EBFT : A Hierarchical and Group-Based Byzantine Fault Tolerant Consensus Algorithm

Wenzheng Li and Mingsheng He
Faculty of Information Technology
Beijing University of Technology
Beijing 100124, China
liwww@bjut.edu.cn

2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS) | 978-1-6654-1506-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICSESS52187.2021.9522233

Abstract—**Consensus algorithm is a key component of the blockchain technology, and also a hot topic in distributed system research. Under the background that blockchain technology has entered the development stage of consortium blockchain, the Practical Byzantine Fault Tolerance (PBFT), proposed by Miguel and Liskov in 1999, occupies an vital position. Although PBFT has many advantages, it does not scale well because of O(n^2) communication complexity. In order to solve this drawback, we present a hierarchical and group-based BFT consensus algorithm — efficient BFT (EBFT). It utilizes a novel network topology to effectively reduce communication times among nodes and provides O(n) communication complexity. The experimental results show that EBFT optimizes the consensus process involving large-scale nodes, so that the consortium blockchain can be applied to a wider range of application scenarios.**

Keywords-PBFT; consortium blockchain; consensus algorithm; high performance

I. Introduction

Blockchain, also known as Distributed Ledger Technology (DLT), is a new distributed infrastructure and computing paradigm which uses a type of "block-chain" data structure to verify and store data, utilizes a distributed consensus algorithm to generate and update data, depends on cryptography to ensure the security of data transmission and access, and uses smart contracts composed of automated script codes to program and manipulate data [1]. Blockchain technology originated from the Merkle-Damgard structure first described by Ralph Merkle in his doctoral thesis in 1979, which is used in many general hash functions. In 2008, Satoshi Nakamoto published a paper "Bitcoin: A Peer-to-Peer Electronic Cash System" [2] at the Cryptography Forum, which fully explained the principles of Bitcoin and blockchain technology.

According to the different attributes and access conditions of the participants, blockchain can be divided into public blockchain, private blockchain and consortium blockchain [3]. The consortium blockchain is jointly managed by multiple institutions, and each institution runs one or more nodes [4]. It has the characteristics of multi-agent, multi-link, and high-value [5]. At the 2019 CCF Blockchain Technology Conference, Chen Chun, an academician of the Chinese Academy of Engineering, proposed that blockchain technology has entered the consortium blockchain development stage since 2015 [6], and the consortium blockchain is currently the most practical and promising technology in China. Although the consortium blockchain has the advantage of partial decentralization, strong controllability, and high-value applications, the high-performance and high efficiency consortium blockchain consensus algorithm is a bottleneck restricting the development of the consortium blockchain, and becomes a difficulty that needs to be broken [7].

The consensus algorithm is the core of the blockchain, which can ensure that all nodes follow the same accounting rules without central control and achieve the consistency of distributed data [8]. The consensus algorithm of the blockchain system operates in a relatively complex, open and trustless Internet environment [9]. The goal of its research is how to achieve a balance between node availability and consistency and achieve high-efficiency certification [10]. The existing consensus algorithms can be divided into the following five types [11]:

1) Proof consensus. The miner nodes must prove that they have a certain ability in each round of consensus. The proof method is usually to competitively complete a task that is difficult to solve but easy to verify. The miner node that wins the competition will obtain accounting rights, such as Proof of Work (PoW) and Proof of Stake (PoS) [12].

2) Election consensus. The miner nodes select the current leader node by voting during each round of consensus. The miner node that first obtains more than half of the votes will get the right to generate a block. Traditional distributed consensus algorithms are mostly such consensus, such as Paxos [13] and Raft [14].

3) Random consensus. A miner node is directly determined as the leader node of a round according to a certain random method.

4) Consortium consensus. The miner nodes first elect a group of representative nodes based on a certain method, and then the representative nodes obtain the accounting rights in turn or by election.

5) Mixed consensus. The miner nodes adopt a mixture of multiple consensus algorithms to select the leader node.

978-1-6654-1506-4/21/$31.00©2021 IEEE

The Practical Byzantine Fault Tolerance (PBFT) algorithm proposed by Miguel Castro and Barbara Liskov in 1999 [15] is a partially decentralized consensus algorithm based on state machine replication. PBFT is composed of the consistency protocol, the view change protocol and the checkpoint protocol and it is the preferred core consensus algorithm for the consortium blockchain currently [16]. PBFT solves the problem of low efficiency of the original Byzantine fault tolerant algorithm and reduces the complexity of the algorithm from exponential to polynomial, making the Byzantine fault tolerant algorithm feasible in practical applications [17]. This algorithm can ensure that the blockchain system can still operate normally when no more than 1/3 of the nodes have a Byzantine error.

