Asynchronous Consensus Algorithm Integrating Dynamic Weight Sharding Strategy

Journal:	*Chinese Journal of Electronics*
Manuscript ID	E230313.R1
Manuscript Type:	Original Article
Keywords:	Blockchain, Sharding technology, Asynchronous consensus
Speciality:	Computer Science [C]
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
main.tex	

Chinese Journal of Electronics
vol.04, no.04, pp.1– 12, 2022
https://doi.org/123.123.123.123

Research Article

Asynchronous Consensus Algorithm Integrating Dynamic Weight Sharding Strategy

Ao Xiong[1], Wang Zhang[1], Yu Song[1], Dong Wang[2], Da Li[2], QingLei Guo[2], and DeSheng Bai[2]

1. *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
2. *State Grid Blockchain Technology (Beijing) Co., Ltd, Beijing 100032, China*

Corresponding author: Yu Song; Email: songyu@bupt.edu.cn.

Received March 22, 2022; Accepted March 22, 2022; Published March 22, 2022.

Abstract — Blockchain technology has broad application prospects in many fields due to its unique characteristics such as decentralization, traceability, and non-tampering, and has become a research hotspot in recent years. As a key component of blockchain technology, consensus algorithm is one of the important factors affecting blockchain performance. However, many consensus algorithms currently used in consortium chains are based on time assumptions and lack horizontal expansion capabilities. That is to say, the consensus algorithm cannot reach a consensus in an asynchronous network where the receiving time of network packets is uncertain, and its efficiency will decrease as the number of nodes increases, which hinders the large-scale application of the alliance chain. In order to solve the above problems, this paper proposes the DS-Dumbo algorithm, an asynchronous consensus algorithm that integrates dynamic sharding strategies, based on the currently excellent DumboBFT asynchronous consensus algorithm. The main work of this paper revolves around how to fragment and optimize the consensus process. First of all, this paper designs a node asynchronous sharding model based on multi-dimensional weights, so that the re-sharding work of each blockchain node can be executed concurrently with the asynchronous consensus algorithm, reducing the conflict between blockchain sharding and asynchronous consensus algorithms. In addition, we also designed an intelligent transaction placement strategy, which calculates the relevance score of each transaction for all shards to determine which shard the transaction is processed in order to reduce the number of complex cross-shard transactions. Finally, we optimized the execution process of the DumboBFT algorithm, converted its internal synchronous working mode to an asynchronous working mode, and reduced the consumption of consensus work to a certain extent. The experimental evaluation shows that the DS-Dumbo algorithm has higher throughput and lower delay than the DumboBFT algorithm, and can increase the throughput with the increase of nodes, and has the ability of horizontal expansion.

Keywords — blockchain; sharding technology; asynchronous consensus

I. Introduction

Blockchain technology has attracted the attention of many fields with its unique technical advantages. This technology has been successfully applied in different fields, such as energy [1], agri-food [2], public administration [3], healthcare [4]. This technology has been successfully applied in different sectors such as. Currently, there are issues with the issuance and control of electronic licenses in the domestic power business, such as lack of standard norms, imperfect interconnection recognition mechanisms, incomplete shared service systems, lack of application channels and independent management platforms for electronic licenses, and lack of credible authorization and certification mechanisms for electronic license application [5].Blockchain technology has advantages such as security, decentralization, and transparency [6], which can provide support for electronic license management in the power industry. However, there are also some problems with blockchain technology, with the most significant issue being the low efficiency of consensus algorithms [7]. At present, the main consensus algorithm used in blockchain is based on Byzantine fault tolerance. Although this algorithm has high security, its consensus efficiency is low, and it is difficult to ensure efficiency and high availability in the face of complex network environment and high concurrent data processing requirements [8].

In order to enable blockchain to be applied to high concurrent data processing needs, it is necessary to improve the performance of blockchain technology, and this paper will use sharding technology to improve the performance of blockchain[9]. The main idea of blockchain sharding technology is to divide the entire blockchain network into multiple independent cell blockchains (shards), each of which is independent of each other and concurrently processes transaction data in the network. Fragmentation technology can effectively reduce the computational complexity of consensus within a single fragment, improve consensus efficiency, and enhance network scalability[10]. Although sharding technology improves network efficiency, it basically keeps the blockchain consensus algorithm unchanged and still relies on the assumption of network synchronization [11], showing low availability in networks with large network delay fluctuations. Therefore, it is necessary to combine asynchronous consensus methods to solve this problem.

Asynchronous consensus technology is a method to ensure the consensus between different nodes even if there is network delay, failure or malicious behavior between nodes. It adopts clock asynchronization [12]. In an asynchronous network, the use of asynchronous consensus satisfies strong consistency and guarantees security in the blockchain networkcite[13]. The use of asynchronous consensus can ensure multi-chain global consensus and solve the problem of low performance and limited reliability of the blockchain[14].

Therefore, this paper combines sharding and asynchronous consensus to solve the high concurrency data processing needs of blockchain in complex network environments. Monoxide proposes asynchronous consensus zones that shard the Proof-of-Work (PoW) blockchain. They propose eventual atomicity to ensure transaction atomicity across zones[15]. MWPoW+ is an asynchronous consensus sharding consensus protocol, which is a strong consensus protocol based on voting, and can ensure the security of the sharding blockchain[16].

