The Journal of Supercomputing (2022) 78:4850–4881
https://doi.org/10.1007/s11227-021-04061-3

Scalable blockchain storage mechanism based on two-layer structure and improved distributed consensus

Chunlin Li[1,2] · Jing Zhang[2] · Xianmin Yang[3]

Accepted: 26 August 2021 / Published online: 10 September 2021

Abstract
Existing public blockchain architectures suffer from the difficulty of scaling to support large-scale networks with high TPS, low latency and security. Using the idea of network fragmentation, an improved Raft-based PBFT consensus mechanism is proposed to solve the problem. The network nodes are grouped, and the group adopts the improved Raft mechanism for consensus, and then, the leaders elected in each group form the network committee, and the network committee uses the PBFT mechanism for consensus within the network committee. The results show that R-PBFT is more scalable than PBFT and Raft in a large-scale network environment because it can guarantee high consensus efficiency while having Byzantine fault tolerance; similarly, to improve the fairness between user experience and TPS in blockchain systems, based on the transaction fairness model, a fairness packing algorithm is proposed for storing transactions. It is based on a two-level model, firstly sorting the transactions in descending order based on GasPrice, moreover considering the fairness model for descending order. The experimental confirmed that all the performance of fairness packing is superior to Ethereum packing.

Keywords Public blockchain · Consensus mechanism · R-PBFT · Raft · Fairness model

✉ Chunlin Li
chunlin74@aliyun.com

1 CAAC Key Laboratory of Civil Aviation Wide Survellence and Safety Operation Management & Control Technology, Tianjin, China

2 School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430063, China

3 Data Recovery Key Laboratory of Sichuan Province, College of Mathematics and Information Science, Neijiang Normal University, Neijiang 641100, People's Republic of China

1 Introduction

Blockchain [1] is a decentralized, traceable, tamper-evident, distributed database maintained by multiple parties. Data are stored as distributed storage on multiple nodes. The core of blockchain includes consensus mechanism, distributed storage, cryptography and smart contracts. Blockchain generates data and updates data through a distributed node consensus algorithm and securing blocks based on cryptographic; blockchain programs and manipulates data through smart contracts automatically generated by script code; blockchain has gained widespread use in areas requiring credit authentication [2]. Among them, the consensus mechanism focuses on solving the consistency problem of distributed systems, aiming to ensure the consistency of data copies maintained by all nodes and avoid the occurrence of data disparity and information asymmetry problems. Different scenarios have different needs for the scalability, consensus efficiency and privacy of consensus mechanisms [3, 4].

Although blockchain has received widespread attention and is rapidly emerging, it is still in the development stage and therefore faces many challenges, such as performance and throughput issues (The average transaction TPS of bitcoin is 7 per second and confirmation delays of up to 1 h, and most blockchain systems are currently unable to meet the throughput requirements of centralized trading systems), energy consumption issues (proof-of-work blockchains often require extremely high arithmetic power consumption), security and convenience issues (news of key loss and theft of cryptocurrency wallets is endless), etc. Blockchain often requires extremely large amounts of arithmetic power consumption, security and convenience issues (cryptocurrency wallets are inundated with news of lost and stolen keys), etc. In addition, blockchain protocols need to make trade-offs and compromise among "scalability," "security" and "decentralization."

The main contributions of this paper are as follows:

(1) Existing public blockchain architectures suffer from the difficulty of scaling to support large-scale networks with high TPS, low latency and security. Based on PBFT and Raft consensus mechanism and introducing a consistent hash algorithm, a two-layer consensus mechanism, R-PBFT, is proposed to solve the problem. the outer layer groups nodes, the group elects Leaders as representatives and designs supervisory nodes to prevent malicious nodes; the inter-group forms a committee together through Leaders and passes consensus through Raft.

(2) In order to improve the fairness between user experience and TPS in blockchain systems, a fair packing algorithm for stored transactions is proposed based on the transaction fairness model by introducing Jain's fairness index [5]. This model first sorted the transactions in descending order based on GasPrice, and then in descending order based on fairness index. Through experiments, it has been experimentally demonstrated that the fairness packing algorithm largely outperforms the Ethereum packing algorithm for different transaction entry rates, block generation times, block sizes and transaction read rates.

(3) For the experimental demonstration of the proposed algorithm, a virtual blockchain environment was constructed through servers and computers. The complexity and diversity of node resources in a blockchain network are represented by the division of many virtual machines with different CPU cores and hard disks.

The rest sections of this paper are organized as follows: The related work is discussed in Sect. 2; A Raft-based improved Byzantine fault-tolerant consensus algorithm for public blockchains is proposed in Sect. 3; a fairness packing algorithm in the Ethernet based on Jain's fairness index is proposed in Sect. 4; the experimental environment and the performance of algorithms are discussed in; Sect. 5; the work of this paper and future work is concluded in Sect. 6.

2 Related work

With the continuous development of blockchain technology, how to use blockchain to replace existing distributed databases has become a hot issue [6]. Blockchain is still in its infancy of development, and although it is developing rapidly there are still many problems in terms of performance, smart contracts and consensus mechanisms.

Currently, Bitcoin [7] has a theoretical throughput of 7 TPS (Transaction per second). Ether [8] has a throughput of 25 TPS, which is not a significant improvement. Hyperledger fabric, a representative of the federated chains, achieves throughputs of more than 2000 TPS, but it is still not enough to replace existing practical solutions. Therefore, analyzing the performance bottlenecks of blockchain and performing targeted optimization has become one of the important directions of current blockchain research.

Blockbench, developed by TTA Dinh et al. at the National University of Singapore, is the first testing method to analyze the performance of private blockchain. Blockchain is abstracted into four layers: data layer, consensus layer, application layer and execution layer [9]. Blockbench evaluated the performance of Hyperledger Fabric and Parity in terms of throughput, scalability, latency and fault tolerance [10]. Thakkar et al. of the National Institute of Technology, India, experimented with Hyperledger Fabric [11] by parameter tuning and proposed some optimization methods accordingly. Gorenflo et al. at the University of Waterloo increased the throughput of Hyperledger Fabric from 3000 TPS to 20,000 TPS by changing the system architecture of the fabric [12]. Pongnumkul et al. at the National Electronics and Computer Technology Center in Thailand tested the performance of Ether and Hyperledger Fabric [13], but with a somewhat homogeneous selection of workload types. Rouhani et al. of the University of Saskatchewan compared the performance of two different Ether clients [14]. Li et al.'s resource management strategy based on cost and user experience in edge cloud environment is a feasible idea in blockchain environment [15, 16]. Ampel et al. at the University of Arizona experimentally identified a number of problems with Sawtooth that still need to be addressed [17]. Hao et al. from Beijing University of Posts and Telecommunications studied how the private blockchains are affected by consensus algorithms [18]. In addition, many other teams have studied the performance of blockchain systems [19–24].

The theory of smart contracts was first proposed in 1994 by cryptographer Nick Szab and defined smart contracts. Current research on the application areas and development prospects of smart contracts is more focused. For example, AmjadQashlan et al. proposed a smart contract model to ensure the security of IoT smart appliances for the application of smart contracts in home life [25]; Li et al. proposed a optimal task scheduling strategy can be obtained according to the Dijkstra shortest path algorithm based on Fibonacci heap [26, 27]; Shi-ChoCha et al. developed a data framework of security for blockchain smart contracts in smartphone information security and proposed countermeasures to establish a risk prevention and control model using blockchain smart contracts [28]; in addition, there are also studies on smart contract regulation such as Markus Knecht, Burkhard Stiller developed a platform for the deployment and management of smart contracts in response to the difficulty of managing smart contracts [29].

As for the consensus mechanism, the existing consensus mechanisms include PoS [30], PoW [31], DPoS [32] Practical Byzantine fault tolerance [33], Paxos [34], Raft [35], etc. Classical blockchain consensus algorithms have many drawbacks, and the most discussed one should be the scalability issue. Therefore, scholars have proposed many solutions, for example, Bitcoin-NG [36] and Byz-Coin [37]. The key block uses the classical PoW algorithm for leader election, and transactions are performed in microblocks. In addition, to prevent double-spending attacks, Bitcoin-NG introduces a "poison transaction" mechanism [38]. ByzCoin is a further improvement based on Bitcoin-NG and PBFT protocols. It separates the leader election and transaction confirmation in the original scheme, where the PoW algorithm remains in charge of conducting the leader election, while the transaction is immediately confirmed with the help of PBFT committee verification. In prototype experiments, ByzCoin achieved Paypal-quantity throughput with a confirmation wait time of 15–20 s [37]. However, they still associate PoW with leader elections and introduce other complex consensus mechanisms (e.g., BFT), which can significantly reduce their efficiency in extreme cases. Li et al. improved PoS by developing a distributed block system that is more adapted to the digital economy [39]; Karakostas et al. prevent attacks on nodes with weaker arithmetic in the PoW consensus by means of checkpointing and timestamping [40]; Howard et al. compared the advantages, disadvantages and differences between the Paxos and Raft algorithms in distributed consensus [40]. Practical Byzantine Fault Tolerance (PBFT) is one of the common blockchain consensus mechanisms [41]. Compared with it, Raft algorithm will be more efficient. Raft is commonly accepted and applied as a consensus algorithm in distributed systems. Its principle is briefly described as follows: each node must be in one of three states: follower, leader or candidate. A follower becomes the leader if it wins the leader election and continues to send heartbeats to other nodes after becoming the leader. If the follower does not receive a heartbeat within the minimum election timeout period, then it enters the leader election process. This process is the core of Raft algorithm and is called leader election. Raft algorithm is simplified from Paxos algorithm [42], which does not include leader election as an important element and is more decentralized and facilitates nodes to balance the load [43, 44]. However, the implementation of the Paxos algorithm requires a complex architecture and is difficult to apply in practice. Raft is widely used because its security is comparable to the Paxos algorithm and it is easy to understand and build the architecture.

