Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7270624, 13 pages
https://doi.org/10.1155/2020/7270624

Research Article

Hybrid Consensus Algorithm Optimization: A Mathematical Method Based on POS and PBFT and Its Application in Blockchain

Yaqin Wu, Pengxin Song, and Fuxin Wang

School of Mechanical Electronic and Information Engineering, China University of Mining and Technology (Beijing), Beijing 100083, China

Correspondence should be addressed to Yaqin Wu; wyqcumt@cumtb.edu.cn

Received 11 December 2019; Accepted 6 March 2020; Published 13 April 2020

Academic Editor: Ricardo Aguilar-Lopez

Blockchain is a new technology for processing complex and disordered information with respect to business and other industrial applications. This work is aimed at studying the consensus algorithm of blockchain to improve the performance of blockchain. Despite their advantages, the proof of stake (POS) algorithm and the practical Byzantine fault tolerance (PBFT) algorithm have high latency, low throughput, and poor scalability. In this paper, a blockchain hybrid consensus algorithm which combines advantages of the POS and PBFT algorithms is proposed, and the algorithm is divided into two stages: sortition and witness. The proposed algorithm reduces the number of consensus nodes to a constant value by verifiable pseudorandom sortition and performs transaction witness between nodes. The algorithm is improved and optimized from three dimensions: throughput, latency, and scalability. The experimental results show that the improved hybrid consensus algorithm is significantly superior to the previous single algorithms for its excellent scalability, high throughput, and low latency.

1. Introduction

With the development of science and technology, blockchain technology has gradually emerged in people's vision and become a focus of research by experts and scholars. It is receiving increasing attentions from industry and academia. People have been aware of the unique remarkable innovation that the blockchain technology can bring about and have invested in researches on business aspects such as finance, insurance, and traceability, which further accelerates maturity and industry landing of the blockchain technology [1, 2]. Currently, the fledgling blockchain technology is developing in China; the market competition pattern and the independent intellectual property system are initially formed. Meanwhile, relevant technologies and applied ecology are forming in areas, such as energy, medicine, and libraries [3–5]. The blockchain technology is a new kind of distributed infrastructure and computing paradigm in which the chain data structure is used to verify the data structure and store data, the distributed node consensus algorithm is used to generate and update data, the cryptography way is used to ensure the security of data transmission and access, and the intelligent contracts composed of automated script code are used to program and manipulate data.

Extensive researches have been conducted on the blockchain. Larimer et al. proposed the delegated proof of stack (DPOS) algorithm [6] that utilizes the entrusted voting of stakeholders to elect multiple super nodes and solves the consensus problem in a democratic and fair way. Wood et al. proposed the proof of authority (POA) algorithm [7] which is essentially an optimized equity proof model. Identity authentication is used as the form of equity to replace digital currency. Authentication is authorized by a preapproved set of nodes to authenticate transactions and blocks in their respective networks. To ensure network efficiency and administrative security, there are usually only a few (typically fewer than 25) validation nodes. Bowers et al. proposed the proof of retrievability (POR) algorithm [8] which is a form of proof from the file system to the client to prove the integrity of the file. Compared with other consensus algorithms, the POR algorithm is of lower communication complexity. Castro and Liskov proposed the practical Byzantine fault

tolerance (PBFT) algorithm [9]. As an algorithm based on a state machine copy replica, the PBFT algorithm provides a fault tolerance guarantee that the failure node does not exceed $(n-1)/3$ under the premise of ensuring safety and activity [10].

This work is aimed at studying the blockchain consensus algorithm and proposing an improved hybrid consensus algorithm that combines advantages of the PBFT algorithm and the proof of stake (POS) algorithm. The proposed improved consensus algorithm boasts both the low latency and high throughput of the PBFT algorithm and the scalability and fairness of the POS algorithm. The main contributions of this work are as follows:

(1) Mainstream blockchain consensus algorithms were studied. Their advantages and disadvantages and specific application scenarios were compared by means of data analysis. The POS algorithm features a consensus time latency and low throughput, while the PBFT algorithm needs to specify the number of nodes in the whole network when running. For a P2P network, the number of nodes does not always remain constant, and nodes will continuously join or exit the network. However, the current number of nodes is critical to a PBFT algorithm, and the number of wrong nodes can lead to consensus failures or even data inconsistencies.

(2) Consensus algorithms were analyzed. An improved hybrid consensus algorithm was put forward by combining advantages of the PBFT algorithm and the POS algorithm. It reduces the number of consensus nodes to a constant value by means of verifiable pseudorandom sortition and conducts transaction witness between nodes. Finally, the improved algorithm was tested and verified from three dimensions, namely, throughput, scalability, and latency. The results reveal that the improved hybrid consensus is pretty advantageous in the three dimensions.

2. Theories

2.1. Overview of Consensus Algorithm. A consensus algorithm is a set of rules that govern the proper functioning of a distributed system. These rules define the basic functions of different parts, the way they cooperate, and the essential conditions required for functioning properly. A consensus algorithm describes the rules to be followed to reach a consensus and usually describes what operations should be performed under what conditions in detail. For example, the proof of work (POW) algorithm defines the following rule: as long as a node can calculate the block that conforms to the target value, it can be verified by the whole network. Generally speaking, a consensus algorithm is a process of solving data synchronization between the nodes which do not trust each other in a distributed system.

2.1.1. POW Algorithm. The POW algorithm [11] was proposed before the appearance of cryptocurrencies and was first used in antispam scenarios. It usually consists of hashing and validation functions. The input of a hash function is a character or a file with a variable length, while its output is a hash value with a constant length. Since many raw and intermediate data are abandoned in the hash process, the hash function is irreversible. The results of hash functions are highly random to ensure fair competition between nodes. Validation functions are used for defining a hash value range that conforms to the rules so as to measure the workload. The value ranges of validation functions are dynamically adjusted to avoid the problems caused by the increase or decrease of network hashrate so that the POW algorithm can run stably and smoothly:

$$\begin{cases} \text{hash}(T) = \text{hash value}, \\ \text{verify}(\text{hash value} \geq \text{target value}). \end{cases} \tag{1}$$

The POW algorithm shown in Eq. (1) relies on mathematical problems that are easy to be verified but difficult to be solved. According to a hash function hash: $N \longrightarrow N$, x can be solved so that $\text{hash}(T + x) \geq \text{target value}$ and T are the Merkel roots of the business lists. Target value is a dynamically adjusted value called the difficulty value; x represents variable parameters, such as trading data and verification random number. Each block in the blockchain with workload proof will contain a solution to solve this problem. The total difficulty of the blockchain is the sum of the difficulties of all blockchains, as shown in the following equation:

$$D(l) = D\left(\sum_i l_i\right) = \sum_i D(l_i). \tag{2}$$

In the process of forking, the highest blockchain is the one with the greatest total difficulty, as shown in the following equation:

$$\Phi(l_1, l_2, \ldots, l_n) = l_m, \quad \text{where } m = \arg\max_{i \in 1,\ldots,n} \left(D(l_i)\right). \tag{3}$$

In bitcoin [12], for example, the hash function is SHA-256, and the validation function is to compare the hash value with the target value until the hash value becomes smaller than the target value. The package equity of the block is obtained by solving the block that meets the requirements. Each node can merely carry out hash value collision by constantly changing variable options such as the random number until the value satisfying the condition is acquired.