To maintain sharding technology's anti-censorship ability, frequent re-sharding is required, which necessitates a nearly uniform working state among the majority of network nodes. But achieving consensus among nodes' working states is challenging within the asynchronous consensus framework[17]. Thus, resolving the re-sharding issue within the asynchronous consensus framework is crucial for successful integration of these technologies. In an asynchronous environment, the state of nodes cannot be unified, so the method and time of selecting shard are crucial. One approach is to do global resharding between multiple consensus shards at a specific time, which simplifies the sharding process. However, this approach leads to huge computational waste, as early nodes have to wait for subsequent nodes to complete their work, which determines the sharding efficiency of the system. Instead, resharding can be done asynchronously to maximize the utilization of computing resources. In order to maximize the use of computing resources and improve efficiency, this paper will use asynchronous resharding.

The paper is organized as follows: Chapter 2 presents work done by other researchers to improve the throughput and asynchronous applicability of consensus algorithms. Chapter 3 presents the overall architecture of the algorithm and describes how nodes are sharded in this solution. Chapter 4 is about the transaction intelligent placement strategy, which aims to reduce the number of cross-shard transactions in the system to speed up the overall consensus process. Chapter 5 introduces the improvement scheme of node asynchronous consensus algorithm. Chapter 6 verifies the performance indicators of the algorithm through experiments. Chapter 7 summarizes the work of the paper and analyzes the shortcomings of the algorithm and looks forward to the future.

II. Related Work

In recent years, many researchers have conducted extensive studies on improving the performance and applicability of blockchain consensus algorithms[18][19], PBFT is a widely used consensus algorithm in consortium blockchains[20]. However, PBFT still has many problems. For example, the time complexity of PBFT is $O(n^2)$, as the number of nodes increases, the communication frequency between nodes also increases squartially, resulting in high network communication costs, network congestion, significant reduction in system efficiency, and lack of scalability[21]. Therefore, the number of nodes in the PBFT cluster will not exceed 100. Based on the shortcomings of the PBFT algorithm, many researchers are studying new technologies, such as sharding technology [22] and asynchronous consensus in use in blockchain[13].

Reference [23] proposed the Elastico20 protocol, which is the first protocol to use sharding technology in blockchain consensus algorithms. The main idea of Elastico20 is to evenly divide or parallelize the mining network into smaller committees, each of which processes a disjoint set of transactions (or ”shards”). However, Elastico is only a rough protocol and does not address issues such as cross-shard transactions, and it only has a 1/4 node Byzantine fault-tolerant capability. Reference [24] proposes a sharding technology called OmniLedger to implement a secure, scalable, decentralized distributed ledger. OmniLedger uses a bias-resistant public randartication protocol to ensure the random allocation of nodes, and the protocol also uses the two-phase client-driven ”locking/unlocking” protocol Atomix to solve the problem of cross-shard transaction processing in sharded blockchains. However, users of OmniLedger need to actively participate in cross-shard transactions, which is difficult to meet the needs of lightweight users, and it only has a 1/4 node Byzantine fault-tolerant capability. Reference [10] proposes a new clustering-based sharding consensus algorithm (KBFT). The KBFT algorithm uses the K-prototype clustering algorithm to shard nodes, and allows transaction consensus to be independently and concurrently executed in each shard. However, the algorithm has a serious problem that the node sharding process is too complex and affects the overall efficiency of the system.

On the other hand, PBFT is based on the assumption of a semi-synchronous network, and its availability will be greatly affected when the network fluctuates significantly. Many researchers have also made contributions to improving the asynchronous network availability of consensus algorithms. Reference [25] proposes an efficient asynchronous secure computing protocol, which can realize the perfect secure computing of any function in the case of at most $f<N$ malicious nodes, that is, does not disclose any redundant information, and guarantees correctness and consistency. But the protocol in the paper also allows halting to occur with non-zero probability, i.e. the possibility of unterminating execution. This kind of downtime probability cannot be eliminated, because if the calculation that must be able to be terminated is required, then the condition of $N<4f$ needs to be satisfied, and the protocol in the paper can only guarantee the condition h of $N<4f$. Reference [26] proposes a series of safe and efficient asynchronous broadcast protocols to implement basic functions such as reliable broadcast, atomic broadcast, and secure causal atomic broadcast in asynchronous networks. These functions are an important part of implementing fault-tolerant replicated services in asynchronous networks. The paper uses the concepts and tools of modern cryptography, gives strict models and definitions, and analyzes the correctness, security and complexity of the protocol. Reference [27] proposes a HoneyBadgerBFT algorithm, which is the first practical asynchronous Byzantine consensus algorithm. The HoneyBadgerBFT algorithm consists of a reliable broadcast protocol and an asynchronous binary public subset agreement. The reliable broadcast protocol replaces the value received for broadcasting among nodes, while the asynchronous binary public subset agreement based on the Ben-or algorithm can ensure that all nodes can eventually reach consensus on the proposals. Reference [28] optimizes and improves the Honey Badger algorithm and proposes the DumboBFT algorithm, which uses MVBA instead of the asynchronous binary subset protocol used in HoneyBadgerBFT, avoiding each member running n instances of the asynchronous binary subset protocol in a single consensus, thereby accelerating asynchronous consensus speed.

