GT-NRSM: efficient and scalable shardingconsensus mechanism for consortiumblockchain

Tao Shen (shentao@kust.edu.cn)
Kunming University of Science and Technology
Tianyu Li (litianyu@stu.kust.edu.cn)
Kunming University of Science and Technology
Zhuo Yu (yuzhuo@sgitg.sgcc.com.cn)
Beijing Zhongdian Puhua Information Technology Company,Ltd.
FenHua Bai (bofenhua@stu.kust.edu.cn)
Kunming University of Science and Technology
Chi Zhang (laxzhang@outlook.com)
Kunming University of Science and Technology

Research Article

Keywords:

DOI: https://doi.org/

Additional Declarations: No competing interests reported.

GT-NRSM: efficient and scalable sharding consensus mechanism for consortium blockchain

Tao Shen[1], Tianyu Li[1], Zhuo Yu[2], FenHua Bai[1*] and Chi Zhang[1]

[1*]Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, 650500, Yunnan, China.
[2]Beijing Zhongdian Puhua Information Technology Company,Ltd., Beijing, 100192, Beijing, China.

*Corresponding author(s). E-mail(s): bofenhua@stu.kust.edu.cn; Contributing authors: shentao@kust.edu.cn; litianyu@stu.kust.edu.cn; yuzhuo@sgitg.sgcc.com.cn; laxzhang@outlook.com;

Abstract

Blockchain is an innovative application of distributed storage, consensus mechanism, cryptographic algorithm and other computer technologies. As the underlying architecture of blockchain, consensus mechanism is the key to realize service-oriented applications of blockchain in terms of its security, efficiency and scalability optimization. In some high complexity consensus mechanism such as Practical Byzantine Fault Tolerance (PBFT), throughput is severely reduced as the number of nodes increases, and even in low complexity algorithms such as Raft, the load on leader is severely affected as the network size increases, which affecting consensus efficiency. To solve these problems, in this paper, we propose a node reliable shard model based on guarantee tree (GT-NRSM) that achieves high scalability while ensuring a certain degree of decentralization and security based on consortium blockchain. Firstly, we design a guarantee mechanism to represent the trust relationship between nodes, and then we design a reliable node selection strategy based on the guarantee mechanism to evaluate the node guarantee results and consensus behavior, determine the node trust status, and identify malicious nodes and select a list of trusted leaders. Secondly, we propose a Dual-Leaders supervision mechanism, where deputy detects the heartbeat of leader while the deputy activity is detected by consensus nodes. Finally, we use guarantee mechanism and reliable node selection strategy to design a network partitioning method to achieve high concurrent consensus for multiple partitions and greatly improve the consensus efficiency. Subsequent experiments show that the throughput of the proposed algorithm improves by 48% over Raft and is much higher than PBFT, which has higher throughput and lower consensus latency.

Keywords: Blockchain, Consensus, Shard, Leader election,Guarantee Tree

1 Introduction

Blockchain is a revolutionary technology that has received strategic attention from various countries [1, 2], it is a traceable and growing distributed ledger database that relies on a cryptographic algorithm, consensus mechanism, and peer-to-peer (P2P) network technology to link blocks together in a chain-like structure. With the combination of blockchain technology with several scenarios such as finance, trade, law, and the Internet of Things, blockchain research and applications have shown an explosive growth trend and have become an emerging technology with equal influence and promise as big data, artificial intelligence, cloud computing, and autonomous driving[3].

Blockchain, as a decentralized and distributed system, has decision making power dispersed among various nodes. Each node needs to agree on the block data. Therefore, there are fundamental and practical obstacles to its wider applicability. Previously, it was generally believed that the most critical issue of blockchain was the trilemma of decentralization, security and scalability. But now with the addition of the concept of trust, trust is very important for blockchain scalability and there is a trade-off between trust and decentralization. A blockchain system with trusted nodes does not compromise security even if the consensus process is reduced. Thus the impossibility triangle theory of distributed systems can be extended to a quadrilateral in blockchain systems, increasing trust. The issue of blockchain scalability requires a deeper study of the trade-offs.

Bitcoin processes about 3-4 transactions per second, while Ether processes only 14 transactions per second, and consensus algorithms designed based on Consortium Blackchain, such as PBFT consensus algorithm throughput decreases severely as the number of users involved in consensus increases. Scalability is the main obstacle facing the implementation of blockchain applications[4]. To further improve the scalability of blockchain systems, existing approaches can be divided into off-chain and on-chain solutions. Off-chain solutions are currently divided into payment channels, side chains, off-chain computing, and cross-chains, avoiding the computational costs of traditional blockchain systems that require each honest node to receive, store, and send relevant data, and to agree among all nodes while ensuring the order of the total amount of valid data. The solution has applicability in some cases, but in many application scenarios, the solution is needed, i.e. to record, verify and save all data. Giulio Malavolta et al.[5] defined an anonymous multi-hop locks(AMHL) cryptographic algorithm for designing privacy-preserving Primary Care Networks(PCNs) that is compatible with multiple cryptocurrencies, allows atomic swapping and interoperability from the bottom, and the algorithm has been applied to PCNs in lightning networks. Rami A. Khalil et al.[6] proposed a non-custodial second layer financial intermediary solution that prevents double spending and guarantees user control over funds by using a decentralized blockchain ledger with smart contracts as a means of dispute resolution. Kalodner et al.[2] proposed a smart contract enabled cryptocurrency system Arbitrum, which frees miners from the execution and verification of smart contracts to improve scalability, with smart contracts done off-chain. J Kwon et al.[7] designed Cosmos, an independent blockchain network that can connect different blockchain platforms and can independently parallelize and interact with each other to achieve interoperability and scalability.

On-chain solutions differ from off-chain solutions where the chain handles the main chain structure by modifying some aspects of it; they do not introduce another chain or perform any task outside the main chain, but focus on the consensus, network and data structure of the blockchain. Currently there are mainly block data processing, Directed Acyclic Graph(DAG) directed acyclic graph structures, sharding techniques, and concurrent solutions. Jeremy Rubin, M. Naik et al.[8] proposed a Bitcoin improvement scheme that converts Bitcoin scripts into Merkle trees of their Merkelized Abstract Syntax Trees(MAST) branches, unused scripts can be removed from the block, and when the Merkle proof returns True, the transaction proponent needs to provide a proof of the Merkle tree with the missing script branch in order to prove spending bitcoins. Sompolinsky et al.[9] proposed a PoW permissionless blockchain PHANTOM based on the DAG structure, which promotes the directed acyclic graph of bit blocks to Satoshi Nakamoto's blockchain. It can distinguish between blocks correctly mined by honest nodes and blocks created by malicious nodes that choose to deviate from the mining protocol. Sharding technology is one of the existing solutions with high research significance. Parwat Singh Anjana et al.[10] proposed a parallel smart contract framework to execute smart contract transactions concurrently using Software Transactional Memory Systems(STM). Miners use multiple threads to execute smart contract transactions simultaneously to generate the final state of the blockchain. Most of the existing network sharding techniques are based on geographic clustering or random number sharding. The sharding method based on geographic clustering can hardly provide effective guarantee for sharding reliability, while the random number sharding method can guarantee slicing reliability, but the sharding communication consumption is large.

This paper proposes an on-chain solution for federated chains that aims to improve the scalability of traditional blockchain systems by weakening a certain degree of security and decentralization. In summary, the main contributions of this paper include the followings:

- We put forward a node-guarantee mechanism that applies the idea of node-guarantee and designs a Maximum Guarantee Tree(MG-Tree) to visualize the trust relationship between nodes.
- We propose a reliable node selection strategy based on node evaluation with guarantee mechanism, which can evaluate node behavior and calculate Trust_Value, including leader election scheme and anomalous node determination and elimination scheme, which reduces both the number of anomalous nodes and the possibility of anomalous nodes becoming leader, so that the total number of malicious nodes does not exceed the maximum number tolerable by the network.
- Dual-Leaders supervision mechanism is proposed, which can quickly achieve leader's rotation when leader is abnormal and reduce the impact of leader abnormalities on the consensus process. In which deputy detects the heartbeat of leader, while deputy itself is detected by consensus nodes. In this paper, we compare Dual-Leaders supervision mechanism with the Raft's leader election, and experiments show that the consensus latency of Dual-Leaders supervision mechanism is improved by 9% compared with Raft.
- We design a consensus partitioning model based on guarantee mechanism and reliable node selection strategy, which divides the shards according to node reliability and trust relationship between nodes, and ensures that the number of nodes between shards is approximately equal. It can effectively improve the transaction processing speed while ensuring shard's reliability.
- We put forward a GT-NRSM applied to consortium blockchain with high performance as well as high scalability, and the communication complexity of the algorithm is O(N).
- We conduct experiments in terms of throughput, node anomaly model, leader's rotation delay, horizontal and vertical of sharding, and block scalability, and compared the experimental results.

The rest of this paper is organized as follows. Section 2 reviews the research results related to blockchain consensus algorithms, trust model, and consensus partitioning techniques. Section 3 introduces the current consensus model. Section 4 describes and analyzes the proposed algorithm in detail. We evaluate the performance of our proposed algorithm in Section 5. Finally, we conclude the paper and discuss future work in Section 6.

2 Related work

Related work areas covered by GT-NRSM: blockchain consensus mechanism, trust model, sharding technology.

2.1 Blockchain consensus algorithm

A blockchain is essentially a distributed ledger. All users in a distributed network can record in the ledger through consistency protocols[11]. There are various kinds of consistency protocols, and consensus algorithms are an umbrella term for these consistency protocols. Consensus algorithms can be broadly classified into two classes: proof class and voting class.

The most famous proof of consensus algorithm is the proof of work (PoW) mentioned by Satoshi Nakamoto in the Bitcoin white paper. In PoW, all users have the same bookkeeping rights. The first user to figure out the PoW problem gets the correct bookkeeping rights[12] and also receives a certain amount of bitcoins as a reward. PoW bookkeeping requires a large amount of computing power resources for value-free computations. Bahri et al.[13] proposed a proof-of-trust (PoT) blockchain in which peer-to-peer trust is evaluated and used as a partial replacement for PoW in the network based on a trust graph that appears in a decentralized manner and is encoded and managed by the blockchain itself, which can reduce resource consumption.