The POW algorithm is a computation-intensive algorithm that consumes considerable energy and resources. Every year, the bitcoin network consumes about 57.6 TWh of electricity, equivalent to the annual electricity consumption of Colombia [13]. In addition, the competition for bitcoin mining is continuing, which means more electricity will be consumed. In spite of its vast energy consumption, this mechanism can guarantee the security and trust of data. In fact, the POW algorithm has some other inherent defects. The huge energy consumption leads to a low degree of personal participation and turns the bitcoin network into an organizational or operational mining pool. As a result, the

bitcoin network gradually tends to be centralized, contrary to the original idea of decentralization.

Excessive concentration of hashrate will bring the risk of "51% hashrate attack." The so-called "51% hashrate attack" does not mean a specific 51% hashrate. Instead, it refers to the fact that if a node holds more than half of the network hashrate, it will pose the risk of attacking the network as it is capable of tampering with the block data by virtue of its own hashrate. However, due to the large number of nodes in the bitcoin network and their distribution all over the world, it is hard to obtain more than half of the hashrate of the whole network, and it is extremely difficult and expensive for individuals to tamper with the data.

2.1.2. POS Algorithm. Different from the POW algorithm, the POS algorithm need not obtain the rights and interests of block packaging through hashrate. Instead, it is conducted in the form of mortgage tokens, and the node with the most security interests will obtain the rights and interests of the next block packaging [14]. The biggest advantage of the POS algorithm is that it does not need an intensive operation. Compared with the POW algorithm, the POS algorithm not only consumes less energy, but also functions more efficiently. Moreover, lower energy consumption makes participation cheaper and allows more people to join the decentralized network.

The POS algorithm has a collection of equity holders. Each time the algorithm selects an equity holder from the collection as a witness through the pseudorandom algorithm. The witness is endowed with the right to produce a block based on the greatest height of the current blockchain. If the witness fails to produce the block within the given time, the second witness will be chosen. The POS algorithm can provide a relatively fair solution. In the pseudorandom algorithm, the probability that an equity holder is selected as a witness increases linearly according to its weight of equity mortgage. That is, an equity holder who owns more equity mortgage is more likely to be selected as a witness. If one node has 10 times the equity of the other, it is going to get 10 times the probability of being selected. POS means that blocks are all packaged by equity holders. If an equity holder is verified to have violated the rules of double mining or double signature, the mortgaged tokens and equity will be confiscated.

2.1.3. PBFT Algorithm. The PBFT algorithm provides the fault tolerance of $|(n-1)/3|$. It aims to solve the problem of inefficiency of the original Byzantine algorithm and can be applied in the actual scenario. In equation (4), n is the total number of nodes directly participating in the consensus in the network and f is the maximum number of malicious nodes allowed to be tolerated. The algorithm requires the participation of at least $3f+1$ nodes. In the extreme case, f nodes can be faulty nodes (different from malicious nodes). To enable the number of normal nodes to be greater than that of malicious nodes $n-2f>f$, it is necessary to satisfy $n>3f$, and at least $3f+1$ nodes are needed. Since at most f malicious nodes exist in the network which contains $3f+1$ nodes, the algorithm can still guarantee consistency and correctness of the data:

$$\begin{aligned} n &= 3f+1, \\ f &= \frac{n-1}{3}. \end{aligned} \tag{4}$$

The three phases of agreement among four nodes are described in Figure 1, where Node 4 is the faulty node and Nodes 1–3 are the normal nodes. First, the client initiates a request, and after receiving the request, Node 1 sends a preprepared message to the other nodes. If the other nodes agree to the request, they will enter the prepare phase; otherwise, they will send a rejection message back. In the prepare phase, if preprepared messages are received from more than $2f$ nodes, the preparation will be completed and the commit phase will be entered. In the commit phase, the commit information is broadcast to other nodes. If committed messages are received from more than $2f$ nodes, the commit phase will be completed, and more than half of the nodes will reach an agreement. Finally, the information will be replied to the client after being processed. The algorithm has set a corresponding time specification for each procedure in the process of reaching an agreement. If the task of this phase fails to be completed within the specified time, the current round of consensus terminates for timeout.

Figure 1 shows that the network message transmission between nodes is very frequent in the PBFT algorithm. As the number of nodes increases, a large number of consensus messages are transmitted through the network, leading to the congestion and even unavailability of the network. The PBFT algorithm needs to specify the number of nodes in the whole network when running. For a P2P network, the number of nodes does not always remain constant, and nodes will continuously join or exit the network. However, the current number of nodes is critical to a PBFT algorithm, and the number of wrong nodes can result in consensus failures or even data inconsistencies. Due to the uncertain number of network nodes and the consumption of network message, the PBFT algorithm cannot be used in P2P networks with many nodes.

2.1.4. Raft Algorithm. The Raft algorithm, a consistency algorithm proposed after the Paxos algorithm, is easier to be understood and implemented than the Paxos algorithm [15]. It is a voting-type consistency algorithm that guarantees data consistency through log synchronization. In order to enhance its comprehensibility, the Raft algorithm separates the process of achieving consistency into four parts: leadership election, log synchronization, member change, and security assurance.

In the Raft algorithm, nodes play different roles, including leaders, candidates, and followers, in distributed systems on the basis of their different states. The nodes play switched roles and undertake different tasks according to the change of system status. The specific role transition is disclosed in Figure 2.

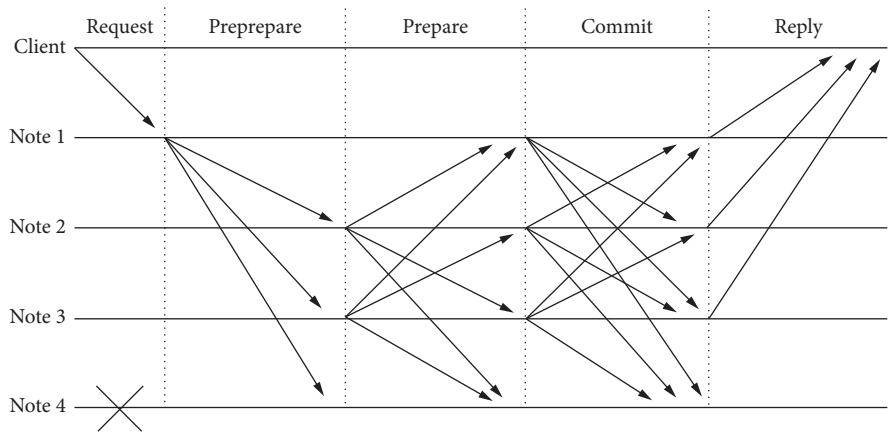

Figure 1: Flowchart of PBFT algorithm.

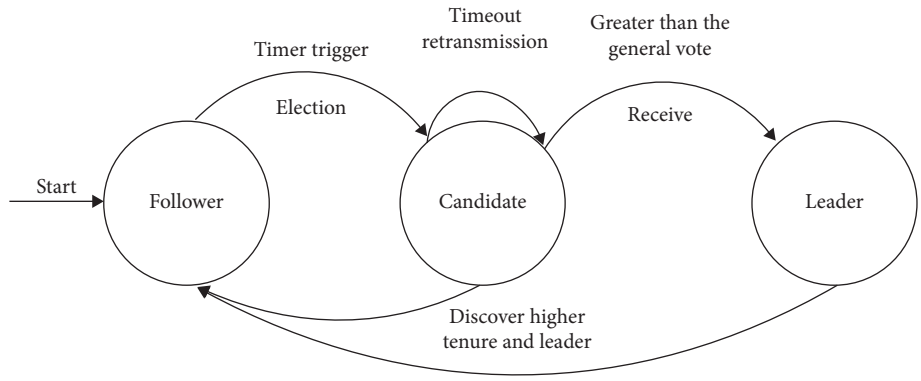

Figure 2: Schematic diagram of role transition character transformation in the Raft algorithm.