Sharding technology and asynchronous consensus are both solutions that can improve the efficiency and availability of blockchain systems, but currently, there is not much research on the combination of sharding technology and asynchronous consensus. This paper explores a comprehensive solution to integrate sharding technology and asynchronous consensus to improve the performance of blockchain consensus algorithms while increasing their availability.

III. Sharding Strategy in An Asynchronous Environment

1. Overall Structure

The structure of the asynchronous consensus algorithm that integrates dynamic sharding strategy is shown in Figure 1.

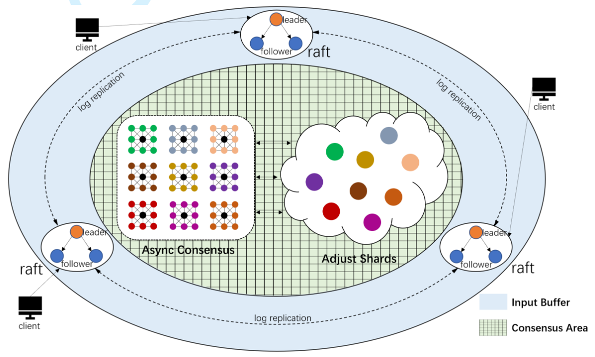

Figure 1 The overall structure of an asynchronous consensus algorithm that incorporates dynamic sharding strategies.

This scheme comprises an input buffer area and an asynchronous consensus area. The system creates an input buffer area that offers clients read and write services, records their write and consensus content, and aids consensus without participating in the final work, fully satisfying CFT (Crash Fault Tolerance). The asynchronous consensus area consists of alliance members who carry out consensus work on write contents made in the buffer area, and record them on the chain, thereby fulfilling BFT (Byzantine Fault Tolerance). The input buffer area has multiple Raft domains, each having one primary node receiving client input and other followers acting as backup nodes. Raft domains in different shards synchronize their commit logs and the resulting synchronized messages are the data to be consensused in the input buffer area. The asynchronous consensus area performs two primary functions: adjusting consensus nodes' shards and carrying out asynchronous consensus algorithms in different shards. Shard adjustment involves considering device capabilities and each node's historical behavior comprehensively, aimed at enhancing the system's flexibility and anti-attack capability.

The proposed plan introduces a dynamic topology to the network by continuously reshuffling the consensus nodes. However, this dynamic topology poses challenges for clients attempting to submit consensus information to the consensus nodes. To address this issue, an input buffer is introduced to simplify the submission process for clients. The input buffer, constructed by the consortium chain system, is isolated from the consensus work of the parties, making it less susceptible to hacker attacks. Nevertheless, unexpected situations, such as downtime, may occur, and therefore, the Raft algorithm is employed to maintain a single input domain. Furthermore, the plan provides multiple input Raft domains to meet the high concurrency requirements of asynchronous consensus algorithms. Clients can choose the node in the nearest input domain to submit consensus information.

The transaction permissions including client identity information and asset authentication are authenticated by the input buffer nodes when a transaction tx is submitted to the input buffer. The input buffer maintains a globally consistent consensus information buffer queue. The buffer queue adds a 64-bit unsigned and monotonically increasing consensus tag number to the validated transactions and fills the consensus information buffer queue with the numbered transaction information. The queue packs a batch of transactions into consensus information ”m” then inputs each one sequentially into the prepared asynchronous consensus node. If the consensus information ”m” fails, it is split into a group of transaction information $\{tx_i, tx_{i+1}, \ldots, tx_{i+n}\}$, re-validated, processed, and then re-inserted into the buffer queue.

2. Node Weight Configuration

Sharding reduces the number of nodes in a single consensus domain, therefore, a reasonable redistribution of the docking nodes is necessary to enhance the fault tolerance of the domain. We have originally introduced a node weight matrix to assist in this task in our plan. Assuming that the entire network has N nodes, then the size of the weight matrix is $N * N$. We evaluate each node's weight value from three aspects: computational capability value cScr, historical performance value hScr, and recommended reputation value rScr. If the weight evaluation of node i for node j at moment t can be represented as Scr_{ij}^{t}, This time t only represents absolute time, which is a moment. The performance indicators of nodes are updated in real time with time t, and there is no need for the t of all node indicators to be aligned with each other. then the weight values of each node at moment t can be represented as follows:

$$SCR^{\mathrm{t}} = \begin{bmatrix} \mathrm{Scr}_{11}^{t} & \mathrm{Scr}_{12}^{t} & \cdots & \mathrm{Scr}_{1N}^{t} \\ \mathrm{Scr}_{21}^{t} & \mathrm{Scr}_{22}^{t} & \cdots & \mathrm{Scr}_{2N}^{t} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Scr}_{N1}^{t} & \mathrm{Scr}_{N2}^{t} & \cdots & \mathrm{Scr}_{NN}^{t} \end{bmatrix} \tag{1}$$

1) Computation power value

The computation power value is dependent on the physical computing power of the node and the reliability of the network. The hardware largely determines the physical computing power and is invariable once node registration is finalized. Network reliability, on the other hand, quantifies the probability of sustaining reliable communication between node i and node j when they interact in a particular network setting. Therefore, the computational power value can be computed using the formula below:

$$cScr_{ij}^{t} = cp^{t} + psc_{ij}^{t} \tag{2}$$

The variable cp^{t} represents the physical computing power of node i at time t, while psc_{ij}^{t} represents the network packet sending capacity between node i and node j.

