performance test results are compared with those obtained for the existing Byzantine algorithm.

4.1. CSPBFT characteristic analysis

This section divides the network communication cost according to the consensus principle of CSPBFT and compares it with the communication cost of the existing Byzantine consensus algorithm. The possibility of malicious node attacks in each phase of the CSPBFT consensus algorithm and corresponding solutions are also analyzed.

4.1.1. Internet communication costs

Taking the total number of nodes in the network to be N, the results of the comparisons of CSPBFT with the existing optimization algorithm in terms of communication complexity, token type, storage cost and fault tolerance are shown in Table 1.

The broadcast stage in the CSPBFT consensus process needs to update the subworker node at the end of the term. The maximum numbers of network communications in each step when the subnetwork votes for the subworker node and performs consensus work, as well as the consensus verification, are determined as follows.

1) Subnet master node generation phase: *request*$_E$: 1; *send*: $\mathbf{N}/\mathbf{C}$; *election*:$(\mathbf{N}/\mathbf{C})^2$; *vote*: $\mathbf{N}/\mathbf{C}$. $\mathbf{C}$ is the number of subnetworks set in the network.

2) Consensus stage: *request*$_P$:1; *pre-prepare*: $\mathbf{C}$-1; *prepare*: $(\mathbf{C}\text{-}1)^2$; *commit*: $\mathbf{C}$-1.

3) Verification phase: *publish*: $\mathbf{C}$-1; *verify*: $\mathbf{C}$-1; *reply*: 1. The round term of each subworker participating in the consensus work is $\mathbf{C}$ times.

In the voting phase, the subworker node has a tenure of **C**. After participating in **C** consensus tasks, it enters the next term's election. The communication cost for CSPBFT to complete a consensus work is $\mathbf{N}^2/\mathbf{C}^3+\mathbf{N}/\mathbf{C}^2+\mathbf{C}^2+3\mathbf{C}$, and the communication complexity is $\mathbf{O}(\mathbf{N}^2/\mathbf{C}^3)$.

According to the comparison between the scheme proposed here and the existing Byzantine consensus, the communication cost of the CSPBFT consensus process is low. This reduces the pressure on the local data storage of consensus nodes and confers strong fault tolerance, which helps to improve the throughput of blockchain network transactions and reduce consensus time delay.

Table 1. Comparison of communication consumption with existing Byzantine consensus algorithms.

Consensus	Communication Complexity	Token Type	Storage Cost	Fault-Tolerant
CSPBFT	$O(N^2/C^3)$	CTN、CS、CAS	Low	49% N
PBFT	$O(N^2)$	Null	High	f (f = N/3-1)
WBFT	$O(N^2)$	Weight	High	1/3N
mPBFT	$O(N^2/3)$ or $O(2N/3)$	Reliability	Middle	1/3 N or 16.7% N
SG-PBFT	$O(N^2/4)$	Score	High	f (f = N/3-1)
DPN-PBFT	$O(N)$	Null	Middle	49% N
SHBFT	$O(N^2/M)$	Null	High	f (f = N/3-1)
tPBFT	$O(N^2)$	Null	Middle	f (f = N/3-1)

4.1.2. Security analysis

There may be malicious or faulty nodes in the CSPBFT consensus network, which will lead to malicious attacks or consensus failures during consensus work. The analysis and processing methods for addressing malicious attacks and consensus failures during the consensus process are as follows.

1) Subnet master generation attack/failure:

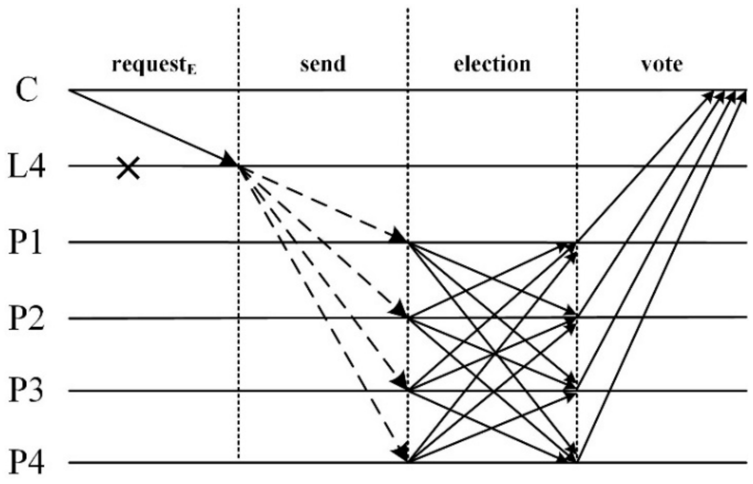

Figure 6. Candidate nodes are malicious/ offline.

Figure 6 shows the situation where the client sends a tenure replacement message to the subworker node in the organization. When the candidate node receiving the message is a malicious node or is in a disconnected fault state, the peer nodes in the subnetwork for the tenure update cannot receive the correct election message. When the candidate node is not a malicious node, it is in the network; when the candidate node is a faulty node, the master node update process will always be in a waiting state. If the client fails to receive the peer node feedback message in the subnet after the waiting timeout, it will resend the term replacement message to the primary node in the next term.

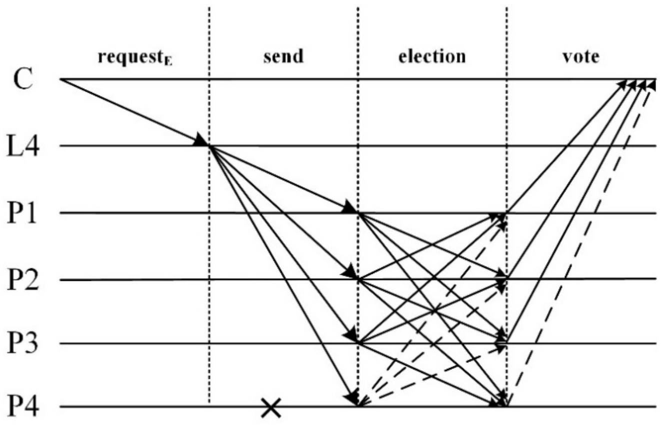

Figure 7. Peer nodes are malicious/ offline.

As shown in Figure 7, after the candidate node sends an election message to the peer nodes in the subnetwork, some peer nodes in the network are faulty nodes or malicious nodes. During the voting phase, some peer nodes have network failure problems; also, the candidate nodes will maintain the network connection to send messages and vote after the peer nodes reconnect to the network to receive messages. During voting selection, there may be malicious nodes in the subnetwork in addition to the faulty node. For example, after receiving the election message, the P4 node in Figure 7 verifies that the content of the message is true and valid, but it publishes the malicious message that the election is false in the network. After the renewal of the tenure has been completed, malicious nodes will be punished by deducting CAS to reduce their ability to carry out malicious activities in subsequent network activities. In this stage, the malicious fault tolerance is 49% and a node needs to control more than half of the node weights in the subnetwork to succeed in malicious attacks.

2) Block consensus phase:

Figure 8 shows a case where the master node responsible for this block production has a network failure after the term update has been successfully completed and cannot send consensus messages to other subworker nodes. The consensus will be in the waiting process. After the client waits for a timeout, it will resend the *request$_P$* message to the subworker node of the next subnetwork and notify the subworker that the offline subnetwork will renew its term in advance.