The running process of the Raft algorithm resembles an election. Specifically, each node can become a candidate who needs to convince most voters to vote for it. Once more than half of the voters support it, it becomes the leader. Each node in the Raft algorithm can be a candidate to send requests to other followers. If the candidate receives more than half of the followers' responses, the election is successful, and the candidate becomes a leader. Meanwhile, a log is copied to the followers. If the leader node is lost or goes down, the election will be reconducted in accordance with the previous steps. Each node is equipped with a timer for determining whether the leader node is normal according to the heartbeat delay. If the heartbeat times out, it means that the leader node is lost and fails to provide services. In this case, the node will convert its role into a candidate and participate in the leader election of a new term.

2.1.5. Algorithm Comparison. Common consensus algorithms in the blockchain include POW, POS, delegated proof of stake (DPOS), and PBFT. These proof algorithms have large performance bottlenecks. Both the early distributed consistency algorithms and the current blockchain consensus algorithms are used for solving the problem of data inconsistency. Of the two types, the former are mostly applied to private networks or controlled network environments, and most of them are only targeted at dealing with node network latency or other hardware failure problems, without considering the problem of malicious hacking or attack to the node data.

As shown in Table 1, the Raft algorithm does not support Byzantine fault tolerance. Besides, it strictly restricts the entry and exit of nodes, and its network is of constant scale and poor scalability. On the contrary, proof algorithms such as POW and POS need not submit their own conclusions to the other nodes in the network. They reach a consensus using proof, and other nodes can perform verification. In addition, proof algorithms, which are public network-oriented, possess relatively lower requirements for participants, allow participants to be anonymous, support Byzantine fault tolerance, and enable the entire network to reach a consensus within minutes. Nevertheless, the POW algorithm mining consumes considerable electricity, and it has the disadvantages of high latency and low throughput. For all its low electricity consumption, the POS algorithm has a long recognition period and low throughput, and it is dependent on its own "digital currency." Although the DPOS algorithm is also less electricity consuming, it is more complicated because the concept of entrusted equity is introduced in the algorithm and the representative node is selected by means of manual voting. The notes are a low participation degree, so the algorithm is not completely decentralized. Although

Table 1: Comparisons of consensus algorithms.

Algorithm	Billing right	Advantage	Disadvantage	Representative application
POW	Competing competition	Decentralization	High consumption, high latency	Bitcoin
POS	Equity competition	Low resource consumption	Low participation	Nxt
DPOS	Equity election	High throughput, high latency	Producer evil	EOS
PBFT	Leader node	High throughput, high latency	Poor scalability	Fabric
Raft	Leader node	High throughput, high latency	No Byzantine fault tolerance	Etcd

PBFT supports Byzantine fault tolerance and boasts high throughput and low latency, the nodes in the whole network are limited. Meanwhile, its scalability is poor, and excessive nodes will lead to a large number of consensus messages and unavailability of the nodes.

2.2. Algorithm Design and Improvement

2.2.1. Formal Definition and Analysis. A blockchain is essentially a state machine driven by transactions, and the system state ($s \in S$) is a collection of stored data states. An operation to change a data state is called a transaction. They are defined as follows:

(i) Transaction state machine: a transaction state machine is a quad (T, S, g, δ) in which T means a collection of transactions; S means a collection of states; $g \in S$ means the initial state; δ means a state transition function, δ: $S \times T \longrightarrow S$. The semantics of this transaction state machine are a discrete migration system $(S, g, \longrightarrow)$ in which $g \longrightarrow \in S \times S$ represents the migration relationship.

(ii) Transaction log: a transaction log, called a ledger in the blockchain, is a set of transactions with abstract data types constructed recursively. It is defined as follows:

$$\begin{cases} l = \Gamma(T_t), \\ l = l1 + l2, \end{cases} \tag{5}$$

where $T_t \in 2^T$ represents a set of transactions; $\Gamma \in 2^T \longrightarrow L$ is a function for constructing a set of transaction construction logs; L is a collection of logs; $\times$: $L \times L \longrightarrow L$ represents the merging of two logs into one. It is noteworthy that a log usually represents a set of transactions rather than a data state. In the blockchain, all transaction operations are orderly and dependent. Modifying one of the transactions requires reconstructing a transaction log, which increases the cost of data tampering. Multiple valid log sets can be constructed based on the received transaction log set, but the variation of transaction order will result in the inconsistent system state, which is often referred to as "data forking."

(iii) Data bifurcation: let $\{T_t, T_t'\} \in 2^T, T^t \subseteq T_t'$. If $l = \Gamma(T_t)$, $l' = \Gamma(T_t')$ and $l \leq l'$ ($\leq$ represents the prefix relationship) are not satisfied, then data forking occurs between l and l'. According to the semantics of the transaction state machine, if no forking occurs, each node will eventually enter the same state. If forking occurs and each node still enters the same state, the case is called pseudo-forking. When forking occurs, each node needs to choose one of the many forking chains in accordance with the established rule. This established rule is the consensus algorithm.

(iv) Consensus algorithms: Φ: $2^T \longrightarrow L$ is a function that receives a collection of transaction logs and returns a unique collection of ordered logs that conform to established rules. A good consensus algorithm should have a high convergence rate which can be achieved based on a reduction of forking probability and a strong ability to prevent malicious attacks.

Assuming that there are c nodes in the system participating in transaction log production, $M = |L|; N = |S|; M_i = |L_i|$, where $L_i = \{l | f(l) = s_i, s_i \in S\}$, then the probability of pseudoforking occurrence is P_f, as shown in the following equation:

$$P_f = \sum_{i=1}^{N} \left(\frac{M_i}{M}\right)^C - \frac{1}{M^{C-1}}. \tag{6}$$

As can be seen from equation (6), the optimization of a consensus algorithm by reducing the probability of pseudoforking can be achieved in two ways. The first is to change the structure of transaction log, that is, to divide the equivalent class of transaction log and construct a log structure with a low forking probability. The second is to reduce C, that is, to reduce the probability of forking and promote the efficiency of consensus by reducing the number of nodes participating in the consensus. The second way is adopted to improve the existing consensus algorithm in this paper.

2.2.2. Algorithm Improvement and Implementation

(1) Improvement of POS and PBFT Algorithm. The study on consensus algorithms reveals that the POS algorithm not only spends a long time in reaching an agreement but also requires mining operations. Even after the transaction is processed, it still needs a long time to confirm the data until the probability enters the acceptable range. The whole process is too time consuming. Therefore, the POS algorithm is not suitable for scenarios requiring quick feedback of results, such as distributed storage.

The complexity of practical Byzantine messages is $O(n^2)$. In a large-scale network, consensus messages will occupy a large number of system resources. Meanwhile, affected by the transaction processing speed of nodes and the network latency, the time consumed in the process of reaching a consensus will be significantly delayed [16]. The time required to reach a consensus under different transaction delays when $f = \{1, 2, 3, \ldots, 33\}$ and $n = \{4, 7, 10, \ldots, 100\}$ is exhibited in Figure 3 where the ordinate is the time required to reach a consensus and the abscissa is the number of nodes participating in the consensus. It can be seen from Figure 3 that the time required to reach a consensus increases with the growth of the number of nodes.

