Journal of King Saud University – Computer and Information Sciences 34 (2022) 8370–8379

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Weighted Byzantine Fault Tolerance consensus algorithm for enhancing consortium blockchain efficiency and security

Hongwu Qin [a,*], Yuntao Cheng [a], Xiuqin Ma [a], Fei Li [a], Jemal Abawajy [b]

[a] Northwest Normal University, Lanzhou 730070, China
[b] Cyber Security Research and Innovation Centre, Deakin University, Geelong, VIC 3220, Australia

ARTICLE INFO

Article history:
Received 16 April 2022
Revised 24 July 2022
Accepted 15 August 2022
Available online 18 August 2022

Keywords:
Blockchain
Consensus Algorithm
Consortium Blockchain
PBFT
Weighted Byzantine Fault Tolerance

ABSTRACT

In blockchain, the consensus algorithm is a core component that governs the trust among the participants in the blockchain activities. However, the exiting consensus algorithms suffer from performance bottleneck such as low throughput, high delay, unstable performance, sustainability issues and vulnerability to targeted attacks. In this paper, we propose a new consortium blockchain consensus algorithm, referred to as Weighted Byzantine Fault Tolerance (WBFT) consensus algorithm that improves system throughput and consensus delay. We introduce a dynamic weighting mechanism for consensus nodes, which enhances the security of blockchain system by weakening the influence of malicious nodes and reduces the probability of malicious behavior. We validate the performance of WBFT experimentally and compare it against Practical Byzantine Fault Tolerance (PBFT) and Reputation-Based Byzantine Fault-Tolerance (RBFT) based approaches. The results show that WBFT substantially outperforms PBFT and RBFT in terms of system throughput, consensus delay and security.

1. Introduction

The invention of Bitcoin (Bitcoin, 2009), a cryptographic digital currency, has led to the development of a blockchain technology. The blockchain offers properties such as transparency, trustworthiness, dependability and immutability through integration of various concepts such as distributed ledger, cryptographic hash, consensus protocol, and smart contract. These properties enable blockchain to offer distributed platform for interested entities to transact without the need for a trusted third party even in the presence of potential malicious participants. The immense potential that the blockchain technology offers coupled with its tamper-proof nature, transparency, and traceability over time has drawn considerable interest from various industries and research communities. As a result, blockchain technology has emerged as one of the important technologies suitable for a wide range of applications such as transportation (Majeed et al., 2021), healthcare (Omar et al., 2019), logistics supply chain (Yang et al., 2019), finance (Gao and Chang, 2020); industrial Internet of Things (Zhang et al., 2020), smart city (Sun and Zhang, 2020); energy trading (Aitzhan and Svetinovic, 2018); cloud computing (Ma et al., 2022) and so on.

In a blockchain, data units, known as blocks, are linked together in chronological order with a chain like structure. For every block, a unique hash value is computed from the transactions in the block and stored in the header of the block. Consequently, any changes to the block will be easily identified because the modification will lead to a different hash value of the block, which provides a strong tamper-resistance to the transaction data. Existing blockchain systems are generally divided, depending on the level of openness, into public blockchain, private blockchain and consortium blockchain. The public chain permits anyone to freely participate in the blockchain activities whereas private chain constraints participation by invitation or permission only (Huang et al., 2020). Participation in the consortium chain is restricted to the members of the consortium as determined by a predefined list of participants. The latter two types of the blockchain are generally known as permissioned blockchains. Consortium blockchain has been widely used in commercial applications. The consensus algorithm is key to the blockchain system (Hu et al., 2020) and mainly responsible

* Corresponding author.
E-mail addresses: qinhongwu@nwnu.edu.cn (H. Qin), 871668647@qq.com (Y. Cheng), maxiuqin@nwnu.edu.cn (X. Ma), 2019221872@nwnu.edu.cn (F. Li), jemal.abawajy@deakin.edu.au (J. Abawajy).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2022.08.017
1319-1578/

for assuring the stability, quality of security and integrity of the blockchain. The efficiency of a consensus algorithm directly affects the security, throughput, availability, and other key indicators of the blockchain system (Wang et al., 2019). A variety of Byzantine Fault Tolerant (BFT) derivative consensus algorithms such as Practical BFT (PBFT) (Castro and Liskov, 1999) are used in consortium blockchain systems. Although exiting consortium blockchain consensus algorithms have improved the operational efficiency of the consortium blockchain, they still suffer from several shortcomings which make them unsuitable for real-time applications (Wang et al., 2019). For example, the performance of PBFT consensus algorithm will decline sharply with the increase in the number of nodes. They also suffer from low throughput, high latency, and low Byzantine Fault-Tolerant rate. Blockchain may also contain one or more malicious nodes that engage in sinister activities such as tampering with the consensus algorithm process, corrupt the transactions, and disrupt the network operation (Yang et al., 2019). However, the BFT based consensus algorithm is vulnerable to malicious nodes because the correct consensus can be reached only when less than one-third of the malicious nodes are presented (Castro and Liskov, 1999). These shortcomings seriously affect the performance of the consortium blockchain and thus impede its wide adoption.

To solve the above problems, this paper proposes a new consortium blockchain consensus algorithm named WBFT. Prior to the commencement of the consensus process, a weighting mechanism is applied to the nodes in the consortium. For a node with malicious behavior, its weight will be reduced, thereby limiting its participation in consensus. Subsequently, the occurrence of malicious behaviors in the system can be reduced. This method also optimizes the communication flow of consensus algorithm and improves the system operation efficiency. We validate the performance of WBFT experimentally. The results show that WBFT outperforms existing algorithms in terms of system throughput, consensus delay and security.

The contributions in this paper can be summarized as follows:

- A new consortium blockchain consensus algorithm named WBFT is proposed. The algorithm introduces a weighting mechanism, which weakens the influence of malicious nodes as a whole, reduces the probability of malicious behavior, and then improves the performance of consortium blockchain system.
- WBFT removes the commit step in PBFT three-stage communication that has no impact on the consensus results, which improves the throughput of the consortium blockchain system.

The rest of the paper is organized as follows. The related work is described in Section II, including a quick overview on blockchain technology and some of the most popular consensus algorithms. The proposed consensus algorithm is described in Section III. The experiments and the discussion of the results are presented in Section IV and Section V respectively. The conclusion and future work are given in Section VI.

2. Related work

Blockchain is a fully distributed network of peers where the peers collectively ensure the validity and consistency of the data and transactions in a decentralized manner. A consensus algorithm is deployed in blockchain to ensure that the peers in the blockchain network reach a collective agreement. A variety of consensus algorithms for blockchain has been put forward and this section will review the existing consensus algorithms.