However, PBFT algorithm still has some shortcomings in terms of performance and usability [18]. In the consensus process of PBFT, each node has to conduct several broadcast communications and frequently perform message signing and signature verification [19]. Therefore, as the number of nodes increases, the system performance will drop sharply, and it is generally not suitable for the situation with a large number of nodes [20].

In this paper, we propose a lightweight consortium blockchain consensus algorithm — Efficient Byzantine Fault Tolerance (EBFT) based on the PBFT. Different from the public blockchain, not all nodes in the consortium blockchain have the possibility of doing evil, and the supervisor and operator nodes can be regarded as trusted nodes. Therefore, we first divide all nodes in the system into different groups based on a clustering algorithm, and propose a hierarchical and group -based blockchain node network topology. After that, we optimized the consensus protocol of the PBFT algorithm to adapt to the proposed system model, and proposed a lightweight consensus process based on the RSA algorithm.

The remainder of this paper is structured as follows: Section 2 provides an overview of relevant works in current. Section 3 looks into the system model and detailed implementation process of consensus. Section 4 presents the evaluation results of EBFT in various performance metrics. Section 5 concludes the paper and discusses the future work.

II. Related Works

In view of the shortcomings of the PBFT algorithm, in recent years, scholars and research institutions around the world have conducted extensive research and proposed a series of new consensus algorithms based on the PBFT algorithm.

Gueta et al. [21] proposed the Scalable Byzantine Fault Tolerance (SBFT) algorithm. It treats a node as a fixed and immutable block producer. Although this scheme has greatly improved the performance of the PBFT algorithm, its message mode is completely similar to the Raft algorithm widely used in the private blockchain. And it cannot bear the Byzantine error of the block producer. Therefore, the performance improvement of the SBFT comes entirely from the sacrifice of the Byzantine fault tolerance of the block producer.

Yin et al. [22] proposed the HotStuff algorithm. It changes the network topology of PBFT to a star network topology, and each communication relies on the central node. The node no longer broadcasts the message to other nodes through the P2P (Peer-to-Peer) network, but sends the message to the central node which processes the message and sent it to other nodes. In addition, the HotStuff algorithm merges the view change process with the normal process, reducing the complexity of view change.

The Bystack team [23] proposed the Bystack Byzantine Fault Tolerance (BBFT) algorithm. It proposes to change the network topology of nodes to a node forest structure connected by multiple node trees. The nodes are divided into consensus nodes, gateway nodes and leader nodes. The numerous communications between nodes are replaced by the aggregate signatures. The BBFT algorithm reduces the amount of consensus communication to O(n) complexity, but the use of aggregate signatures also puts forward new requirements on the computing performance of nodes.

Amir et al. [24] proposed the Prime algorithm to reduce the impact of view change on the PBFT algorithm caused by the malicious leader node. It introduces an extra stage in the three-stage consistency protocol and its performance is lower than the PBFT algorithm in normal scenarios, but it has more advantages in error scenarios.

Hao et al. [25] proposed the Dynamic PBFT algorithm to realize the dynamic property of the PBFT algorithm. It defines a special trusted node to manage the information of all replica nodes. The trusted node is responsible for judging whether the node can register and join the blockchain system. At the same time, the trusted node will also maintain and record the node's identity information and status. However, Dynamic PBFT relies too much on the existence of trusted nodes, and does not explain how the system operates normally and realizes dynamic properties when the trusted node goes down or even commits evil.

It can be seen from the above research that with the PBFT algorithm as a starting point, researchers have made many improvements to it. The network topology of blockchain nodes has undergone a process of change from mesh to star and then to hybrid. The purpose of changing the network topology is to reduce the number of communications between nodes in the consensus process and ease the computational pressure. However, the existing research does not propose an effective network partition method, especially in the alliance chain scenario. In addition, in the consortium blockchain, trusted nodes can lead the consensus process, be responsible for block generation, signature collection and verification, and no longer need to verify each other between every two nodes.