In the PBFT algorithm, the greater the number of nodes participating in the consensus is, the safer the system is, but the longer the delay is [17, 18]. Although the POS algorithm is strongly expansible without restrictions on network nodes, its operation confirmation relies on the confirmation of subsequent nodes and there exists a risk of rollback [19, 20]. Therefore, the POS algorithm is improved by integrating advantages of the PBFT algorithm. Specifically, the nodes directly participating in the Byzantine agreement are controlled in a small fixed range. The improvement can not only ensure low latency and high throughput of the PBFT algorithm but also enable fair participation of vast network nodes into the consensus so that the network scale is not limited by the PBFT algorithm.

Equity refers to the proportion of value provided by a single node in the blockchain network. The specific algorithm for calculating the value can be selected in accordance with the actual scenario, and the value can be comprehensively evaluated from multiple perspectives such as whether the node is trustable, network latency, hard disk free space, CPU utilization rate, and response latency. In the existing blockchain applications, the concept of digital currency is usually introduced to represent the equity for the convenience of calculation, transfer, and exchange.

"Dynamic" is mainly reflected in the following two aspects: (1) all nodes in the whole network can freely join or exit the network; that is, the number of network nodes can be dynamically expanded or reduced without affecting the system consensus dynamic; (2) the witness node changes dynamically and is dynamically designated by the POS algorithm according to its security interest and whether it is a malicious node.

In light of the above ideas, the algorithm is optimized. The network nodes are divided into witness nodes and ordinary nodes. Among them, witness nodes are selected from ordinary nodes by the pseudorandom algorithm of POS at intervals and are granted the right of witness to initiate the Byzantine consensus and package the block. Ordinary nodes are responsible for synchronizing and verifying block data and providing storage space, but they do not participate in the Byzantine consensus. Witness nodes are selected asynchronously. The next round of witness nodes is selected in advance. In other words, the switch of witness nodes can be completed in a short time without waiting for the asynchronous result of POS. The process is presented in Figure 4.

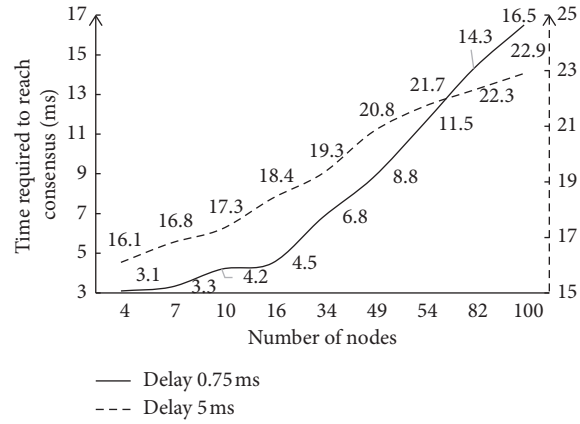

Figure 3: The time required to reach consensus and the number of nodes.

(2) Implementation of Decentralized Verifiable Cryptographic Sortition

(1) Selection and witness stages in algorithm: based on the previous planning, the algorithm can be divided into two stages. The first stage is the "sortition stage" in which the POS algorithm selects a group of nodes as witness nodes through the pseudorandom algorithm based on node equity. The second stage is the "witness stage" (Figure 5), in which the witness nodes begin to process the received operational data, initiate the consensus, and reach an agreement with other witness nodes in accordance with the current status. Finally, the witness nodes package the block and synchronize the data to other nodes. If a witness node fails to reach a consensus due to repeated downtime, timeout, inconsistent data, etc., it will be switched in advance.

(2) Recognition of consensus nodes: the algorithm manages the network nodes through dynamic selection so that the number of nodes participating in the Byzantine consensus remains constant and each node owns a fair chance to participate in data packaging. The improved algorithm still maintains the advantages such as low latency and high throughput of the PBFT algorithm. The result of the request is returned synchronously, but it needs to wait for the block confirmation of subsequent nodes.

The key to the algorithm is how to select consensus nodes accurately and efficiently. The decentralized verifiable cryptographic sortition is used to process this selection. Consensus nodes are randomly selected from many nodes according to their weight, and the cryptographic sortition algorithm [21] randomly selects a subset of nodes according to their weight. The nodes participating in the sortition need to have a pair of public-private keys (*PK* and *SK*) for proving to other nodes that they have been selected. Supposing that there is a set of nodes with a weight of w_i, and the weight of all nodes in the whole network is $W = \sum_i w_i$, then the probability that the node i can be selected is P, $P = w_i/W$.

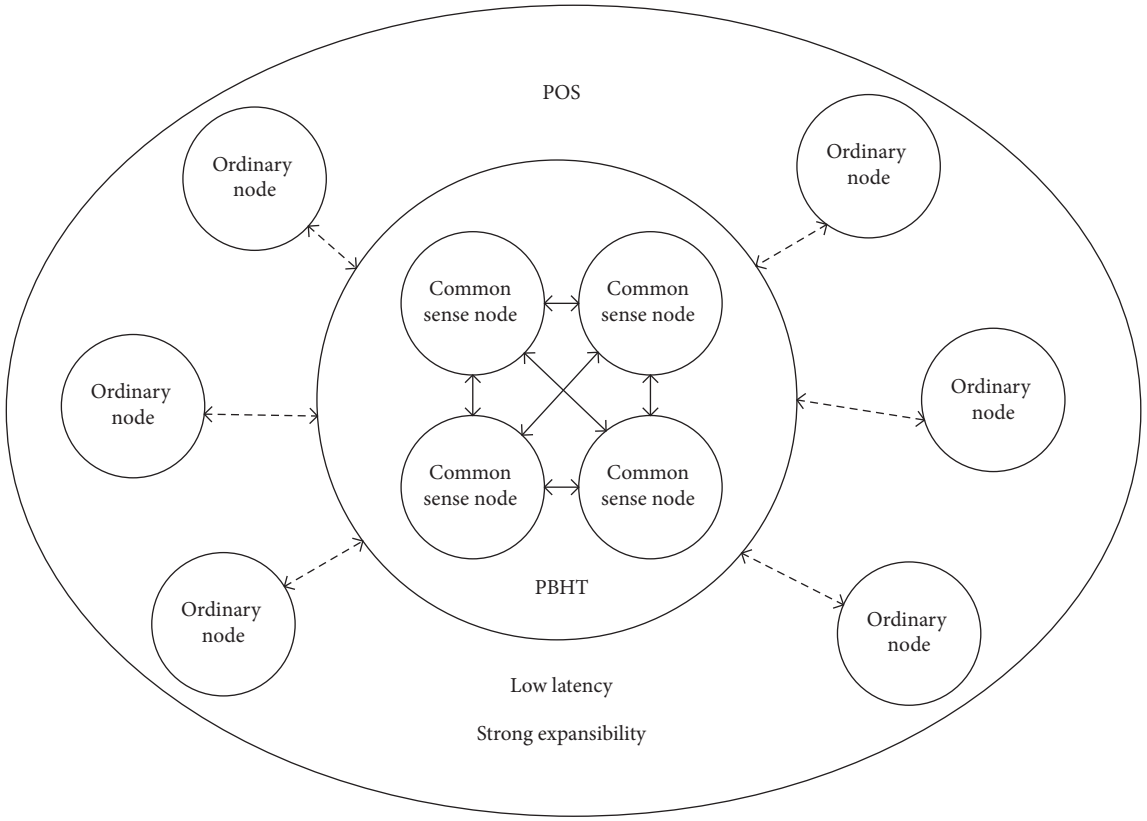

Figure 4: Node sortition for participating in the consensus.