In proof of stake(PoS), the block producer's decision is determined by the value of equity held by the user. The higher the equity value, the more likely it is to become a billing node, but the vast majority of equity resources are still controlled by a few users. The rich get richer. Cheng et al.[14] introduced behavioral credits in the PoS mechanism based on the original PoS, establish credit ratings, and interact with monetary age. In this way, the income distribution of the nodes involved in the generation of the PoS mechanism is made more fair and reasonable, improving social stratification and resisting the tendency of system centralization.

Unlike proof algorithms, voting algorithms do not require a lot of hashing operations and require a lot of communication between nodes. Raft consensus algorithm is an early distributed consistency algorithm that achieves consensus from the perspective of multi-replica state machines by managing the log replication of multi-replica state machines. However, it only considers fault tolerance for erroneous nodes and has no Byzantine fault tolerance. Huang et al.[15] proposed a two-level consensus mechanism RBFT with supervisory nodes for federated chains. First, the network nodes are grouped and the group adopts the improved Raft mechanism for consensus, and then the leaders elected from each group form a network committee and the network committee adopts the PBFT mechanism for consensus within the network committee. RBFT has Byzantine fault tolerance and can has Byzantine fault tolerance and high consensus efficiency, and thus has higher scalability.

The PBFT consensus algorithm was first implemented in Fabric v0.6.0. This consensus algorithm selects a master node from the whole network nodes responsible for creating blocks, and then excludes malicious nodes from influencing the consensus after three stages of voting, but the O(N^2) communication complexity makes the system performance degrade as the network size increases. Gao et al.[16] proposed a novel optimized practical Byzantine fault-tolerant consensus algorithm T-PBFT based on the eigentrust model, where the algorithm evaluates node trust through transactions between nodes to select high-quality nodes in the network to build consensus groups. It can optimize Byzantine fault tolerance and reduce the probability of perspective change and communication complexity. The consensus algorithm makes the blockchain very useful in many scenarios. The following table1 compares the various consensus algorithms mentioned above and analyzes the advantages and disadvantages of each consensus algorithm from different aspects.

Table 1 Performance comparison of different consensus algorithms

Name	Raft[17]	PoW[18]	PoS[19]	PBFT[20]	GT-NRSM
Decentralization	Medium	High	High	Medium	Medium
Security	Low	High	High	Medium	Medium
Consistency	Medium	Weak	Weak	Strong	Medium
Scalability	Medium	Low	Low	Medium	High
Communication complexity	N	N	N	N^2	N
Leader election	Medium	/	/	Slow	Fast
Resource consumption	Less	more	less	less	less

2.2 Trust model

The trust model was first systematically discussed by Marsh, who first formalized trust with a mathematical model and proposed a model for trust metrics in terms of relevant attributes such as the content and degree of trust. Trust models can be classified into centralized trust models and distributed trust models based on the presence or absence of third-party management. The engentrust[21] and peertrust[22] are two classical trust evaluation models in distributed networks for trust evaluation of distributed network nodes to avoid untrustworthy transaction objects and improve the security of transactions. The engentrust iteratively calculates the global trust value based on the direct trust value between nodes, and proposes that the recommendation behavior of nodes with high trust is biased and trustworthy; the peertrust calculates the direct reputation of nodes based on the interaction evaluation between nodes, and uses the feedback evaluation between nodes, the number of transactions, the trustworthiness of the feedback evaluation, the transaction time, the amount and the environment as calculation factors to calculate the reputation value of nodes.

Song et al.[23] proposed a multidimensional trust index system and evaluation mechanism (MDTEM) for fintech built on blockchain to design blockchain-based index system and evaluation mechanism for direct trust, indirect trust, recommendation trust and feedback trust in fintech, which can effectively improve the secure application of fintech trust mechanism. Pal, S et al.[24] proposed a blockchain-based trust management framework that allows independent trust providers to implement different trust metrics on a common set of trust evidence and provide individual trust values. We use geographic location as proof of interaction. Zhang et al.[25] proposed a sharding model to optimize fault tolerance, number of cross-sharding transactions, and sharding trust. Its application to a blockchain system can help it improve the trustworthiness of its sharded transaction processing and reduce its transaction latency, thus enhancing the scalability of existing technologies. Lahbib et al.[26] proposed an IoT-based blockchain trust architecture to evaluate device trustworthiness, define trust scores for each device, and securely store and share them with other devices in the network by embedding them in blockchain transactions. It can effectively ensure system resilience against tampering and attacks, reliability, and low complexity of IoT scenarios and applications. Shala[27] et al. proposed a new approach to integrate blockchain technology into the trust assessment process and presented the concept of using blockchain for decentralized Machine to Machine (M2M) application service delivery within the system. The data integrity is verified by introducing a P2P overlay combined with a blockchain network.

2.3 Sharding technology

Traditional blockchains have serious throughput issues. Originally a technique used for databases, the great advantage of sharding is that each sub-shard is parallel to the others and different transactions are processed between the shards.

At present, blockchain mainly includes three types of sharding: network sharding, transaction sharding, and state sharding, among which, network sharding is the foundation and state sharding is the bottleneck[28]. Network sharding divides the whole network into different shards by a certain organization, and each shard processes some transactions in the whole blockchain in parallel, and each part of the transactions is completely different, so that multiple transactions can be verified at the same time. Transaction sharding enables each network shard to have stronger processing capability for transactions. It divides the client's cross-sharding transactions into several related sub-transactions, and the cross-sharding transactions of different shards can be processed in parallel. In order to reduce the pressure of storing the ledger in each node, state sharding stores each part of the completely different ledger in each shard, and the whole shard network forms a complete ledger. Under the current technical conditions, both network sharding and transaction sharding have a more desirable implementation, while there are still many technical problems in implementing state sharding.

Since 2016, LUU et al.[29] proposed a sharding Elastico algorithm to expand the capacity in the database, in which the node sharding technology is combined with blockchain technology, and PoW sharding with Byzantine Fault Tolerance(BFT) consensus is used to reduce the number of consensus users and improve the transaction processing speed. Zamani et al.[30] proposed a RapidChain public blockchain sharding protocol that does not require a trusted initial setup and enables parallel transaction processing, storage and communication, which can tolerate up to 33% of failed nodes across the network species, uses a cross-sharding communication protocol that avoids relaying transactions to the entire network nodes, and uses Visual_Secret_Sharing (VSS) to achieve randomness and ultimately greater throughput and scalability. Since then blockchain sharding technology has started to become one of the mainstream methods of blockchain scaling. Kokoris-Kogias et al.[31] proposed Omniledger to add committee refactoring to Elastico and proposed an anti-locking solution for cross-slice transactions, which effectively solved the problems of low Elastico operational efficiency and cross-sharding transactions. Chainspace was proposed by Al-Bassam et al.[32] to build a smart contract application platform on the basis of sharding, fostering communication sharding and computational sharding for transactions and smart contracts. Sel-Daniel et al.[28] proposed an approach to solve the data availability problem (DAP) in the context of slicing by reusing Casper's validator to achieve inter-node collaboration. In addition, Dange et al.[33] uses database sharding to achieve scalability of the blockchain and uses trusted random numbers generated by trusted hardware as the criteria for node assignment, which ensures the activity and security of the whole protocol to some extent. Kwak et al.[34] proposed a hierarchical negotiation mechanism based on service area sharding to improve the performance degradation due to increased system traffic. Yang et al.[35] used sharding techniques combined with a hybrid model to build an online e-voting system that guarantees the security of the voting process.

3 Frameworks and model

In this section, we provide problem statements for the general framework of GT-NRSM, network model, and node model.

3.1 Algorithmic framework

As shown in Figure 1, GT-NRSM is essentially a state machine replication algorithm. A transaction is distributed by the client, and leader generates a block and distributes it to the consensus nodes within shard. The consensus nodes verify the block, cast a ballot, and execute a log replication request if the block is valid. Otherwise, they will discard the block. the goal of GT-NRSM is to achieve data consistency across state machines efficiently and securely for client-side transactions. GT-NRSM consists of three parts: (1) Reliable node selection strategy, (2) Dual-Leaders supervision mechanism, (3) Consensus partitioning model. The reliable node selection algorithm is mainly used for malicious node identification and leader node selection, Dual-Leaders supervision mechanism is mainly used to ensure leader availability and achieve consistent and robust data consensus within shard, and consensus partitioning model is mainly used to divide the consensus domain and achieve reliable and efficient high concurrent consensus.

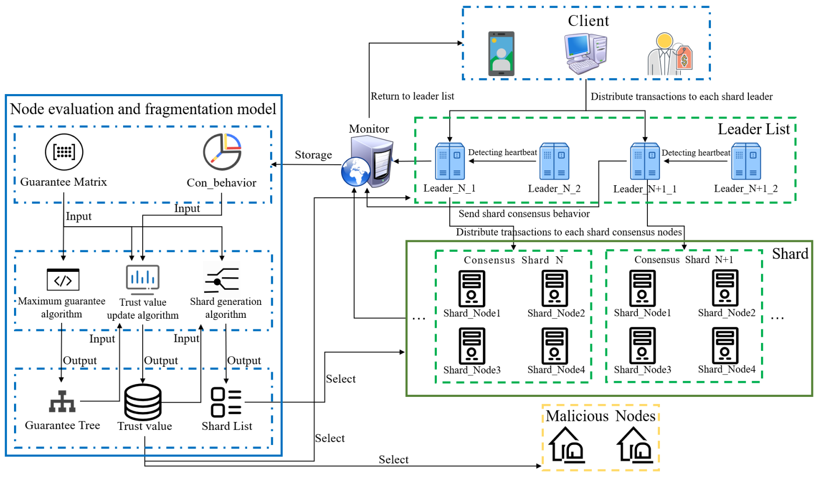


Fig. 1 GT-NRSM system framework structure

The reliable node selection strategy achieves node evaluation by calculating Trust_Value, which is implemented based on the guarantee mechanism. The guarantee mechanism establishes a link between all consensus nodes and other nodes once before the consensus cycle starts, tests the link duration as the communication cost, calculates the guarantee value to other nodes based on the communication cost and the consensus behavior of the target node, and finally generates a MG-Tree according to the maximum guarantee principle.