The comprehensive survey examining the advantages and disadvantages of the existing blockchain consensus algorithms are presented in (Wang et al., 2019; Bamakan et al., 2020). Existing blockchain consensus algorithms can be generally classified into proof-based algorithm and voting-based algorithm. The classical Proof-of-X (PoX) consensus algorithms include Delegated Proof of Work (Gervais et al., 2016) (POW), Proof of Stake (POS) (Proof of stake, 2020), and Delegated Proof of Stake (DPOS) (Yang et al., 2019). These classical algorithms are generally based on a reward system for engaging in the blockchain activities such as appending a block or get involved in the mining work. However, each potential participant is required to solve a complex mathematical puzzle to earn the honor to append a block or get involved in the blockchain mining process. POW is the consensus algorithm used in Bitcoin (Bitcoin, 2009). POW uses hash operation to prevent distributed denial of service (DDoS) attacks (Chonka and Abawajy, 2012). The problem with POW is that it wastes a large amount of computing power and consumes excessive energy (Sun et al., 2020). POS tries to reduce the large amount of resource waste in POW. It takes the percentage of the total number of held COINS as an important reference and includes the time to own COINS to determine the right to keep accounts. However, POS has no cost benefit, which makes it easy to fork. DPOS, on the other hand, adds the election mechanism on the basis of POS and conducts blockchain consensus according to the equity election agent. This method improves the efficiency of consensus. However, it still relies on token mechanism and has few business application scenarios. The Raft consensus algorithm mainly consists of the selection of the leader node, which makes the decisions. However, it can only be applied to private chains (Lei et al., 2019). The PoX consensus algorithms are naturally suitable for public blockchain where any participant can enter or exit the blockchain mining process. However, PoX algorithms suffer from low efficiency, high computational power, and excessive energy consumption.These shortcomings make PoX algorithms very challenging to meet the needs of real-time transactions on the energy Internet (Lei et al., 2019).

The Byzantine Fault Tolerant (BFT) consensus algorithms, such as Raft (Hu et al., 2020), Practical BFT (PBFT) (Castro and Liskov, 1999) and Reputation-Based BFT (RBFT) (Lei et al., 2019); are in the class of voting-based consensus algorithm. These algorithms are appropriate for consortium blockchain or private blockchain. PBFT is a replicator algorithm of state machines and is currently the most widely used algorithm for consortium blockchain. Fig. 1 shows the PBFT three-stage consensus algorithm process with 4 workers nodes and one master node that is elected through a rotation and election mechanism. Also, a threshold F is used for the number of nodes expected to agree on the consensus. In the request phase, the client sends its request message to the master node. Then the master node distributes the request message to other nodes. That is the pre-prepare phase. In the next two stages, prepare and commit, message sending and reply continue between the nodes. In the reply stage, the requestor checks to see if it has

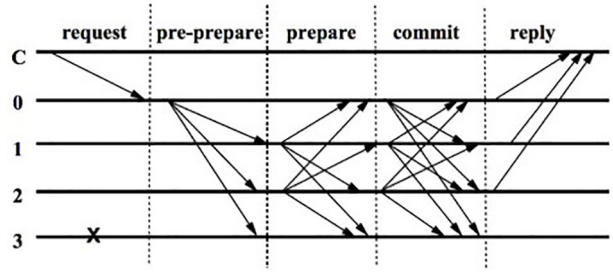

Fig. 1. Workflow of PBFT consensus algorithm with 5 peers. C (requesting node), 0 (master node), 1, 2, 3 represent the peer nodes where peer 3 is considered faulty/malicious node.

received the same response message from at least F + 1 different nodes, and if so, saves it as a consensus result. That means the consensus has succeeded and the consensus result is the outcome of PBFT algorithm (Lamport et al., 2019).

PBFT is superior to other consensus algorithms for commercial use and can significantly reduce the computing waste problem exhibited by the classical blockchain consensus algorithms. It also provides a fault-tolerant guarantee that the failure node does not exceed (N-1) /3 under the premise of ensuring safety and activity. Although PBFT algorithm achieves the coexistence of availability and security to a certain extent, it has several shortcomings. First, since PBFT was not designed specifically for the blockchain at the beginning, its traditional client/server response form has limited application scenarios, which cannot meet the complex application requirements. The traditional PBFT algorithm is not suitable for direct application to the peer-to-peer (P2P) network in the high concurrency blockchain environment (Bhushan et al., 2021). Second, PBFT lacks efficient node joining and exiting mechanism. At present, it is needed to restart the whole system before the messages of node joining and exiting can be perceived. This method has low flexibility and scalability. Third, PBFT does not restrict the malicious nodes, but is compatible with the malicious nodes existing in the system through its fault tolerance (N-1)/3, where N is the maximum number of blockchain nodes in the system. Also, it does not take corresponding intervention measures for the performance impact of the malicious nodes on the system. Finally, three-stage broadcast protocol used in PBFT consumes much bandwidth unnecessarily. Although the pre-prepare stage is single-node broadcasting, the prepare and commit stages are all full-node broadcasting, which consumes much network resources. Business applications have huge nodes and thus it is very necessary to optimize the three-stage broadcast protocol.

Several extensions of the original PBFT consensus algorithm have been proposed in the literature. He and Hou (He and Hou, 2019) proposed e-PBFT in which the main node mechanism in the consensus process is removed and the power is assigned equally to each node in the system. But there is no effective restriction on the malicious behavior of nodes in the system. Miller et al. (Miller et al., 2016) proposed Honey Badger BFT, which can solve the low efficiency problem caused by abnormal network state, but at the same time increase the complexity of consensus communication. SBFT (Gueta et al., 2018) have improved the performance of PBFT algorithm to some extent, but with different entry points and limited promotion dimensions. Unfortunately, PBFT consensus algorithm and its variants are vulnerable to many attacks against the primary node and barely able to detect and eliminate faulty nodes in the blockchain system (Lei et al., 2019).

To address these problems, Lei et al. (Lei et al., 2019) proposed a Reputation- Based Byzantine Fault-Tolerance (RBFT). The system as a whole is similar to PBFT process and based on three stages broadcast protocol as in PBFT, but there are new mechanisms and applications. RBFT removes the view mechanism in the PBFT and sets a fixed period for generating new consensus fast. RBFT also establishes credit value for each node to reduce system failure caused by view switching or malicious behavior. By judging the system nodes through credit value, the discourse power of the nodes with higher credit value can be enhanced, the participation of the nodes with lower credit value can be weakened, and the failure risk of the main node can be reduced. Although RBFT reduces the impact of malicious behavior on the consensus process to a certain extent, it has limited restrictions on malicious nodes, and there is still much room for improvement on the restrictions on the master node. The mechanism of weakening discourse power does not affect the number of malicious behaviors in the system with high probability. The problem of information delay caused by malicious behavior to the system still exists and needs to be improved. As a result, the two above existing methods have the high delay and the low throughput and unstable performance.