2) Historical performance value

The significance of past performance values diminishes as the temporal distance from the present point expands. Conversely, the relevance of past reputation values increases as the temporal proximity from the present point intensifies. As a result, the evaluation of past performance values is closely linked to a dynamic alteration factor. We can establish a dynamical calculation function for this factor as elucidated below:

$$\theta(x, t_k) = e^{-x(t-t_k)}, 0 \le x \le 1, 0 \le t_k < t \tag{3}$$

Here, x represents the rate of change adjustment factor, and $t_k < t$ represents a specific past point in time. Historical performance is calculated using a set of historical performance value according to the following equation 4:

$$h\,\mathrm{Scr}_{ij}^{t} = \frac{\sum_{hScr>Scr'} \theta\left(x, t_k\right) hScr_{ij}^{t_k}}{\sum_{hScr>Scr'} \theta\left(x, t_k\right) hScr_{ij}^{t_k} + \sum_{hScr\leq Scr'} \theta\left(x, t_k\right) hScr_{ij}^{t_k}} \quad (4)$$

Here, Scr' represents the consideration threshold of the historical performance value, and only the values above the threshold will be finally calculated in the new historical performance value. In order to prevent the value of a certain node from being too high, when calculating the historical performance value, it will be divided by the sum of all historical performance values.

3) Recommended reputation value

Typically, a recommending node would be required to have had prior interactions with the node being evaluated. Furthermore, the greater the weight that the rated node assigns to the recommending node, the more reliable the recommending node is deemed to be. In the event that a network comprises L recommending nodes, denoted by the set $\{n_1, n_2, ..., n_l\}$, with respect to node j, the formula for computing the weight of node j based on recommended nodes is as follows:

$$r\,\mathrm{Sc}\,r_{ij}^{t} = \sum_{l=1}^{L} \frac{c\mathrm{Scr}_{il}^{t}}{\sum_{j=1}^{N} c\,\mathrm{Sc}\,r_{ij}^{t}} c\,\mathrm{Sc}\,r_{ij}^{t} \quad (5)$$

4) Comprehensive weight

Finally, by computing a weighted sum of the computation power value, historical performance value, and recommendation reputation value, we can derive a comprehensive weight value:

$$\mathrm{Scr}_{ij}^{t} = \alpha \times c\,\mathrm{Scr}_{ij}^{t} + \beta \times h\mathrm{Scr}_{ij}^{t} + \gamma \times r\mathrm{Scr}_{ij}^{t} \quad (6)$$

$$\alpha + \beta + \gamma = 1 \quad (7)$$

The values of α , β and γ are adjusted in due course with the working conditions of the overall blockchain network to balance the proportions of computation power value, historical performance value, and recommendation reputation value d in the final node comprehensive weight.

5) Shard weight

The weight of a specific shard is defined as the sum of the weight values of all nodes within the shard. The formula for calculating the weight value of shard n at time t is as follows:

$$SW_n^t = \sum_k \sum_{j=1}^{N} \mathrm{Scr}_{kj}^{\mathrm{t}}, k \in n \quad (8)$$

Here, k represents the node ID within shard n.

3. Node Dynamic Sharding Scheme

After discussing the node weight calculation method, we can further design the node dynamic sharding scheme according to the node weight. Before designing, several basic goals of the sharding algorithm need to be determined. First, it is necessary to ensure that each node is allocated to one and only one shard; second, the specific allocation of nodes must be unpredictable to prevent attackers from predicting which shard a node may be allocated to before allocation. Therefore, in the distribution process, random elements need to be introduced to ensure the randomness of the distribution process. Finally, it is necessary to ensure that each shard has similarity, that is, the sum of node weight scores in each shard is roughly equal. This can prevent unreasonable distribution of shard nodes, resulting in almost all faulty nodes or malicious nodes in a shard, and ultimately destroying the shard consensus. Based on these three goals, the following fragmentation scheme is designed.