In addition, blockchain-based distributed storage has gained widespread attention and application [45, 46], and it has been quite a reliable means of secure data storage. In terms of existing research results, Rizun theoretically analyzed the input costs and operational benefits of data nodes and empirically constructed the demand and supply curves of blockchain systems and their equilibrium states [47]. Li proposed a content placement method, which can effectively solve the content placement problem by considering cost and latency factors and have an inspiring effect on blockchain storage [48]; Moser and Bohome constructed a service processing priority model using queuing theory and empirically examined the effect of service cost on the waiting time for service requests [49].

3 An improved PBFT consensus mechanism based on Raft

3.1 Introduction to R-PBFT consensus mechanism

In order to provide a consensus algorithm with high scalability, low latency, high security and high TPS, this section proposes an improved PBFT consensus mechanism based on Raft —R-PBFT, by combining the security of PBFT with the advantages of high consensus efficiency of Raft. The schematic illustration is shown in Fig. 1.

R-PBFT consensus mechanism groups the blockchain network nodes, and the leaders are elected to form a committee using Raft within each group, and PBFT is used for consensus within the committee. In addition, supervisory nodes are introduced to ensure the security of the algorithm.

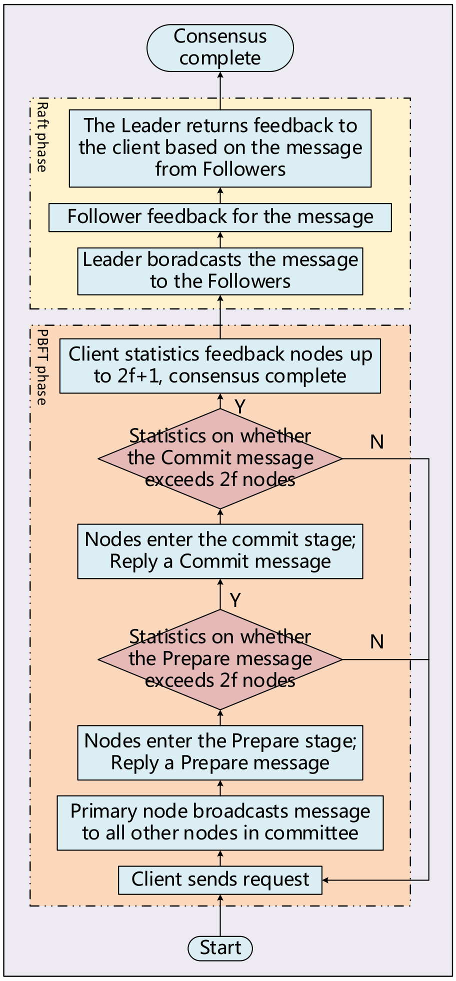
3.2 Node grouping strategy

This section introduces the design idea of consistent hashing and utilizes the characteristics of balance, spread and monotonicity of it to propose the node grouping strategy applicable to R-PBFT.

The result of the consistent hashing is a 32-bit type integer, according to which the hash values of the node mapping can be distributed on a circle from 0 to 2^{32}. The consistent hashing algorithm ensures that the nodes are distributed as evenly as possible and equalizes the load of leaders within each group, and its monotonicity reduces the impact of node joining and exiting on the group.

The nodes grouping strategy using the consistent hashing algorithm is shown in Figure 2. Let every supervisory node be assigned to each r groups. The supervisory node s is no less than 2r. Groups are set to 6 (with three virtual nodes), and every supervisory node is assigned to each three groups, then $s \geq 2$. When the number of groupings is too

Fig. 1 R-PBFT consensus mechanism


Consensus complete
Raft phase
The Leader returns feedback to the client based on the message from Followers
Follower feedback for the message
Leader boradcasts the message to the Followers
PBFT phase
Client statistics feedback nodes up to 2f+1, consensus complete
Y
Statistics on whether the Commit message exceeds 2f nodes
N
Nodes enter the commit stage; Reply a Commit message
Y
Statistics on whether the Prepare message exceeds 2f nodes
N
Nodes enter the Prepare stage; Reply a Prepare message
Primary node broadcasts message to all other nodes in committee
Client sends request
Start

little to cause data skewing problem due to uneven enough slicing, it means that some of the slices are carrying most of the nodes, resulting in load imbalance. Therefore, this paper also introduces the concept of virtual nodes of consistent hashing algorithm to expand the number of groupings by setting multiple virtual groupings for the actual groupings to make the groupings more uniform.

A grouping form of R-PBFT is given here as Fig. 3.

The node grouping algorithm based on consistent hashing algorithm is presented as below.

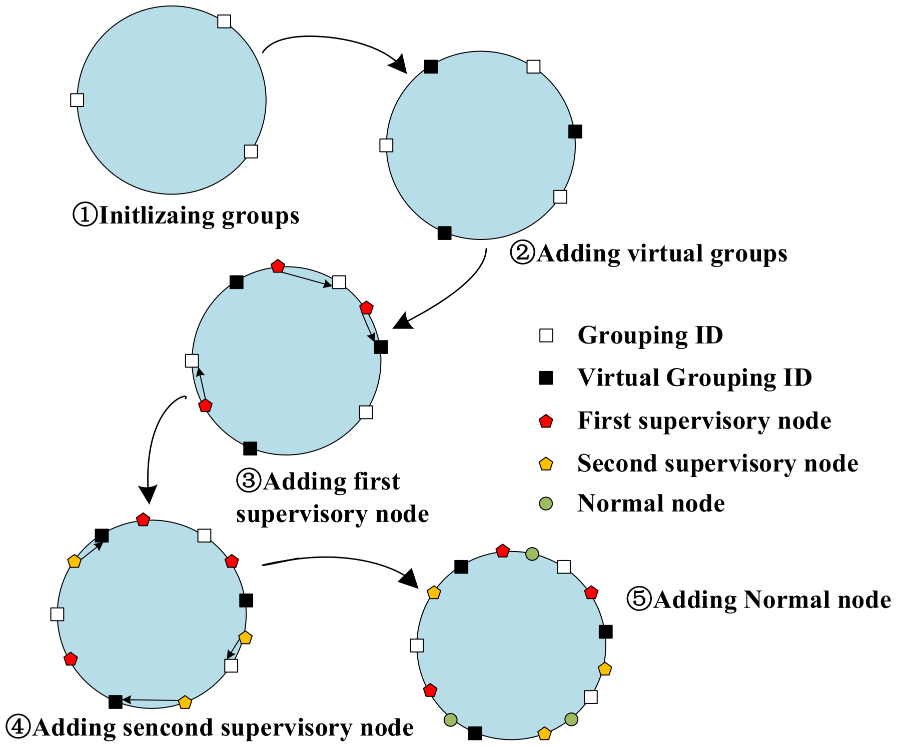


Fig. 2 Nodes grouping strategy

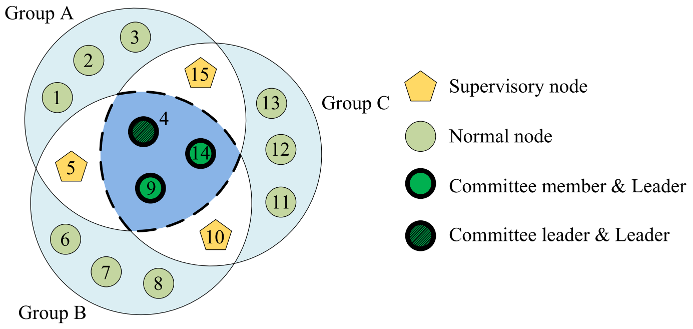


Fig. 3 Example of node grouping: each group contains a leader node, two supervisory nodes and three normal nodes; each supervisory node is assigned to two groups; Group A contains nodes 1, 2, 3, 4, 5, 15; Group B contains nodes 5, 6, 7, 8, 9, 10; Group C contains nodes 10, 11, 12, 13, 14, 15

Algorithm 1: Node grouping algorithm

```
1. Init_hash_ring: The hash ring is initialized according to the number of groups $g$ and the
   location of the group is assigned according to the ID;
2. Grouping supervision nodes to groups:
   for (i=0; r; ++1)
      Hash (nID + randomString);
      Find the group ID clockwise according to the supervisory node position, and map the
      supervisory node to the corresponding multiple different groups;
   end for
3. Grouping for normal nodes:
   Computing Hash (nID + randomString), Mapping normal nodes to a hash ring based on
   results;
4. Reasonableness verification:
   Based on the condition that the number of nodes n is greater than g, $n \geq g$ and the
   condition that each group contains at least one supervisory node. if it is, go to step 5; if
   not, go to step 2;
5. Grouping completed;
```

3.3 Supervision strategy for supervisory nodes

For Raft algorithm, the supervision node plays an important role in improving its security. The supervision node enables Raft to resist Byzantine malicious nodes.

Supervisory nodes exist anonymously in Raft; one or more supervisory nodes may exist within a group after grouping; supervisory nodes also participate in multiple groups and do not participate in Leader elections; supervisory nodes ostensibly follow the Leader as Follower to prevent the Leader from targeting them for fraud.

Raft adds a signature verification part to the consensus. The Leader signs the message when sending it to the Follower. Supervisory nodes verify the signatures and compare the contents after receiving messages from different Leaders as a way to determine whether they are Byzantine malicious nodes.

