A Blockchain Communication Resource Optimization Consensus Method

Jingchang Yu*
Faculty of Information Engineering and Automation, Kunming University of Science and Technology
yujingchang@stu.kust.edu.cn

Tao Shen*†
Faculty of Information Engineering and Automation, Kunming University of Science and Technology
shentao@kust.edu.cn

Fenhua Bai*
Faculty of Information Engineering and Automation, Kunming University of Science and Technology
bofenhua@stu.kust.edu.cn

Zhuo Yu‡
Beijing Zhongdian Puhua Information Technology Co., Ltd
yuzhuo@sgitg.sgcc.com.cn

Jianzhao Luo§
Yunnan Key Laboratory of Blockchain Application Technology
luojianzhao@bhynii.com

ABSTRACT

Since the emergence of Bitcoin, the blockchain technology behind it has been gradually gaining attention from all walks of life. As the core of blockchain technology, the consensus algorithm determines the security, scalability and decentralization of the blockchain and many other important characteristics. The efficiency of the current blockchain consensus algorithm still needs to be improved. To address this issue, this paper proposes a consensus algorithm for communication resource optimization (CCRO), which divides consensus nodes into different domains, calculates the trust of nodes by several trust factors, and assigns different roles to nodes according to the trust. It also introduces a new class of nodes, i.e., communication nodes, which are responsible for the delivery of messages in the consensus process, and achieves the communication resource optimization of the blockchain system. By building an experimental platform, we verify the performance of CCRO, which downscales the inter-node consensus communication protocol, optimizes the regional data synchronization protocol, reduces the network communication overhead, and effectively improves the consensus efficiency.

CCS CONCEPTS

• **Security and privacy**; • **Systems security**; • **Distributed systems security**;

*Jingchang Yu, Tao Shen, Fenhua Bai are with the Faculty of Information Engineering and Automation, Kunming University of Science and Technology of China, Kunming, China (e-mail:yujingchang@stu.kust.edu.cn; shentao@kust.edu.cn; bofenhua@stu.kust.edu.cn;).
†Corresponding author: Tao Shen, e-mail (shentao@kust.edu.cn)
‡Zhuo Yu is with the Beijing Zhongdian Puhua Information Technology Co., Ltd(e-mail:yuzhuo@sgitg.sgcc.com.cn)
§Jianzhao Luo is with the Yunnan Key Laboratory of Blockchain Application Technology of China, Kunming, China(e-mail:luojianzhao@bhynii.com)

KEYWORDS

Blockchain, Consensus algorithm, Communication resource

ACM Reference Format:
Jingchang Yu, Tao Shen, Fenhua Bai, Zhuo Yu, and Jianzhao Luo. 2022. A Blockchain Communication Resource Optimization Consensus Method. In *2022 4th Blockchain and Internet of Things Conference (BIOTC 2022), July 08–10, 2022, Tokyo, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3559795.3559810

1 INTRODUCTION

In 2008, Nakamoto first proposed Bitcoin [1], and the digital currency entered a new stage of development, with its underlying blockchain technology receiving increasing attention from people in all walks of life. Blockchain systems can be analogized to a distributed database [2] technology, and more broadly, blockchain belongs to a decentralized [3] digital bookkeeping technology with tamper-evident, decentralized and privacy-preserving features. The block refers to the data block, the data is written into the block after the node vote, and the generated blocks are articulated into a chain in chronological order. From a technical perspective, blockchain is a unique data recording format where blocks record all transactions and status results that occur over a period of time, a consensus on the current state of the ledger. Transaction data is generated and stored in blocks that are chained together in the order in which they occur. Importantly, the data in the blockchain system is jointly maintained by all nodes, each maintaining node gets a copy of the complete data, and the blocks of the blockchain are closely connected, if a malicious node wants to tamper with the data in a certain data block, it needs to change all the subsequent blocks of that block, so that the gain from tampering with the data is much less than the cost spent, thus ensuring the security of data storage [4] in the blockchain.

The consensus algorithm is the core technology of blockchain [5], which not only ensures the security and stability of blockchain, but also is the critical factor affecting the efficiency of the whole blockchain system. The consensus mechanisms commonly used in alliance chains are PBFT, Raft, and Paxos, but these algorithms still cannot meet the requirements of existing application scenarios in terms of consensus speed, and the high-performance blockchain mechanisms need further research, therefore we propose a consensus algorithm for communication resource optimization (CCRO).

The consensus algorithm for communication resource optimization (CCRO) research idea contains four core concepts of trust, regionalization, communication resource optimization, and message logging. Each of these four concepts is explained below.