To solve the above problems, this paper proposes an improved Byzantine consensus algorithm based on a dynamic weighting mechanism. The details of the proposed algorithm are described in the next section.

3. The proposed algorithm

In this section, we describe the proposed Weighted Byzantine Fault Tolerance (WBFT) consensus algorithm. The network architecture, dynamic weighting mechanism, and three-stage communication optimization deployed in WBFT are described in detail. Table 1 describes the symbols used in the paper and their meanings.

3.1. System model

Fig. 2 shows the overall architecture and workflow of the WBFT algorithm. Unlike the PBFT algorithm that uses the traditional client server model, WBFT is based on peer-to-peer (P2P) network model. P2P network architecture can greatly improve the efficiency of communication. We used Java T-IO framework implementations at the Manager layer to build in the upper layer of system to manage all nodes. By sending regular heartbeat packets, nodes joining and exiting in the system are detected. We reduce system resource wastes by realizing the dynamic states of nodes.

Each node N_i in the blockchain system maintains a base (initial) weight (W) and a running weight (W_i). The running weight of a node represents the degree of trust and the level of activity of the node in the consensus domain. The management node decides whether a node participates in the consensus process according to the threshold of the node. The threshold value TH_i of node N_i is computed as follows:

$$TH_i = \frac{W_i}{W} \tag{1}$$

The nodes with higher TH_i are selected for message consensus, and the nodes with lower TH_i will have limited participation in the consensus. This leads to a rapid consensus in the consortium chain, which has a positive effect on the final consensus efficiency. In order to further improve the communication efficiency between the nodes in the consortium chain as well as eliminate faulty nodes and malicious nodes, WBFT uses a feedback mechanism in which the nodes in the system express their opinion regarding the behavior and the activity of the nodes surrounding them. Based on the

Table 1
Symbols used in the paper.

Symbols	Explanation
N	Number of nodes in the system
M	Number of nodes participated in the consensus process
N_i	The ith node in the system
W	The base (initial) weight of a node
W_i	The running weight of the ith node
TH_i	The threshold of the ith node
NB	The set of malicious behaviour types of nodes
T_i	The consensus completion time of the ith node
CCT	The set of T_i
FI	Feedback information of all nodes participated in the consensus process
P	The set of penalty level
bh_i	The behaviour type of the ith node
T_{avg}	The average consensus completion time
R_i	Indicate if the ith node reported malicious behaviour

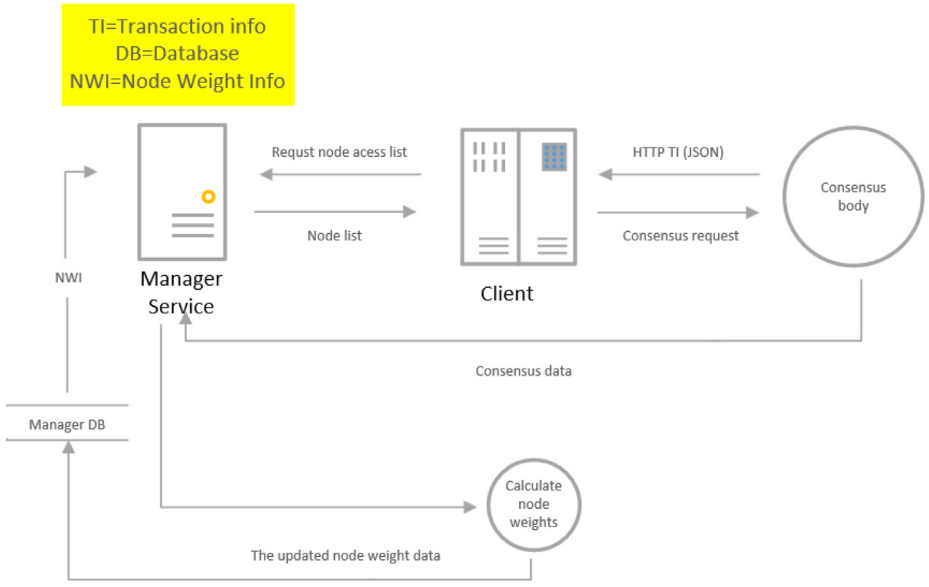

Fig. 2. The architecture and workflow of WBFT. "TI" means "Transaction Info", DB represents "Database", NWI refers to "Node Weight Info". WBFT is based on peer-to-peer (P2P) network model, which can greatly improve the efficiency of communication.

opinion of other nodes in the system, the algorithm increases/decreases the weight W_i of a node. In the following subsection, we will explain the weighing mechanism in detail.

3.2. Dynamic weighting mechanism

The weight value of each node is dynamically changed at the end of each consensus process. When a node is more active or honestly conducts the consensus activities, its weight value is increased. If the node's activity decreases or maliciously interferes with the consensus process, its weight will be reduced after the consensus ends. Therefore, the participation rate of malicious nodes in the consensus process can be reduced by dynamically adjusting the node weights, and finally the node consensus in the consortium chain will reach a stable level, so as to improve the efficiency of consensus. The weight updating process is shown in Algorithm 1.

Algorithm 1: Weight decision updating algorithm

Input: CCT, FI
Output: $W_i, i = 1, \cdots, N.$
1. Initialization
2. Management node determines bh_i according to FI
3. $P = \{P_1, P_2, P_3\}$
4. For each N_i Do // Punishment step
5. If (N_i participated in the consensus && $bh_i \in \{b_1, b_2, b_3\}$) Then
6. $W_i = W_i - P_i$
7. EndIf
8. EndFor
9. $T_{avg} = \frac{1}{M}\sum_{i=1}^{M} T_i$// Average consensus completion time
10. For each N_i Do // Reward step
11. If (N_i participated in the consensus) Then
12. $time = T_i - T_{avg}$
13. If (($time < 0$) || ($R_i == true$)) Then
14. $W_i = W_i + 1$
15. EndIf
16. EndIf
17. EndFor
18. Return W_i

The inputs to the algorithm include *CCT* and *FI*. *CCT* is the set of consensus completion time of all nodes participated in the consensus process. *FI* is the feedback information of all nodes participated in the consensus process. The feedback information of a node includes the behavior information of its neighbor nodes and the evidence that it propagates the received messages. The management entity determines the behavior type of a node according to the feedback information of its all neighbors. The behavior type reported by most of the neighbors is finally adopted. For example, suppose node N_i has four neighbors, if three of them report that N_i did not disseminate message, the management entity will set the behavior type of N_i to b_2. If a node reports the malicious behavior of a neighbor node, the feedback information must be accompanied by evidence, such as the timestamp of the received message. The management node will identify these evidences and determine the behavior of the neighbor node, that is, set a value to bh_i. The weights of the nodes are adjusted by analyzing the behavior and the activity of the nodes. The algorithm will output the updated weight value of a node (W_i). We will now explain the dynamic permission adjustment aspect of the WBFT algorithm.