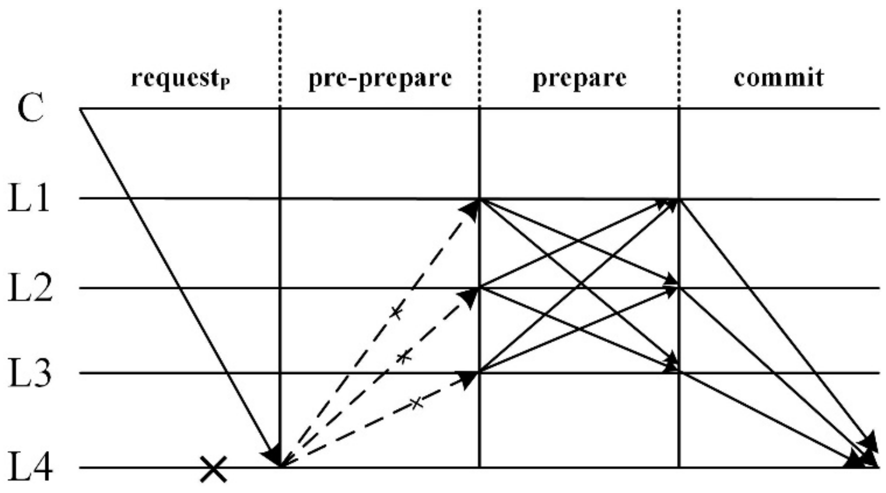

Figure 8. Master node offline.

As shown in Figure 9, when the subworker verifies the message of the master node that produced the block, it needs to verify the content of the block and check the node that produced the block. Other nodes in the network send and receive messages to and from each other, and the allowable node failure rate in the *prepare* phase is 49%. In order to avoid consensus network failure caused by too few subworker nodes in the consensus network and partial failures that cannot continue to generate blocks, the nodes of the same subnetwork are allowed to temporarily continue to generate blocks during the *prepare* phase verification. However, in 2**S**/3 consecutive blocks, the number of blocks produced by a subworker node cannot exceed two; otherwise, the subnetwork responsible for producing blocks needs to be updated in advance, and the new subworker node will perform the task of producing blocks.

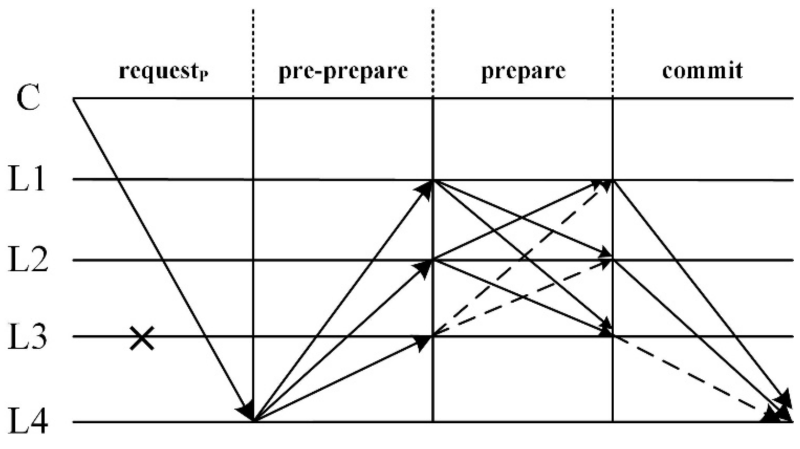

Figure 9. Subworker node is malicious.

4.1.3. Feasibility analysis

According to the needs of data security and data management efficiency in the community digital governance scenario, the data security and consensus efficiency of CSPBFT were analyzed. In the context of community digital governance, blockchain technology is used to manage data such as user information, incentive mechanisms and problem handling. It can ensure the security and integrity of data and avoid data loss and tampering. This section presents the experimental test results for CSPBFT. When the number of nodes is between 400 and 600 or exceeds 600, the CSPBFT algorithm still has high throughput and low latency. While realizing efficient access to data, it can provide high concurrency for community digital autonomy with efficient data management operations. The CSPBFT algorithm can meet the requirements for real-time interaction of users in a community digital governance scenario. In existing research, Decred [20] utilized a hybrid consensus mechanism that can satisfy community application scenarios with 1000 nodes, a 10-second consensus delay and 200 M monthly transaction volume. On the basis of ensuring that the blockchain management community data are feasible, the ability of the blockchain to process data has been improved by CSPBFT. Finally, the feasibility of this scheme is verified by scheme comparison and performance delay testing.