The supervision strategy of the supervisory nodes is shown in Fig. 4, which is divided into three stages: evidence collection, evidence presentation and verification.

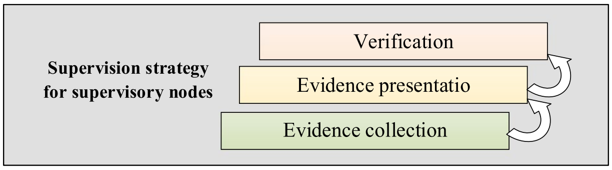


Fig. 4 Supervision strategy for supervisory nodes

(1) Evidence collection: When performing Raft intra-group consensus, supervisory nodes can receive messages sent by Leader to Follower in multiple groups at the same time. First, verify their signatures and then compare whether the messages sent by multiple Leader nodes are the same. If there is a Leader node with different messages from other Leader nodes, it can be judged as a malicious node and enter the evidence presentation stage.
(2) Evidence presentation: The supervisory node packages the messages of the malicious node into a verification message $<V, t, i, No>$ and sends the message to the node management service. Where V is the message identifier, t is the timestamp, i is the message, and No is the node number of the supervisory node.
(3) Verification: The node management service judges the public key and other message of the cited malicious node according to the content of the cited evidence to verify and cite the legality and makes a verdict on whether to remove the malicious node.

3.4 R-PBFT consensus algorithm

3.4.1 Pre-Prepare stage

R-PBFT consensus is divided into two stages before and after PBFT consensus and Raft consensus. The PBFT stage is divided into several smaller stages, which will be explained below.

In the Pre-Prepare stage, the master node broadcasts a Pre-Prepare message to the other nodes. $<PP, v, No, t, i, st>$ is the message format. PP is the message identifier of Pre-Prepare stage; v is view; st is node's signature. The same parameters will not be stated twice.

The second step is the Prepare stage, where the Prepare messages are fed back after the other nodes receive the information from the master node. $<P, v, No, t, i, st>$ is the message format. After the master node receives more than $2f$ messages, Prepare-Collect messages will be broadcast to other nodes. $<PC, v, No, t, info1, st>$ is the message format. $info1$ is all messages received by the master node, t is the timestamp. After the node receives the message and verifies that it is correct, it proceeds to the next stage.

Finally, there is the commit phase. The node that passed the verification in the previous step sends a message to the master node. $<C, v, No, t, i, st>$ is the message format, i.e., Commit message. The master node will broadcast the message when it receives more than $2f$ messages. $<PC, v, t, info2, st>$ is the message format. After passing the validation of other nodes, new blocks are generated into the blockchain.

If there are $2f$ Prepare messages pass the verification, the node enters the Commit stage. The process of the Commit stage is similar to the Prepare stage. After the node enters the Commit stage, it sends a Commit message to the primary node p in the format of $<C, v, No, t, i, st>$, where v is the view number, No is the message sequence number.

When the primary node p receives $2f+1$ messages from different consensus nodes (including itself), view number v, and sequence number No, it will broadcast Commit-Collect message to all consensus nodes. The message format is $<PC, v, t, info2, st>$. After the consensus node n receives the Commit-Collect message, it confirms whether there are $2f+1$ correct Commit messages *info2* sent by different consensus nodes. After successful confirmation, the block is added to the local blockchain to complete the consistency process.

3.4.2 Raft consensus stage

After entering the Raft consensus stage, the Leader broadcasts a message to the Follower, the Follower receives the message and gives feedback, the Leader determines whether consensus is reached based on the feedback message and submits the log. After the consensus is completed, reply to the client.

3.4.3 R-PBFT consensus algorithm

Algorithm 2: R-PBFT consensus algorithm

```
1.  PBFT-Stage：The client sends a data request;
2.  Primary node broadcast Pre-Prepare messages;
3.  Sub node broadcasts <P, v, No, t, i, st> and counts the number of Prepare messages and
    denotes it as count1;
4.  if (count1 ≥ 2f)
        Sub node broadcasts <C, v, No, t, i, st>;
    end if
5.  Sub node counts Commit messages and denotes it as Count2;
6.  if (Count2 ≥ 2f+1)
        Enter the Raft-Stage;
    end if
6.  The leader broadcasts Raft consensus message <Rcm, No, i> to the followers;
7.  Followers reply Rcm Prepared messages to leader;
8.  The leader counts the number of Rcm Prepared messages received and denotes it as
    Count3,
9.  if (Count3 ≥ N/2)
        commit Log;
    end if
10. Reply client;
```

4 Block generation and blockchain storage based on fairness packing

4.1 Block generation

Transactions are stored on blocks, and blocks are successfully generated to mark the end of the transaction cycle. Block generation is the process by which mining nodes pack the received transactions into blocks and find the random number nonce to meet the mining difficulty by hash collision, which mainly includes two parts: selecting the transactions and PoW mining. A legitimate block is required to gain consensus from other nodes, and in order to guarantee the legitimacy of the block, the legitimacy of the transactions must be guaranteed, so the mining nodes will only choose legitimate transactions to pack into blocks. Figure 5 shows the node mining process.

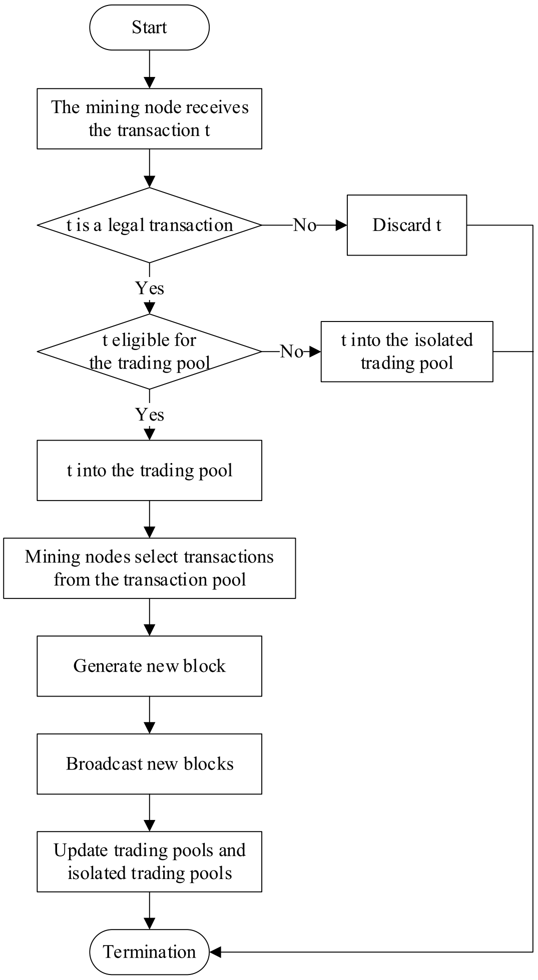
To verify the legitimacy of the new block. It will broadcast to the neighboring nodes. Mining nodes are rewarded for generating new blocks, including a fixed block reward and a variable transaction fee reward. When mining nodes generate blocks, packing as many transactions into the blocks as possible can maximize the mining nodes' transaction fee income while achieving block expansion. This chapter focuses on how nodes verify the legitimacy of transactions and blocks, and how mining nodes select transactions to maximize transaction fee revenue to achieve block expansion.

4.2 Transaction legitimacy verification

Newly generated transactions are passed to neighboring nodes via broadcasts, and the legitimacy of the transactions is then verified through them. If the transaction is legitimate, it will be put into the transaction pool or isolated transaction pool and wait to be packed into the block and will also broadcast the transaction to neighboring nodes. Of course, the transaction will be discarded if it is not legal. The legitimacy validation rules are as follows; a transaction needs to satisfy all of the following rules at the same time to be considered legitimate and is not legitimate if it does not satisfy one of the rules.

(1) Transactions are not packed into block.
(2) Transaction data conform to the version specification, including the way the transaction data is organized, the content of the fixed fields, the length of other fields, etc.
(3) The volume of the transaction is less than or equal to the largest transaction volume and greater than or equal to the smallest transaction volume.
(4) Validate the transaction input. The transaction input list cannot be empty; if the input list is not empty, go to validate the transaction input. For each transaction input, the UTXO used is not used by an exchange in the blockchain, i.e., the UTXO used does not exist in the UTXO pool; the unlock script provided in the transaction input is verified to be valid; any of the transaction inputs is verified

Fig. 5 Mining node block generation flow chart


Start
The mining node receives the transaction t
t is a legal transaction
No
Discard t
Yes
t eligible for the trading pool
No
t into the isolated trading pool
Yes
t into the trading pool
Mining nodes select transactions from the transaction pool
Generate new block
Broadcast new blocks
Update trading pools and isolated trading pools
Termination

to be valid; no two transaction inputs can use the same UTXO in the transaction input. If the transaction input references a Coinbase transaction, then the Coinbase transaction needs to get more than 100 confirmations, as well as the signature length of the Coinbase field needs to be between 2 and 100 bytes.

(5) Verify the transaction output. If the transaction output list is empty, the transaction is not legitimate. If the output list is not empty, then go ahead and validate the transaction output. For each transaction output, if the transaction output amount is negative or less than the specified minimum transaction output amount, the transaction output is invalid. Of course, the transaction output amount cannot be arbitrarily large, or at least cannot exceed the amount of coins in the whole network. If there exists a transaction output that is verified to be invalid, then the whole transaction will also be considered illegitimate.

(6) The output amount of the transaction is not less than the transaction input amount.