3.3. Dynamic behavioral analysis for node punishment

The behavioral analysis of the nodes in the consensus process aims to determine the negative weight adjustment level of a node. Three possible unacceptable behaviors, NB = {b1, b_2, b_3}, are defined below:

b_1: the node delays information it received for a while.

b_2: the node never disseminated the information it received.

b_3: the node alters the information and broadcast the altered information.

Each behaviour b_i, corresponds to a penalty level P_i, $P_i \in P, P = \{P_1, P_2, P_3\}$, P_i represents the severity of punishment, and $P_3 > P_2 > P_1$.

A new round of weight adjustment process is conducted after each consensus process. Specifically, the behavioral information of a given node N_i in the blockchain system is collected from its surrounding nodes. At the end of each round of consensus, the nodes participated in the consensus process provide information

feedback to the management node through HTTP request. Based on the opinion of the surrounding nodes, the weight of node N_i is adjusted as follows:

$$W_i = W_i - P_i \tag{2}$$

In other word, node N_i is penalized by deducting P_i from its weight. For example, assume that each node has a base weight 10 (W = 10) and P= {P_1 = 5, P_2 = 8, P_3 = 10}. If node N_i is judged to have engaged in b_1 behaviour, its weight will be reduced by 5. Thus, the new weight (W_i) of node N_i will be W_i = 5. If node N_i is judged to have exhibited behavior b_3, its weight will be zero.

3.4. Activity analysis for rewarding nodes

In WBFT, the nodes participated in the consensus process are rewarded based on their activity. Specifically, the activity between nodes is measured by the consensus time (T_i) of node N_i and the average consensus time (Tavg). Tavg is defined as follows:

$$T_{avg} = \frac{1}{M}\sum_{i=1}^{M} T_i \tag{3}$$

A node N_i is rewarded by increasing its weight value in two cases. If the consensus time of a node is less than the average consensus time of all nodes, its weight will be appropriately increased. If the node reports malicious or down behavior of other nodes through the feedback mechanism, the node can also get the corresponding weight reward. The management node can manually and automatically verify feedback information by running logs on the chain. The weight updating formula is as follows:

$$W_i = W_i + 1 \tag{4}$$

3.5. Algorithm complexity analysis

In the algorithm, every node in the blockchain is visited twice to adjust its weight. The basic operations involved in the procedure mainly include addition and subtraction. Hence, the number of times of basic operations is approximately equal to 2 *N*. The algorithm's time complexity is $O(N)$. The space overhead of the algorithm includes the storage of *N* weights and several variables. Hence, the algorithm's space complexity is also $O(N)$.

3.6. Three-stage communication optimization

The number of messages exchanged during consensus algorithm is very important as it can affect the performance of the algorithm. In the application scenario of consortium chain, the main purpose of each node on the chain is to reach a consensus on the transaction information, without considering the message ordering. That means the commit stage can be removed without affecting the consensus result. We exploit this knowledge to optimize the three-stage broadcast protocol used in PBFT algorithm. Fig. 3 shows the protocol used in the proposed algorithm for consensus process.

Prior to starting the consensus algorithm, the weight inspection of the nodes is performed to exclude nodes with low weights from the participation in the consensus process. Specifically, when a request is received by the primary node, it checks the threshold value of all nodes and excludes the nodes with lower threshold from the participation in the consensus. For example, in Fig. 3, when the primary node 0 receives a request, it checks the threshold values of node 1, node 2 and node 3. The weight of node 3 is lower than the other two nodes. Thus node 3 is excluded from the consensus process. The primary node 0 then forwards the request to node 1 and node 2. Obviously, this method will reduce the communication cost and will not affect the consensus result. The prepare state is similar to the PBFT except that some nodes are excluded. If the number of consensus nodes in the system is lower than the minimum value, the management system will prompt in time. After entering the replay state, the active nodes and the primary node communicate their reply to the sender node. Based on the protocol, the maximum number of consensus communications after modification is:

$$C = N \times (N - 1) \tag{5}$$

Note that the number of nodes participating in the consensus can be reduced proportionately for the nodes which have the lower weight. Thus, our method can reduce the number of malicious behaviors on the premises of guaranteeing the node quantity and system security, which improves overall system performance.

4. Performance analysis

In this section, we validate the proposed algorithm WBFT through performance analysis. We also compared WBFT against RBFT and PBFT. The experiment is based on the virtual machine plus the master node and takes the consortium chain as the background.

4.1. Experimental setup

The equipment used in the experiment consists of 6 computers and 16 virtual machine nodes in the server. The T-IO network communication framework in Java is used to realize the P2P communication mode in the system. We randomly determine whether a node produces malicious or failure behavior during the current consensus process. The experimental environment is shown in

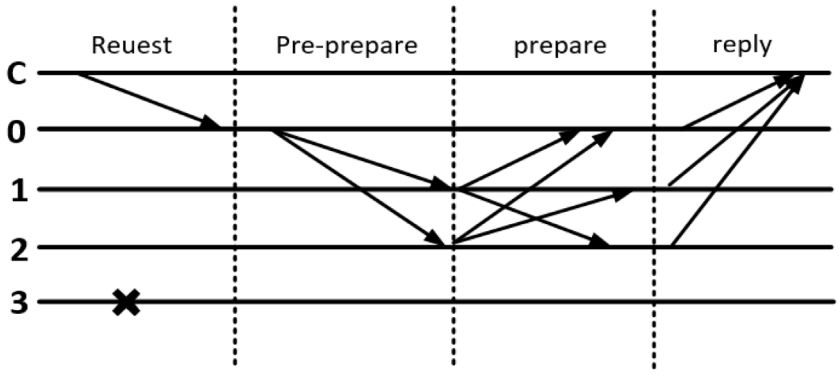

Fig. 3. Improved communication protocol. It shows the protocol used in the proposed algorithm for consensus process.

Table 2
Experimental environment.

Software/Hardware	Version
Operating System	Centos7
Memory	16 GB RAM
CPU	Intel Core i7-7700 K 4.2 GHz
JDK	8

Table 2 and the overall experimental frame is shown in Fig. 4. Algorithm parameters are set as follows: $W = 10$, $P_1 = 5$, $P_2 = 8$, $P_3 = 10$. When the threshold TH_i of a node is lower than 0.8, the node will be not invited to participate in the consensus process. In order to balance the participation of nodes, we set the maximum value of W_i to 20.