4.2. Performance testing

To assess the consensus performance of the CSPBFT algorithm, three main tests were performed: a time-delay test for reaching consensus between nodes in the network, a throughput test for network transactions and a performance comparison with the existing Byzantine algorithm. The time delay and transaction throughput for nodes to reach consensus represent the consensus performance of the CSPBFT algorithm and affect the efficiency of online community governance. In the simulation test of the CSPBFT consensus algorithm, monitoring nodes subworker nodes and peer nodes were set to A, D, F in each subnetwork for testing. Transaction information was sent in scenarios with different numbers of nodes for simulation testing.

4.2.1. Experimental environment settings

In order to verify the feasibility of applying the CSPBFT algorithm to online community governance, the subworker node and monitoring node were simulated on a PC through the docker container, and the peer nodes in each subnetwork were simulated by opening different ports on the PC. The hardware configuration and virtual machine configuration of the PC used in the experimental test were as follows: PC (AMD Ryzen 7 5800H, 3.20 GHz, 16 G memory) virtual machine (four-core CPU, main frequency 2.0 GHz, 4 GB memory). Other software environmental settings are shown in Table 2. The experimental process occurred as follows. Start the client of the CSPBFT network. Send consensus messages to subworker nodes, subnet master nodes, candidate nodes and monitoring nodes through the client. Use scripts to send transaction notifications to the client, simulating the CSPBFT consensus process for multiple transactions. The aim was to test the consensus delay and throughput of CSPBFT consensus and compare the performances of different schemes. The consensus delay is the time of blockchain consensus data, which determines the response efficiency of user access. Throughput is the number of transactions processed by the zone system per unit time, usually expressed as TPS. It determines the maximum ability of the system to process data instantaneously. The TPS is an important indicator used to measure the ability of a blockchain to process multiple transactions concurrently.

Table 2. Environment configuration.

Software configuration	Software version
Window	v 10
Ubuntu	v 20.04
Golang	v 1.17.5
Nodejs	v 16.13.1
IPFS	v 0.12.0
MySQL	v 8.0
Docker	v 20.10.7
Docker-compose	v 1.25.0

4.2.2. Consensus latency

Four network scenarios were simulated with different numbers of nodes to test the consensus time delay; the test involved the simulation of sending 1000, 2000, 3000, ..., 10,000 transactions to the network, and 20 simulation tests were conducted for each scenario. The consensus time delay for each scenario was as follows (Figure 10). The number of nodes of each identity in the four scenarios was set as follows: **Sc1** (**A**: 5; **D**: 40; **F**: 4), **Sc2** (**A**: 7; **D**: 60; **F**: 6), **Sc3** (**A**: 11; **D**: 80; **F**: 8), **Sc4** (**A**: 13; **D**: 100; **F**: 8).