- Trust. Due to the strict entry and exit characteristics of the alliance chain [6], the state stability of the nodes in the blockchain system is required to be high. The trust is not a value subjectively determined, but an objective evaluation of the ability of nodes in the blockchain system, which plays a non-negligible role in judging the quality of nodes in the system. Nodes with high motivation and high consensus completion frequency will have high trust, while a node with long response time and error in consensus process will have low trust. Nodes play different roles according to their trust, and thus perform different tasks in the consensus process.
- Regionalization. Cross-platform interactions within enterprises, between enterprises, and between enterprises and governments are a local phenomenon. This is because the data of each unit involves commercial secrets. Based on this, this paper regionalizes the nodes in the blockchain system and divides them into several domains that do not interfere with each other. The nodes within each domain only need to interact with the internal nodes, which reduces the message complexity and enhances the scalability of the blockchain [7]. All domains will share the same blockchain.
- Communication resource optimization. The common consensus mechanism in alliance chains is the Practical Byzantine Fault Tolerance algorithm, PBFT. In the PBFT consensus process, all nodes are involved and the communication overhead is extremely high. Therefore, this paper proposes a special class of nodes, i.e., communication nodes, which are elected by the blockchain system according to the trust and are responsible for the message transmission between consensus nodes. Communication nodes greatly reduces the message complexity in the consensus process and reduces the communication overhead, thus achieving the optimization of communication resources in the blockchain system.
- Message log. All messages during the consensus process are recorded in the message log, which is equivalent to a ledger. All nodes perform log backups and verify voting information. The function of the message log is to prevent CCRO from becoming centralized due to the presence of communicating nodes.

Section 1 and section 2 of this article mainly discuss the related work of blockchain technology and consensus mechanism research. Section 3 proposes a consensus algorithm for communication resource optimization (CCRO). Section 4 verifies the proposed communication resource optimization consensus method through experiments. Moreover, a conclusion is drawn in section 5.

2 RELATED WORKS

Consensus algorithm [8] is the core of blockchain system, however, the current consensus algorithm has become a bottleneck in the performance of the whole blockchain system. Consensus refers to a consistent result reached by multiple mutually independent participants on the same issue. Consensus in blockchain means that on an open distributed computer system, certain measures are taken to enable most nodes to agree on the same block. However, it also consumes huge computational, storage and communication resources, which makes the performance of the blockchain system low, and the more the number of nodes, the more prominent this drawback is.

In response to these shortcomings, relevant studies have been conducted. The literature [9] proposed a DPBFT algorithm for dynamic trustworthiness evaluation of blockchain energy transactions. Xiang F et al [10] proposed Jointgraph, a DAG-based Byzantine consistency algorithm for federated chains. The literature [11] proposed a blockchain consensus protocol based on quality of service (Qos). Wang Y et al [12] proposed Credit Delegation Byzantine Fault Tolerance protocol (CDBFT) which can motivate the reliable nodes. The literature [13] proposed a scalable dynamic multi-body hierarchical PBFT (SAMA-PBFT) algorithm, which reduces the communication overhead from $O(n^2)$ to $O(n*k*log_k n)$. Lu X et al [14] proposed a blockchain energy internet distributed energy trading scheme based on SDN. The literature [15] developed an optimization model and a blockchain-based architecture to manage the operation of crowdsourced energy systems. Sheikh A et al [16] focused on the process of energy transactions between electric vehicles and distributed networks under a Byzantine-style blockchain consensus framework.

Although the above research has solved the problem of low node motivation in the consensus process, the problems of low throughput and high transaction latency have not been completely solved to meet the needs of existing application scenarios. Therefore, this paper proposes a consensus algorithm for communication resource optimization.

3 CONSENSUS MODEL

3.1 Trust

3.1.1 Definition of trust. This paper experiments with a blockchain system containing $n(n \in N,\ n \geq 6)$ nodes. Nodes can be assigned to$m$ mutually independent domains at the logical level, each with $(3f+3)(f \in N)$ nodes. The trust R is used to denote the current trust of nodes in the blockchain network with a value domain of $[1, 10]$. R_{ij} denotes the trust of node j in domain i. The value of R_{ij} is positively related to the stability and reliability of nodes. The initial trust level of the node is R_{init} and the default value is 5.

The nodes are classified into 3 roles according to the trust level. $R_{primary}$ denotes the lower limit of trust as a primary node, R_{comm} denotes the lower limit of trust as a communication node, and R_{normal} denotes the lower limit of trust as a consensus node, and these values satisfy $R_{primary} > R_{comm} > R_{normal}$. The default value of $R_{primary}$ is 9, the default value of R_{comm} is 8, and the default value of R_{normal} is 4. If a node maintains stable operation and reliable performance during the consensus process, then as the number of consensus rounds increases and its trust grows, the node will gradually be upgraded to a communication node or even a primary node; on the contrary, if a node runs unstable in the consensus process, its trust will decrease gradually to below

R_{normal} and cannot participate in the consensus process. It can only improve the trust by the response time when storing blocks.

3.1.2 Trust Impact Factor. The calculation of trust is based on an objective assessment of nodes' behavior in the consensus process, and nodes' behavior can be evaluated from different aspects. For example, some nodes operate stably and have a long uptime after accessing the system, while some nodes are not highly motivated and have consensus message timeouts, etc. These different aspects can be reflected by factors such as node uptime, frequency of participation in consensus, etc. These factors are collectively referred to as trust impact factors. These impact factors have different weights in the trust evaluation, and the resulting trust can objectively evaluate the capability of nodes.