Before conducting the experiments, the consortium chain was started to simulate information trading and malicious behavior every day, so as to make the node participation weight in the system conform to the real application environment as far as possible.

4.2. Performance metrics

For measuring the performance, we used several metrics commonly used in the literature. The first metric is the average consensus delay time (D_{avg}), which refers to the overall average time required to complete a round of the consensus process and defined as follows:

$$D_{avg} = \frac{1}{N}\sum_{i=1}^{n} T_{delayPeriod} \tag{6}$$

where $T_{delayPeriod}$ is the consensus delay time and computed from the time the client request is originated ($T_{request}$) to the time that the consensus completed (T_{finish}). It is defined as follows:

$$T_{delayPeriod} = T_{finish} - T_{request} \tag{7}$$

Generally, a lower delay can improve the commercial performance of the consortium chain system and complete more business processing at the same time.

Throughput capacity is the second performance metric we used, which is defined as follows:

$$TPS = \frac{Transaction\,volume}{Averageresponsetime} \tag{8}$$

Transaction volume refers to the total number of processed transactions when a block is generated. Average response time refers to the average time of generating a block.

Throughput measures the traffic flow of the network communication within a period of time, and the size of throughput represents the total amount of transactions processed by the blockchain system within a unit time. In blockchain systems, TPS is typically used to represent the number of successful transactions per second.

Security is also an important evaluation index for the consensus algorithms. For security analysis of the consensus algorithms, we examined the capability of the system to withstand the denial of service attacks in blockchain distributed systems. We do this by examining the number of node participation in malicious behavior to subvert the integrity of the system and subsequently cause denial of service attack.

5. Discussion of results

5.1. Consensus delay analysis

In this section, we examine the performance of WBFT, RBFT and PBFT in terms of the consensus delay time. Fig. 5 shows the average consensus delay time of the three algorithms as the number of nodes vary when the block generation time is 5 s (Fig. 5a), 10 s (Fig. 5b), 15 s (Fig. 5c) and 20 s (Fig. 5d) respectively. Table 3 shows the average delay by three algorithms when block generation time are set to 5 s, 10 s, 15 s and 20 s, respectively.

From the above results, we observe that the consensus delay of the system increases with the increase of the block generation time. Compared with PBFT, the average consensus delay of the proposed method is decreased by 18.01 %. Compared with RBFT, our

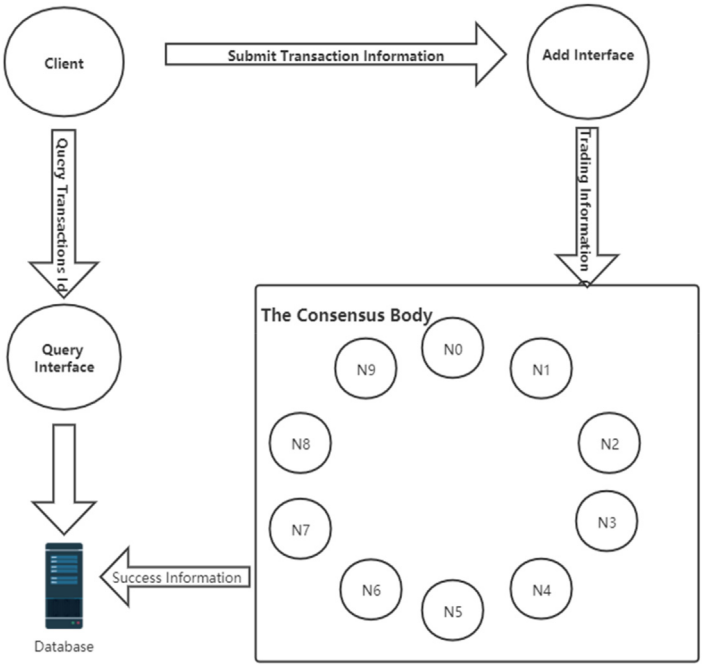

Fig. 4. Experimental frame diagram. In the experiment, the client node sends a request to the consensus body through the interface. The request contains the required JSON-formatted data. When the consensus process completes, the results are stored in the database. The client directly queries the database through the query interface.

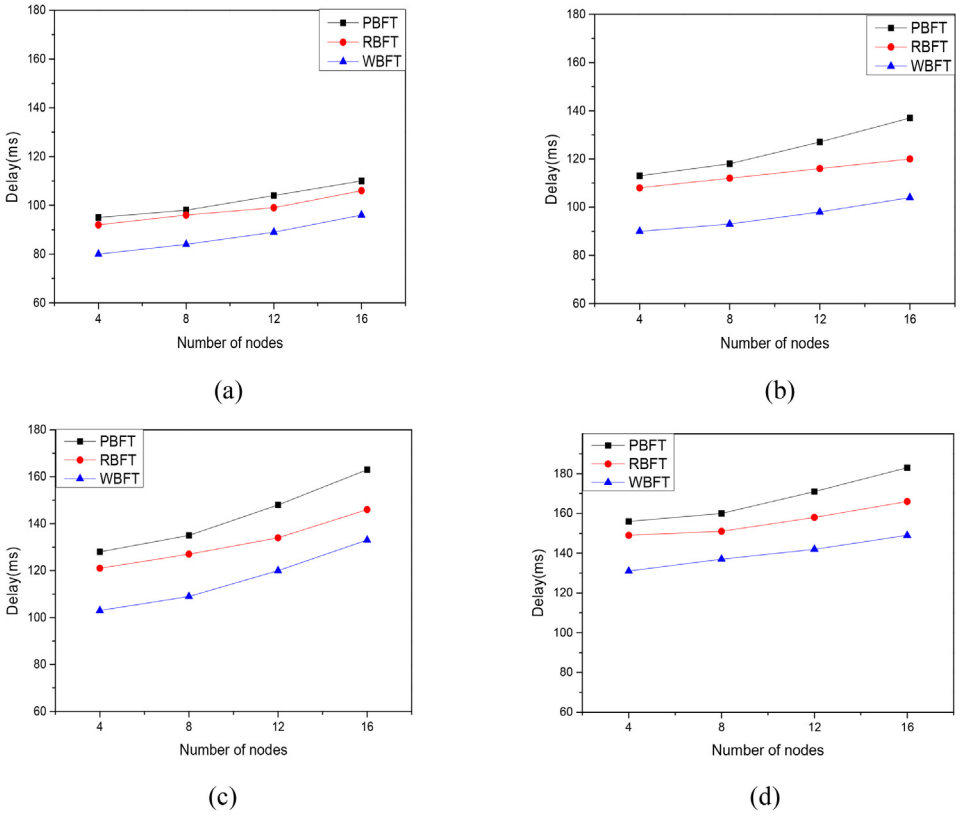

Fig. 5. WBFT consensus delay (Different time). It shows the average consensus delay time of the three algorithms as the number of nodes vary when the block generation time is 5 s (Fig. 5a), 10 s (Fig. 5b), 15 s (Fig. 5c) and 20 s (Fig. 5d) respectively.