Dual-Leaders supervision mechanism is to add a deputy in place of the leader in each shard. The leader is responsible for receiving client-initiated transactions, packaging the transactions and initiating consensus requests to the consensus nodes in the shard. Deputy periodically receives heartbeats from leader, and if there is a significant delay in the heartbeat or no heartbeat is received at all, deputy then directly broadcasts instead of leader, and the consensus node within shard then initiates a new deputy campaign request. This mechanism improves the reliability of the system and ensures that consensus does not block due to leader performance issues.

Consensus partitioning model provides a network shard partitioning method, which first delineates the list of leader and the number of shards by a reliable node selection algorithm, and then introduces a guarantee mechanism, in which each shard is connected to a consensus node according to the maximum guarantee principle, while ensuring that the number of nodes in each shard is approximately equal. This mechanism reduces the communication pressure of the system and effectively improves the consensus speed.

3.2 Network model

The system model defines the number of nodes as n=3f+1. The system can be guaranteed to operate correctly with no more than f Byzantine fault nodes and f down nodes. Byzantine fault nodes are subject to malicious interference with consensus decisions or downtime, such as not sending messages or sending error messages. Network communication is a reliable p2p communication network. Consensus nodes can receive messages from other nodes. Broadcast messages cause the sender to send the message to all nodes in the network including itself. In this paper, a semi-synchronous network model is used. An upper bound is set on the message transmission time $\triangle t$. It is the time interval between sending and receiving. If the communication time exceeds $\triangle t$, the message is no longer processed.

3.3 Node Model

As shown in Figure 1, we illustrate the relationship between the nodes by defining the architecture diagram. The node states in the traditional consensus algorithm are mainly divided into leader and consensus nodes, and in GT-NRSM designed in this paper, the concept of monitor is added, and the detailed description of each definition is as follows.

Definition 1: (Consensus Nodes) The replicas in the system that participate in consensus are called consensus nodes. For the node numbered i sent by leader is defined as N_i. As shown in Equation 1, the set of all nodes involved in consensus is denoted by the symbol G. Node N_i(i=[1,n],$N_i \in G$) is aligned with the other nodes of G through the consensus mechanism.

$$G = (N_1, N_2, \cdots, N_i, \cdots, N_n) \tag{1}$$

$$N_i = (G_i, st_i, h_i, v_i, l_i, S_i, C_i) \tag{2}$$

As shown in Equation 2, where G_i denotes the consensus node ID and IP list saved by the node except itself, st_i denotes the state the node itself is in, where $st_i \in< leader, candidate, follow >$, and the tasks of each node in different states will be introduced in detail in Section 4. h_i indicates the height of the block whose node has reached the latest state. v_i indicates the view number. The view will be updated when the consensus cycle ends and leader pair changes. l_i indicates leader index, and the value may change only when the consensus cycle ends or when deputy initiates a campaign. S_i denotes the list of shards in which the node is located, and the consensus node only makes decisions on the transactions initiated within the shard. C_i denotes the guarantee value of consensus node for other nodes, and updates the guarantee value for other nodes whenever consensus node receives a guarantee request from monitor.

Definition 2: (Leader) Leader is selected from the consensus nodes, and leader is divided into two types of leaders as well as deputy, such as the leader of the shard is defined as l_{i_1} and deputy is defined as l_{i_2}, and the set of all leaders is represented by L. As shown in Equation 3, Leader l_i (i=[1,n],$l_i \in L$) is maintained by deputy and kept consistent with other nodes.

$$L = (l_1, l_2, \cdots, l_i, \cdots, l_n) \tag{3}$$

$$l_i = (N_i, B_i, SB_i, FB_i, \triangle t) \tag{4}$$

As shown in Equation 4, where N_i is the variable built in for leader as the consensus node, and B_i is the node malicious behavior judgment made by the leader based on the final consensus result for the decision of the consensus node within the shard. SB_i is the number of correct decisions made by consensus nodes within shard in the cycle, and FB_i is the number of malicious interference or wrong decisions made by the shard consensus nodes in the cycle. $\triangle t$ is the upper limit time for deputy to receive heartbeats from leader.

Definition 3: (Monitor) In order to implement node evaluation and consensus sharding, GT-NRSM algorithm introduces the concept of monitor, which is not involved in consensus decision, but only responsible for generating Leader_List and the Shard_List, and monitor is defined as M.

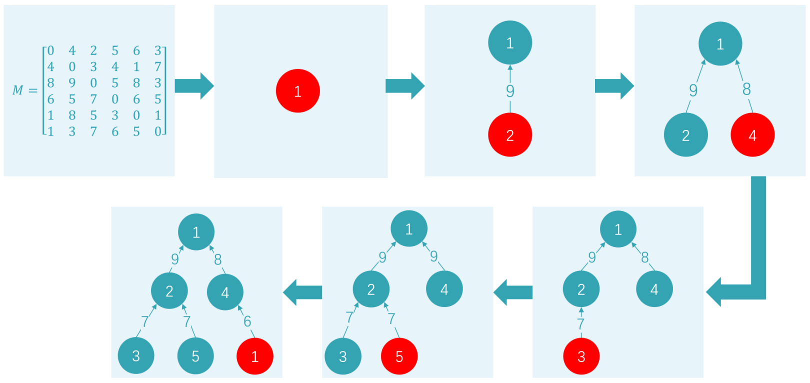
$$M = (G, L, GM, B, TV, S) \tag{5}$$

As shown in Equation 5, where G denotes the list of IP addresses of consensus nodes, L denotes the node index of leader, GM denotes the guarantee matrix composed of guarantee values, which is reset and updated by monitor when requesting guarantee values from consensus nodes, B denotes the node consensus behavior, which is identified by leader for consensus node decisions after each round of consensus in the cycle and broadcasted uniformly by monitor, TV denotes nodes's Trust_Value, which is updated after the consensus cycle based on the guarantee tree and consensus behavior, S denotes the shard's index list where each node is located, based on which each node determines its own shard index and consensus node list within shard.

4 Proposed algorithm

Existing blockchain consensus algorithms, such as PBFT, use multiple N-N broadcasts to experiment with data consistency, and this approach makes the blockchain consensus algorithm communication complexity of O(N^2) , and up to O(N^3) for PBFT if view transformation occurs[36]. Then, the distributed system consensus algorithm Raft, although the algorithm communication complexity is O(N), does not operate in a Byzantine environment and its leader needs to meet the high concurrent broadcasting conditions, and leader downtime resulting in leader rotation still slows down the consensus process, which in turn affects the normal efficiency of the blockchain system[37].

In order to solve the problems of insufficient security and inefficiency of the existing blockchain consensus layer. Firstly we first put forward a node guarantee mechanism to represent the trustworthy relationship between nodes. We design a reliable node selection strategy based on this to establish a node evaluation model, which can effectively evaluate nodes based on their performance and behavior. Secondly, we put forward a Dual-Leaders supervision mechanism, in which Dual-Leaders supervision mechanism each other to perform transaction distribution operations in turn. We also propose a consensus partitioning model, which selects leader based on a reliable node selection strategy and implements partitioning through a guarantee mechanism. Finally, we put



$M = \begin{bmatrix} 0 & 4 & 2 & 5 & 6 & 3 \\ 4 & 0 & 3 & 4 & 1 & 7 \\ 8 & 9 & 0 & 5 & 8 & 3 \\ 6 & 5 & 7 & 0 & 6 & 5 \\ 1 & 8 & 5 & 3 & 0 & 1 \\ 1 & 3 & 7 & 6 & 5 & 0 \end{bmatrix}$

Fig. 2 Example of MG-Tree generation process with M matrix

forward a GT-NRSM secure and scalable consensus algorithm with O(N) time complexity, which can support concurrent consensus while implementing malicious node identification and screening, and can effectively solve the problem of low consensus throughput and security.

4.1 Reliable node selection strategy

Most of the traditional consensus mechanisms are implemented based on weak centralized ideas, that is, the implementation of random leader rotation. Leader is the core role in the consensus process, leader to send and receive transactions. Copy nodes to verify transactions, if there are malicious nodes in the network to interfere with the consensus, there is a probability to affect the final decision. Even extreme cases if the malicious node is elected leader, the system will have a great security vulnerability resulting in the consistency of the blockchain system failure.