Definition 1 (Node Uptime Value) The ratio of the node uptime to the total time after it is connected to the blockchain system, which reflects the stability of the node. The node uptime value I_1 is expressed as:

$$I_1 = \sqrt{\frac{T_{normal}}{T}} \tag{1}$$

Where T_{normal} denotes the uptime of the node in the system without failure, T denotes the total time from the node's access to the system to its exit from the system, and a larger value of I_1 indicates higher node stability.

Definition 2 (Node Positivity) Refers to the frequency of node participation in consensus after accessing the system, which reflects the degree of node aggressiveness. Node positivity I_2 is expressed as:

$$I_2 = e^{\frac{E-E_{avg}}{E_{avg}}} \tag{2}$$

Where, E denotes the number of nodes participating in consensus, E_{avg} denotes the average of the number that all nodes in the system participate in consensus. If the difference is positive, indicating that the number of nodes participating in consensus is greater than the average value, and the larger I_2 indicates that the nodes are more active.

Definition 3 (Consensus Completion) The frequency of nodes participating in consensus and completing consensus, which reflects the operation status of nodes in the blockchain system. The consensus completion degree I_3 is expressed as:

$$I_3 = \frac{M}{D} \times 100\% \tag{3}$$

Where, M denotes the number of consensus nodes participated and completed, D denotes the total number of consensus nodes participated, and larger I_3 denotes superior node performance.

Definition 4 (Response evaluation value) The primary node waits while the node votes on the preparatory block and stores the formal block. To avoid long waiting times, R′ is introduced. The response evaluation value R′ is expressed as:

$$R' = \begin{cases} R_{last} - 1.2, & (t_1 + t_2) \text{ in the top } 20\% \\ R_{last}, & (t_1 + t_2) \text{ in the middle } 60\% \\ R_{last} + 1.2, & (t_1 + t_2) \text{ in the bottom } 20\% \end{cases} \tag{4}$$

Among them, in the experimental platform of this paper, the time to generate a block is about 120ms, i.e. 0.12s. We use this time as the maximum of the primary waiting time. In order to eliminate the slow responding nodes as soon as possible, we set the weight of this value to 10, and the calculated weight is 1.2. R_{last} indicates the trust of the nodes after the last round of consensus. t_1 indicates the voting time of each node for the preparatory block. t_2 denotes the storage time of each node for the formal block. The sum of t_1 and t_2 is used as the response time of the node. The longer the response time is the longer the waiting time of the primary node. The value is sorted in descending order. If a node is ranked in the first quintile of all nodes, its trust will be reduced by 1.2, if it is ranked in the second quintile, the trust will be increased by 1.2, and if it is ranked in the middle quintile, no change will be made.

Definition 5 (Trust Influence Factor Weights) The weights of the trust impact factors are W_1, W_2 andW_3, which correspond to the three influence factors I_1, I_2 and I_3. The weight values are 0.4, 0.2 and 0.4, respectively. The higher the weight value, the more important the impact factor is.

3.1.3 Trust Calculation. At the end of each consensus process, the system evaluates the behavior of each node in the current consensus round. The trust R_{ij} of node j in domain i is expressed as:

$$R_{ij} = I_1 \cdot R_{init} + \frac{\sum_{u=1}^{3} I_u \cdot W_u}{\sum_{u=1}^{3} I_u} + \left(R' - R_{last}\right) \tag{5}$$

Where, R_{init} is the initial value of node trust. R_{last} denotes the trust of nodes after the last round of consensus. I_1 is the value of node uptime. I_2 is the node positivity. I_3 is the consensus completion degree. R′ is the response evaluation value. W_1, W_2, W_3 are the weights occupied by the influence factors.R_{ij} can be accurately used as an objective evaluation of nodes' behavior in the current round of consensus. If nodes are highly motivated and have a good consensus completion degree, or respond quickly to consensus messages, both will result in a high level of trust for the node. Conversely, if nodes are less motivated and respond to consensus messages overtime, they will gain a lower trust level.

In CCRO, the concept of trust is introduced to prevent all nodes from participating in consensus, so that richer resources can be devoted to the consensus process. Thus, the system has more abundant resources to devote to the consensus process, speeding up the consensus operation and increasing the speed of generating blocks.

3.2 CCRO-related role definitions

3.2.1 Monitoring Node. The monitoring node is proposed in the CCRO consensus algorithm, the role is defined as a regulator, which is introduced specifically to ensure the strict entry and exit characteristics of the alliance chain. The monitoring node is responsible for monitoring the behavior of other nodes, which can significantly save the resources of the blockchain system and improve efficiency. The main responsibilities of the monitoring node include:

(1) Identify node status: Nodes in the blockchain system will send heartbeat data regularly. Monitoring nodes identify node status from heartbeat data. Only nodes with normal status can participate in the consensus process;

(2) Domain splitting operation: After identifying the nodes in the normal state of the blockchain system, the monitoring node randomly divides the nodes into several domains with at least 3f + 3 nodes in each domain;

(3) Management of blockchain system nodes: New nodes entering the blockchain system first authenticate themselves to the