In this system, it is stipulated that each consensus node re-performs the sharding operation after the consensus reaches δ times. The value of δ is different for different nodes, and it will decrease with the decrease of its weight score. The purpose is to increase the re-sharding frequency of Byzantine nodes and faulty nodes, and prevent them from gathering in a certain shard. For a certain consensus node, when the consensus number of this round is δ, it will automatically submit a re-sharding application to the system, and submit the work log to update the node weight matrix and the sharding weight vector. For the newly generated shard weight vector, according to the weight of each shard in the weight sum of all shards, calculate its target threshold range and randomly assign it a target threshold range. The calculation process is as follows. For a system with n shards, the sum of the shard weights is W, and there is a shard s with a weight of w_s. Then, the target threshold range of the shard s is as follow:

$$\text{Score}_{\,s} = \left[\text{Score}_{\,s-1},\ \text{Score}_{\,s-1} + \frac{(W - w_s)}{(n-1)} W\right) \quad (9)$$

After the calculation is complete, the target threshold ranges of all shards add up to [0,1], where the ranges of every two shards will not intersect. Finally, according to the node weight score, it is divided into good nodes, faulty nodes or malicious nodes for processing respectively. For a good node, introduce a random number R, put it on the digital identity information of the node for hash operation, convert the operation result into a decimal number of 0 1, and finally judge which fragment the decimal number falls into the target threshold Range, that shard is the new working shard of the node; for the faulty node, it is directly transferred to the shard with the highest comprehensive weight; mark the malicious node, if the number of marks does not exceed the threshold, it will be regarded as a faulty node , when the number of tokens exceeds the threshold, its consensus node qualification will be deleted, and the identity of the consensus node can only be restored after the qualification review.

IV. Shard-based Transaction Consensus Placement Strategy

While blockchain sharding itself can help improve the transaction confirmation process, a large number of cross-shard transactions will inevitably reduce the throughput of the system to some extent. Therefore, in order to improve the transaction processing capacity, this section designs an intelligent transaction placement strategy to reduce the number of cross-shard transactions and improve the consensus throughput.

In order to quantify which shard a transaction should be assigned to, an indicator is designed to measure the matching score between a newly arrived transaction and each shard, and the score is used to determine which shard a transaction should be placed in for processing. This builds a transaction placement method based on matching degree calculation, and its workflow is as follows:

(1) Build a transaction relationship diagram. First of all, it is necessary to convert all transactions and their relationships into the form of nodes and edges, and construct a transaction relationship graph. Nodes in the graph represent transactions, and edges represent reference relationships between two transactions. For example, if the output of transaction A is referenced by the input of transaction B, then there is an edge between A and B in the transaction graph. In this method, the transaction relationship is expressed in the form of graph $G = (V, E)$, V represents a node as a transaction, and E represents an edge as a reference relationship between transactions.

(2) Calculate the transaction relevance score. In order to determine which transaction to put into a certain shard, we need to calculate the correlation between transactions to predict whether they will be consumed at the same time in the future. We can use a network-based algorithm to calculate the relevance score, such as calculating the score of each transaction based on its in-degree and out-degree. The more frequently a transaction is used, the higher its score. The transaction correlation score calculation process is shown in Equation 10:

$$TR(u,i) = \alpha \sum_{j \in \ln(u)} \frac{TR(j)M_{ji}}{\deg(j)} \tag{10}$$

Among them, $\ln(u)$ indicates the set of transactions pointing to u, M_{ji} is the state from node j to node i in the adjacency matrix, and $deg(j)$ indicates the degree of node j (that is, the number of nodes connected to other nodes), N represents the number of all transactions, and is an attenuation factor used to control the calculation rule of the correlation score.

(3) Calculate the shard load. In order to evenly distribute transactions across shards and ensure that each shard does not have too many transactions causing uneven load, we need to know the current load of each shard. This is achieved by computing the total mass of transactions in each shard, which is the sum of the weights of all transactions included. This weight can be the sum of the amounts per transaction or the sum of the transaction sizes. Mathematically, assuming that W_i represents the load of the i shard, the shard load is calculated as shown in Equation 11:

$$D_i = \sum_{t \in S_i} w(u) \tag{11}$$

Among them, S_i refers to the set of transactions contained in the i shard, and $w(u)$ represents the weight of the transaction. In the actual algorithm implementation process, we need to calculate the load rate of each fragment, that is, the utilization rate of its link hardware resources. We can calculate the load rate of shard i by the following Equation 12:

$$L_i = \frac{D_i}{B_i} \tag{12}$$

Where B_i represents the capacity of the i fragment.

(4)Calculate the transaction matching score. To determine which shard to place a transaction into, we need to compute for each shard its match score for the transaction. For this, we will consider the correlation of transactions and the load balancing of shards. For a transaction u and a shard i, the transaction matching score is calculated as shown in Equation 13:

$$MS(u,i) = L_i * TR(u) \tag{13}$$

(5) Determine the transaction placement location. Finally, we need to assign each transaction to the corresponding shard based on the match score. We choose the shard with the highest score as the shard where the transaction is located while ensuring that the capacity

of the shard does not exceed the threshold. If all shards with the highest score exceed the threshold, the transaction is assigned to the shard with the smallest capacity. After a transaction is placed, we need to update the shard load as well as the relevance score for that transaction to ensure load balancing and proper relevance.