First, we put forward a guarantee mechanism among nodes, where all consensus nodes will calculate the guarantee value to other nodes and finally generate an adjacency matrix containing the guarantee values of all consensus nodes to other nodes, and finally find a critical path with maximum guarantee sum in this matrix to form a MG-Tree. The guarantee value represents the trust of nodes to other nodes. In the network ordinary consensus nodes do not have the ability to judge whether other nodes are trustworthy. This paper stipulates that in the case that nodes do not show incorrect behavior, nodes give priority to trust nodes with similar addresses or shorter communication time, and in unstable network conditions, nodes will be down or malicious behavior in the consensus process, and then the consensus behavior is added to one of the trust value measurement parameters. Firstly, the communication cost is calculated as:

$$T_i = t_i - t \tag{6}$$

As shown in Equation 6, t is the time when consensus node initiates the link request, and the consensus node replies to Trust_Value after receiving the request from other nodes, and the time when Trust_Value is received is t_i. The guarantee value is calculated as follows:

$$m_i = \frac{(T_S - t_i)}{(T_S - t)} - \gamma(\sum_{x=1}^{V} \rho_A + \sum_{x=1}^{V} \rho_A L) \tag{7}$$

As shown in Equation 7, where m_i denotes the guarantee value to node i, and T_S is the deadline for receiving true, i.e., consensus node is not processing link replies after T_S time. γ is the influence factor of node consensus behavior on the guarantee, the larger the value of γ the greater the influence of consensus behavior on the guarantee value. m_i decreases as t_i increases. If the replica node does not receive the link reply message, or the received message times out again or the node has too many wrong consensus times resulting in a negative guarantee value, the guarantee value of the node is set to 0. After that monitor collects the guarantee value of each node to generate the guarantee matrix as shown in Equation 8, and the guarantee tree is generated with the guarantee matrix M as an example.

$$M = \begin{bmatrix} 0 & 4 & 2 & 5 & 6 & 3 \\ 4 & 0 & 3 & 4 & 1 & 7 \\ 8 & 9 & 0 & 5 & 8 & 3 \\ 6 & 5 & 7 & 0 & 6 & 5 \\ 1 & 8 & 5 & 3 & 0 & 1 \\ 1 & 3 & 7 & 6 & 5 & 0 \end{bmatrix} \tag{8}$$

In the guarantee matrix, each row of values represents the evaluation made by the node to other nodes, for example, m_{ij} represents the evaluation made by node i to node j. The node cannot generate the guarantee value to itself. Monitor adjusts the guarantee adjacency matrix before calling MG-Tree generation algorithm, resetting the main diagonal element, i.e., m_{ii} element, to 0. Reliable node selection strategy generates a collateral mechanism, and the behavior of each node affects its guaranteed node reputation value, so each consensus node will eventually guarantee only one consensus node. But each consensus node can be guaranteed by multiple consensus nodes at the same time, and it is necessary to try to ensure that the node guaranteed by each node is the node with the largest guarantee value. MG-Tree generation process is shown in Figure 2.

The algorithm first selects consensus node with the largest guaranteed value as the source, then establishes the graph topology sequence, links consensus node with the largest guaranteed value for that node into the tree, and sequentially finds the node with the largest guaranteed value for the node on the guaranteed tree into the guaranteed tree. Finally forming a MG-Tree with all nodes out of degree 1. The pseudo code is as follows.

To avoid the appearance of mutual guarantees, MG-Tree is generated based on the maximum guarantee principle, i.e., the sum of the guarantee values of

Algorithm 1 Guarantee tree generation algorithm

```
Input: $OtherNodeID$ $TrustValue$ $LeaderSum$ $LeaderTerm$ $Behavior$ $GuarantMatrix$
Output: $TrustValue$
1: for $(id = 0;\ id < len(OtherNodeId);\ id++)$ do
2:     $Punishment[id] = -(2 * TrustValue[id]/PI) * (arcsin(Behavior[id]/LeaderSum * LeaderTerm))$
3:     $Reward[id] = (LeaderSum * LeaderTerm - Behavior[id])/(LeaderSum/LeaderTerm)$
4:     $UpdateTrustValue[id] = -Punishment[id] + Reward[id]$
5: end for
6: for $i = 0;\ i < len(OtherNodeId);\ i++$ do
7:     for $j = 0;\ j < len(OtherNodeId);\ j++$ do
8:         if $GuarantMatrix[i][j]/textgreatermax\&i! = j$ then
9:             $s = jandMax = GuarantMatrix[i][j]$
10:        end if
11:    end for
12: end for
13: vis[s] = true;
14: for $(i = 0;\ i < len(OtherNodeId);\ i++)$ do
15:     for $(j = 0;\ j < len(OtherNodeId);\ j++)$ do
16:         for $(k = 0;\ k < len(OtherNodeId);\ k++)$ do
17:             if $GuarantMatrix[j][k]{>}max\&vis[j] == false\&vis[k] == true\&j! = k$ then
18:                 $p = j$
19:                 $q = k$
20:                 $max = GuarantMatrix[j][k]$
21:             end if
22:         end for
23:     end for
24: end for
25: $GuarantTree[p][q] = GuarantMatrix[p][q]$
26: $vis[p] = true$
27: $GuarantRisk[p] = 1/(1 + exp(-GuarantTree[p][q]/Max))$
28: $UpdateTrustValue[p] = UpdateTrustValue[p] + (GuarantRisk[p] * UpdateTrustValue[p])$
29: $TrustValue = TrustValue + UpdateTrustValue$
30: return $TrustValue$;
```

all nodes accessed is guaranteed to be the maximum, instead of guaranteeing the maximum guarantee value of each node accessed. i.e., each node has the possibility to guarantee a node that is not the node with the maximum guarantee value. The reliable node selection strategy determines the reliability of each node by the guarantee value made by the node before consensus and the behavior of the node in the consensus phase, which is called Trust_Value.

If the guaranteed node is a consensus node and has no abnormal behavior in the consensus process, then both the guaranteed node and the guaranteed node will be rewarded with the corresponding Trust_Value; on the contrary, if the node has abnormal behavior, then a chain effect will be formed and the abnormal node and its children will be punished by reducing their corresponding Trust_Value. If a node behaves abnormally, a chain effect is formed, and both the abnormal node and its children will be penalized by reducing its Trust_Value. If the guaranteed node is only a normal consensus node in the consensus cycle, guaranteed nodes are rewarded with Trust_Value.

The change in Trust_Value of node A is shown in Equation 9:

$$T_A = T_A + F_A * G_B + G_A \tag{9}$$

Where, T_A is Trust_Value of node A, F_A is the guarantee function, G_B is the influence function of A's guarantee node B, and G_A is the influence function of node A's own behavior on the Trust_Value during the consensus process, the model considers Trust_Value change from two directions: node itself and guarantee node. For the newly joined nodes, Trust_Value will be initialized, and the nodes whose Trust_Value drops to 0 will be relegated to observer nodes, and to prevent the nodes from growing without limit, Trust_Value reward for the nodes with high Trust_Value will be reduced and the penalty will be increased. The guarantee function for Trust_Value of node is shown in Equation 10:

$$F_A = \frac{1}{1 + e^(- P/MAX)} \tag{10}$$

Where P denotes the guarantee value given by node A to the previous node in the maximum guarantee spanning tree, and MAX denotes the upper limit of the guarantee value. In this formula, the larger the value of P, the larger the F_A, while controlling the size of F_A between 0.5 and 0.75 can solve the problem of reducing the usability of the model due to the implication of too large guarantee weight or too small guarantee weight.

The node A's own consensus behavior affects the function G_A as shown in Equation 11:

$$G_A = R_A - P_A \tag{11}$$

Where R_A represents the normal consensus reward for consensus set nodes and P_A represents the penalty function for downtime and malicious behavior of consensus set nodes during the consensus process.

The node's Trust_Value reward function R_A is shown in Equation 12:

$$R_A = \frac{\sum_{x=1}^{V} \sigma_A}{E_A * V} \tag{12}$$

To ensure the balance of node's Trust_Value, the reward function needs to ensure that the normal consensus node's Trust_Value will not grow indefinitely. So the design is limited by the number of consensus cycle rounds that nodes

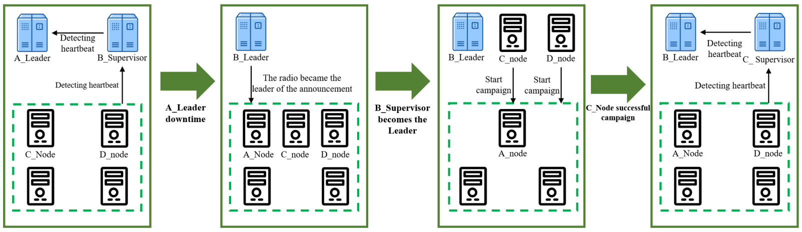


Fig. 3 Leader rotation process in the Dual-Leaders supervision mechanism

participate in to limit Trust_Value growth, the more nodes participate in consensus conversely the lower node's Trust_Value reward. The node's Trust_Value penalty function P_A is shown in Equation 13:

$$P_A = -\frac{2 * T_A}{\pi} * arcsin(\frac{\sum_{x=1}^{V} \rho_A}{V}) - (e^(\sum_{x=1}^{V} \rho_A L - 1)) \tag{13}$$

If a single or a small number of consensus nodes are down or malicious behavior has a small impact on the consensus process, ARCSIN function will increase the penalty according to the number of node problems, and to control the node's Trust_Value, it will be punished according to the percentage of existing Trust_Value, so the higher Trust_Value, the more Trust_Value will be deducted when there is a problem. If the node shows abnormal behavior in the whole consensus cycle, Trust_Value will be cleared 0; if leader fails to have a greater impact on the consensus process, leader rotation will slow down the consensus efficiency, so the penalty of its Trust_Value is larger when doing evil to leader, and monitor will increase exponentially with the number of consensus.

The reliable node selection strategy is based on the principle of high penalty and low reward, and the cost of node failure or evil is high. In the design of the node evaluation algorithm, in order to prevent misjudgment that normal nodes are wrongly identified as malicious nodes, only nodes with frequent abnormalities in a single consensus cycle will be demoted due to Trust_Value clearing 0. As long as the node's Trust_Value is not clear 0, it still has the right to participate in the guarantee and consensus as a consensus node.

4.2 Dual-Leaders supervision mechanism

Leader plays a central role in the consensus mechanism, and is responsible for sending and receiving transactions and determining the consensus behavior of nodes in GT-NRSM, if leader is down and leader rotation occurs, it will seriously affect the running process of the consensus algorithm. In order to increase the fault tolerance of the consensus algorithm, we propose a reliable and efficient Dual-Leaders supervision mechanism. As shown in Figure 3,the following describes Dual-Leaders supervision mechanism's supervision and rotation process.