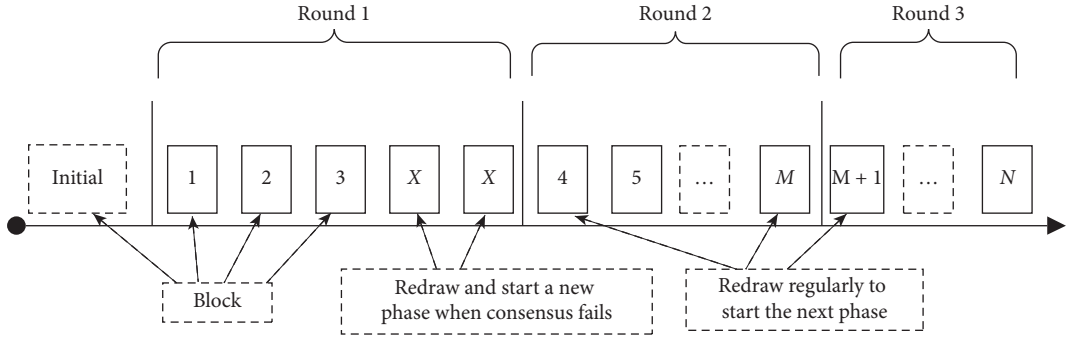

Figure 5: Witness the node switching.

That is to say, the weight of a node is proportional to the probability of its being selected.

The implementation of the cryptographic sortition algorithm depends on the verifiable random function [22]. Given a value x, the holder of the key needs to calculate the value of the function $y = F_{sk}(x)$ and verify $P_{sk}(x)$. Each node can be verified by the public key $PK = g^{sk}$. The algorithm is implemented in the following steps.

First, a pair of public-private keys (PK and SK) is generated to calculate random numbers, as shown in the following equation:

$$(PK, SK) = \text{generate}_{\text{key}}(). \tag{7}$$

Next, the hashing operation is performed with the private key and the result is obtained. The input and output values are represented by x and r, respectively. For the same input value, the output result is determined. The proof value with the private key is calculated and expressed with pi:

$$\begin{cases} r = \text{hash}_{\text{VRF}}(SK, x), \\ \text{pi} = \text{prove}_{\text{VRF}}(SK, x). \end{cases} \tag{8}$$

For verifiable random functions (VRF), anyone can directly use the proof pi to verify the result, and the equation is as follows:

$$r = \text{proof}_{\text{VRF}}(\text{pi}). \tag{9}$$

Taking r as the condition, pi is substituted into equation (9) to obtain

$$\text{hash}_{\text{VRF}}(SK, x) = \text{proof}_{\text{VRF}}(\text{pi} = \text{prove}_{\text{VRF}}(SK, x)). \tag{10}$$

The conclusion is

$$\frac{\text{True}}{\text{False}} = \text{Verify}_{\text{VRF}}(PK, x, \text{pi}). \tag{11}$$

For the sortition algorithm, each round of sortition requires a random number seed which can be repeatedly and publicly calculated and can be verified. The seed of random number is obtained according to the random number of the previous round. That is, the random number seed of the r round is determined based on that of the $r-1$ round. The equation of random number seed transfer is shown in the following:

$$\langle \text{seed}_r, \pi \rangle \longleftarrow \text{VRF}_{sk_u}(\text{seed}_{r-1} \parallel r). \tag{12}$$

In equation (12), π, which is selected by u during random number calculation, is used for VRF proof. Even if u may be a malicious node, the generated seed is still a random value. For $r = 0$, the initial participants use the distributed random function to generate distributed random numbers.

(3) Improved Algorithm Flowchart. The process of the improved algorithm is displayed in Figure 6. At the beginning of the program, it is initialized into Round 1 and node sortition is conducted. A distributed verifiable random function is used to draw lots among ordinary nodes, after which the selected nodes are endowed with the power of witness nodes. Then, the witness nodes reach a Byzantine consensus and package the transaction into a block. Other witness nodes and ordinary nodes are responsible for verifying the block, including transaction data, block data, public-private key signature, and sortition random number. If the block succeeds in passing the verification, it will be broadcast to the whole network and the next block will be verified; otherwise, it will be subjected to timeout detection. A fixed timeout limit is set for each round of consensus. If the timeout limit is exceeded, the witness node will be reselected through drawing lots; otherwise, the witness nodes will try to reach a consensus again.

3. Results and Discussion

In order to verify the improved algorithm, the length of each transaction is set to 50 bytes after serialization, and a transaction is filled with characters if its length is smaller than 50 bytes. When the number of test nodes is large, the CPU and network consumption of the computer are relatively high, and most computer resources are spent on signature verification. To solve this problem, the system adopts a sleep mode, testing and verifying the consensus algorithm from three dimensions: throughput, latency, and scalability.

3.1. Throughput Test. Transaction throughput is an important performance metric for measuring distributed systems [23]. The throughput of a system is usually determined by the number of concurrent transactions and the number of transactions per second (TPS). The time spent on a transaction is the time consumed from the client request to the result returned by the server. The overall performance of the system depends on the module with the weakest transaction processing ability in the system. In a distributed system, TPS refers to the number of transactions that the system can handle in a period:

$$\text{TPS}_{\Delta_t} = \frac{\text{Sum}_{\Delta_t}(\text{Transactions})}{\Delta_t}, \tag{13}$$

where Δ_t is the time spent on transaction processing; $\text{Sum}\Delta_t(\text{Transactions})$ refers to the total number of transactions processed in the Δ_t period; and TPS_{Δ_t} refers to the value of TPS in the Δ_t period.

Bitcoin and Ethereum are representative projects of the POW algorithm. Therefore, in this test, the TPS value of the POW algorithm was obtained by counting block data of Bitcoin and Ethereum. Bitcoin and Ethereum nodes were set up to synchronize block data. Detailed blockchain information was obtained, and transaction data were counted through the RPC interface. The data were counted every day. On each day, the number of newly generated blocks and the number of transactions included in each block were calculated, so was the TPS average. The TPS data in the past three months (2019.1.22–2019.4.21) are listed in Figure 7 where the TPS value of Bitcoin was smaller than 5 and fluctuated between 2.2 and 4.65 while that of Ethereum fluctuated between 5 and 9.

Nxt is the representative application of the POS algorithm [24]. The Nxt node test network was built for acquiring the number of transactions in the block and in each blockchain every day, based on which TPS was calculated. The TPS data in the past three months (2019.1.22–2019.4.21) are listed in Figure 8 where the TPS value of Nxt fluctuated between 30 and 70.

Due to the differences among the consensus algorithms, the intervals between blocks differ notably. To acquire more accurate test data, the total number of transactions and the TPS value were counted, with 1 minute taken as the test interval and 60 minutes taken as a cycle. The transaction construction client was implemented, in which each transaction was filled with 50 bytes and sent to the service node at the speed of 3000 TPS. During the test, the client need not pay attention to whether the transaction was successful. After the test, the TPS value was uniformly counted and calculated based on the block data. The Raft algorithm, the PBFT algorithm, and the improved algorithm were subjected to the test. The test results are disclosed in Figure 9.