V. Asynchronous Consensus Algorithms in Sharding

The asynchronous consensus part of this paper adopts the current excellent DumboBFT algorithm and improves on its existing work. The DumboBFT algorithm usually involves three phases:

(1) Value Broadcast Phase: Each node submits its own input value to a protocol called Provable Reliable Broadcast (PRBC). The PRBC protocol is responsible for broadcasting the input value to all nodes and waiting for a certain number of Finish messages.

(2) Output Value Phase: When the node receives the required number of Finish messages, the node invokes the Multi-Value Byzantine Agreement (MVBA) protocol. The MVBA protocol uses the output of PRBC as an input vector and will output a decision vector.

(3) Output Signature Phase: The node waits for a period of time to ensure that it receives enough Output Value Phase results, and performs signature processing according to the protocol requirements of the subsequent phase.

The main work focuses on the message broadcast phase of the PRBC protocol and the consensus phase of the MVBA protocol. However, these two phases are synchronized and interdependent to complete, resulting in the algorithm being performance-limited and vulnerable to network delays and errors. To overcome these problems, the proposal introduces the concept of input buffers. In the asynchronous consensus stage, the input buffer serves as middleware to provide input to it, as shown in Figure 2. The result of the message broadcast phase is stored in the input buffer, and the asynchronous consensus phase obtains the result of the message broadcast from the input buffer as input. With the distributed storage and recording capability of the input buffer, the first-stage message broadcast of the DumboBFT algorithm and the asynchronous consensus in the subsequent stages can be decoupled, allowing parallel execution on the same node. This greatly improves the throughput of the algorithm.

In the initial stage of message dissemination, a dependable broadcasting mechanism utilizing the sequence number carried protocol, referred to as PRBC, is employed. Each node's proposal information is as-

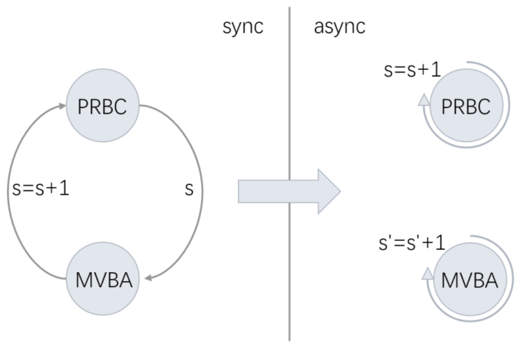

Figure 2 Node working status in synchronous mode and asynchronous mode.

signed through the input buffer. Subsequently upon successful broadcast completion within the shard, the outcome is directed to the input buffer. This result's confirmation can then be authenticated by one honest node within the shard, thereby ensuring proposal information integrity. During the following multi-value consensus stage, the input buffer agent is utilized. Nodes are exempted from waiting on any trailing multi-value consensus results. Upon the completion of slot $s's$ proposal message broadcast during the previous interval, nodes are permitted to continue accepting fresh proposals from the input buffer in a bid to broadcast their proposal message work of time slot $s+1$. Similarly, the second stage of multi-value consensus occurs autonomously for each node. Specifically, when a completed proposal message broadcast of time slot s exists within the input buffer, and the node finishes the multi-value consensus process of phase $s-1$, the input buffer forwards the multi-value consensus task of time slot s to the corresponding node. The algorithmic process is depicted in Figure 3.

It is important to note that the introduction of the input buffer could amplify the algorithm's complexity and necessitate modifications to various sections of the DumboBFT algorithm. For example, the adjustment of the message passing and multi-value consensus protocol could be required to accommodate the input buffer and transaction cache. In the subsequent section, the message passing protocol and multi-value consensus protocol implemented in DS-Dumbo will be discussed.

1. PRBC with Input Buffer

This protocol is used to ensure that the proposal of each node can be correctly broadcast to the whole network, and at the same time prevent malicious nodes from tampering or reviewing the proposal. The main idea is to use threshold digital signature technology to let each

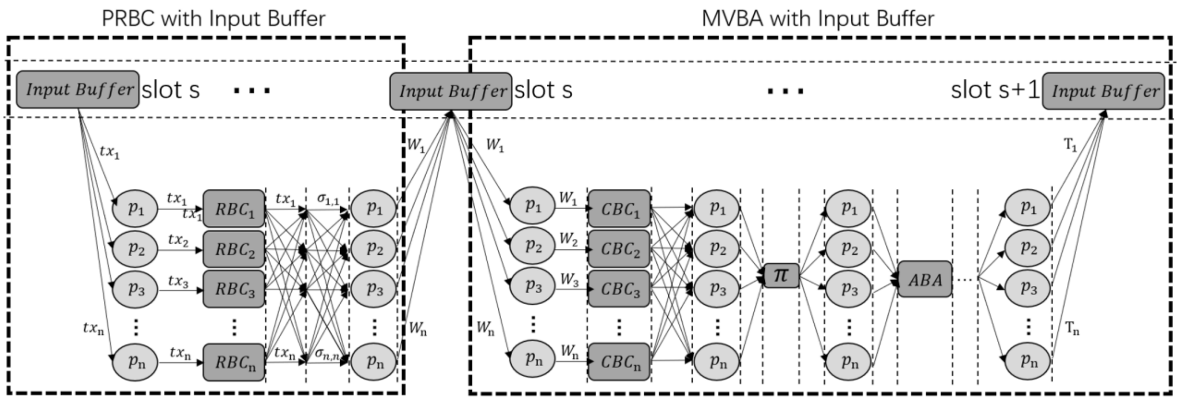

Figure 3 DS-Dumbo Algorithm Process.

node encrypt its own proposal and send the encrypted proposal to other nodes. When a node receives the same proposal from $f+1$ different nodes, it considers the proposal reliable and forwards it to other nodes. When a node receives the same proposal from $2f+1$ different nodes, it considers the proposal valid and takes it as output. In this way, even if there are f malicious nodes trying to forge or intercept proposals, they cannot affect the correct nodes to reach a consensus.