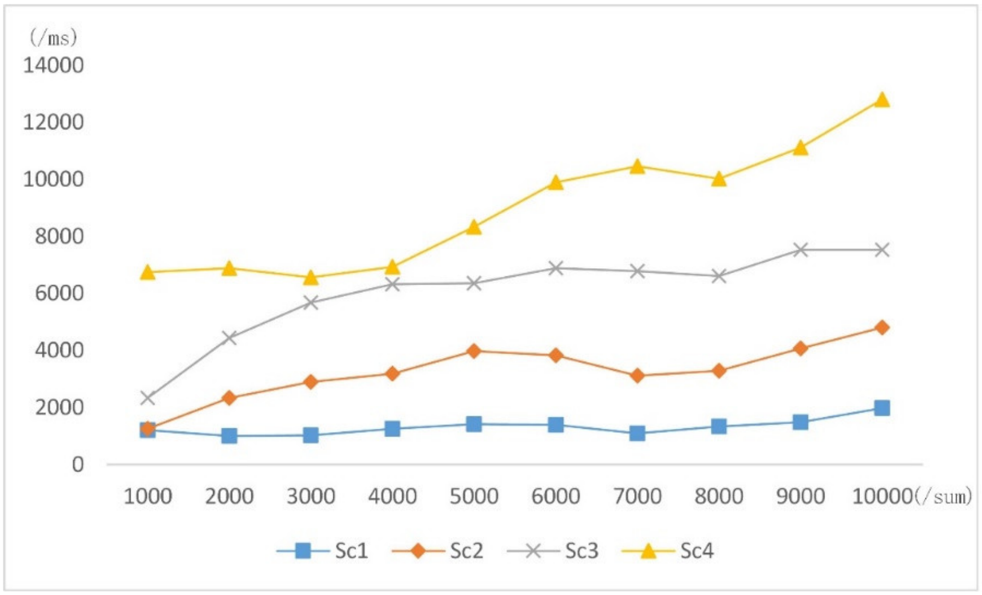

Figure 10. Consensus time delay test.

According to the test results, when the transaction volume was 6000–8000, the time delay of CSPBFT consensus work changed relatively steadily with increasing transaction volume, and the increase was substantial. When the number of subworker nodes reached 100 in **Sc4**, the speed of consensus delay improvement was greater than that in scenarios with fewer than 100 subworker nodes.

4.2.3. Throughput

Four network scenarios were simulated with different numbers of nodes to test the consensus time delay; the simulation involved sending 1000, 2000, 3000, ..., 10,000 transactions to the network, with 20 simulation tests for each situation; the throughput of transactions in each scenario was processed as shown in Figure 11. The number of nodes of each identity in the four scenarios was set as follows: **Sc1** (**A**: 5; **D**: 40; **F**: 4), **Sc2** (**A**: 7; **D**: 60; **F**: 6), **Sc3** (**A**: 11; **D**: 80; **F**: 8), **Sc4** (**A**: 13; **D**: 100; **F**: 8).

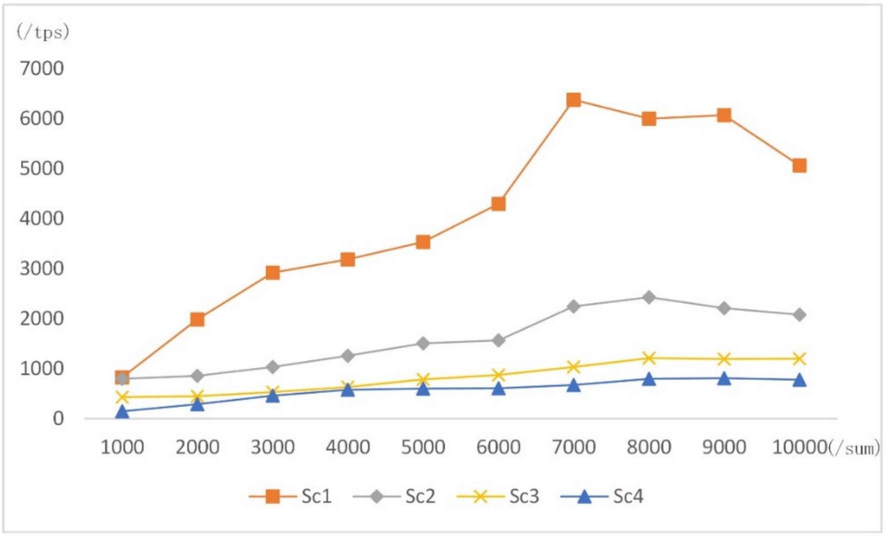

Figure 11. Throughput test.

According to the test results, when the transaction volume was 6000–8000, the transaction throughput using CSPBFT in each scenario reached its highest value; when the transaction volume was 1000–6000, the processing throughput increased slowly. When the volume was more than 8000, the throughput started to decrease as the transaction amount increased. When the number of subworker nodes reached 100 in **Sc4**, the speed of consensus delay improvement was greater than that in scenarios with fewer than 100 subworker nodes.

4.3. Performance comparison

According to the performance test results for CSPBFT, as the number of nodes in the network increase, the performance will deteriorate. Simulations were performed in which there were 7000 transactions to networks with a total of 200, 400, 600 and 800 nodes, with 20 simulation tests for each network. The test results for performance were compared with those obtained with PBFT, WBFT, mPBFT and SG-PBFT. The final comparison results are shown in Figures 12 and 13.