As shown in Figure 9, the TPS of the Raft algorithm fluctuated between 1500 and 2000; that of the PBFT algorithm oscillated with large amplitudes between 700 and 1500; and that of the improved algorithm oscillated with small amplitudes between 1100 and 1400. The comparison

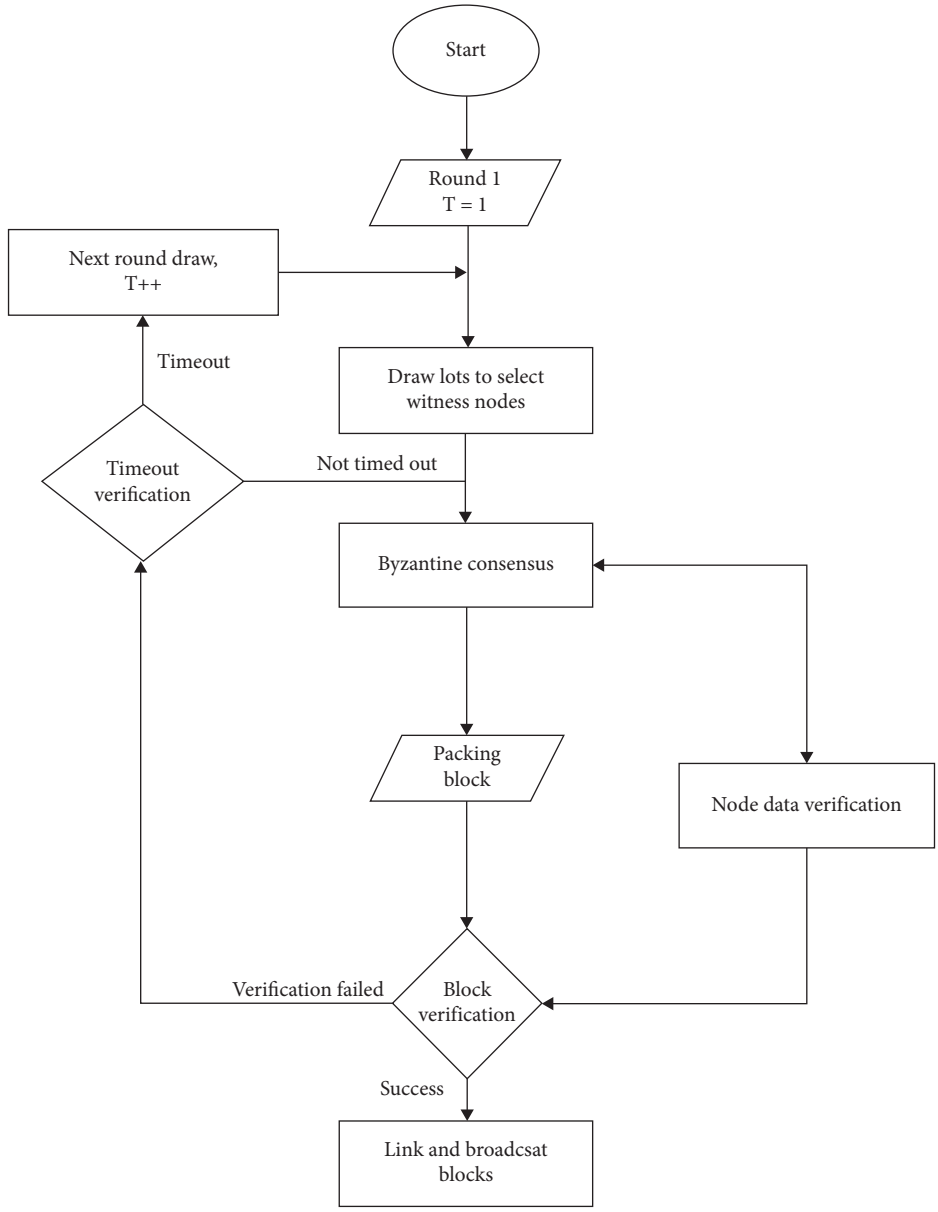

Figure 6: Algorithm flowchart.

among the three algorithms reveals that the Raft algorithm has larger TPS values than the other two, as it is not characterized by Byzantine fault tolerance. Nevertheless, it can achieve higher performance, because leader nodes exist while other nodes synchronize data as redundant nodes without causing network consensus costs in its running process. The comparison between the PBFT algorithm and the improved algorithm suggests that the average TPS values of the two just differ slightly, but the TPS oscillation amplitude of the latter is much smaller than that of the former. It can be known through analysis that, during the running of the PBFT algorithm, the online or downline of nodes significantly affects the consensus, which is likely to lead to network latency or consensus failure.

The TPS values of various algorithms are presented in Figure 10 for comparison. The Raft algorithm has high performance, but it does not support Byzantine fault tolerance. The PBFT algorithm and the improved algorithm are comparable in terms of performance, but the latter boasts a smaller TPS fluctuation range and is more stable.

3.2. Scalability Test. Scalability, divided into horizontal expansion and vertical expansion, is critical to a distributed system [25]. Horizontal expansion refers to promoting the overall performance of the system by adding the number of machines. Its bottleneck lies in the limitation of multi-machine management. Vertical expansion refers to enhancing the overall performance of the system by improving the performance of a single machine. Its bottleneck is that there is an upper limit on the performance of a single machine. Since distributed systems are usually composed of

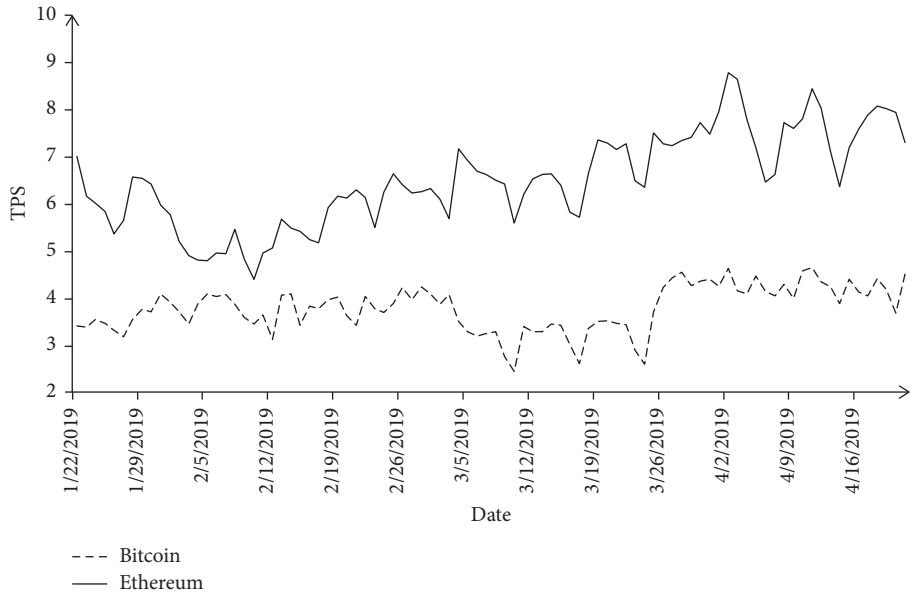

Figure 7: TPS comparison between Ethereum and Bitcoin.