Therefore, we conducted in-depth research on the system model of the consortium blockchain and the consensus protocol of the PBFT algorithm, and proposed a high-performance consortium blockchain consensus algorithm EBFT based on the PBFT algorithm, making it a multi-centralized, multi-level blockchain consensus. The proposed algorithm improves the efficiency of consensus and reduces the number of communications in the consensus process without losing security, aiming to make the blockchain model applicable to a wider range of application scenarios.

III. Efficient Byzantine Fault Tolerance Algorithm

A. System Model

We propose a multi-centralized, multi-level, "layered and grouped" hybrid blockchain architecture. There are two problems, one is the improvement of the network topology of blockchain nodes, and the other is the division of node functions. We improved the mesh network topology of the PBFT algorithm to a "layered and grouped" hybrid structure to reduce the resource consumption of blockchain nodes to maintain peer to peer (P2P) connections and consensus processes. As shown in Fig. 1, all blockchain nodes are divided into two categories: master consensus nodes and slave consensus nodes. All master consensus nodes constitute a master consensus cluster, and each master consensus node corresponds to multiple slave consensus nodes which constitute a slave consensus cluster. In particular, in each round of consensus, a leader node is selected from the master consensus cluster to be responsible for packaging transactions to generate blocks, and the remaining master nodes are responsible for message transmission, while slave consensus nodes only need to verify the block and sign.

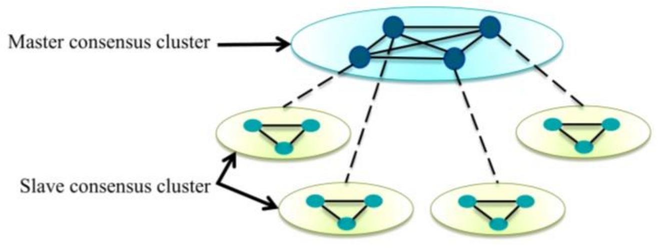

Fig.1 Blockchain system model

We call a node byzantine node if it behaves arbitrarily. It may delay communication, modify messages, replay messages and forge signatures. We assume that the faulty nodes are computationally bound so that it is impossible to subvert the cryptographic techniques used for system security.

B. Network Zoning

Suppose there are n nodes with p-dimensional features as a clustered data set. Use $v_i = \{v_{i1},v_{i2},v_{i3},...,v_{ip}\}$ $(i = \{1,2,...,n\})$ to represent a coordinate value of the nodei mapped in the p-dimensional feature space R^P. The set $V = \{v_1,v_2,...,v_n\}$ of all the above nodes can be expressed as:

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1p} \\ v_{21} & v_{22} & \vdots & v_{2p} \\ \vdots & \vdots & & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{np} \end{bmatrix} \tag{1}$$

K-medoids algorithm uses Euclidean distance to express the similarity of two nodes, and uses formula (2) to express the distance between two nodes v_i and v_k.

$$d(v_i, v_k) = \| v_i - v_k \| = \sqrt{\sum_{j=1}^{p} (v_{ij} - v_{kj})^2} \tag{2}$$

The goal of K-medoids algorithm is to cluster the dataset into K clusters, and each cluster is represented by V_i. Each V_i has a ci representing the cluster center node, and the center node set is represented by $c = \{c_1,c_2,...,c_k\}$. The node v_i belonging to the V_i cluster has higher similarity with other nodes in the same cluster, and lower similarity with nodes in other clusters.

The K-medoids algorithm needs to repeatedly calculate the Euclidean distance from each node in the cluster to other nodes in the iteration, which increases the computational complexity of the algorithm [21]. Taking into account the application scenarios of the consortium blockchain, operators and regulators may not want to waste the computing resources of the consensus cluster to frequently replace the center of the cluster after selecting the initial cluster center node. Therefore, the initial K central nodes are not randomly selected from all samples, but strictly reviewed by upper-level supervisors or operators. In this case, we do not want to adjust the central node.

The detailed process of grouping based on K-medoids algorithm is as follows.

Step 1 Select K nodes in the data set V as the initial cluster center nodes, and initialize the change coefficient P $(0<P<1)$.

Step 2 Calculate the distance function between each node and K cluster centers according to equations (2), and divide the nodes into clusters with the smallest distance.

Step 3 For all nodes in each cluster, calculate the sum of the distances to all other points in the cluster, and use the node with the smallest sum of the distances as the new cluster center node to be determined, and obtain the set of new cluster center nodes to be determined.