4.3 Blockchain storage based on transaction packing fairness

To address the problems encountered in using Ethereum as blockchain technology for data storage, this section establishes a transaction fairness model, designs an Ethereum blockchain packing algorithm, optimizes the Ethereum blockchain transaction processing process and improves the fairness of the system in the process of conducting storage transactions.

In addition, this section establishes a transaction fairness model, designs a packing algorithm, optimizes the Ethereum blockchain transaction processing flow and proposes a solution to the problems encountered in the process of data storage by Ethereum blockchain technology.

5 Transaction processing in the Ethereum blockchain

5.1 Transaction fairness model

First, Jain 's Fairness Index is defined:

$$J(x_1, x_2, \dots x_n) = \left(\sum_{i=1}^{n} x_i\right)^2 \Big/ n \cdot \sum_{i=1}^{n} x_i^2 \tag{1}$$

Transaction fairness is defined according to it. Assuming that transaction ti is uploaded at time s_i , confirmed at time v_i. the response time r_i for t_i is defined as

$$r_i = v_i - s_i \tag{2}$$

Assume that n transactions that have been processed by the system $t_1, t_2, \dots, t_n$ Response time is $r_1, r_2, \dots, r_n$. The fairness of the system is defined as follows:

$$F\left(t_1, t_2, \dots, t_n\right) = \frac{(\sum_{i=1}^{n} r_i)^2}{n \sum_{i=1}^{n} r_i^2} \tag{3}$$

Assuming that a submitted transaction t_i is submitted at time s_i, the waiting time w_i for the transaction at time t_w is

$$w_i = t_w - s_i \tag{4}$$

Given the waiting time of n transactions $t_1, t_2, \dots, t_n$ in the transaction cache pool as $w_1, w_2, \dots, w_n$ each block is packed with m transactions, and packing the first m transactions with the longest waiting time can achieve the maximum fairness.

Proof Suppose the transactions in the given transaction cache pool are listed in descending order of waiting time, and the set of transactions is obtained as $T = \left\{t_1, t_2, \dots, t_n\right\}$ where the waiting time of transactions is $w_1, w_2, \dots, w_n$. Define *Strategy1* as the above packing policy, i.e., pack the first x transactions with the longest wait time.

Now suppose there exists another packing strategy Strategy2, which achieves greater fairness. The order of packaged transactions for this Strategy2 is $p_1, p_2, \dots, p_x$. It is the order of arrangement different from the previous packaged approach. Define y as the number of blocks obtained by this packing method. Based on the above assumptions, Eq. (4) can be obtained.

$$\sum_{i=1}^{x} w_i > \sum_{i=1}^{x} w_{pi} \tag{5}$$

According to the two packing strategies, x transactions are chosen to pack into blocks each time, so the set of transactions can get $\lceil n/x \rceil$ blocks. Equation can be obtained according to the packing methods of the two strategies.

$$\forall 2 < y < n/x, \sum_{i=1}^{x} w_i > \sum_{i=1}^{x} w_{pi} \tag{6}$$

Although the packing methods are different, the objects being packed are all transaction sets T, so Eq. (6) can be obtained.

$$\sum_{i=1}^{x} w_i > \sum_{i=1}^{x} w_{pi} \tag{7}$$

Suppose the time to generate a batch of transactions into a block is t_g, the response time per transaction using the packing Strategy1 is

$$w_1 + t_w, w_2 + t_w, \dots, w_{x+1} + 2t_w, \dots, w_n + \lceil n/x \rceil t_w \tag{8}$$

t response time per transaction using the packing *Strategy2* is

$$w_{p_1} + t_w, w_{p_2} + t_w, \dots, w_{p_{x+1}} + 2t_w, \dots, w_{p_n} + \lceil n/x \rceil t_w \tag{9}$$

Thus, the fairness results of the two strategies are shown in Equation (9)and (10).

$$F_{\text{Strage1}} = \frac{\left(\sum_{i=1}^{x} 2(w_i + \lceil i/x \rceil t_w)\right)^2}{x \sum_{i=1}^{x} (w_i + \lceil i/x \rceil t_w)^2} \tag{10}$$

$$F_{\text{Strage2}} = \frac{\left(\sum_{i=1}^{x} 2(w_{pi} + \lceil i/x \rceil t_w)\right)^2}{x \sum_{i=1}^{x} (w_{pi} + \lceil i/x \rceil t_w)^2} \tag{11}$$

Because greater fairness is assumed to be achieved by *Strategy2*, it follows that:

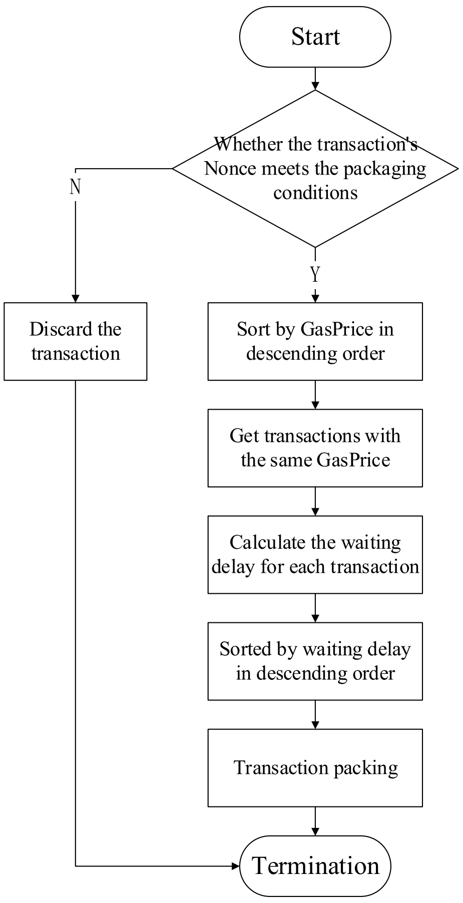

Start
Whether the transaction's Nonce meets the packaging conditions
N
Y
Discard the transaction
Sort by GasPrice in descending order
Get transactions with the same GasPrice
Calculate the waiting delay for each transaction
Sorted by waiting delay in descending order
Transaction packing
Termination

Fig. 6 Blockchain packaging algorithm flowchart based on transaction fairness model

$$F_{\mathrm{Strage1}} < F_{\mathrm{Strage2}} \tag{12}$$

Since the two strategies act on the same set of transactions, it follows that:

$$\sum_{i=1}^{x} w_i^2 > \sum_{i=1}^{x} w_{pi}^2 \tag{13}$$

$$\sum_{i=1}^{x} \left(w_i + \lceil i/x \rceil t_w\right) > \sum_{i=1}^{x} \left(w_{pi} + \lceil i/x \rceil t_w\right) \tag{14}$$

According to Eqs. (9), (10), (11) and (13), it can be known that:

$$x \cdot \sum_{i=1}^{x} \left(w_i + \lceil i/x \rceil t_w\right)^2 > x \cdot \sum_{i=1}^{x} \left(w_i + \lceil i/x \rceil t_w\right)^2 \tag{15}$$

Expanding Eq. (14) yields Eq. (15):

$$\begin{gathered}\sum_{i=1}^{n} w_i^2 + \sum_{i=1}^{n} \left(\lceil n/x \rceil t_w\right)^2 + 2\sum_{i=1}^{n} \left(w_i \lceil n/x \rceil t_w\right) > \\ \sum_{i=1}^{n} w_{pi}^2 + \sum_{i=1}^{n} \left(\lceil n/x \rceil t_w\right)^2 + 2\sum_{i=1}^{n} \left(w_{pi} \lceil n/x \rceil t_w\right)\end{gathered} \tag{16}$$

From Eqs. (12) and (15):

$$\sum_{i=1}^{n} \left(w_i + \lceil n/x \rceil\right) > \sum_{i=1}^{n} \left(w_{pi} + \lceil n/x \rceil\right) \tag{17}$$

By adding up Eqs. (4), (5):

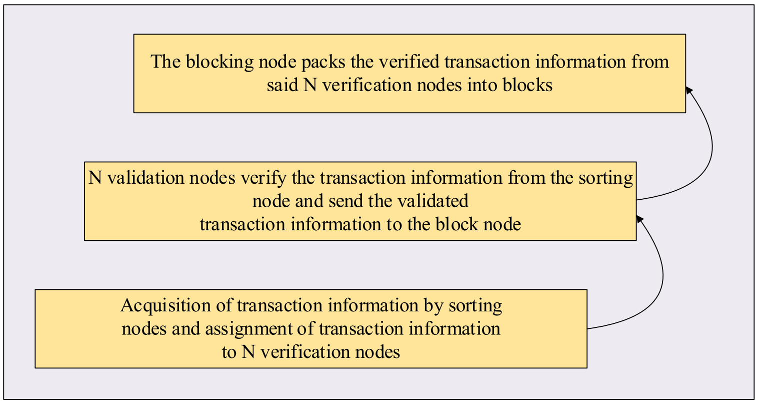

The blocking node packs the verified transaction information from said N verification nodes into blocks
N validation nodes verify the transaction information from the sorting node and send the validated transaction information to the block node
Acquisition of transaction information by sorting nodes and assignment of transaction information to N verification nodes

Fig. 7 Block-out process

$$\sum_{i=1}^{\lceil n/x\rceil-1}\sum_{j=1}^{ix} w_j > \sum_{i=1}^{\lceil n/x\rceil-1}\sum_{j=1}^{ix} w_{pj} \tag{18}$$

By adding up equations (16), (17):

$$\begin{aligned}\lceil n/x\rceil\sum_{i=1}^{n} w_i = \sum_{i=1}^{\lceil n/x\rceil-1}\sum_{j=1}^{ix} w_j + \sum_{i=1}^{x}\left(w_i\lceil n/x\rceil\right) > \\ \sum_{i=1}^{\lceil n/x\rceil-1}\sum_{j=1}^{ix} w_{pj} + \sum_{i=1}^{x}\left(w_{pi}\lceil n/x\rceil\right) = \lceil n/x\rceil\sum_{i=1}^{n} w_i\end{aligned} \tag{19}$$

Simplify Eq. (18) to get formula (19):

$$\sum_{i=1}^{x} w_i = \sum_{i=1}^{x} w_i \tag{20}$$

Obviously, Eq. (19) contradicts Eq. (6) and therefore concludes that there is no fairness strategy greater than Strategy1, i.e., sorting transactions by wait time and selecting the transaction with the longest wait time for packaging can achieve maximum system fairness.