Node p_i can divide a received transaction message into $n-f$ mini-blocks and combine them with $2f$ erasure code blocks to form the broadcast set M for this message. While delivering the message, node p_i sends each block of M to all nodes in the network, and other nodes, upon getting a message block from p_i, relays the block to the remaining nodes. Upon receiving $n-f$ message blocks, a node restores and verifies the initial message using the erasure code. If it is accurate, the node dispatches an Echo message, which upon gathering enough Echo messages, confirms the message receipt. By using erasure codes to deliver messages, the algorithm effectively reduces message complexity and reduces the network burden during transmission.

The advantage of this protocol is that it can realize the function of reliable broadcast without relying on any random source. It can also effectively reduce the communication complexity, since each node only needs to send and receive $O(|B|)$ bits of data, where $|B|$ is the proposed size.

2. MVBA with Input Buffer

The MVBA algorithm is a random-based Byzantine fault-tolerant algorithm that seeks to bypass the limitations of deterministic algorithms in addressing Byzantine fault-tolerance difficulties in asynchronous networks. This algorithm integrates the threshold cryptography technique, mandating a minimum of 2f+1 sincere nodes to legitimize a message, thereby reinforcing the algorithm's security. Threshold cryptography, which is a group-based signature technology, necessitates node cooperation beyond a previously decided limit to decipher ciphertext.

The MVBA algorithm divides into three phases: message synchronization, leader selection, and ABA algorithm execution. In the message synchronization phase, each node submits proposals, and a simplified consensus propagation algorithm is used to transmit them. During the leader selection phase, the node that requests consensus for the round becomes identified. Next, the algorithm repeatedly runs the ABA algorithm to reach an agreement on the primary node's proposal among all nodes. If no consensus is established among the nodes, another ABA instance is repeatedly executed until reaching the final result of 1.

(1) For each node i, send a consensus message W_i to all other nodes;

(2) When node i receives the message W_i from other node j, node i encrypts the message W_i using the threshold encryption function SigShare, obtains the signature $sigma_i$, and sends it to node j;

(3) When node i receives $2f+1$ signatures from other nodes, node i uses the SigShare function to encrypt its own private key sk_i, obtains the consensus signature sig_i, and sends it to all node;

(4) When node i receives consensus signatures from other nodes, node i adds the received signatures to its own signature set s_i, and adds the message W_i to its own message set S;

(5) In each cycle, node i uses ABA algorithm for consensus on an element S_r in the message set S, and sends the consensus result I_i to all nodes;

(6) If node i receives the same consensus result from

Table 1 symbolic labels

f	Number of Byzantine nodes
N	Number of summary points of the consensus network
t	An absolute global moment
Scr_{ij}^t	the weight evaluation of node i for node j at moment t
$cScr_{ij}^t$	computational power value of node i for node j at moment t
cp_i^t	the physical computing power of node i at moment t
psc_{ij}^t	the network packet sending capacity between node i and node j at moment t
$\theta(x, t_k)$	Historical performance decay factor
$hScr_{ij}^t$	historical performance value
$Scr^{'}$	the consideration threshold of the historical performance value
$rScr_{ij}^t$	Recommended reputation value
SW_n^t	The weight of shard n
δ	Resharding rounds
$Score_s$	target score of shard s
W	The weight sum of the total shards
w_s	weight of shard s
$TR(u,i)$	Relevance score of transaction u relative to shard i
$TR(j)$	Relevance score of node j in transaction set u
M_{ji}	the state from node j to shard i in the adjacency matrix
$deg(j)$	degree of node j
D_i	The load of shard i
B_i	the capacity of shard i
L_i	the load rate of shard i
$MS(u,i)$	The transaction matching score of transaction u for shard i

$2f+1$ nodes, node i takes the result as the consensus result S and returns it.

The symbolic labels in this paper are shown in Table 1

VI. Results

The main purpose of this section is to experimentally verify the horizontal scalability and asynchronous environmental usability of DS-Dumbo. The working process of DS-Dumbo is simulated using Python3. DS-Dumbo uses one process each to handle the message broadcasting and concurrent multi-value consensus work. Use the gevent library to create concurrent coroutine tasks in each process, so that each node can make full use of system resources. Voting for consensus proposals in the MVBA phase is implemented using Boldyreva's pairing-based threshold signatures. To implement a reliable fully meshed asynchronous peer-to-peer channel, each pair of nodes is connected by a (persistent) unauthenticated TCP connection in the experiment. The network layer runs on two separate processes: one for message receiving and the other for message sending. If a TCP connection is dropped (and unable to send messages), reconnection will be attempted.The parameters are set to $\alpha = 0.2$, $\beta = 0.5$, $\gamma = 0.3$, and the following experiments are performed with these parameter values.