After the consensus node with index value ID_k receives Trust_Value list S_L from monitor, it first updates Trust_Value list S_{L_j} of the shard nodes and sorts them, and the two nodes with the highest Trust_Value broadcast the node campaign request $VoteReq =< State, ID_k, LastLog_k, term_k, S_{v_k} >$ in the shard. where state is a binary value, with leader and deputy representing the primary node campaign type, indexed by ID_k after receiving VoteReq, the message is first verified as follows.

$$\begin{cases} ID_k \in S_{L_j} \\ LastLog_k \geq LastLog_j \\ term_k \geq term_j \\ S_{v_k} \geq S_{v_j} \end{cases} \tag{14}$$

The validation process is shown in Equation 14, if the validation does not pass, it returns $False$ signal representing against the vote, if it passes, it returns $True$ signal, if node receives the same type of leader request initiated by multiple nodes at the same time, after each campaign node has passed the validation, it compares the node's Trust_Value and returns $True$ signal for the node with the maximum Trust_Value, while the other nodes return $False$ signal. If the campaign node receives more than 1/3 affirmative votes, it initiates a leader campaign success notification to the replica nodes within shard as well as monitor. Inside the shard, when the consensus node receives the notification, it modifies local leader index and returns True_Value to leader to indicate that it has accepted the fact that the other party is leader. After monitor receives $2 * Shard\_Num$ leader campaign success notifications, it returns the client Leader_List.

In the consensus process, deputy within the shard will regularly receive the heartbeat from leader of sending and receiving transactions, as well as periodically broadcast the heartbeat to replica nodes within shard. If deputy does not receive the heartbeat from the leader within t time, it will initiate the notification to the replica nodes within the shard and the client of its own election as leader. After receiving the notification, the replica nodes will first verify whether the if not, it returns $False$ signal, if yes, it returns $True$ directly, and the client updates Leader_List after receiving the request for node replacement. If replica node does not receive the heartbeat request from deputy within t time, it broadcasts deputy node campaign request within the shard, if it receives more than 1/3 node consensus request, the node broadcasts the notification of elected deputy, after the replica node receives the notification, it modifies the local deputy's index and returns $True$ signal.

4.3 Consensus partitioning model

To solve the problem of low scalability of consensus algorithm in general, this paper put forward a consensus partitioning model based on guarantee mechanism and dynamic weights. Shard idea is one of the most effective means to enhance the consensus algorithm at present, the throughput is limited by

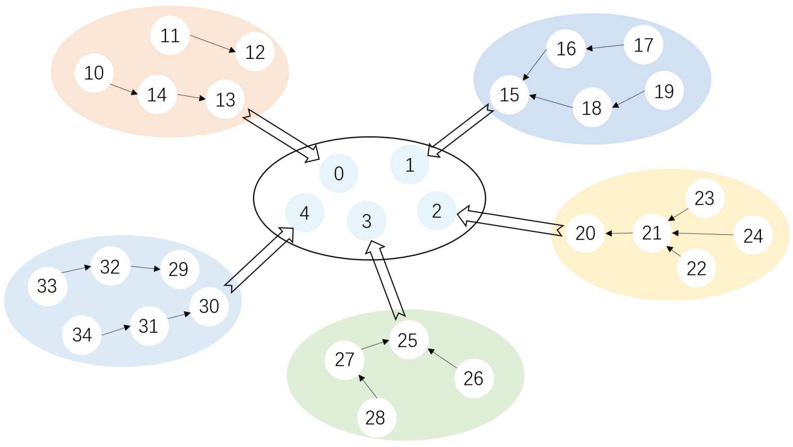


Fig. 4 Consensus partitioning model

network bandwidth and communication delay, the blockchain is adding new blocks, it must ensure that the last block has got a certain synchronization rate in the whole network. In the process of transaction verification and execution, leader must perform state machine replication with every consensus node in the whole network, and each consensus node must store the state associated with every account and every DAPP in the whole network in memory for access during transaction validation. The "Impossible Triangle" theory of blockchain suggests that the most effective way to obtain a design paradigm with high capacity and high throughput rate is to scale horizontally. Figure 4 illustrates the network partitioning model designed in this paper. Therefore, this paper proposes a consensus partitioning mode based on node guarantees and dynamic weights. The algorithm follows the principle of maximum credible guarantee within shard while ensuring a balanced number of nodes in the shard. The algorithm selects the node with the highest Trust_Value according to the number of shards by the reliable node selection algorithm and puts it into the shard as the initial node, and then invokes the guarantee mechanism to build a tree topology by linking the guarantee values in each shard.

First, we designed the second guarantee mechanism, pseudo code as shown in the figure, assuming that the number of replica nodes in the network is N and the number of shards is N_S, as shown in the figure to M matrix example in the network in the condition of the provision of two shards sub-tree generation process. After monitor finishes updating Trust_Value, N_S nodes are selected in descending order and put in each shard respectively. Finally, in this matrix, assuming the number of nodes is N and the number of shards is N_S, monitor updates Trust_Value and selects 2*N_S nodes in descending order and puts them in each shard respectively. In the guaranteed value adjacency matrix generated by consensus node with N_S nodes with the highest Trust_Value as the root of the tree, establish the graph topology sequence, link the consensus node with the largest guaranteed subtree root to the corresponding sub-tree, and sequentially find the node with the largest guaranteed value for nodes on all subtrees to link to the tree, and the number of nodes in each shard shall not exceed (N/N_S)+1 If the number of nodes exceeds this value, the shard is full and no more node access is received, finally forming a maximum guaranteed directed tree with all nodes except the root node with degree 1. As shown in Figure 5, the M matrix is used as an example to specify the generation process when the number of shard is 2.

Algorithm 2 Reliable node selection algorithm

```
Input: $OtherNodeID\ LeaderList\ TrustValue\ GuarantMatrix$
Output: $ShardList$
1: for $(id = 0;\ id < len(OtherNodeId);\ id++)$ do
2:     $ShardList[id] = -1$
3: end for
4: for $i = 0;\ i < len(LeaderList);\ i++$ do
5:     $ShardList[id] = i$
6: end for
7: for $j = 0;\ j < len(OtherNodeId - LeaderList);\ j++$ do
8:     $max = 0$
9:     $p = -1$
10:    $q = -1$
11:    for $j = 0;\ j < len(OtherNodeId);\ j++$ do
12:        for $k = 0;\ k < len(OtherNodeId);\ k++$ do
13:            if $GuarantMatrix[j][k] > max\ \&\ ShardList[j] == -1\ \&$
                 $ShardList[k]! = -1\ \&\ len(OtherNodeId)/len(LeaderList) > via[ShardList[k]]$
               then
14:                $p = j$
15:                $q = k$
16:                $max = GuarantMatrix[j][k]$
17:            end if
18:        end for
19:    end for
20:    if $p == -1$ then
21:        $break$
22:    end if
23:    $ShardList[p] = ShardList[q]$
24:    $via[ShardList[q]] += 1$
25: end for
26: return $ShardList$;
```

4.4 GT-NRSM secure and scalable consensus algorithm

The algorithm applies reliable node selection strategy, Dual-Leaders supervision mechanism, and consensus partitioning model, which solves the problem

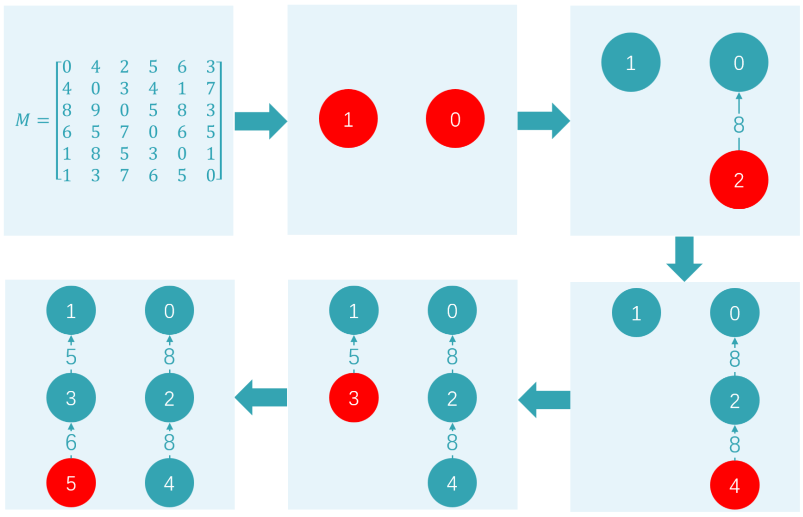


Fig. 5 The process of shard based on the guarantee mechanism

of unreliable consensus nodes and inefficient consensus, and the time complexity of the algorithm is O(N). It supports node activity detection as well as node fault tolerance. First, we describe the execution logic of each node, and then analyze the security, activity and complexity of the algorithm. The flowchart of the algorithm is shown in Figure 6. The algorithm includes four phases, Request: Feedback, Commit, and Reply, where Request and Feedback are executed before the consensus cycle starts, and Commit and Reply will be executed cyclically in the cycle. monitor, leader, and consensus nodes in the figure can represent different roles of nodes.