5.1.1 Blockchain packing algorithm based on transaction fairness model

The blockchain packing algorithm based on the transaction fairness model consists of two main parts: The first part is based on the Ethereum packing transaction process, which sorts the set of waiting packing transactions that satisfy the nonce value in descending order of GASPrice, and this part still sorts the set of transactions according to the principle that Ethereum packs transactions with large GASPrice first. The second part is to consider the fairness packing. To achieve maximum fairness in the system, the transactions with the same GASPrice are sorted in reverse order according to the waiting time, and the transactions with long waiting time are packed first. After the above two parts, the returned transaction set sequence is the final packaged transaction sorting result considering both the principle of GASPrice

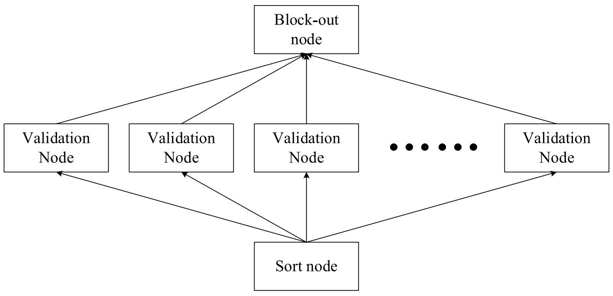


Fig. 8 Block-out system

priority packing and for achieving the maximum fairness of the system. The pseudocode is presented in Algorithm 3, and its flowchart is shown in Fig. 6.

Algorithm 3: Blockchain packaging algorithm based on transaction fairness model

Input: Set of transactions $t = \{t_1, t_1, \dots, t_n\}$; gas_i means GASPrice of t_i; $s = \{s_1, s_1, \dots, s_n\}$ indicates the sending time of each transaction in the pool; t_c means the current time.

Output: Set of transactions $t' = \{t_1, t_1, \dots, t_n\}$。

```
1  for (i=0; n; i++)
2      sortG(t) // Sort the transactions to be packaged in descending order by GASPrice.
3  end for
4  for (i=0; n; i++)
5    if $gas_i == gas_{i+1}$
6        for (;a=i;)
7            if $gas_a > gas_{a+1}$
8                b=a; // Get the same portfolio of trading transactions as GASPrice
9            else
10               a++;
11           end if
12       end for
13       $sortT(i, t, s, t_c)$; // Sort the transactions with the same GASPrice in descending order of waiting time delay
14       i=a;
15   end if
16 i++;
17 end for
18 return t'; // Return the updated set of packaged transactions
```

Table 1 Experimental environment

Equipment parameters	Related properties
CPU	Intel(R) Xeon(R) CPU E5-2630 v4
Hard disk	16.00 GB RAM
Blockchain platform	Hyper ledger fabric v1.1
System environment	Linux, Ubuntu 16.04
Container	Apache 2.0 protocol, Docker
Development language	Python, C + +, go
Testing tools	Hyper ledger Caliper

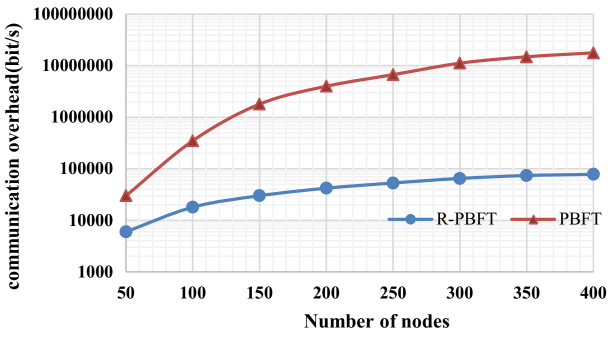


Fig. 9 Comparison of the communication overhead between R-PBFT and PBFT

5.2 Block-out method based on node verification

In the previous section, a fair packing algorithm is proposed to pack the transactions; in this section, how to broadcast the blocks obtained from packing is studied (see Figure 7).

Step 1: The transaction information is obtained by the sorting node and assigned to the N verification nodes.

(1) In this scheme, the transaction information in the system is first detected by the sorting node and assigned to verification nodes, for example, sorting node assigns transaction information detected within a period of time (e.g., 10 min) to the N verification nodes.

(2) Among them, the sorting node may assign the transaction information to the N verification nodes according to the predetermined assignment rules, for example, the sorting node assigns the transaction information to the N verification nodes in order of the time of receiving the transaction information; or the sorting node divides the acquired transaction information into N copies and assigns them to the N verification nodes, respectively.

Step 2: The N verification nodes verify the transaction information from the sorting node and send the verified transaction information to the block node.

1. When a new transaction is generated, the transaction initiator broadcasts the transaction, and the sorting node receives the generated transaction and validates the transaction information. That is, when a new transaction is generated in the system, the legitimacy of the transaction initiator needs to be verified as well as verifying that the transaction initiator's wallet is not being used fraudulently.

Unlike the prior art system where each node validates the transaction information, in this scheme, each transaction information assigned is validated by the N validation nodes assigned to the transaction information. The verification of the transaction information by the verification nodes includes the verification of the transaction initiator's wallet address and the verification of the initiator's signature.

2. The legitimacy of the transaction initiator's wallet can be verified by the address of the initiator's wallet, whereas the address of the initiator's wallet is related to the public key, verifying transaction initiator's wallet means verifying it's public key. Optionally, transaction initiator's wallet address can be verified through a collection of public keys maintained by the system.
3. Further, to prevent the wallet from being used fraudulently, the initiator signs the initiated transaction. Then public key of the transaction initiator can be used by verification node to verify the signature, thereby verifying the legitimacy of the transaction information.

In this scheme, N verification nodes perform the above verification on each received transaction message, and after the verification is completed, the verified transaction message is sent to the outgoing block node.

Step 3: The verified transaction information from said N validation nodes is packaged and blocked by the blocking node.

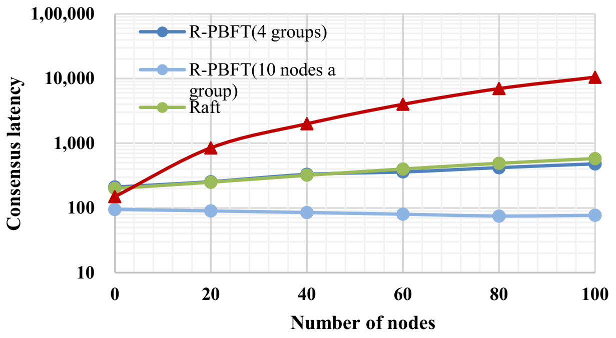


Fig. 10 Comparison of the consensus latency between R-PBFT and PBFT

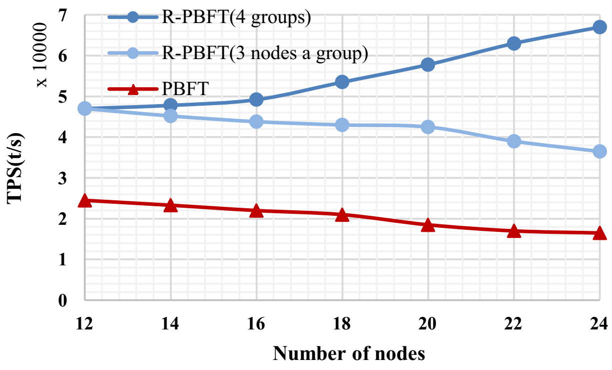


Fig. 11 Comparison of the TPS between R-PBFT and PBFT

(1) The block-out node receives the verified transaction information from the N verification nodes, and since the transaction information has already been verified by the verification nodes, the block-out node can directly package the transaction information for block-out (see Fig. 8).

In this scheme, the above-mentioned sorting nodes, verification nodes and block-out nodes are selected from M alternative nodes, and M is greater than the sum of the number of sorting nodes, verification nodes and block-out nodes. There are many alternative nodes set up in the blockchain. Sorting nodes, verification nodes and block-out nodes are determined from the alternative nodes.

(2) The above alternative nodes are all nodes that satisfy certain conditions, such as asset balance, activity and integrity value all satisfy the preset values. Asset balance refers to the parameter used to measure the available assets of a node, such as the balance of virtual currency held by the node.
(3) Activity refers to the parameters used to measure the node's activity in logging into the blockchain system and participating in block issuance, such as logging time and number of block issuance, etc. Integrity value refers to the parameters used to measure the node's trustworthiness, such as whether there are forged transactions, etc.