The throughput comparison between the CSPBFT algorithm and the existing Byzantine algorithm is shown in Figure 12. The transaction throughput of the CSPBFT algorithm is better than that of other algorithms. After the number of nodes in the network increases to 600TPS, the throughput of the existing algorithm will decrease to 100TPS, the CSPBFT algorithm can guarantee more than 1000

TPS, and, as the number of nodes in the network increases, the decrease in throughput tends to be flat. It can meet the data concurrency requirements of the daily work of large communities.

The results of the comparison of the consensus time delay of the CSPBFT algorithm with those of existing Byzantine algorithms are shown in Figure 13. The time delay of the CSPBFT algorithm for consensus work was much shorter than those of other algorithms. After the number of nodes in the network exceeded 400, the time delay caused by the consensus rapidly increased for some algorithms; Compared with other algorithms with slower growth rates, CSPBFT had a smaller consensus time delay. The CSPBFT algorithm can thus ensure consensus efficiency on the basis of reducing the hardware requirements for online community governance.

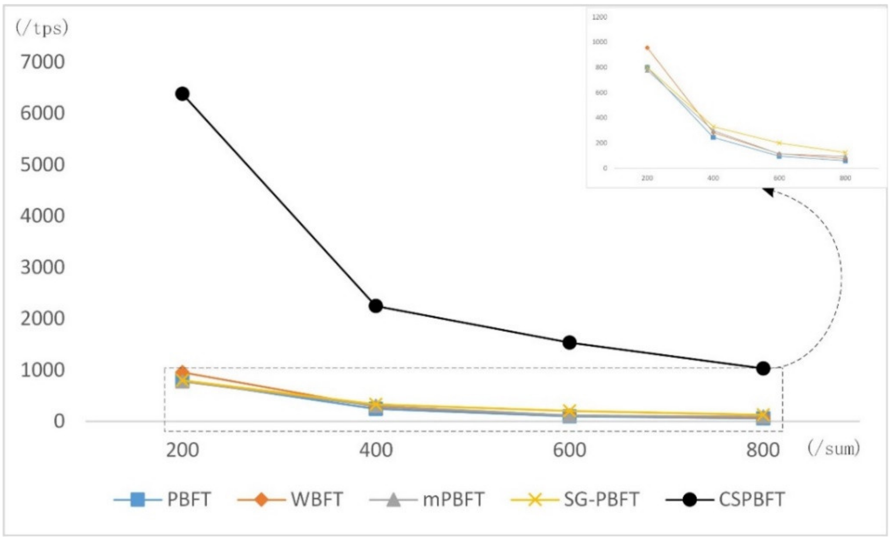

Figure 12. Byzantine consensus throughput comparison.

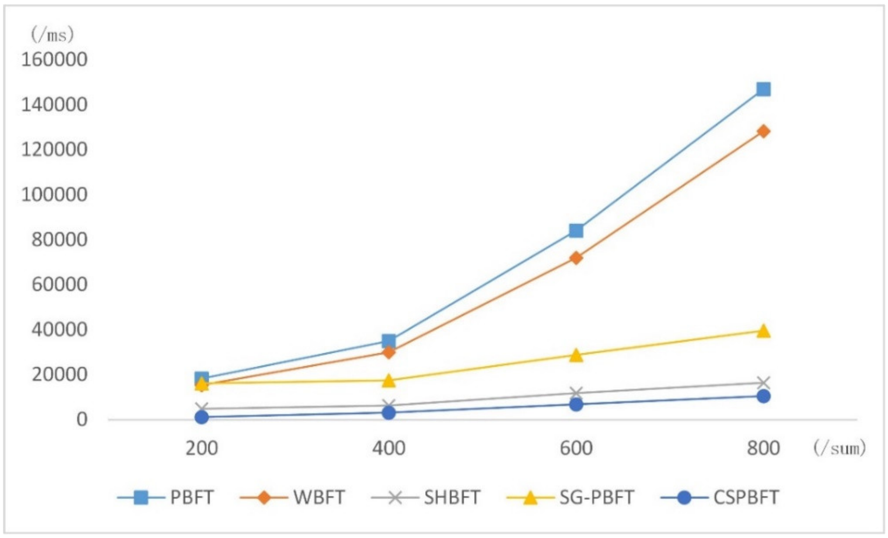

Figure 13. Byzantine consensus time delay test.