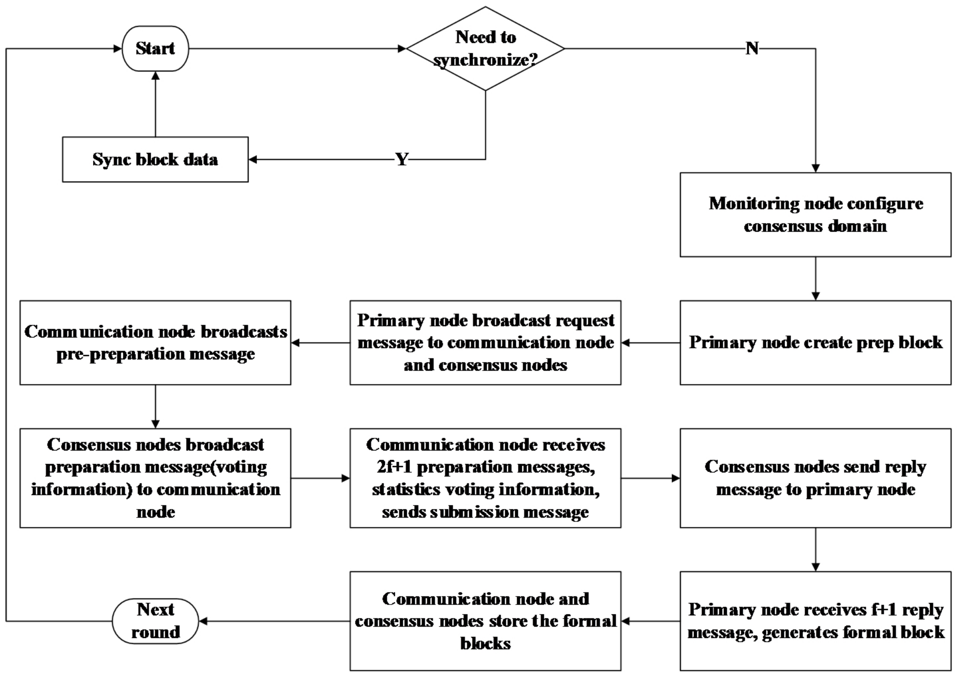
Start
Need to synchronize?
N
Y
Sync block data
Monitoring node configure consensus domain
Primary node create prep block
Primary node broadcast request message to communication node and consensus nodes
Communication node broadcasts pre-preparation message
Consensus nodes broadcast preparation message(voting information) to communication node
Communication node receives 2f+1 preparation messages, statistics voting information, sends submission message
Consensus nodes send reply message to primary node
Primary node receives f+1 reply message, generates formal block
Communication node and consensus nodes store the formal blocks
Next round

Figure 1: CCRO consensus algorithm flow.

monitoring node. They can participate in the consensus only after the monitoring node has passed the verification;

(4) Role authorization: Monitoring node can assign roles to all nodes based on trust, such as primary, communication and consensus nodes, and the domains they belong to;

(5) Scan domain list: The monitoring node scans the status of all nodes in the domain. If all nodes are normal, it can be used as the domain for consensus;

(6) Verify data synchronization: Monitoring node verify whether the block heights are consistent after each node data synchronization is completed. The specific algorithm is shown in Algorithm 1.

Algorithm 1 Block verification

```
nodes list $\{N_1, N_2, N_3, \ldots, N_m\}$
for each node in $Domain_N$
    get $N_n\_blockheight$ from $Domain_N$
      If $N_n\_blockheight == \{N_1, N_2, N_3, \ldots, N_m\}_{blockheight}$
        return true
      else
        synchronization block
    end
end
```

3.2.2 Primary Node. The primary node is the node with trust greater than $R_{primary}$ and normal status. The primary node sends a request message to the communication node and consensus nodes, which contains the preparatory block. Preparatory blocks with transaction data to be voted on. After the voting is completed, the primary node sends the generated formal block to each node for storage.

3.2.3 Communication Node. Communication node is a normal node with trust greater than $R_{comm}$and less than $R_{primary}$. Communication node is mainly responsible for consensus message delivery between consensus nodes. After a new round of consensus starts, the communication node receives the request message from the primary node, broadcasts the pre-prepared message to consensus nodes, then receives and counts the prepared messages from consensus nodes, and finally broadcasts the submission message to all consensus nodes.

3.2.4 Consensus Node. Consensus node is a normal node with trust greater than R_{normal} and less than R_{comm}. The consensus node is responsible for receiving the pre-preparation message broadcasted by the communication node, voting on the preparation block, sending the preparation message to the communication node, after that receiving the submission message, and finally sending the reply message to the client.

3.3 CCRO consensus flow

The CCRO consensus protocol flow is shown in Figure 1, with the following steps:

Data synchronization phase: Monitoring nodes scan consensus nodes $\{N_1, N_2, N_3, \ldots, N_n\}$and determine whether the block heights are consistent. If they are inconsistent, they must synchronize the block heights and then participate in the consensus. The specific algorithm is shown in Algorithm 2

Request phase: After the monitoring node opens a new consensus round, the primary node sends a request message to the

Algorithm 2 Data synchronization

```
nodes list $\{N_1, N_2, N_3, \ldots, N_m\}$ verification
preparation block height $Pre\_blockheight$
for $\{N_1, N_2, N_3, \ldots, N_m\}_{blockheight}$
if $\{N_1, N_2, N_3, \ldots, N_m\}_{blockheight}$==$Pre\_blockheight$
    return true
else
    Temp=$Pre\_blockheight$
    $\{N_1, N_2, N_3, \ldots, N_m\}_{blockheight}$=Temp
    end
end
```

communication node and consensus nodes. The request message contains the generated preparatory blocks. The primary node then writes the request message to the message log.