As shown in Figure 7,the detailed steps for each stage are as follows:

- Request phase: we assume that the current state consensus cycle is R_i, in this phase, first the client sends $Request_1 < LeaderReq = True >$ to monitor to request for Leader_List, monitor receives the signal and starts to call the node based on the obtained consensus behavior state and the guarantee tree generated by the R_{i-1} cycle The selection policy updates the node Trust_Value, selects Leader_List of R_i cycle and malicious nodes based on updated Trust_Value, stores the list in the cache and initiates the request for guarantee value generation broadcast $Request_2 < ID_m, R_i, B_i, M_i, GuaranteeReq = True >$, sets the timeout time T_r, and to reduce the communication consumption, we compressed the communication flow by broadcasting Leader_List as well as the malicious node broadcast simultaneously with the guarantee request.
- Feedback phase: After receiving $Request_2$ message, node N_j verifies and processes the message in three steps, 1) determine whether ID_m is correct and whether R_i is greater than the latest consensus cycle R_j that the node is in, if ID_m is less than the locally saved ID_m^j then return an error processing

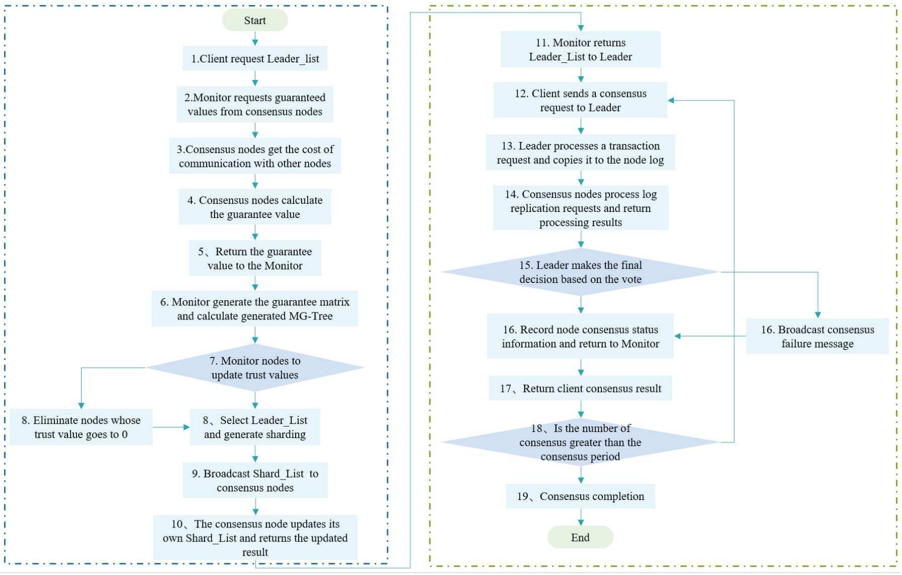

Start
1.Client request Leader_list
2.Monitor requests guaranteed values from consensus nodes
3.Consensus nodes get the cost of communication with other nodes
4. Consensus nodes calculate the guarantee value
5、Return the guarantee value to the Monitor
6. Monitor generate the guarantee matrix and calculate generated MG-Tree
7. Monitor nodes to update trust values
8. Eliminate nodes whose trust value goes to 0
8、Select Leader_List and generate sharding
9. Broadcast Shard_List to consensus nodes
10、The consensus node updates its own Shard_List and returns the updated result
11. Monitor returns Leader_List to Leader
12. Client sends a consensus request to Leader
13. Leader processes a transaction request and copies it to the node log
14. Consensus nodes process log replication requests and return processing results
15. Leader makes the final decision based on the vote
16. Record node consensus status information and return to Monitor
16. Broadcast consensus failure message
17、Return client consensus result
18、Is the number of consensus greater than the consensus period
19、Consensus completion
End

Fig. 6 GT-NEFM algorithm consensus flow

request $Feedback_1 < GuaranteeRespond = False, ErrorID, ErrorT_i >$ eliminate malicious nodes and update the local consensus node list according to the malicious node index provided by monitor, 2) establish a link with each node by transmitting the true signal, test the link length as the communication cost, and generate a consensus value of the node based on the communication cost and the consensus behavior of the node to other nodes guarantee value, 3) return monitor ID_j and guarantee value list $Feedback_1 < GuaranteeRespond = False, ID_j, Gu_i^j >$. After receiving $Feedback_1$ message from ID_j, monitor deposits the guarantee value into the guarantee matrix Gu_i in the cache pool according to the node index, and after reaching T_r time, monitor stops receiving $Feedback_1$ message, does the guarantee tree generation algorithm on the guarantee matrix, puts the guarantee tree into the cache, and is called before R_{i+1} starts. Call shard algorithm to generate the final Shard_List by Leader_List and the guarantee matrix, and broadcast the shard $ShardReq < ID_m, S_L, T_V >$ to the consensus node list concurrently, after the consensus node receives the ShardReq, it will update its own shard listS_L_j and the shard's node Trust_Value list T_V_j, and return the client Leader_List : $Feedback_2 < ID_m, Le_i >$. After the consensus node receives Shard_List, it updates its own Shard_List and modifies leader index.

- Commit and Reply phase: After receiving $Feedback_2$, the client takes the transaction from the transaction pool, sorts the transaction and starts to concurrently distribute the transaction to the incoming and outgoing transaction nodes in the shard, leader receives the transaction set, parses the transaction and generates the block, sends the transaction to the replica

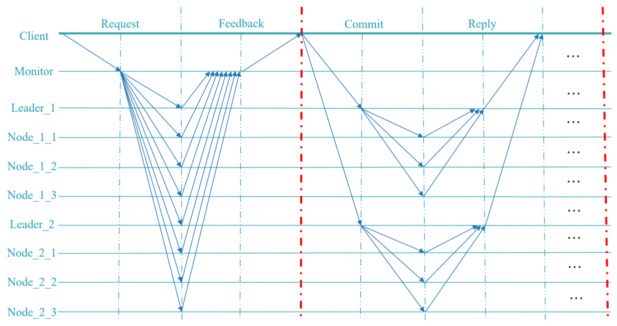


Fig. 7 GT-NRSM process main communication flow

node in the shard. Reply stage, after receiving the transaction, the node verifies the transaction, if the block passes, it returns True, if the verification fails, it returns False, if leader receives more than f+1 passed votes, it means the block is legal, if it receives no more than f+1 passed votes or no more than f+1 failed votes, it means the block is not legal and discards the block. After leader makes the final decision, it counts the replica node votes, records the replica node downtime in the shard and whether it is inconsistent with the final decision, sends the node consensus behavior to monitor, which records it and finally returns the decision to the client. After receiving the return message from all leaders, the client determines whether the consensus count reaches the upper limit of consensus cycle, if not, it continues to repeat the above steps, and if it reaches the upper limit, it requests a new round of Leader_List from monitor.

5 Experiments and results analysis

5.1 Experimental platform and experimental environment

In this section, we analyze GT-NRSM from four aspects: throughput comparison, malicious node identification, leader rotation delay, and shard experiment. The specific go language is used to build, the remote call is implemented by GRPC, the data persistence is implemented by MySQL, the block generation interval and the node sending heartbeat time are reasonable values.

5.2 TPS comparative analysis

Transaction throughput is an important performance metric for distributed systems. The throughput of a system is usually determined by the number of concurrent transactions and the number of transactions per second (TPS), as shown in Equation 15. We use it to test and compare the performance of

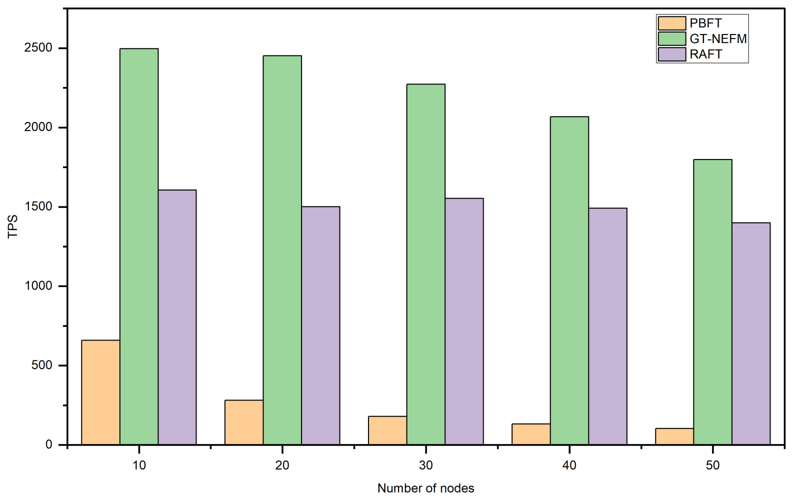


Fig. 8 Comparison of the average TPS of different consensus algorithms

several algorithms.

$$TPS = \frac{Tran-len}{\triangle t} \tag{15}$$

Where Tran-len is the number of transactions contained in a block, $1T$ is the generation time of a block, and the GT-NRSM protocol requires Trust_Value updates and partitioning outside the consensus cycle, so the communication consumption outside the consensus cycle is calculated in the consensus TPS in the experiment.

$$1T = Time_{consensus} + TrustValue_{update} + Time_{Shard} \tag{16}$$

Due to the different consensus algorithms, the intervals between blocks are also different. To ensure the accuracy of the experiment and the stability of the system, we specify consensus on 100 blocks * 1000 simulated transaction data. After the test, the transaction time is calculated uniformly. We conducted the experiment by fixing the amount of data and changing the number of nodes, and the experimental results are shown in Figure 8. The experimental results show that both GT-NRSM and Raft have high throughput for the same data volume and different number of nodes, and GT-NRSM improves 48% over Raft in terms of throughput and is much higher than PBFT due to the concurrent consensus of sharding.

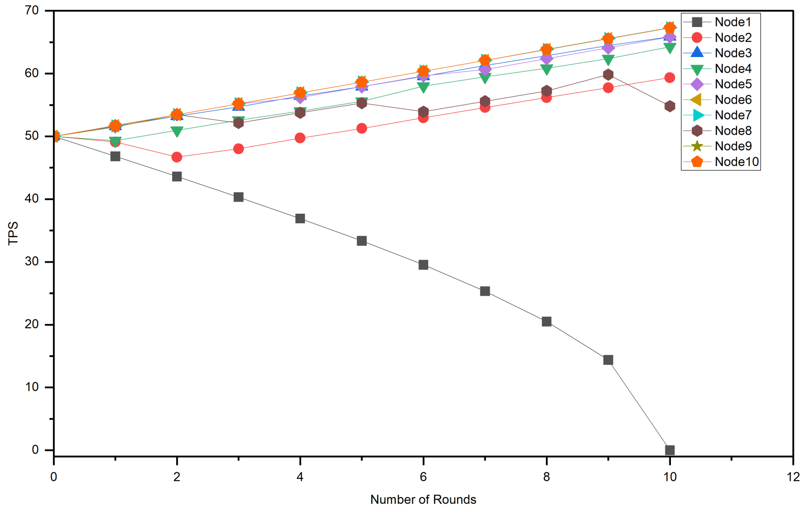


Fig. 9 Change of Trust_Value of 10 nodes

5.3 Malicious node identification analysis

The reliability of nodes is ensured by the reliable node election strategy in the GT-NRSM protocol, i.e., the nodes are judged to be malicious nodes based on their cumulative malicious consensus count, and when the reliability of a node is lower than a threshold value, the node is judged to be malicious and all consensus nodes are broadcasted to reject the value index, thus optimizing the network environment. In this experiment, 10 nodes are deployed for Trust_Value reward and punishment test and the number of consensus cycles is assumed to be 10. where node 1 is a continuous malicious node and node 8 is a discontinuous malicious node. The node Trust_Value variation is shown in Figure 9. It can be observed from the figure that the average Trust_Value of the nodes is increasing and the difference in Trust_Value is not significant, in which node 1 drops to 0 after successive rounds of malicious behavior and is judged by the algorithm as an abnormal node and removed from the network. Node 8 performs evil behavior in round 1, round 6 and round 10, respectively, and its penalty cost still increases accordingly; node 2 and node 4 do not perform evil behavior, but they wrongly vouch for malicious nodes, so the final Trust_Value is lower than most normal nodes. Overall the reliable node selection strategy can meet the requirements for malicious node identification as well as dividing the shard.