Table 3
Average delay (the generating block time are set to 5 s, 10 s, 15 s and 20 s, respectively).

Generating block time	Algorithm	4	8	12	16	Average delay	Improvement/PBFT	Improvement/RBFT
5 s	PBFT	95	98	104	110	101.75	–	–
	RBFT	92	96	99	106	98	3.69 %	–
	WBFT	80	84	89	96	87.25	14.25 %	10.97 %
10 s	PBFT	113	118	127	137	123.75	–	–
	RBFT	108	112	116	120	114	7.88 %	–
	WBFT	90	93	98	104	96.25	22.22 %	15.57 %
15 s	PBFT	128	135	148	163	143.5	–	–
	RBFT	121	127	134	146	132	8.01 %	–
	WBFT	103	109	120	133	116.25	18.99 %	11.93 %
20 s	PBFT	156	160	171	183	167.5	–	–
	RBFT	149	151	158	166	156	6.87 %	–
	WBFT	131	137	142	149	139.75	16.57 %	10.42 %

method is decreased by 12.22 %. When block generation time is more than a certain value, the consensus delay will increase exponentially. The increase in block generation time means that more transaction information will be generated in this time interval, which leads to the increase of the consensus delay. Due to the transaction information forwarding in the consensus process, when the amount of data acceptable to the system node is exceeded, the system consensus will become unavailable, and the consensus delay exponentially rises. Compared with RBFT and PBFT, WBFT implements the node reward and punishment mechanism, optimizes the consensus process, and thus reduce the steps of consensus communication.

In the next experiment, we adopt a fixed amount of transaction information, set the same block generation time, and conduct 20 consensus experiments. Fig. 6 shows the results. It can be seen that, compared with PBFT and RBFT algorithms, WBFT significantly reduces the consensus delay and improves the consensus efficiency by optimizing the consensus mechanism and consensus communication process.

5.2. Throughput capacity analysis

We now examine the performance of WBFT, RBFT and PBFT in terms of the throughput. Fig. 7 shows the throughput of the three algorithms as the number of nodes varies when the block generation time is 5 s (Fig. 7a), 10 s (Fig. 7b), 15 s (Fig. 7c) and 20 s (Fig. 7d) respectively. Table 4 shows the average throughput by three methods when block generation time are set to 5 s, 10 s, 15 s and 20 s, respectively.

The experimental results show that with the gradual increase of consensus nodes, the throughput of WBFT decreases gradually on the whole, but the performance of WBFT consensus algorithm is

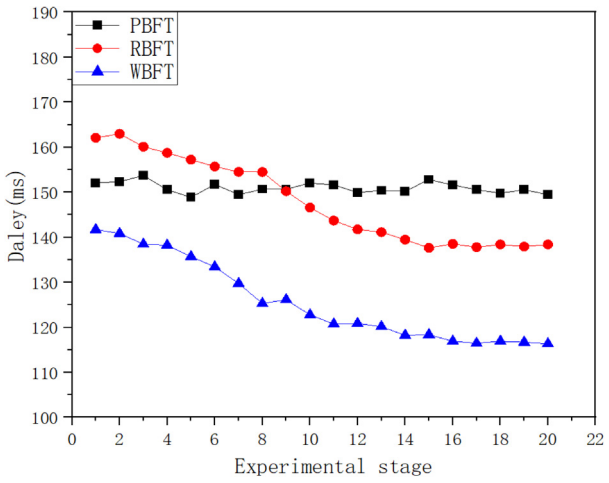

Fig. 6. Consensus delay. It shows the delay in the consensus processes of WBFT, RBFT, and PBFT in 20 consensus experiments.

better than that of RBFT and PBFT in the same block time, node number and other environments. Compared with PBFT, the average throughput of WBFT is increased by 17.18 %. Compared with RBFT, WBFT is increased by 8.54 %. It is true that the number of consensus nodes, block generation time and transaction data volume of the system will have a major impact on the throughput of the system.

In the next experiment, we adopt a fixed amount of transaction information, set the same block generation time and number of nodes, and conduct 20 consensus experiments. Fig. 8 shows the results. It can be seen that, the average throughput of WBFT is obviously higher than PBFT and RBFT algorithms.

5.3. Security analysis

Security is also an important evaluation index for the consensus algorithms. In the current complex network environment, to ensure the safe and stable operation of the blockchain is the focus of current research. The security analysis of the WBFT based blockchain system is as follows:

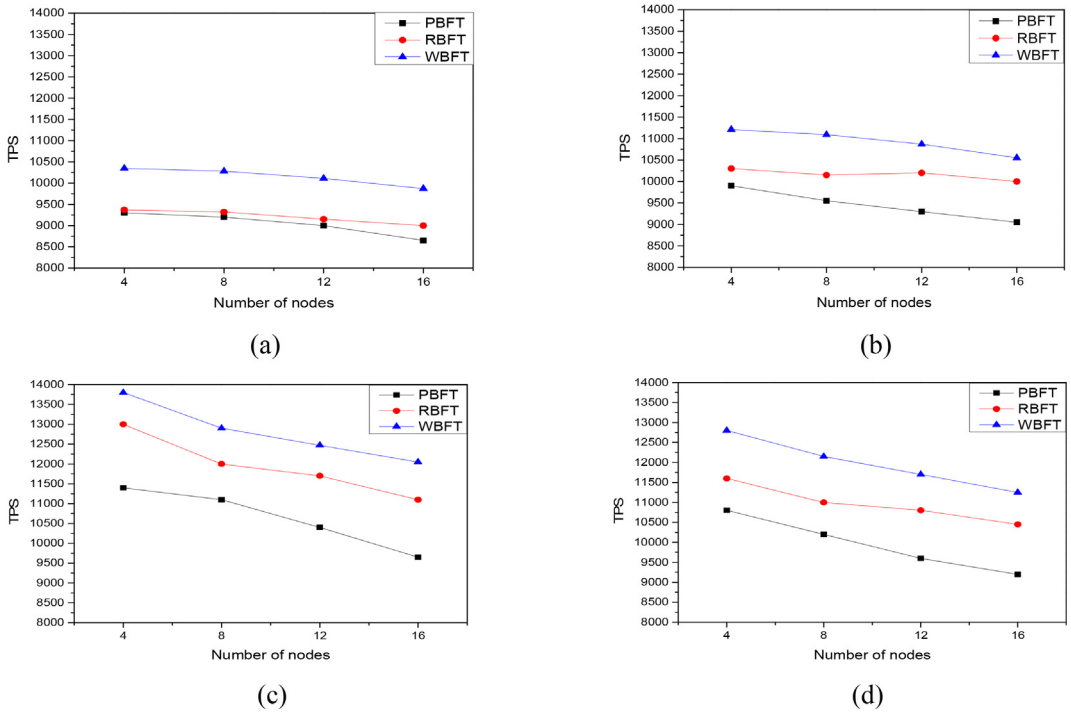

Fig. 7. The throughput of WBFT, RBFT, and PBFT. It shows the throughput of the three algorithms as the number of nodes vary when the block generation time is 5 s (Fig. 7a), 10 s (Fig. 7b), 15 s (Fig. 7c) and 20 s (Fig. 7d) respectively.