Pre-preparation phase: The communication node receives the request message initiated by the primary node and enters the pre-preparation phase. It broadcasts the pre-preparation message to consensus nodes and writes the pre-preparation message to the message log.

Preparation phase: The consensual nodes receive the pre-preparation message and enter the preparation phase. They broadcast a preparation message to the communicating nodes, which contains voting information. The preparation messages are written to the message log.

Submission phase: The communication node receives 2f+1 preparation messages from different nodes and enters the submission phase. It counts the voting results and sends the commit message to consensus nodes. The commit message is written to the message log.

Reply phase: After the consensus node receives the submission message, it enters the reply phase. Consensus nodes send reply messages to the primary node. After the primary node receives f + 1 reply messages, the request gets completed at this time. The primary node then performs the block-out operation. The reply message is written to the message log.

Storage phase: After the primary node generates the formal block, the communication node and consensus nodes store the formal block. The primary node opens the next round of consensus.

The above is a detailed consensus process. In the CCRO consensus algorithm, the consensus process is separated from the execution of the block-out process which makes the blockchain system have more abundant storage resources and improve the consensus efficiency.

A comparison of the message complexity of CCRO algorithm and PBFT algorithm is shown in Figure 2. The addition of communication node greatly reduces the message complexity in the consensus process and decreases the communication pressure of the system. It further shortens the consensus process cycle and effectively improves the efficiency of consensus.

3.4 Security analysis

3.4.1 Malicious Nodes Tampering with Block Information Will Not Take Effect. CCRO is a consensus algorithm based on alliance chain optimization, which has a strict access and exit mechanism, so the probability of nodes doing evil is small. The monitoring node is

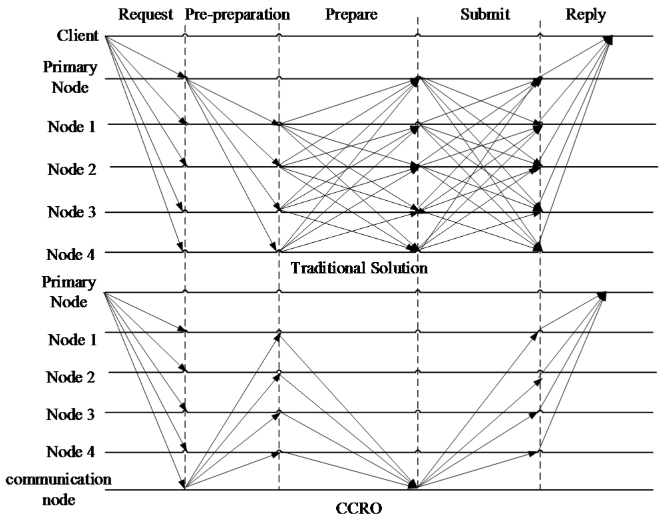

Figure 2: Comparison of message complexity between CCRO and PBFT

set up for the supervision department, so the malpractice of the monitoring node is not considered. The monitoring node will check the message log and the block data during the consensus process. If the node's block data is tampered, the node block information verification will not pass and the node will be kicked out of the system. The block information tampered by the malicious node will not be valid.

3.4.2 System Fault Tolerance Analysis. Assuming that the number of system nodes is n and the number of malicious nodes is f, we consider two extreme cases.

The first and best case scenario, where f nodes are both malicious and faulty nodes. Then, according to the principle of majority rule, only $f + 1$ normal nodes are needed for the system to reach consensus. Considering the presence of primary and communication node, the number of nodes in each domain is at least $2f + 3$, i.e., $n \geq 2f + 3$. The maximum number of malicious nodes supported in this case is $(n - 3)/2$.

In the second and worst case, there are f malicious nodes and f faulty nodes in the system. In the CCRO algorithm, malicious nodes are kicked out of the system, leaving f faulty nodes. Similarly, the number of normal nodes should be at least $f + 1$. Therefore, the number of all types of nodes is $3f + 3$, i.e., $n \geq 3f + 3$. The maximum number of malicious nodes supported in this case is $(n - 3)/3$.

Combining the two cases, the maximum number of malicious nodes supported by CCRO is $(n - 3)/3$, so we set the number of nodes in each domain to $n \geq 3f + 3$, where f represents the upper limit of the number of malicious nodes. In combination with 3.4.1, it is impossible for a malicious node in CCRO to tamper with the blockchain successfully.

3.4.3 CCRO can Effectively Prevent Replay Attack and Bifurcation. Each node in the CCRO consensus algorithm signs the messages it sends and verify the signatures of the messages it receives. Each node maintains a public-private key pair, the private key is used to sign the sent messages and the public key is used as the node ID to identify and verify the signatures. In the preparation phase, the communication node determines whether there are duplicate preparation messages, checks whether the block parent hash contained in the preparation message is the highest block hash of the current node, and whether the block height of the preparation block is equal to the highest block height plus one. Therefore, CCRO can effectively prevent bifurcation and replay attacks.