Step 4 If the new cluster center node set to be determined is the same as the original center node set, the algorithm terminates and returns the final clustering result; if the new cluster center node set to be determined is not exactly the same as the original center node set, the central node set will be replaced with the new cluster central node set to be determined in step 3 under the probability of P and return to step 3. The algorithm terminates and returns the final clustering result under the probability of $(1-P)$.

C. System Initialization

First, assume that each node has a unique *ID* and there is a key center (KC). The node needs to register at KC before joining the blockchain system. KC generates a public key and a private key of the node based on the RSA algorithm, and sends the private key through a secure secret channel to the node. The public key is disclosed to all nodes. The RSA algorithm is adopted because of its high security and a high position in the industry. RSA's security is based on the difficulty of decomposition of large integer.

The RSA algorithm is described as follows:

First select two large prime numbers p and q, and calculate $n=p*q$. Since p and q are large prime numbers, according to Euler's function: $\varphi(n) = (p-1)(q-1)$. Then randomly select e so that it satisfies gcd $(e, \varphi(n)) = 1$ and satisfies $1<e<\varphi(n)$. Finally, calculate d such that $e \bullet d = 1 \bmod \varphi(n)$, that is, e and d are multiplicative inverses modulo $\varphi(n)$. Therefore, we get the public key (PU) = (e, n), and the private key (PR) = (d, p, q).

RSA signature generation and verification

1) Signature algorithm: for a known message, calculate the signature $s = M^d \pmod n$;

2) Signature verification: For a known message signature (M, s), verify whether s^e is equal to $M \pmod n$, if it is true, the signature is valid, otherwise it is invalid.

D. *Consensus Peocess*

Since the master nodes are all trusted nodes, and the signature is unforgeable and verifiable. Therefore, the process of mutual verification between nodes in PBFT is not necessary, and the consensus process can be led by the leader node of each round of consensus. Under normal case, the algorithm consists of three stages: distribute, collect, and finalize as described in the following:

1) Distribute process. The leader node L selects transactions from its transaction pool and packages them into the next proposed block B. The proposal is broadcasted to all nodes in the master consensus cluster. And then, master consensus nodes start to pass it down along the topology tree edges to its corresponding slave consensus nodes.

2) Collect process. At the leaf level, upon receipt of the proposal, slave consensus node i verifies the block and signature, signs the block, and sends the result to its master consensus node. Every master consensus node collects the signatures generated by slave nodes and sends all signatures (including its own signature) to the leader node.

3) Finalize process. The leader node verifies all signatures received. If the block is verified by more than half of the nodes in the system, the leader node will add the new block to the end of the blockchain and send and broadcast a synchronization message to other master nodes. Each master node will broadcast synchronization messages to all its corresponding slave nodes. The consensus node synchronizes the block after receiving the synchronization message and verifying the message.

Assume that there is a two-layer topology consensus network composed of 7 nodes, and the consensus process is shown in Figure 2.

T0: The leader or S0 node sends the block and the signature to S1.

T1: S1 forward the block and signature to their slave consensus nodes.

T2: The slave consensus nodes verify the block and signature and sends the signature to S0 and S1.

T3: S1 will send all signatures to the leader node after collecting the signatures.

T4: The leader node verifies all the signatures received. If more than x (half of the total number of nodes) nodes verify the block successfully, it sends a synchronization block message to S1.

T5: S0 and S1 send the synchronization block message to their slave consensus nodes.

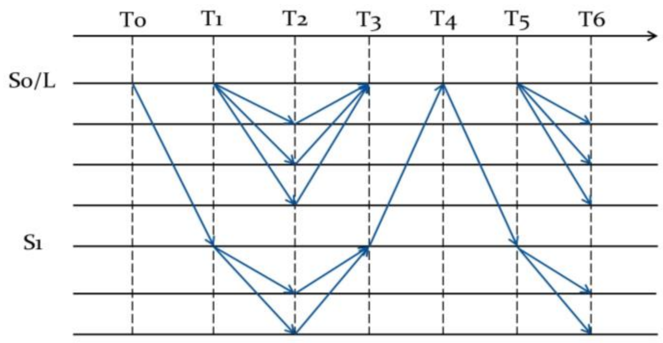

Fig.2 Consensus process of EBFT

IV. SIMULATION AND ANALYSIS

A. *Preliminaries*

We implemented simulations using Python language to compare the performance of EBFT and PBFT algorithms. The experiment was based on one single computer, nodes were simulated by multiple threads and communications between threads were realized by Socket. The experiment results described in the rest of this section were performed within a MacBook Air, Intel Core i5 2.0 GHz and 4 GB of random access memory (RAM). The operating system was Mac OS X Mountain Lion 10.8.5.