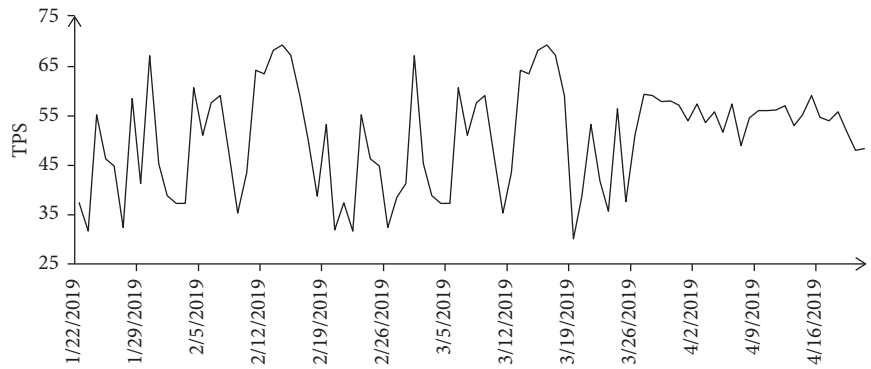

Figure 8: TPS value of Nxt.

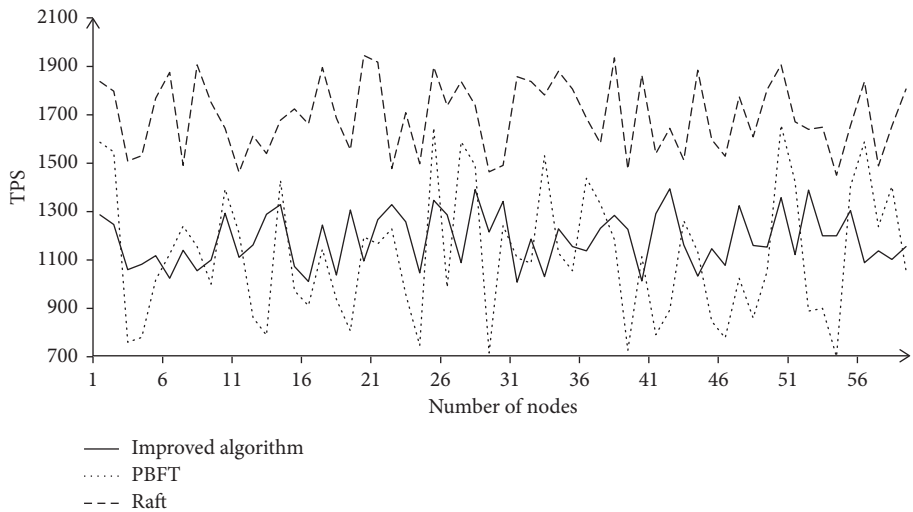

Figure 9: PBFT, Raft, and improved algorithm TPS comparison.

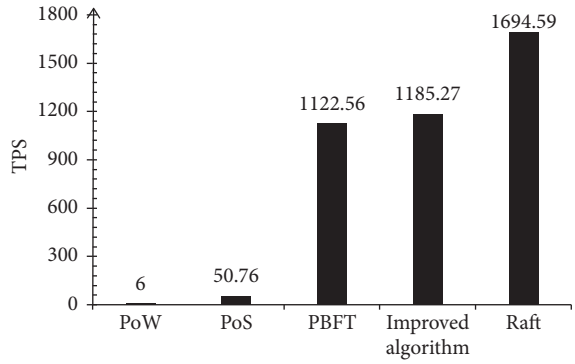

Figure 10: TPS under different consensus algorithms.

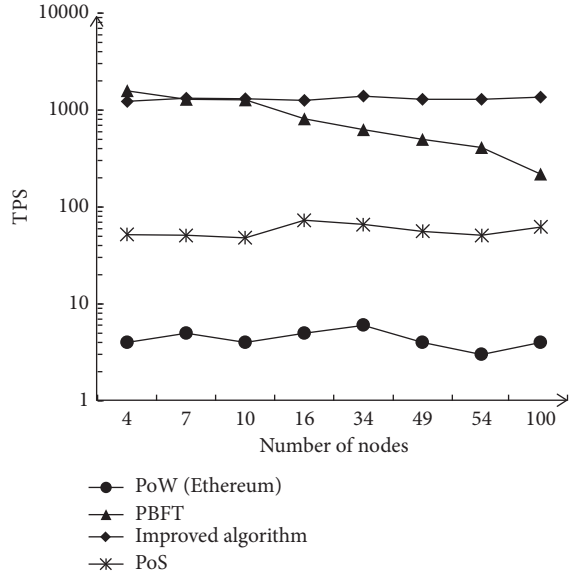

Figure 11: Influence of the number of nodes on the TPS.

inexpensive and efficient nodes, horizontal expansion is generally adopted to promote system performance.

Excellent scalability enables a system to make dynamic adjustment according to the current task status. When the system is under a high load, it can raise the processing capacity by increasing the number of nodes. When the system is under a low load, it can improve the utilization by reducing the number of nodes. Whether a distributed system possesses strong scalability depends on whether the consensus algorithm supports the dynamic joining or exiting of nodes.

In the scalability test, the influence of the increase or decrease in node dynamics during the operation of the consensus algorithm was evaluated by measuring TPS. Just as the throughput test, the client still sent the transaction filled with 50 bytes at the speed of 3000 TPS. The difference lies in that, during this test, the number of network nodes continuously increased by the timed script, and the TPS under the current number of nodes was counted and calculated in the unit of 1 minute. Meanwhile, the transaction processing performance of the system under different numbers of nodes was analyzed. The POW algorithm, the PBFT algorithm, the POS algorithm, and the improved algorithm were subjected to this test.

As shown in Figure 11, the performance of the PBFT algorithm decreases significantly as the number of nodes increases. The TPS values of the PBFT algorithm are about 1200 and 218 when there are 4 nodes and 100 nodes in the system, respectively, which means that the performance under 100 nodes is nearly 7 times as high as that under 4 nodes. The performances of the POW algorithm, the POS algorithm, and the improved algorithm are relatively stable. The increase in the number of nodes does not greatly affect the performances of the consensus algorithms. It is concluded that the PBFT algorithm has poor scalability, while the other three algorithms have strong scalability.

3.3. Latency Test. Latency is an important indicator for evaluating the performance of a distributed system and measuring communication delays and algorithm delays between nodes in a distributed system [26, 27]. A system with low latency is able to quickly send back the transaction processing result and achieve good user experience, while one with high latency is unable to send back the transaction result processing in time (the result can only be obtained asynchronously) or achieve good user experience. As shown in equation (14), latency of a distributed system usually consists of three parts: the time required for the transaction request data to be transmitted in the network (T_{Request}), the time required for the witness nodes to reach a consensus ($T_{\text{Consensus}}$), and the time required for the processing result to return (T_{Response}):

$$\text{Delay}_{\text{Transaction}} = T_{\text{Request}} + T_{\text{Consensus}} + T_{\text{Response}}. \quad (14)$$

For the purpose of accurately measuring the consensus time, transaction requests are randomly sent to different nodes and broadcast over the whole network. The client end records the timestamp at two moments: when it receives a request and after receiving the feedback. The latency for processing a transaction can be obtained by calculating the difference between the two timestamps.

For the POW algorithm and the POS algorithm, the latency is limited by the program rules. For example, the block-producing times of bitcoin, Ethereum, and Nxt program are regulated to be about 10 minutes, 2 minutes, and 1 minute, which means that it takes an average waiting time of 5 minutes, 1 minute, and 30 seconds for a transaction to be successfully written to the blockchain, respectively. It can be seen that the two algorithms both have long and uncertain latencies ranging from seconds to minutes.

In this test, the latencies of the PBFT algorithm and the improved algorithm were calculated and counted in the unit of 1000 transactions. The client end constructs 50 byte-long transactions which were then sent to the service nodes at random time intervals. Finally, the average transaction processing latencies were obtained according to the results, as exhibited in Figure 12.