Table 4
Average throughput (the generating block time are set to 5 s, 10 s, 15 s and 20 s, respectively).

Generating block time	Algorithm	4	8	12	16	Average TPS	Improvement/PBFT	Improvement/RBFT
5 s	PBFT	9300	9200	9000	8650	9037.5	–	–
	RBFT	9370	9320	9150	9000	9210	1.91 %	–
	WBFT	10,350	10,280	10,110	9870	10152.5	12.34 %	10.23 %
10 s	PBFT	9900	9550	9300	9050	9450	–	–
	RBFT	10,300	10,150	10,200	10,000	10162.5	7.54 %	–
	WBFT	11,210	11,090	10,870	10,550	10,930	15.66 %	7.55 %
15 s	PBFT	11,400	11,100	10,400	9650	10637.5	–	–
	RBFT	13,000	12,000	11,700	11,100	11,950	12.34 %	–
	WBFT	13,800	12,900	12,470	12,050	12,805	20.38 %	7.15 %
20 s	PBFT	10,800	10,200	9600	9200	9950	–	–
	RBFT	11,600	11,000	10,800	10,450	10962.5	10.18 %	–
	WBFT	12,800	12,150	11,700	11,250	11,975	20.35 %	9.24 %

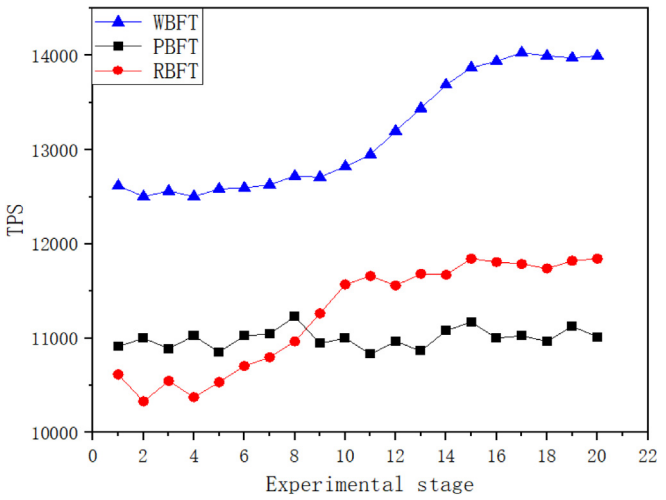

Fig. 8. Throughput of WBFT, RBFT, and PBFT. It shows the throughput of WBFT, RBFT, and PBFT in 20 consensus experiments.

(1) Preventing denial of service attacks in a blockchain system - Blockchain is a special kind of distributed database system, in which each node has a replica of trading information. When a node in the blockchain is attacked, other nodes can validate the trading information and perceive the error, and thus guarantee the correctness of the overall data. Therefore, the system is immune to attacks against one node.
(2) Robustness guarantee - Compared with the existing methods, our method can reduce the number of node participation and malicious behavior, which leads to the higher robustness and then improves the system performance.
(3) The proportion of malicious ACTS - The three algorithms were put into the simulated environment to perform experiment. They were given the same system parameters, and 20 transactions of consensus were conducted. The occurrence of malicious behaviors in the system was recorded respectively. Fig. 9 shows the proportion of malicious behaviors under three consensus algorithms in 20 consensus experiments.

The average failure behavior rates are 30 %, 18.725 % and 15.655 % under PBFT, RBFT and WBFT, respectively. It can be seen from Fig. 9 that WBFT has obvious advantages in greatly reducing the occurrence of malicious behaviors in the system, and thus improves the system security.

6. Conclusions

This paper proposes a new consortium blockchain consensus algorithm named WBFT based on a weighting mechanism. WBFT gives each blockchain node a weight, and limits malicious or faulty nodes to participate in the consensus process by dynamically adjusting the value of the weight. The proposed algorithm weakens the influence of malicious nodes as a whole, reduces the probability of malicious behavior, and then improves system throughput, consensus delay and enhances the security of blockchain system. We compare the proposed method with the two existing algorithms such as PBFT and RBFT. Experimental results show that compared with PBFT, the average delay of the proposed method is decreased by 18.01 %. Compared with RBFT, our method is decreased by 12.22 %. In terms of average throughput, our method is 17.18 % better than PBFT; compared with the RBFT method, the proposed method improved by 8.54 %. We can find that the average failure behavior rates are 30 %, 18.725 % and 15.655 % by PBFT, RBFT and WBFT, respectively. We can draw the conclusion that our algorithm outperforms the two existing algorithms from system throughput, consensus delay and security.

WBFT is validated to be effective in the experimental environment. In the future work, we will further test WBFT algorithm in the real consortium blockchain environment, improve the algorithm to adapt to the real consortium blockchain application. In addition, the consortium blockchain system will generate a large number of operation logs after a long period of consensus operation. These logs include detailed information of malicious behavior.