3.4.4 CCRO can Effectively Prevent Sybil and Eclipse Attacks. In blockchain networks, sybil attacks fakes out multiple identities for malicious vote brushing. An eclipse attack modifies the node's configuration file to separate the normal node from the network.

In this paper, each node must obtain permission to join the blockchain. Each node has a unique identity. Messages in consensus must be signed and verified using key pairs. If the verification is not passed, the message is invalid. In addition, once a node's configuration file is modified, the monitoring node will kick it out of the system. Based on the above mechanism, CCRO can effectively prevent attacks such as Sybil and Eclipse.

3.4.5 CCRO can Effectively Prevent the Primary Node and Communication Node from Abnormal. In CCRO, if the primary node or communication node is down, the system will reselect the node to continue the previous operation. And we design a dynamic reward and punishment mechanism based on trust, where the trust of nodes goes up and down according to their performance in consensus, so that the primary node and communication nodes are not fixed.

4 EXPERTMENT AND ANALYSIS

In order to analyze the performance of the CCRO algorithm, we built an experimental platform using the Java language. Experimental platform using Redis as a queuing tool for transactional sorting and MySQL for data persistence. We use the slicing technique to achieve the division of domains. After the monitoring node gets the transaction data from Redis, it slices and encodes it based on the code parameters in the database. The database and Redis are run separately in the docker environment. In order to reduce the interference of network quality in this experiment, we deploy a small local area network. 18 nodes are deployed in this experiment, the CPU frequency of the monitoring node is 2.90GHz and the memory is 16G, the CPU frequency of the rest of the nodes is 2.20GHz and the memory is 8G. The communication between the nodes is established based on the gRPC framework. The standard values of PBFT were obtained from the literature [17].

4.1 Node trust change

After running the CCRO algorithm on the experimental platform for a period of time, the monitoring node calculated the trust of six nodes based on the trust impact factor, as shown in Figure 3.

After each node joins the system, the trust changes as the number of consensus rounds increases, and the monitoring nodes have different choices of primary and communication nodes. Take node 1 for example, in the first two rounds it runs stably, but in the third round there is a failure and the value of the impact factor I_1 decreases, so the trust level decreases and node 5 replaces it as the primary node, with node 1 as the consensus node. After a period of

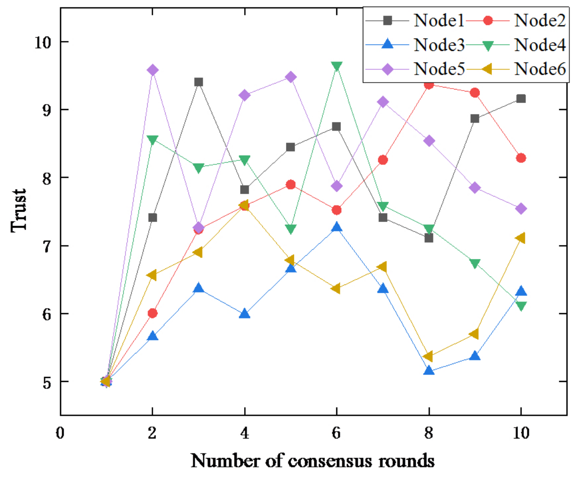

Figure 3: Change of node trust.

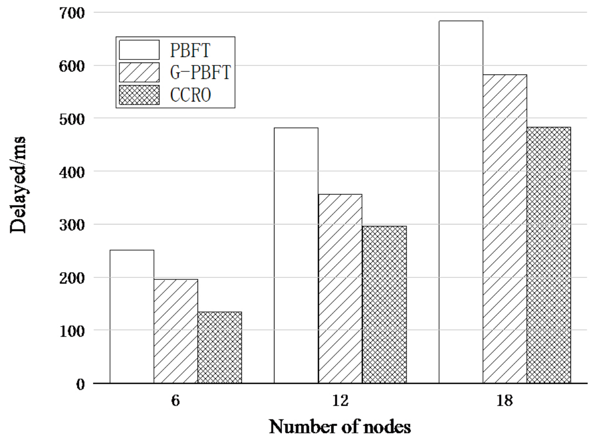

Figure 4: System delay under different consensus algorithms.

normal operation, its trust rises, and in the sixth round it takes too long to respond and the response evaluation value R′ decreases, leading to a decrease in trust. So the roles of each type of nodes are not constant in CCRO.