5. Conclusions

The combination of blockchain and community digital governance solves the problems of low data security, difficulty in sharing and tracing and low enthusiasm on the part of multiple subjects regarding participation in community governance. However, it cannot provide efficient concurrent performance. The participants in the community governance process are composed of relevant departments and users, and the governance data do not require all nodes in the network to participate in the consensus. Previous studies have optimized various consensus algorithms, but it has still been difficult to meet the requirements for high-concurrency and low-latency services.

Therefore, we have proposed a PBFT optimization algorithm based on community contribution. The consensus work is divided into three stages: election stage, consensus process and result verification. The consensus network is divided into two levels. Based on the contribution of nodes in the consensus network and community governance, the nodes of each subnetwork participating in the block are selected, and the monitoring node is set to review and verify the work of the block node. This improves the accuracy of consensus while reducing the communication cost of each round of consensus, so as to provide higher concurrency for community governance. The performance of the CSPBFT algorithm was tested experimentally, and we found that it is feasible to use the CSPBFT algorithm to manage community governance data.

However, this study was not without limitations. CSPBFT can provide high concurrent access, but the optimization of data access only compresses the consensus data. Considering the long-term accumulation of data, it is necessary to design a more optimized data storage method. In addition, data access security needs to be considered when optimizing data storage. Therefore, future work will address the following aspects.

1) Owing to the large-scale user base environment, the upper-level service has high data storage requirements that could exceed current hardware storage limitations. Sharding technology [21,22] could be used to store CSPBFT data and reduce hardware requirements for node services. Future work will further improve the interaction rate between users and consensus networks and improve user efficiency.

2) When users participate in consensus to access data, data security is controlled through user authentication policies such as authority management and identity verification to ensure data security. Future work will eliminate the risk of data leakage and improve the credibility of community governance data.

Acknowledgments

This work was part of the research projects of the National Natural Science Foundation of China (No. 62001262); and the Key Research and Development Foundation of Shandong Province of China (No. 2019GGX101017), as supported by the Department of Science & Technology of Shandong province.

Conflict of interest

All authors declare no conflict of interest regarding the publication of this paper.