6 Results of experiments

6.1 Experimental result of R-PBFT consensus algorithm

The PBFT consensus mechanism requires two nodes to communicate, and the communication traffic is $O(n^2)$. The communication traffic of the Raft consensus mechanism is $O(n)$, and through network partitioning, the communication traffic of R-PBFT decreases from $O(n^2)$ to $O(n/k)+\mathrm{O}(p^2)$ compared to the PBFT consensus

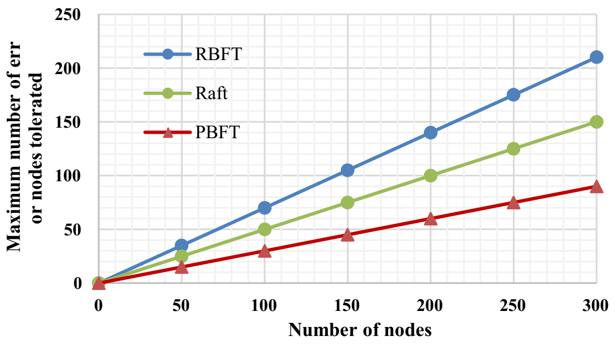


Fig. 12 Comparison of the fault tolerance between R-PBFT and PBFT

mechanism. The simulation sets the message content $m=256$ bit, message size $r=80$ bit for replying client, and heartbeat packet size $h=64$ bit for Raft consensus process in one consensus process.

The consensus latency of this algorithm is the time from when the request was initiated to when the request is acknowledged and uploaded to the chain. The average of 30 consecutive measurements is taken as the consensus latency of the algorithm, and the data of the proposed algorithm R-PBFT and the comparison algorithms PBFT and Raft are recorded separately for the same network size.

In blockchain systems, *TPS* is defined as the number of transactions *M* divided by the transaction processing time t, i.e., each node can listen to requests from clients, so a client program is bound for each node. All requests are finally collected and packaged by the master node and propagated to other nodes through the consensus mechanism. The block output interval is set to 10 s, and the stable 10 sets of data in the database are taken after 5 min of system operation. The average of the 10 stable data sets in the database is taken as the system *TPS* for this test.

$$TPS = \frac{M}{t} \tag{21}$$

Let f_1, f_2 be integers greater than 0. Raft nodes' groups m$\geq 2f_1+1$, and groups $g \geq 3f_2+1$. Each r groups are assigned a supervisory node ($s \geq 2$), and assuming the same number of nodes in groups, the total number of nodes N satisfies:

$$N = gm - \left[\frac{g}{s}(s-1)\right] - [l(s-1)] \tag{22}$$

$$l = \left\{ \begin{array}{c} 0, g \bmod \ s = 0 \\ 1, g \bmod s \neq 0 \end{array} \right\} \tag{23}$$

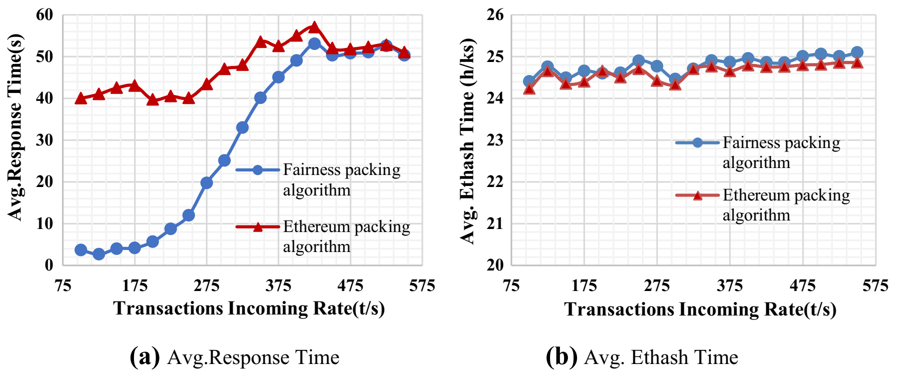


Fig. 13 The influence of the transactions incoming rate on the experimental

Maximum fault tolerance for the PBFT consensus stage is (g-1)/3, and the maximum fault tolerance for the Raft stage is (m-1)/2. With the participation of supervisory nodes, the maximum fault tolerance of R-PBFT is:

$$
\begin{aligned}
&F \le \frac{g-1}{3}m + \left(\mathrm{s} - \frac{g-1}{3}\right)\left(\frac{m-1}{2} - 1\right) + \\
&\left[\left(g - \frac{g-1}{3}\right) - \left(s - \frac{g-1}{3}\right)\right]\left(\frac{m-1}{2}\right) \\
&= 4f_1f_2 + 2f_2 + f_1 - s
\end{aligned}
\tag{24}
$$

The simulation is set to assign one supervisory node to each three groups (i.e., $s=3$) and assume that the amount of nodes contained in all groups is the same. Table 1 shows the experimental environment.

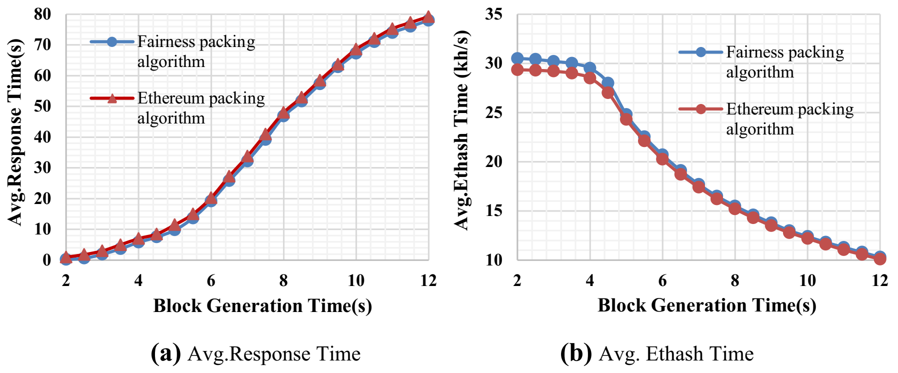


Fig. 14 The influence of the block generation time on the experiment

6.1.1 Communication overhead

Figure 9 shows a comparison of the communication overhead between R-PBFT and the PBFT consensus mechanism. It can be seen that the communication overhead of R-PBFT is much smaller than that of PBFT in the whole blockchain network. For example, when the amount of network nodes is 117, the communication overhead of the PBFT is much smaller than that of the blockchain network. The communication overhead of the classical PBFT consensus mechanism is 3.49×10^6 bit. The communication overhead of R-PBFT is 4.11×10^4 bit, and the communication overhead is reduced by 98.8%. As the amount of nodes increases, R-PBFT saves more communication overhead than PBFT.

6.1.2 Consensus latency

The results of the consensus latency of the PBFT and the comparison algorithm are presented in Fig. 10. Consensus latency grows gradually as the increasing nodes. PBFT algorithm grows the fastest, and Raft grows the slowest. For R-PBFT, increasing nodes and keeping nodes constant while increasing the number of groups causes higher consensus latency compared to keeping the groups constant while increasing the nodes, as shown in Fig. 10 for R-PBFT (amount of nodes in the group: 10) and R-PBFT (number of groups: 4). This is because increasing the groups essentially expands the committee size and increases the consensus elapsed time in the PBFT stage, so consensus latency is more affected by increasing groups. However, the latency of R-PBFT is still much smaller than that of PBFT for the same node. From Fig. 8, latency of PBFT increases sharply with the increase in node size, while the latency of R-PBFT increases at a slower rate. Therefore, R-PBFT can still ensure high consensus efficiency when the node size increases.

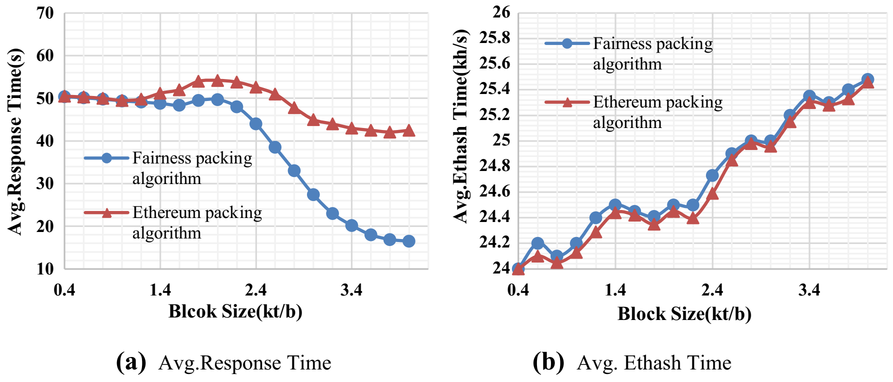


Fig. 15 The influence of the block size on the experiment

6.1.3 TPS(transaction per second)

The TPS test results of R-PBFT and PBFT are shown in Fig. 11. Consensus efficiency is the main factor affecting TPS, and the higher the consensus efficiency, the higher the transaction processing capacity. In addition, node concurrency is another factor affecting TPS; the higher the node concurrency, the higher the transaction traffic. However, the increase in block size requires higher bandwidth. (The experiment uses a virtual machine to simulate the environment and does not reach the upper limit of bandwidth, which can be equated to infinity.) Factors affecting TPS also include the ability of each node to handle concurrent data, I/O read/write capability to the database, etc.

Figure 11 presents that the increase in nodes enhances the concurrency of nodes, but transaction processing takes longer and TPS decreases as PBFT decreases consensus efficiency when nodes increase. For R-PBFT, nodes are divided into four groups, and when increasing nodes in the group, Sect. 6.1.3 presents that the algorithm consensus efficiency is less affected and node concurrency is the main factor affecting TPS. As nodes increases, TPS gradually increases. When nodes in a group are fixed and groups are increased, consensus efficiency becomes the main factor affecting TPS. As groups increase, TPS gradually decreases. However, in the same network size, the TPS of R-PBFT is about 300%~400% of that of PBFT. Therefore, R-PBFT is more suitable for federation chain application scenarios with higher TPS requirements.