It can be observed from Figure 12 that the PBFT algorithm has a low transaction processing latency when there

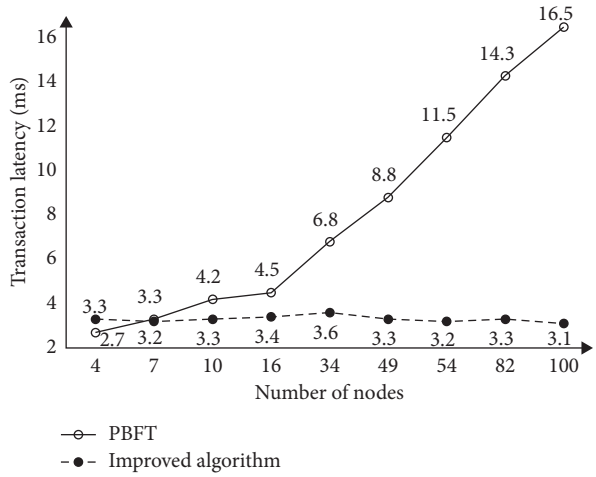

Figure 12: Impact of different number of nodes on transaction latency.

are 4 nodes in the system, but its latency increases as the number of nodes grows. In contrast, the transaction processing latency of the improved algorithm is more stable without varying greatly as the number of nodes grows because its number of witness nodes remains constant at 4 in this test.

In short, the POW algorithm and the POS algorithm have long and uncertain latencies, while the PBFT algorithm and the improved algorithm have shorter latencies. Besides, compared with the PBFT algorithm, the transaction processing latency of the improved algorithm is less affected by the number of nodes.

4. Conclusions

With the continuous development of blockchain technology, consensus algorithms have gradually become a new research hotspot. The pros and cons of a consensus algorithm directly affect the performance and function of the blockchain. A good consensus algorithm should feature good scalability, low latency, high throughput, and decentralization. Based on the existing consensus algorithm, an improved hybrid consensus algorithm based on the PBFT algorithm and the POS algorithm is proposed in this work. It dynamically selects the consensus node in the form of verifiable cryptographic sortition, which not only allows a large number of nodes to participate in the consensus fairly but also ensures the low latency and high throughput of the consensus algorithm. In the future, we will continue to study the blockchain consensus algorithm and improve the consensus algorithm in light of the idea of reducing block forking. Through the improvement and optimization of the blockchain ledger structure, the ledger will be reconstructed in the form of a directed acyclic graph to avoid the performance loss caused by the ledger forking. In this way, the performance of the blockchain consensus algorithm can be improved.

Data Availability

The coding data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

Thanks are due to Dr. Wei Zhao in China University of Mining and Technology (Beijing) for the useful comments on the manuscript preparation. This research was funded by the National Natural Science Foundation of China (Grant no. 51874314) and Fundamental Research Funds for the Central Universities (Grant no. 2011YJ15).

References

[1] C. Pop, T. Cioara, M. Antal, I. Anghel, I. Salomie, and M. Bertoncini, "Blockchain based decentralized management of demand response programs in smart energy grids," *Sensors*, vol. 18, no. 2, p. 162, 2018.

[2] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: a survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[3] M. Andoni, V. Robu, D. Flynn et al., "Blockchain technology in the energy sector: a systematic review of challenges and opportunities," *Renewable and Sustainable Energy Reviews*, vol. 100, pp. 143–174, 2019.

[4] M. B. Hoy, "An introduction to the blockchain and its implications for libraries and medicine," *Medical Reference Services Quarterly*, vol. 36, no. 3, pp. 273–279, 2017.

[5] W. Zhao, Y. Cheng, Z. Pan, K. Wang, and S. Liu, "Gas diffusion in coal particles: a review of mathematical models and their applications," *Fuel*, vol. 252, pp. 77–100, 2019.

[6] D. Larimer, "Delegated proof-of-stake (Dpos)," 2014, https://bitcointalk.org/index.php?topic=558316.0.

[7] G. Wood, "Polkadot: vision for a heterogeneous multi-chain framework," 2016, https://polkadot.network/PolkaDotPaper.pdf.

[8] A. Juels and B. S. Kaliski Jr., "PORs: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 584–597, Alexandria, VA, USA, November 2007.

[9] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," *OSDI*, vol. 99, pp. 173–186, 1999.

[10] G. Bracha and S. Toueg, "Asynchronous consensus and broadcast protocols," *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 824–840, 1985.

[11] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols (extended abstract)," in *Secure Information Networks: IFIP-The International Federation for Information Processing Preneel B. ed*, vol. 23, pp. 258–272, Springer, Boston, MA, USA, 1999.

[12] R. Böhme, N. Christin, B. Edelman, and T. Moore, "Bitcoin: economics, technology, and governance," *Journal of Economic Perspectives*, vol. 29, no. 2, pp. 213–238, 2015.

[13] WorldData.info. Energy Consumption in Colombia, https://www.worlddata.info/america/colombia/energy-consumption.php.

[14] S. King and S. Nadal, "Ppcoin: peer-to-peer crypto-currency with proof-of-stake," 2012, https://decred.org/research/king2012.pdf.

[15] J. Chiefari, Y. K. Chong, F. Ercole et al., "Living Free-Radical Polymerization by Reversible Addition–Fragmentation Chain

Transfer: The RAFT Process," *Macromolecules*, vol. 31, no. 16, pp. 5559–5562, 1998.

[16] H. Sukhwani, J. M. Martínez, X. Chang et al., "Performance modeling of Pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *Proceedings of the 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 253–255, Hong Kong, China, September 2017.

[17] Y. Yuan, X. Ni, S. Zeng, and F. Wang, "Blochchain consensus algorithms: the State of the art and future trends," *Acta Automatica Sinica*, vol. 44, no. 11, pp. 2011–2022, 2018.

[18] C. Cachin, S. Schubert, and M. Vukolić, "Non-determinism in byzantine fault-tolerant replication," 2016, https://arxiv.org/abs/1603.07351.

[19] Y. Gilad, R. Hemo, S. Micali et al., "Algorand: scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, ACM, Shanghai, China, October 2017.

[20] J. Kwon, "Tendermint: consensus without mining: draft V. 0.3," 2014, http://digitalslab.buaa.edu.cn/__local/C/8D/68/C0A97F7766C07EBAE618D8CF318_304F91EF_AAD6F.pdf.

[21] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 120–130, IEEE, New York City, NY, USA, October 1999.

[22] Y. Dodis, "Efficient construction of (distributed) verifiable random functions," in *Proceedings of the International Workshop on Theory & Practice in Public Key Cryptography*, Miami, FL, USA, January 2003.

[23] G. F. Coulouri, D. Jean, and K. Tim, *Distributed System Concept and Design*, Pearson Education, London, UK, 2004.

[24] S. Popov, "A probabilistic analysis of the nxt forging algorithm," *Ledger*, vol. 1, pp. 69–83, 2016.

[25] F. Xu, G. Yang, and D. Ju, "Dsign of distributed storage system on peer-to-peer structure," *Journal of Software*, vol. 15, no. 2, pp. 268–277, 2004.

[26] X. Qin, Z. Han, and L. Pang, "Real-Time scheduling with fault-tolerance in heterogeneous distributed systems," *Chinese Journal of Computers*, vol. 1, pp. 49–56, 2002.

[27] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and iot integration: a systematic survey," *Sensors*, vol. 18, no. 8, p. 2575, 2018.