4.2 Comparative analysis of system delays under different consensus algorithms

To verify whether CCRO improves the performance of the blockchain system, this subsection uses system latency as a metric to examine. For comparative analysis, we use the performance data of the standard PBFT algorithm as a baseline and also implement the G-PBFT algorithm that uses grouping alone. As can be seen from Figure 4, the PBFT algorithm performance is inversely proportional to the number of nodes, and the average transaction request latency increases from 251ms to 680ms. The performance of the grouping-based G-PBFT consensus algorithm is slightly improved due to the fact that grouping reduces the network-wide consensus

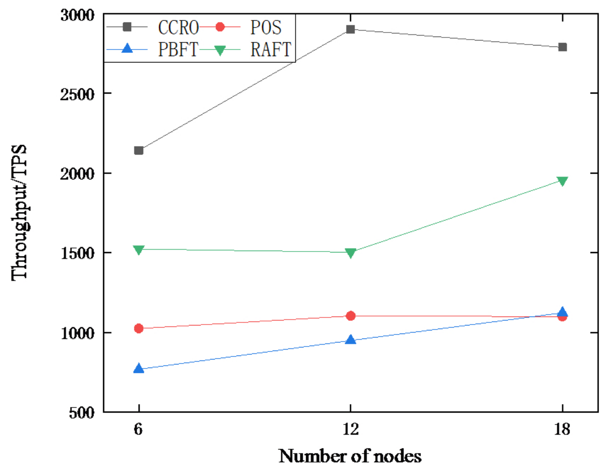

Figure 5: TPS comparison under different number of nodes

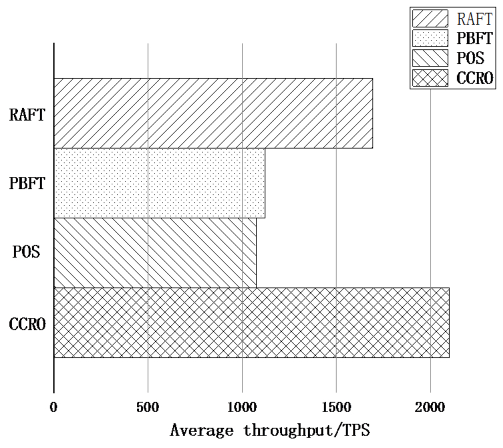

Figure 6: Average TPS comparison

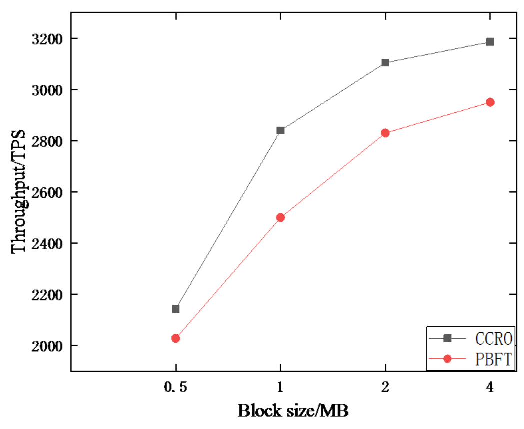

Figure 7: System throughput of different block sizes

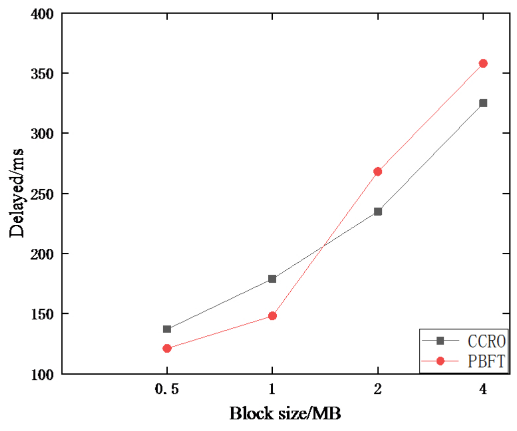

Figure 8: System delays of different block sizes

message complexity. The average latency of CCRO transaction requests at 18 nodes is 489ms, which is close to the latency of PBFT algorithm at 12 nodes. The experiment demonstrates the good scalability of the CCRO consensus algorithm, which does not degrade performance due to the dramatic increase in consensus messages when the number of nodes in the blockchain system increases.

4.3 TPS comparative analysis

System throughput is a performance metric for evaluating distributed systems. In general, system performance is evaluated in terms of the number of transactions completed per second (TPS). We use TPS to compare the performance difference of several algorithms.

$$\text{TPS} = \frac{\text{TX\_length}}{\text{process\_time}} \tag{6}$$

Where TX_length is the number of transactions contained in a block; process_time is the time to generate a block.

Due to the differences among consensus algorithms, we send 1000 simulated transactions per second to Redis clients considering the accuracy of the experimental data and the carrying capacity of the system. We count the time to perform 20 consecutive rounds of consensus for Raft algorithm, PBFT algorithm, POS algorithm and CCRO of this paper with 6, 12 and 18 nodes, respectively. The results are shown in Figure 5 and Figure 6.

As can be seen from Fig. 5 and Fig. 6, the peak and average TPS of CCRO outperforms other consensus algorithms, and it is observed that a certain round of consensus takes 142ms, generating a preparatory block and storing a formal block takes 45ms, and the voting process on the data occupies 68.3% of the consensus cycle. The CCRO in this paper substantially reduces the consensus message complexity and significantly improves the TPS.