6.1.4 Fault tolerance

Consensus efficiency and TPS are not affected by the amount of supervisory nodes because the supervisory nodes are located in multiple groups and their communication is independent in consensus process. Therefore, the communication strength of supervisory nodes increases, the overall communication strength of the consensus algorithm remains unchanged. In terms of fault tolerance, an error of a supervisory

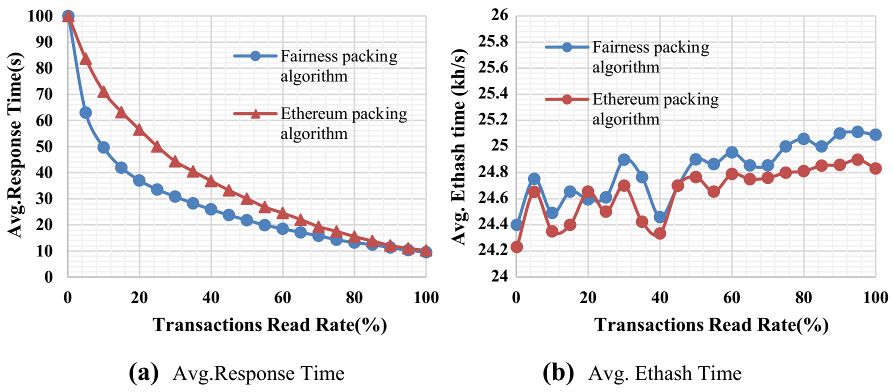


Fig. 16 The influence of the transactions read rate on the experiment

node is equivalent to an error of a node in all the groups containing that supervisory node, and the maximum fault tolerance of the PBFT algorithm is given by Eq. (4) in Sect. 5.2.4. Maximum fault tolerance performance of R-PBFT with PBFT and Raft is compared in Fig. 12.

In Fig. 12, R-PBFT has higher fault tolerance than PBFT and Raft. The maximum fault tolerance is 2 for PBFT alone, 3 for Raft, and 4 for R-PBFT (equivalent to the limit of three consensus groups of two nodes each, with one supervisory node for each group). As the size of the nodes increases, the R-PBFT fault tolerance increases. Therefore, R-PBFT has higher security.

6.2 Experimental result of fairness packing algorithm

Based on the transaction fairness model, blockchain packing algorithm is experimented to evaluate its performance. In addition to fairness packing algorithm, another classical blockchain packaging algorithm is selected for comparison.

The following four performance metrics are defined as important parameters for the performance of the algorithm, respectively. They are transaction incoming rate, transactions read rate, block generation time, block size, respectively. Transaction incoming rate is adjusted by changing the PoW difficulty. Three other parameters are adjusted by setting specific parameters and avg.response time (s) and avg.ethash time (Number of hashes calculated per 1000 s, h/ks) of different algorithms are then derived to compare the performance of the algorithms.

6.2.1 Transaction incoming rate

This section focuses on the effect of transaction incoming rate on the fairness packing algorithm. Set the block generation time to 5 s, block size to 2kt/b, and transaction read rate to 100%. Transaction incoming rate in the experiment is set in the range of 100 ~ 550(t/s), and the amount of change per step is set to 25(t/s).

When transaction incoming rate continues to increase, there will be more transactions in the transaction buffer pool, and when the speed of transaction dealing cannot keep up with transaction incoming rate, the transaction buffer pool will continue to store more transactions. As shown in Fig. 13, the avg.response time of the fairness packing algorithm has advantage over the Ethereum packing algorithm under different transaction incoming rates. Avg.response time grows slowly with the increase in task volume in the range of 100–175 (t/s) and then grows rapidly in the range of 175–550 (t/s) and it is speculated that when the transaction incoming rate is less than 175 (t/s), there is no transaction in the transaction buffer pool, while the transaction incoming rate is greater than 175 (t/s), it starts to be greater than the transaction dealing speed. The transaction buffer pool starts to accumulate a backlog of transactions, resulting in a decrease in response speed.

The average ethash time of fairness packing algorithm and Ethereum packing algorithm will continue to grow with the transaction incoming rate growth, and eventually stop growing around 400 (t/s); The fairness packing algorithm has some advantages over the Ethernet packing algorithm, but the advantages are not large.

The reason is relatively intuitive: transaction incoming rate at 100–175 (t/s) does not reach the upper limit of the node's processing capacity, so no matter how many transactions incoming, they are packed into blocks; The transaction incoming rate continues to grow, the node's packing speed has reached the processing limit, and the transaction starts to pile up in the transaction buffer pool and cannot be processed in time, so the average ethash time reaches the maximum and cannot continue to rise.

6.2.2 Block generation time

How average response time and average ethash time are affected by block generation time is studied in Fig. 14. The range of it is chosen from 2 to 12 s, and we set the transactions incoming rate to 500 t/s, block size to 2kt/b, and transactions read rate to 100%.

Under the premise of constant block size, the longer the block generation time, the fewer transactions can be dealt per unit time, the speed of transaction dealing cannot keep up with transaction incoming rate. The transaction buffer pool will constantly stacked with new incoming transactions, making the avg.response time longer. Overall, the response time of fairness packing algorithm is slightly shorter than that of Ethereum packing algorithm, and the difference in performance between the two algorithms is not significant.

The amount of transactions and ethash computation that can be dealt decreases as the increase in block generation time. For example, when it is greater than 6 s, because of the amount of dealt transactions is far from the upper limit of transaction processing capacity, the performance gap between the two algorithms is not large; when it is less than 6 s, it is close to the upper limit of transaction processing capacity, which is equal to approaching the bottleneck of ethash computing capacity, and the growth rate of avg.ethash time slows down greatly, while fairness packing algorithm compared with Ethereum packing algorithm, the computational advantage of ethash is also revealed.

In general, the performance difference between the two algorithms under different block generation time conditions is not much, and the advantage of the fairness packing algorithm is not significant for different block generation times.

6.2.3 Block size

Block size is a major factor affecting the transaction packing algorithm. In Fig. 15, set the transactions incoming rate to 500 t/s, block generation time to 5 s, and transactions read rate to 100%. The range of block size is set to 0.4–4.0kt/b, and the step size is 0.2kt/b.

Firstly, the effect of block size on avg.response time can be summarized by Figure a. At 0.4–1.2 kt/b, the difference between the two algorithms is small; however, after the block size rises larger, because the fairness packing algorithm for transaction priority allocation is based on two-level allocation, it significantly outperforms the Ethereum packing algorithm in terms of performance.

Secondly, with the increase in block size, we can see that the average ethash time has a small increase, although it intuitively feels that the average ethash time is greatly improved due to the axis setting; in addition, the average ethash time of fairness packing is slightly greater than the Ethereum packing.

6.2.4 Transactions read rate

The impact of transaction read rates on the two packing algorithms is studied in Fig. 16. Set the transactions incoming rate to 500 t/s, block generation time to 5 s and block size to 2kt/b. Transactions read rate is set from 0 to 100% in steps of 5%.

The first is the effect of transactions read rate on the average response time. Because the transactions read rate affects volume of transactions dealt simultaneously provided that transactions incoming rate is determined. Therefore, as the transactions read rate increases and the volume of dealt transactions becomes larger, the avg.response time gradually decreases; when transactions read rate is close to 0, the average response time becomes very large, and the result is unacceptable. Therefore, the transactions read rate is better than 50%; it can be seen that the fairness packing algorithm outperforms the Ethereum packing algorithm in terms of average response time, especially when the transactions read rate is in the range of 0 to 50%. When the transactions read rate is close to 100%, the difference between the performance of the two algorithms on average response time gradually decreases, so when testing the influence of other parameters, the transaction read rate is usually set to 100% to control the impact of the transaction read rate.

The fairness packing algorithm performs better than the Ethereum packing algorithm for the average ethash time. Because the transaction volume does not reach the upper limit of ethash computation, the increase in transaction read rate also increases avg.ethash time; after that, the ethash computation capacity is limited and eventually level off. The fairness packing algorithm performs much better than the Ethereum packing algorithm in average response time for transactions read rate range 0–70%. The advantage in average ethash time is not obvious, but the difference in algorithm performance can still be seen.

7 Conclusion and Future Work

Based on Raft and Jain's fairness index, an improved PBFT consensus mechanism, R-PBFT and a two-layer structured fair packing mechanism are proposed in this paper.

R-PBFT consensus mechanism groups the blockchain network nodes, and the leaders are elected to form a committee using Raft within each group, and PBFT is used for consensus within the committee. Because R-PBFT is unable to counter Byzantine malevolence, supervisory nodes are introduced to improve its security. It is experimentally demonstrated that R-PBFT outperforms the classic PBFT consensus mechanism in terms of communication overhead, consensus latency, TPS and fault tolerance.

This paper establishes a transaction fairness model, designs an Ethereum blockchain packing algorithm, optimizes the Ethereum blockchain transaction processing process and improves the fairness of the system in the process of conducting storage transactions, to address the performance issues encountered in the use of Ether as a blockchain technology for data storage. In addition, this paper optimizes the Ethereum blockchain transaction processing flow and proposes a solution to the problems encountered in the process of data storage by Ethereum blockchain technology. The fairness packing algorithm handles transaction priorities with the classical Ethereum packing algorithm in the case of different GASPrice; in the case of the same GASPrice, the transaction priorities are handled with the fairness packing algorithm. The superiority of this algorithm in terms of performance has been experimentally demonstrated. The fairness packing algorithm experiments with four parameters, namely transaction incoming rate, transactions read rate, block generation time, block size. Fairness packing algorithm basically outperforms the classical Ethereum packing algorithm under different selection conditions of each parameter through experiment.