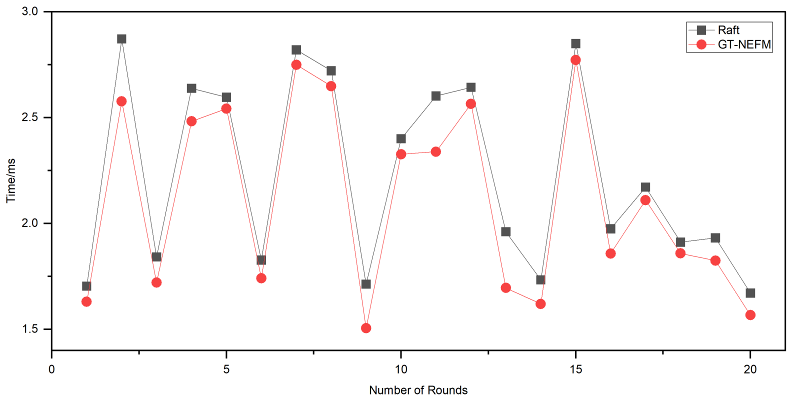
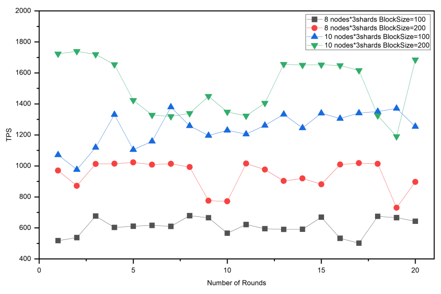


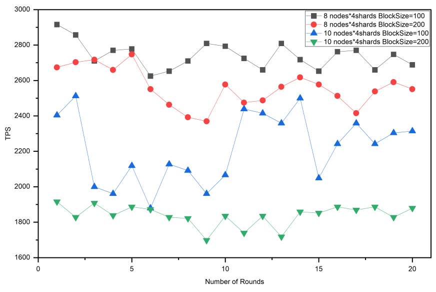
Fig. 10 Comparison of leader rotation delay for different consensus algorithms

5.4 Leader rotation latency comparison analysis

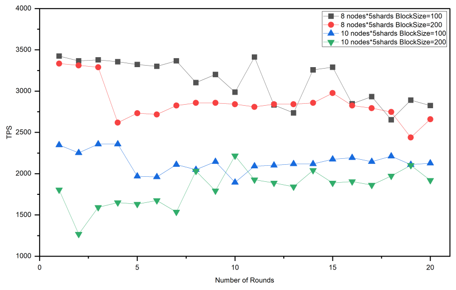
In the Raft consensus protocol, if the replica node does not receive the heartbeat of the incoming autonomous node, it will determine that it has failed and will adjust its identity status to that of a candidate and initiate a leader election request. GT-NRSM protocol applies a Dual-Leaders rotation supervision strategy to reduce the communication consumption of leader election, if the heartbeat of leader is detected by the supervisor node, the heartbeat of the supervisor node is detected by the replica node. If the heartbeat of leader is detected by the supervisor node, the heartbeat of the supervisor node is detected by the replica node. Since the time required for leader rotation is extremely short, in order to achieve an accurate comparison, we put leader rotation algorithm of Raft's algorithm and leader rotation algorithm of GT-NRSM of this paper into the same sample. We test both algorithms for 20 groups to calculate their time delays, and the experimental results are shown in Figure 10, where the horizontal coordinates represent the number of rounds and the vertical coordinates represent the time delays. The experimental results show that Dual-Leaders supervision mechanism of GT-NRSM outperforms Raft algorithm in terms of running time. GT-NRSM can achieve the principal node turnover in a shorter time after the downtime of leader and reduce the impact on the consensus process.



(a) 3 shards



(b) 4 shards



(c) 5 shards

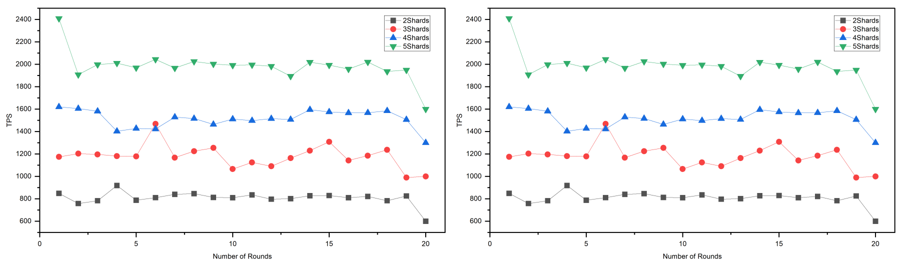
Fig. 11 Comparison of TPS for 3 (left), 4 (right), and 5 (bottom) shards

5.5 Comparative analysis of the effect of shard

Transactions in the fast confirmation network are collected by slicing as transaction blocks. Compared to the longer wait time required for the confirmation of a Bitcoin transaction block, the shard can also reach consensus on the transaction block and add it to the blockchain in a shorter period of time. As a result, the transactions contained in the transaction block are also confirmed.

5.5.1 Vertical scalability analysis

The common problem of current mainstream consensus algorithms is that the consensus process requires a large number of node communications. the GT-NRSM algorithm reduces the number of consensus node communications in a consensus process, but inter-node communications still consume a lot of resources. To test the impact of the number of nodes on the performance of the blockchain system, we test TPS variation of the number of consistent nodes within a shard with different block sizes with a constant number of shards, as shown in Figure 11, where the horizontal coordinates indicate the number of rounds and the vertical coordinates indicate the TPS. It can be found that with a fixed number of shards, TPS of GT-NRSM is lower as the number of nodes



(a) 40 Nodes (a) 50 Nodes

Fig. 12 TPS comparison of 40 (left) and 50 (right) nodes

increases, which is because in GT-NRSM, each additional node increases the thread communication of leader sending log preparation requests and collecting voting information. So the higher the number of nodes in the shard, the longer the time to maintain inter-node consistency, the higher the communication consumption with leader, and the longer the time required in the consensus process.

5.5.2 Horizontal scalability analysis

In this paper, GT-NRSM improves the performance of blockchain systems by dividing consistent resources. To test the horizontal scalability of GT-NRSM, we test the relationship between the number of shards and TPS for different number of nodes when the block size is 1000. As shown in Figure 12, the horizontal coordinates indicate the number of rounds and the vertical coordinates indicate the TPS. It can be found that the consensus speed of GT-NRSM increases as the number of shards increases, while TPS increases. However, the increase becomes slower and slower, and TPS tends to be stable. This is because while keeping the number of nodes within a shard constant, each additional shard adds a concurrent consensus process to process the data, thus increasing the consensus speed of the blockchain system in the consensus phase. As the number of shards increases, the time required to monitor to receive consensus behavior, execute the shard algorithm, etc. also increases. So the growth of TPS becomes smaller and smaller until it is flat. Therefore, with the constant amount of consensus data, the overall consensus time cost decreases as the number of shards increases, and TPS of GT-NRSM protocol increases until it stabilizes.

5.6 Block scalability analysis

In the GT-NRSM protocol, the number of data entries in a block can be configured according to business needs in order to optimize the efficiency of the

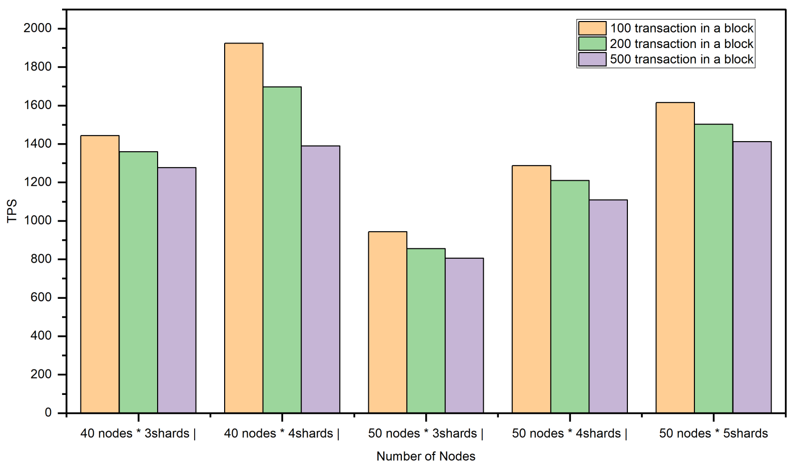


Fig. 13 Comparison of TPS for different number of transactions within a block

blockchain system. In order to study the impact of block size on the processing speed of the blockchain system, the scalability of blocks was tested and analyzed. Specifically, we tested the processing speed of different transaction volumes in a block under different conditions. Twenty rounds of testing were conducted for each block size under different conditions. TPS was averaged, and the value of the error bar is the standard deviation value of the throughput of the 20 blocks, as shown in Figure 13. The horizontal coordinates indicate the different numbers of nodes and shards, and the vertical coordinates indicate TPS.