4.4 The impact of block size on system performance

In order to compare the performance difference of the system under different block sizes, four values of 0.5MB, 1MB, 2MB and 4MB are chosen respectively, which are also the four values often chosen when actually deploying blockchain systems. From Figure 7 and Figure 8, it can be seen that as the block size increases to 2MB, both CCRO and PBFT throughput increase, while the CCRO algorithm system throughput increases more than PBFT. At this time, the PBFT algorithm latency exceeds that of the CCRO algorithm, and the average latency of CCRO algorithm transaction requests increases from 137ms to 310ms.This is because as the block size increases, the consensus count decreases, improving the transaction processing capacity of the system, but for some individual requests, the waiting time for data to be packed increases the length of the request. When the block size increases from 2MB to 4MB, the CCRO algorithm system throughput increases from 3104TPS to 3185TPS, which is only a 2% increase, and the system latency also increases. It is easy to see that increasing the block size at this point is no longer able to significantly improve system performance. Through comparative analysis, as the block size increases, the CCRO algorithm outperforms the PBFT algorithm in terms of both throughput and system latency. 2MB is the most suitable block size for the CCRO algorithm, and the blockchain system has the best performance at this time.

5 CONCLUSION

To address the efficiency problem of current blockchain consensus algorithms, this paper proposes a consensus algorithm for communication resource optimization)CCRO(. Experimental results show that CCRO can significantly reduce the loss of communication resources in the consensus process and effectively improve the efficiency of the blockchain system. In addition, for CCRO, we will do further work on security and efficiency, such as assigning roles to nodes based on game theory rather than trustworthiness, and investigating alternative monitoring nodes to deal with contingencies.

ACKNOWLEDGMENTS

The authors thank the Major Scientific and Technological Projects in Yunnan Province (no. 202002AB080001-8), Yunnan Key Laboratory of Blockchain Application Technology (funding no. 202105AG070005, open project no. YNB202115).

REFERENCES

[1] Nakamoto S J C. Bitcoin: A Peer-to-Peer Electronic Cash System[J].

[2] P. A. Bernstein, J. B. Rothnie, N. Goodman and C. A. Papadimitriou, "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," in IEEE Transactions on Software Engineering, vol. SE-4, no. 3, pp. 154-168, May 1978, doi: 10.1109/TSE.1978.231494.

[3] P. R. Pagilla, N. B. Siraskar and R. V. Dwivedula, "Decentralized Control of Web Processing Lines," in IEEE Transactions on Control Systems Technology, vol. 15, no. 1, pp. 106-117, Jan. 2007, doi: 10.1109/TCST.2006.883345.

[4] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," in IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 5, pp. 847-859, May 2011, doi: 10.1109/TPDS.2010.183.

[5] M. Belotti, N. Božić, G. Pujolle and S. Secci, "A Vademecum on Blockchain Technologies: When, Which, and How," in IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3796-3838, Fourthquarter 2019, doi: 10.1109/COMST.2019.2928178.

[6] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang and E. Hossain, "Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains," in IEEE Transactions on Industrial Informatics, vol. 13, no. 6, pp. 3154-3164, Dec. 2017, doi: 10.1109/TII.2017.2709784.

[7] Q. Zhou, H. Huang, Z. Zheng and J. Bian, "Solutions to Scalability of Blockchain: A Survey," in IEEE Access, vol. 8, pp. 16440-16455, 2020, doi: 10.1109/ACCESS.2020.2967218.

[8] F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong and M. Zhou, "Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism," in IEEE Access, vol. 7, pp. 118541-118555, 2019, doi: 10.1109/ACCESS.2019.2935149.

[9] Cai W, Jiang W, Xie K, *et al.* Dynamic reputation–based consensus mechanism: Real-time transactions for energy blockchain[J], 2020, 16(3): 1550147720907335.

[10] Xiang F, Huaimin W, Peichang S, *et al.* Jointgraph: A DAG-based efficient consensus algorithm for consortium blockchains[J], n/a(n/a).

[11] Yu B, Liu J, Nepal S, *et al.* Proof-of-QoS: QoS based blockchain consensus protocol[J]. Computers & Security, 2019, 87: 101580.

[12] Wang Y, Cai S, Lin C, *et al.* Study of Blockchains's Consensus Mechanism Based on Credit[J]. IEEE Access, 2019, 7: 10224-10231.

[13] Feng L, Zhang H, Chen Y, *et al.* Scalable Dynamic Multi-Agent Practical Byzantine Fault-Tolerant Consensus in Permissioned Blockchain[J]. Applied Sciences, 2018, 8(10).

[14] Lu X, Shi L, Chen Z, *et al.* Blockchain-Based Distributed Energy Trading in Energy Internet: An SDN Approach[J]. IEEE Access, 2019, 7: 173817-173826.

[15] Wang S, Taha A F, Wang J, *et al.* Energy Crowdsourcing and Peer-to-Peer Energy Trading in Blockchain-Enabled Smart Grids[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49(8): 1612-1623.

[16] Sheikh A, Kamuni V, Urooj A, *et al.* Secured Energy Trading Using Byzantine-Based Blockchain Consensus[J]. IEEE Access, 2020, 8: 8554-8571.

[17] M. Hu, T. Shen, J. Men, Z. Yu and Y. Liu, "CRSM: An Effective Blockchain Consensus Resource Slicing Model for Real-Time Distributed Energy Trading," in IEEE Access, vol. 8, pp. 206876-206887, 2020, doi: 10.1109/ACCESS.2020.3037694.