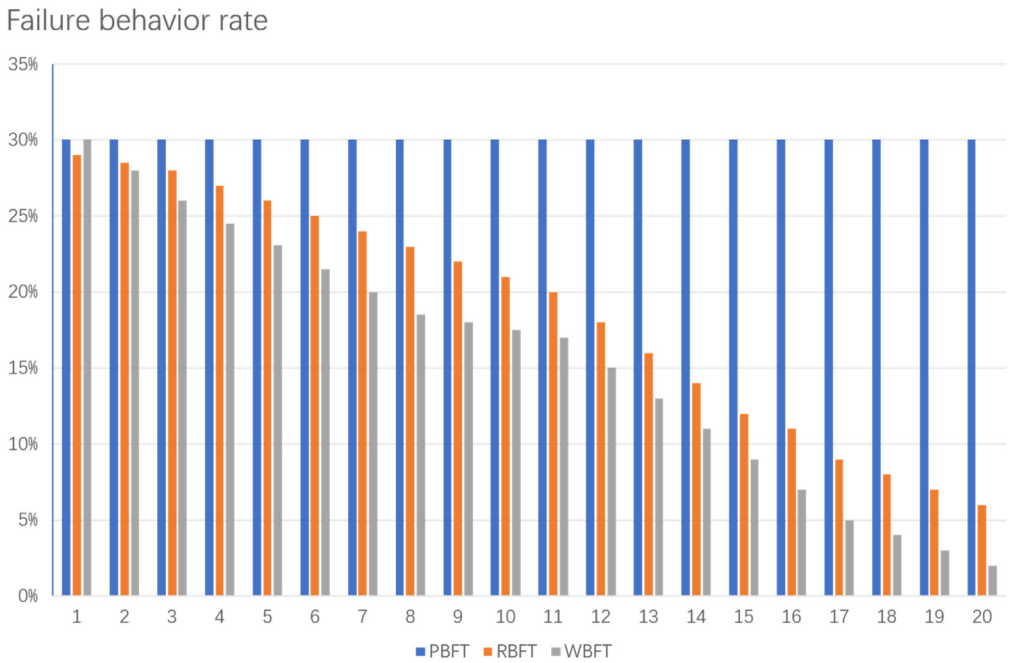

Fig. 9. Proportion of malicious ACTS. It shows the proportion of malicious ACTS of PBFT, RBFT and WBFT in 20 consensus experiments.

How to make rational use of these information and whether it is necessary to open these malicious information to other node are problems which are worth to study in the future.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (62162055) and the Gansu Provincial Natural Science Foundation (21JR7RA115).

References

Aitzhan, N.Z., Svetinovic, D., 2018. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE Trans. Dependable Secure Comput. 15 (5), 840–852.

Bamakan, S.M.H., Motavali, A., Babaei Bondarti, A., 2020. Alireza Babaei Bondarti, A survey of blockchain consensus algorithms performance evaluation criteria. Expert Syst. Appl. 154.

Bhushan, B., Sinha, P., Sagayam, K.M., J, A., 2021. Untangling blockchain technology: A survey on state of the art, security threats, privacy services, applications and future research directions. Comput. Electr. Eng. 90.

Nakamoto S. Bitcoin: a peer-to-peer electronic cash system [OL]. https://bitcoin.org/bitcoin.pdf, 2009.

Castro, M., Liskov, B., 1999. Practical Byzantine fault tolerance. USENIX Symposium on Operating Systems Design and Implementation (OSDI) 99, 173–186.

Chonka, A., Abawajy, J., 2012. Detecting and mitigating HX-DoS attacks against cloud web services. In: 15th International Conference on Network-Based Information Systems, pp. 429–434.

Gao, W., Chang, S.u., 2020. Analysis on blockchain financial transaction under artificial neural network of deep learning. J. Comput. Appl. Math. 380, 112–991.

Gervais, A., Karame, G.O., Wüst, K., et al., 2016. On the security and performance of proof of work blockchains. ACM SIGSAC Conference on Computer & Communications Security. ACM.

Gueta G G, Abraham I, Grossman S, et al. SBFT: a Scalable Decentralized Trust Infrastructure for Blockchains, 2018.

He, L., Hou, Z., 2019. An improvement of consensus fault tolerant algorithm applied to consortium chain. In: 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), pp. 1–4.

Hu, M., Shen, T., Men, J., Yu, Z., Liu, Y., 2020. CRSM: an effective blockchain consensus resource slicing model for real-time distributed energy trading. IEEE Access 8, 206876–206887.

Huang, D., Ma, X., Zhang, S., 2020. Performance analysis of the raft consensus algorithm for private blockchains. IEEE Trans. Syst. Man Cybern. Syst. 50, 172–181.

Lamport, L., Shostak, R., Pease, M., 2019. The Byzantine generals problem. In: Malkhi, D. (Ed.), Concurrency: the Works of Leslie Lamport. Association for Computing Machinery.

Lei, K., Zhang, Q., Xu, L., et al., 2019. Reputation-based byzantine fault-tolerance for consortium blockchain. 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS).

Ma, X., Xu, D.u., Wolter, K., 2022. Blockchain-enabled feedback-based combinatorial double auction for cloud markets. Future Gener. Comput. Syst. 127, 225–239.

Majeed, U., Khan, L.U., Yaqoob, I., Kazmi, S.M.A., Salah, K., Hong, C.S., 2021. Blockchain for IoT-based smart cities: Recent advances, requirements, and future challenges. J. Network Comput. Appl. 181, 103007.

Miller, A., Xia, Y., Croman, K., et al., 2016. The honey badger of BFT protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 31–42.

Omar, A.A., Bhuiyan, M.Z.A., Basu, A., Kiyomoto, S., Rahman, M.S., Abdullah, Bhuiyan, Alam, Md Zakirul, Basu, A., Kiyomoto, S., Rahman, Mohammad Shahriar, 2019. Privacy-friendly platform for healthcare data in cloud based on blockchain environment. Future Gener. Comput. Syst. 95, 511–521.

Proof of stake [EB/OL].[2020-05-25].https://en.bitcoin.it/wiki/Proof-of-Stake.

Sun, G., Dai, M., Sun, J., Yu, H., 2020. Voting-based decentralized consensus design for improving the efficiency and security of consortium blockchain. IEEE Internet Things J. 99, 1.

Sun, M., Zhang, J., 2020. Research on the application of blockchain big data platform in the construction of new smart city for low carbon emission and green environment. Comput. Commun. 149, 332–342.

Wang, B., Dabbaghjamanesh, M., Kavousi-Fard, A., Mehraeen, S., 2019. Cybersecurity enhancement of power trading within the networked microgrids based on blockchain and directed acyclic graph approach. IEEE Trans. Ind. Appl. 55 (6), 7300–7309.

Wang, W., Hoang, D.T., Xiong, Z., et al., 2019. A survey on consensus mechanisms and mining management in blockchain networks. IEEE Access 7, 22328–22370.

Yang, A., Li, Y., Liu, S., Li, J., Zhang, Y., Wang, J., 2019. Research on logistics supply chain of iron and steel enterprises based on blockchain technology. Future Gen. Comput. Syst. 101, 635–645.

Yang, F., Zhou, W., Wu, Q., Long, R., Xiong, N.N., Zhou, M., 2019. Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism. IEEE Access 7, 118541–118555.

Zhang, W., Wu, Z., Han, G., Feng, Y., Shu, L., 2020. LDC: A lightweight dada consensus algorithm based on the blockchain for the industrial Internet of Things for smart city applications. Fut. Gener. Comput. Syst. 108, 574–582.