6 Conclusion and future work

The scalability issue of consensus mechanism is currently the main obstacle to the implementation of coalition chain applications. In this study, we propose the GT-NRSM to solve the current problems of low blockchain scalability, poor node reliability and balanced quadrilaterality. GT-NRSM is a consensus protocol mainly for reliability evaluation and consensus network slicing applied to coalition chains. Firstly, we introduce the concept of guarantee and build the node evaluation model and network shard model based on the guarantee mechanism to ensure the reliability of shard while achieving the expansion of the blockchain. Secondly, we propose leader supervision mechanism to effectively reduce the rotation delay caused by leader downtime and reduce the impact of leader downtime on consensus process. The experimental results show that GT-NRSM has higher fault tolerance, faster consensus efficiency, greater system throughput and better scalability, and has obvious advantages over the applied consensus algorithms. GT-NRSM cannot effectively identify many types of network adversary attacks, and in future research, we plan to specifically investigate how to further study the shardng model using artificial intelligence techniques to deepen the network sharding scheme. In addition, an optimized scheme will be designed to determine the optimal block size and interval to improve consensus efficiency through deep reinforcement learning.

References

[1] Makhdoom, I., Abolhasan, M., Abbas, H., Ni, W.: Blockchain's adoption in iot: The challenges, and a way forward. Journal of Network and Computer Applications **125**, 251–279 (2019). https://doi.org/10.1016/j.jnca.2018.10.019

[2] Kalodner, H., Goldfeder, S., Chen, X., Weinberg, S.M., Felten, E.W.: Arbitrum: Scalable, private smart contracts. In: Proceedings of the 27th USENIX Conference on Security Symposium. SEC'18, pp. 1353–1370. USENIX Association, USA (2018)

[3] Liu, Y., He, D., Obaidat, M.S., Kumar, N., Khan, M.K., Raymond Choo, K.-K.: Blockchain-based identity management systems: A review. Journal of Network and Computer Applications **166**, 102731 (2020). https://doi.org/10.1016/j.jnca.2020.102731

[4] Singh, A., Click, K., Parizi, R.M., Zhang, Q., Dehghantanha, A., Choo, K.-K.R.: Sidechain technologies in blockchain networks: An examination and state-of-the-art review. Journal of Network and Computer Applications **149**, 102471 (2020). https://doi.org/10.1016/j.jnca.2019.102471

[5] Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. Cryptology ePrint Archive, Paper 2018/472. https://eprint.iacr.org/2018/472 (2018). https://eprint.iacr.org/2018/472

[6] Khalil, R., Gervais, A.: Nocust-a non-custodial 2nd-layer financial intermediary. (2018)

[7] Kwon, J., Buchman, E.: A network of distributed ledgers. Cosmos, dated, 1–41 (2018)

[8] Rubin, J., Naik, M., Subramanian, N.: Merkelized abstract syntax trees. XP055624837, Dec **16**, 3 (2014)

[9] Sompolinsky, Y., Wyborski, S., Zohar, A.: Phantom ghostdag: a scalable generalization of nakamoto consensus: September 2, 2021. In: Proceedings of the 3rd ACM Conference on Advances in Financial Technologies, pp.

57–70 (2021)

[10] Anjana, P.S., Kumari, S., Peri, S., Rathor, S., Somani, A.: An efficient framework for optimistic concurrent execution of smart contracts. In: 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 83–92 (2019). IEEE

[11] Michael, M., Mills, S.: The missing links in the chains? mutual distributed ledger (aka blockchain) standards. Long Finance, SSRN Electron. J., 1–75 (2016)

[12] Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Annual International Cryptology Conference, pp. 139–147 (1992). Springer

[13] Bahri, L., Girdzijauskas, S.: When trust saves energy: a reference framework for proof of trust (pot) blockchains. In: Companion Proceedings of the The Web Conference 2018, pp. 1165–1169 (2018)

[14] Cheng, Y., Hu, X., Zhang, J.: An improved scheme of proof-of-stake consensus mechanism. In: 2019 4th International Conference on Mechanical, Control and Computer Engineering (ICMCCE), pp. 826–8263 (2019). IEEE

[15] Huang, D., *et al.*: Rbft: byzantine fault-tolerant consensus mechanism based on raft cluster. J. Commun **42**(03), 209–219 (2021)

[16] Gao, S., Yu, T., Zhu, J., Cai, W.: T-pbft: An eigentrust-based practical byzantine fault tolerance consensus algorithm. China Communications **16**(12), 111–123 (2019)

[17] Fu, W., Wei, X., Tong, S.: An improved blockchain consensus algorithm based on raft. Arabian Journal for Science and Engineering **46**(9), 8137–8149 (2021)

[18] Cao, B., Zhang, Z., Feng, D., Zhang, S., Zhang, L., Peng, M., Li, Y.: Performance analysis and comparison of pow, pos and dag based blockchains. Digital Communications and Networks **6**(4), 480–485 (2020)

[19] Saad, S.M.S., Radzi, R.Z.R.M.: Comparative review of the blockchain consensus algorithm between proof of stake (pos) and delegated proof of stake (dpos). International Journal of Innovative Computing **10**(2) (2020)

[20] Li, W., Feng, C., Zhang, L., Xu, H., Cao, B., Imran, M.A.: A scalable multi-layer pbft consensus for blockchain. IEEE Transactions on Parallel and Distributed Systems **32**(5), 1146–1160 (2020)

[21] Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th International Conference on World Wide Web, pp. 640–651 (2003)

[22] Xiong, L., Liu, L.: Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE transactions on Knowledge and Data Engineering **16**(7), 843–857 (2004)

[23] Song, Y., Sun, C., Peng, Y., Zeng, Y., Sun, B.: Research on multi-dimensional trust evaluation mechanism of fintech based on blockchain. IEEE Access (2022)

[24] Pal, S., Hill, A., Rabehaja, T., Hitchens, M.: Veriblock: A blockchain-based verifiable trust management architecture with provable interactions. In: 2022 International Conference on Computer Communications and Networks (ICCCN), pp. 1–7 (2022). IEEE

[25] Zhang, P., Zhou, M., Zhen, J., Zhang, J.: Enhancing scalability of trusted blockchains through optimal sharding. In: 2021 IEEE International Conference on Smart Data Services (SMDS), pp. 226–233 (2021). IEEE

[26] Lahbib, A., Toumi, K., Laouiti, A., Laube, A., Martin, S.: Blockchain based trust management mechanism for iot. In: 2019 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–8 (2019). IEEE

[27] Shala, B., Trick, U., Lehmann, A., Ghita, B., Shiaeles, S.: Blockchain-based trust communities for decentralized m2m application services. In: International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 62–73 (2018). Springer

[28] Sel, D., Zhang, K., Jacobsen, H.-A.: Towards solving the data availability problem for sharded ethereum. In: Proceedings of the 2Nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, pp. 25–30 (2018)

[29] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30 (2016)

[30] Zamani, M., Movahedi, M., Raykova, M.: Rapidchain: Scaling blockchain via full sharding. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 931–948 (2018)

[31] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford,

B.: Omniledger: A secure, scale-out, decentralized ledger via sharding. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598 (2018). IEEE

[32] Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., Danezis, G.: Chainspace: A sharded smart contracts platform. arXiv preprint arXiv:1708.03778 (2017)

[33] Dang, H., Dinh, T.T.A., Loghin, D., Chang, E.-C., Lin, Q., Ooi, B.C.: Towards scaling blockchain systems via sharding. In: Proceedings of the 2019 International Conference on Management of Data, pp. 123–140 (2019)

[34] Kwak, J.-Y., Yim, J., Ko, N.-S., Kim, S.-M.: The design of hierarchical consensus mechanism based on service-zone sharding. IEEE Transactions on Engineering Management **67**(4), 1387–1403 (2020)

[35] Abuidris, Y., Kumar, R., Yang, T., Onginjo, J.: Secure large-scale e-voting system based on blockchain contract using a hybrid consensus model combined with sharding. Etri Journal **43**(2), 357–370 (2021)

[36] Sun, Y., Xue, R., Zhang, R., Su, Q., Gao, S.: Rtchain: A reputation system with transaction and consensus incentives for e-commerce blockchain. ACM Transactions on Internet Technology (TOIT) **21**(1), 1–24 (2020)

[37] Tan, P., Zou, W., Tang, W.: A consensus algorithm with leadership transfer-ltraft. In: China Conference on Wireless Sensor Networks, pp. 235–249 (2021). Springer

Declarations

Funding This work was supported in part by the Major Scientific and Technological Projects in Yunnan Province under Grant 202002AB080001-8; in part by the Yunnan Key Laboratory of Blockchain Application Technology under Grant 202105AG070005 and Project YNB202109 and YNB202115; in part by the National Natural Science Foundation of China under Grant 61971208; in part by the Yunnan Reserve Talents of Young and Middle-Aged Academic and Technical Leaders (Shen Tao) under Grant 2019HB005; in part by the Yunnan Young Top Talents of Ten thousand Plan (Shen Tao, Zhu Yan, Yunren Social Development) under Grant 2018 73; and in part by the Scientific Research Fund Project of Yunnan Provincial Department of Education under Grant 2022Y160.

Competing Interests The authors have no relevant financial or non-financial interests to disclose.

Author Contributions All authors contributed to the study conception and design. TS and TL did the ideal development and most of the implementation and evaluation. The first draft of the manuscript was written by TL. ZY and CZ contributed to the technical discussion as well as contributed to the manuscript writing and proofreading. As the corresponding author, FB supervised the entire process from idea development to implementation, experimentation and evaluation, and paper writing.

Data Availability Available upon request.

Conflict of interest The authors declare that they have no conflicts of interest.

Ethical approval This material has not being published in whole or in part elsewhere.

Research invovling human and/or animal participants This study does not involve human participants and/or animal research, which is used for non-life science journals.