R-PBFT gains advantages such as high consensus rate and high throughput because of the architecture, but also because of the architecture, there may be some difficulties in application implementation. On the one hand, it requires higher stability of hardware, and on the other hands, it may cause difficulties in understanding. Therefore, further optimization and simplification are needed to implement it at the application level. Similarly, the fairness packing algorithm has some limitations and faces the same problems as the consensus mechanism: The architecture is more complex and difficult to implement for applications. Therefore, making approach more lightweight and easier to apply is a problem that should be addressed in the future work.

Acknowledgements The work was supported by the National Natural Science Foundation (NSF) under grants (No. 61873341, 62171330), Key Research and Development Plan of Hubei Province (No. 2020BAB102), Open project of CAAC Key Laboratory of Civil Aviation Wide Surveillance and Safety Operation Management & Control Technology (No.202001), Open Research Fund Program of Data Recovery Key Laboratory of Sichuan Province (Grant No. DRX2001). Any opinions, findings and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

References

1. Berdik D, Otoum S, Schmidt N et al (2021) A survey on blockchain for information systems management and security. Inf Process Manag 58(1):102397
2. Li Y, Zheng K, Yan Y et al (2017) EtherQL: a query layer for Blockchain System[M]. In: Candan S, Chen L, Pedersen TB, Chang L, Hua W (eds) Database systems for advanced applications. Springer, Cham
3. Brewer EA (2000) Towards robust distributed systems. In: Proceedings of the nineteenth annual ACM symposium on principles of distributed computing. ACM Press, New York, 343502. https://doi.org/10.1145/343477
4. Wang WB, Dinh TH, Xiong ZH et al. (2018) A survey on consensus mechanisms and mining management in blockchain networks. ar-Xiv Preprint, arXiv:1805.02707
5. Jain RK, Chiu DMW, Hawe WR (1984) A quantitative measure of fairness and discrimination. Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA

6. Yuan Y, Wang FY (2016) Blockchain: the state of the art and future trends. Acta Autom Sin 42(4):481–494
7. Nakamoto S (2019) Bitcoin: a peer-to-peer electronic cash System. consulted
8. Wood G (2014) Ethereum: a secure decentralised generalised transaction ledger. Ethereum Proj yellow Pap 2014 (151):1–32
9. Dinh TTA, Wang J, Chen G, Liu R, Ooi BC, Tan KL (2017) Blockbench: a framework for analyzing private blockchains. In: Proceedings of the 2017 ACM international conference on management of data, pp 1085-1100
10. Kotewicz M (2021) Parity [EB/OL]. https://github.com/openethereum/parity-ethereum. Accessed 29 Mar
11. Thakkar P, Nathan S, Viswanathan B (2018) Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS). IEEE 8: 264–276
12. Gorenflo C, Lee S, Golab L, Keshav S (2020) Fastfabric: scaling hyperledger fabric to 20 000 trans- actions per second. Int J Netw Manag 30(5):e2099
13. Pongnumkul S, Siripanpornchana C, Thajchayapong S (2017) Performance analysis of private blockchain platforms in varying workloads. In: 26th International conference on computer communication and networks (ICCCN). IEEE, pp 1–6
14. Rouhani S, Deters R (2017) Performance analysis of ethereum transactions in private blockchain. In: 8th IEEE international conference on software engineering and service science (ICSESS). IEEE, pp 70–74
15. Li C, Bai J, Yi C et al (2020) Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system. Inf Sci 516:33–55
16. Li C, Song M, Zhang M, Luo Y (2020) Effective replica management for improving reliability and availability in edge-cloud computing environment. J Parallel Distrib Comput 143:107–128
17. Ampel B, Patton M, Chen H (2019) Performance modeling of hyperledger sawtooth blockchain. In: IEEE international conference on intelligence and security informatics (ISI). IEEE, pp 59–61
18. Hao Y, Li Y, Dong X, Fang L, Chen P (2018) Performance analysis of consensus algorithm in private blockchain. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 280–285
19. Suankaewmanee K, Hoang DT, Niyato D, Sawadsitang S, Wang P, Han Z (2018) Performance analysis and application of mobile blockchain. In: International conference on computing, networking and communications (ICNC). IEEE, pp 642–646
20. Chen S, Zhang J, Shi R, Yan J, Ke Q (2018) A comparative testing on performance of blockchain and relational database: Foundation for applying smart technology into current business systems. In: International conference on distributed, ambient, and pervasive interactions. Springer, pp 21–34
21. Baliga A, Subhod I, Kamat P, Chatterjee S (2018) Performance evaluation of the quorum blockchain platform. arXiv preprint arXiv:1809.03421
22. Huang D, Ma X, Zhang S (2019) Performance analysis of the raft consensus algorithm for private blockchains. IEEE Transact Syst Man Cybern Syst 50(1):172–181
23. Gervais A, Karame GO, Wüst K, Glykantzis V, Ritzdorf H, Capkun S (2016) On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 3-16
24. van Moorsel A (2019) Benchmarks and models for blockchain: consensus algorithms. ACM SIGMETRICS Perform Eval Rev 46(3):113–113
25. Qashlan A, Nanda P, He X (2020) Automated ethereum smart contract for block chain based smart home security. In: smart systems and IoT: innovations in computing. Springer, Singapore, pp 313–326
26. Li C, Zhang Y, Zhiqiang H et al (2020) An effective scheduling strategy based on hypergraph partition in geographically distributed datacenters. Computer Netw 170:107096
27. Li C, Tang J, Ma T, Yang X, Luo Y (2020) A workflow job scheduling algorithm based on load balancing in distributed cloud. J Netw Comput Appl 152:102518
28. Cha SC, Peng WC, Huang ZJ et al (2017) On design and implementation a smart contract-based investigation report management framework for smartphone applications. In: International conference on intelligent information hiding and multimedia signal processing, Springer, Cham, pp 282–289

29. Knecht M, Stiller B (2017) Smartdemap: A smart contract deployment and management platform. In: IFIP international conference on autonomous infrastructure, management and security. Springer, Cham 159–164
30. Li A, Wei X, He Z (2020) Robust proof of stake: a new consensus protocol for sustainable blockchain systems. Sustainability 12:2824
31. Karakostas D, Kiayias A (2021) Securing proof-of-work ledgers via checkpointing. In: IEEE international conference on blockchain and cryptocurrency (ICBC), pp 1-5. https://doi.org/10.1109/ICBC51069.2021.9461066
32. Howard H, Mortier R (2020) Paxos vs Raft: Have we reached consensus on distributed consensus?. In: Proceedings of the 7th workshop on principles and practice of consistency for distributed data, pp 1–9
33. Castro M, Liskov B (1999) Practical byzantine fault tolerance. In: Proceedings of the third symposium on operating systems design and implementation. ACM Press, New York, pp 173–186
34. Lamport L (1998) The part-time parliament. ACM Trans Comput Syst 16(2):133–169
35. Ongaro D, Ousterhou TJ (2014) In search of an understandable consensus algorithm. In: Proceedings of USENIX ATC'14: 2014 USENIX annual technical conference. USENIX association, Berkeley, pp 305–320
36. Eyal I, Gencer AE, Sirer EG et al (2016) Bitcoin-ng: A scalable blockchain protocol. In: 13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16), pp 45–59
37. Kogias EK, Jovanovic P, Gailly N, et al. (2016) Enhancing bitcoin security and performance with strong consistency via collective signing. In: 25th {usenix} security symposium ({usenix} security 16), pp 279–296
38. Eyal I, Birman K, Van Renesse R (2015) Cache serializability: reducing inconsistency in edge transactions. In: 2015 IEEE 35th international conference on distributed computing systems. IEEE, pp 686–695
39. Buterin V (2014) A next-generation smart con-tract and decentralized application platform. Etherum 1:1–36
40. Larimer D (2014) Delegated proof of stake consensus[R]. [2020–08–19]
41. Xu X, Sun G, Luo L et al (2021) Latency performance modeling and analysis for hyperledger fabric blockchain network. Inf Process Manag 58(1):102436
42. Castro M, Liskov B (1999) Practical Byzantine fault tolerance. In: Proc Symp Oper Syst Design Implement, New Orleans, LA, USA, pp 173–186
43. Lamport L (1998) The part-time parliament. ACM Trans Comput Syst 16(2):133–169
44. Moraru I, Andersen DG, Kaminsky M (2013) There is more consensus in egalitarian parliaments. In: Proc ACM Symp Oper Syst Principles (SOSP), Farmington, PA, USA 358–372
45. Reyna A et al (2018) On blockchain and its integration with IoT. Challenges and opportunities. Future Gener Comput Syst 88:173–190
46. Liang W et al (2020) An industrial network intrusion detection algorithm based on multi-feature data clustering optimization model. IEEE Trans Ind Informat 16(3):2063–2071. https://doi.org/10.1109/TII.2019.2946791
47. Rizun PR (2015) A transaction fee market exists without a block size limit. Block size limit debate working paper
48. Li C, Song M, Yu C, Luo Y (2021) Mobility and marginal gain based content caching and placement for cooperative edge-cloud computing. Inf Sci 548:153–176
49. Möser M, Böhme R (2015) Trends, tips, tolls: a longitudinal study of Bitcoin transaction fees. In: International conference on financial cryptography and data security. Springer, Berlin, Heidelberg, pp 19–33