We set one client make a request to the consensus cluster every 50 ms. We also set up one block is generated per second, and each block contains 200 transactions at most.

B. *Evaluation Critetia*

This experiment evaluated EBFT and PBFT in terms of delay and throughput. Delay refers to the time from transaction submission to transaction completion. In the blockchain system, delay is a standard for measuring network performance and the running time of consensus algorithms. The lower the delay, the faster the confirmation of the transaction, while reducing the clients' waiting time. Throughput represents the ability of a blockchain system to process transactions per unit of time, and is generally represented by Transaction Per Second (TPS). Throughput refers to the ratio of the number of confirmed transactions to the time consumed. The higher the throughput, the stronger the cluster's ability to handle clients' requests.

C. *Algorithm Performance Analysis*

Fig. 3 shows an average delay in different number of nodes. We sent 1000 requests to all nodes in the blockchain network randomly and calculated the average delay. The delay increases for both the EBFT and the PBFT as the number of nodes increases. However, the EBFT has a significant large delay compared to the PBFT because of its exponential three-stage consensus process. Therefore, the average delay of the EBFT is on average 84.08% smaller than PBFT.

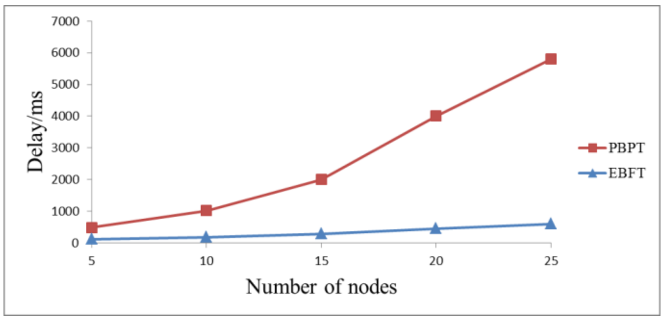

Fig. 3 Average delay according to the number of nodes

Fig. 4 shows the throughput of EBPT and PBFT in different number of nodes. We recorded the time from the client sending the first request to the cluster giving the response of the 1000th request and TPS. Obviously, as the number of nodes increases, the delay of a single transaction increases, so the throughput of EBPT and PBFT decrease. In particular, the average throughput of PBFT drops by 45.92% on average, while the throughput of EBFT only decreases by 8.69% on average. Apparently, in the same number of nodes, EBPT shows higher throughput than PBFT because of the lower delay.

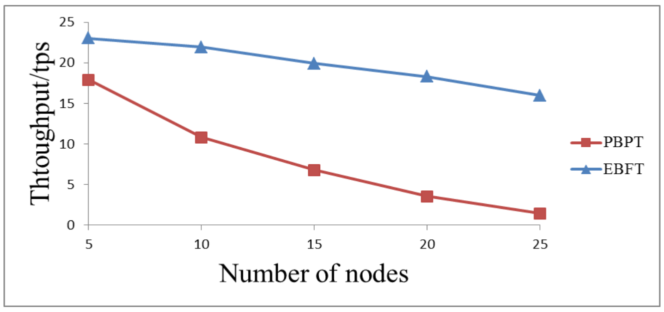

Fig. 4 Throughput according to the number of nodes

According to the analysis of the experimental results, it can be basically drawn that EBFT can effectively reduce consensus time in a single round of consensus and still achieve high throughput in the presence of large-scale nodes.

V. Conclusions

In this paper, we proposed a novel consensus algorithm EBFT for consortium blockchain. We divide the undifferentiated nodes in the PBFT algorithm into different layers and groups and propose a hierarchical and group-based blockchain model. Then, we modify the consensus protocol of PBFT based on the model proposed. Two main performance, delay and throughput, were compared between EBFT and PBFT. The experimental results show that EBFT is superior to traditional PBFT algorithm in consensus efficiency and more suitable for consortium blockchain application scenarios where large-scale consensus nodes participate. Future research will aim at improving the efficiency of the generation and verification of the digital signature and implementing the EBFT in practical consortium blockchain based applications.