References

1. Y. W. Chai, W. B. Guo, Smart management and service of communities in Chinese cities, *Prog. Geogr.*, **34** (2015), 466–472. https://doi.org/10.11820/dlkxjz.2015.04.008
2. F. Zhang, Study on the holistic governance of mega city community, *Urban Dev. Stud.*, **28** (2021), 1–4+10.
3. C. F. Han, Research on innovating mechanisms for community governance based on blockchain, *Frontiers*, **2020** (2020), 66–75. https://doi.org/10.16619/j.cnki.rmltxsqy.2020.05.007
4. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2022. Available from: https://assets.pubpub.org/d8wct41f/31611263538139.pdf.
5. N. Elisa, L. Yang, F. Chao, Y. Cao, A framework of blockchain-based secure and privacy-preserving E-government system, *Wireless Netw.*, 2018. https://doi.org/10.1007/s11276-018-1883-0
6. Y. Li, H. Y. Duan, Y. Y. Yin, H. H. Gao, Survey of crowdsourcing applications in blockchain systems, *Comput. Sci.*, **48** (2021), 12–27.
7. S. Zhu, Z. Cai, H. Hu, Y. Li, W. Li, Zkcrowd: A hybrid blockchain-based crowdsourcing platform, *IEEE Trans. Ind. Inf.*, **16** (2019), 4196–4205. https://doi.org/10.1109/TII.2019.2941735
8. H. Qin, Y. Cheng, X. Ma, F. Li, J. Abawajy, Weighted Byzantine Fault Tolerance consensus algorithm for enhancing consortium blockchain efficiency and security, *J. King Saud Univ. Comput. Inf. Sci.*, **34** (2022), 8370–8379. https://doi.org/10.1016/j.jksuci.2022.08.017
9. Y. Min, The modification of pBFT algorithm to increase network operations efficiency in private blockchains, *Appl. Sci.*, **11** (2021), 6313. https://doi.org/10.3390/app11146313
10. J. Martins, B. Fernandes, I. Rohman, L. Veiga, The war on corruption: the role of electronic government, in *EGOV 2018: Electronic Government*, (2018), 98–109. https://doi.org/10.1007/978-3-319-98690-6_9
11. J. A. Garcia-Garcia, N. Sánchez-Gómez, D. Lizcano, M. J. Escalona, T. Wojdyński, Using blockchain to improve collaborative business process management: systematic literature review, *IEEE Access*, **8** (2020), 142312–142336. https://doi.org/10.1109/ACCESS.2020.3013911
12. Y. Meshcheryakov, A. Melman, O. Evsutin, V. Morozov, Y. Koucheryavy, On performance of PBFT blockchain consensus algorithm for IoT-applications with constrained devices, *IEEE Access*, **9** (2021), 80559–80570. https://doi.org/10.1109/ACCESS.2021.3085405
13. H. Xiong, M. Chen, C. Wu, Y. Zhao, W. Yi, Research on progress of blockchain consensus algorithm: a review on recent progress of blockchain consensus algorithms, *Future Internet*, **14** (2022), 47. https://doi.org/10.3390/fi14020047
14. H. Samy, A. Tammam, A. Fahmy, B. Hasan, Enhancing the performance of the blockchain consensus algorithm using multithreading technology, *Ain Shams Eng. J.*, **12** (2021), 2709–2716. https://doi.org/10.1016/j.asej.2021.01.019
15. M. Pandey, R. Agarwal, S. Shukl, N. K. Verma, Reputation-based PoS for the restriction of illicit activities on blockchain: algorand usecase, preprint, arXiv:2112.11024.
16. G. Xu, H. Bai, J. Xing, T. Luo, N. N. Xiong, X. Cheng, et al., SG-PBFT: A secure and highly efficient blockchain PBFT consensus algorithm for Internet of vehicles, *J. Parallel Distrib. Comput.*, **164** (2022), 1–11. https://doi.org/10.1016/j.jpdc.2022.01.029

17. Y. Na, Z. Wen, J. Fang, Y. Tang, Y. Li, A derivative PBFT blockchain consensus algorithm with dual primary nodes based on separation of powers-DPNPBFT, *IEEE Access*, **10** (2022), 76114–76124. https://doi.org/10.1109/ACCESS.2022.3192426
18. Y. Li, L. Qiao, Z. Lv, An optimized Byzantine fault tolerance algorithm for consortium blockchain, *Peer-to-Peer Networking Appl.*, **14** (2021), 2826–2839. https://doi.org/10.1007/s12083-021-01103-8
19. S. Tang, Z. Wang, J. Jiang, S. Ge, G. Tan, Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain, *Sci. Rep.*, **12** (2022), 4426. https://doi.org/10.1038/s41598-022-08587-1
20. Decred btcsuite developers, in US and CA, *Decred Documentation*, 2023. Available from: https://docs.decred.org/getting-started/project-history/.
21. J. L. Wang, X. Wang, Y. M. Shen, X. Y. Xiong, W. H. Zheng, P. Li, et al., Building operation and maintenance scheme based on sharding blockchain, *Heliyon*, **9** (2023), E13186. https://doi.org/10.1016/j.heliyon.2023.e13186
22. H. W. Huang, W. Kong, X. W. Peng, Z. B. Zheng, Survey on blockchain sharding technology, *Comput. Eng.*, **48** (2022), 1–10. https://doi.org/10.19678/j.issn.1000-3428.0063887