References

[1] Zheng Z, Xie S, Dai H, et al. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends.2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 2017: 557-564.

[2] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System [Online] , available: http://bitcoins.info/bitcoin.pdf, June 26, 2020.

[3] Conoscenti M , Antonio V, Martin J C D. Blockchain for the Internet of Things: a systematic literature review. 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Nov 29-Dec 2, 2016.

[4] Y. Jiang and S. Ding, "A High Performance Consensus Algorithm for Consortium Blockchain," 2018 IEEE 4th International Conference on Computer and Communications (ICCC), Chengdu, China, 2018, pp. 2379-2386.

[5] S. Malik, V. Dedeoglu, S. S. Kanhere and R. Jurdak, "TrustChain: Trust Management in Blockchain and IoT Supported Supply Chains," 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 2019, pp. 184-193.

[6] X. Zeng, N. Hao, J. Zheng and X. Xu, "A consortium blockchain paradigm on hyperledger-based peer-to-peer lending system," in China Communications, vol. 16, no. 8, pp. 38-50, Aug. 2019.

[7] X. Wang, Q. Feng and J. Chai, "The Research of Consortium Blockchain Dynamic Consensus Based on Data Transaction Evaluation," 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2018, pp. 214-217.

[8] U. Bodkhe et al., "Blockchain for Industry 4.0: A Comprehensive Review," in IEEE Access, vol. 8, pp. 79764-79800, 2020.

[9] S. Gao, T. Yu, J. Zhu and W. Cai, "T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm," in China Communications, vol. 16, no. 12, pp. 111-123, Dec. 2019.

[10] S. Pahlajani, A. Kshirsagar and V. Pachghare, "Survey on Private Blockchain Consensus Algorithms," 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), CHENNAI, India, 2019, pp. 1-6.

[11] Yuan Yong, Ni Xiao-Chun, Zeng Shuai, Wang Fei-Yue. Blockchain consensus algorithms: the state of the art and future trends. Acta Automatica Sinica, 2018, 44(11): 2011-2022.

[12] Proof of stake[Online]. https://en.bitcoin.it/wiki/Proof of Stake . 2018

[13] A. Ailijiang, A. Charapko and M. Demirbas, "Consensus in the Cloud: Paxos Systems Demystified," 2016 25th International Conference on Computer Communication and Networks (ICCCN), Waikoloa, HI, 2016, pp. 1-10

[14] Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. 2014 USENIX Annual Technical Conference, Jun 19-20, 2014, Philadelphia, USA. 2014

[15] Castro M, Liskov B. Practical Byzantine fault tolerance. 3rd Symposium on Operating Systems Design and Implementation, New Orleans, USA, 1999:173-186.

[16] Clement A, Wong E L, Alvisi L, et al. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. 2009, 9: 153-168.

[17] Aublin P, Ben Mokhtar S, Quema V. RBFT: Redundant Byzantine Fault Tolerance. 2013 IEEE 33rd International Conference on Distributed Computing Systems, Philadelphia, PA, USA, 2013: 297-306.

[18] Gao S, Yu T, Zhu W, et al. T-PBFT: An EigenTrust-Based Practical Byzantine Fault Tolerance Consensus Algorithm. China Communications, 2019, 16(12): 111-123.

[19] Crain T, Gramoli V, Larrea M, et al. DBFT: Efficient Leaderless Byzantine Consensus and its Application to Blockchains. IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 2018: 1-8.

[20] Zhao Q, Sun Y, Zhang P. Design of Trust Blockchain Consensus Protocol Based on Node Role Classification. 2019 IEEE International

Conference on Service Operations and Logistics, and Informatics (SOLI), Zhengzhou, China, 2019:104-109.

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

[22] Yin M, Malkhi D, Reiter M, et al. HotStuff: BFT Consensus in the Lens of Blockchain. 2018.

[23] ByStack Team. BBFT: a Hierarchical Byzantine Fault Tolerant Consensus Algorithm [Online], available: https://github.com/bystackcom/bbft, December 26, 2020.

[24] Amir Y, Coan B, Kirschr J, et al. Prime: Byzantine Replication under Attack. IEEE Transactions on Dependable and Secure Computing, 2011, 8(4): 564-577.

[25] Hao X, Yu L, Liu Z, et al. Dynamic Practical Byzantine Fault Tolerance. 2018 IEEE Conference on Communications and Network Security (CNS), Beijing, China, 2018:1-8.