The results of Experiment 1 are shown in Figure 4. Throughput is the metric used to measure the algorithm's ability to process transactions quickly, and the higher the throughput, the greater the load that the algorithm can handle. In a test network consisting of eight nodes, two of which are Byzantine, the algorithm's throughput was evaluated. Figure 4 show the test results, where the horizontal axis represents the batch size of each round of transactions processed and the vertical axis shows the average throughput in tps (transactions per second) after ten rounds. Because the DS-Dumbo algorithm mainly extends the DumboBFT algorithm by sharding, it shows significant improvement in throughput compared to DumboBFT. The test results indicate that the throughput of the DS-Dumbo algorithm is approximately three times that of the DumboBTF algorithm, and their throughput trends are similar.

The Experiment 2 tested the relationship between the latency of the DS-Dumbo algorithm for a single shard and the number of nodes. In this experiment, the delay status of the system was set at 20 and 100 nodes. As shown in Figure 5, the latency of the DS-Dumbo algorithm increases as the number of nodes in the network grows. This suggests that, as the number of nodes increases in the algorithm, the network load also increases, leading to increased time costs for data exchange and communication, thereby increasing latency. However, because the DS-Dumbo algorithm parallelizes the two-phase work of DumboBFT,

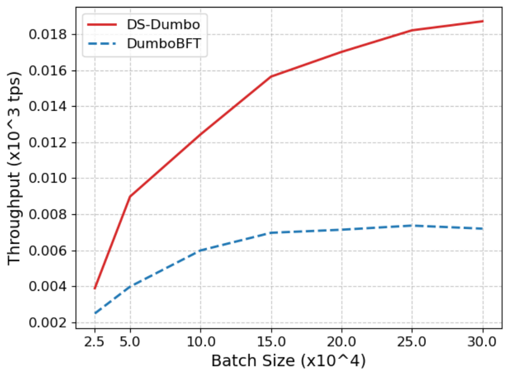

Figure 4 Throughput comparison.

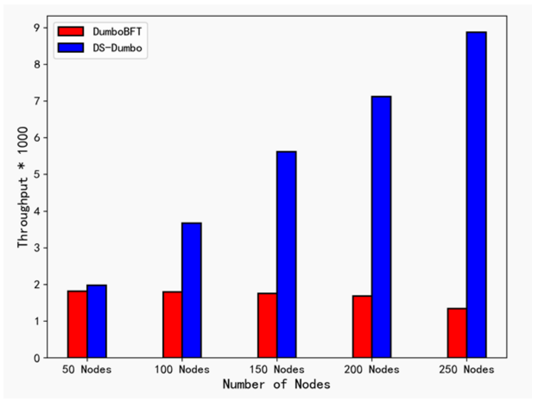

Figure 6 Scalability comparison.

the improved algorithm can better utilize computing resources and reduce latency. Therefore, DS-Dumbo's single shard can also tolerate more consensus node counts.

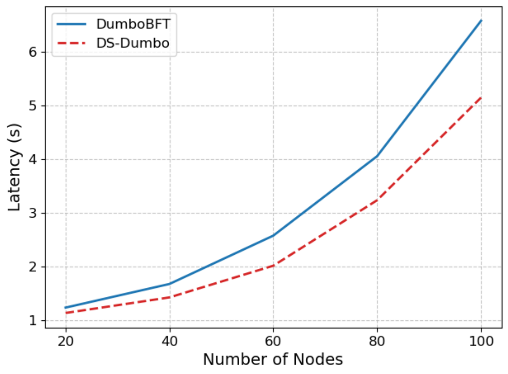

Figure 5 Latency comparison.

In the Experiment 3, each shard was set to contain 50 nodes. The experiment evaluated the performance of the consensus algorithm in multiple consecutive epochs, each epoch adding an average of 50 new nodes to the consensus algorithm. The era average throughput was used as the evaluation metric, which represents the average throughput obtained after multiple measurements in an era. As shown in Figure 6, the throughput of the DS-Dumbo algorithm using the sharding technique increases with the number of nodes, while the throughput of the DumboBFT algorithm decreases as the number of nodes increases. This indicates that consensus algorithms that incorporate sharding techniques have good scalability.

The Experiments 4 were conducted on the improved algorithm's ability to reduce cross-shard transactions. By continuously increasing the number of shards and adding completely random transactions to the network for consensus, the ratio of cross-shard transactions in the total number of transactions in the network was calculated. The experiment compared the transaction intelligent placement strategy of the DS-Dumbo algorithm with the transaction random placement strategy of the OmniLedger algorithm. 10,000 random transactions were added to the network for different shard sizes. The results are shown in Figure 7, which demonstrates that, for the DS-Dumbo algorithm, the transaction intelligent placement strategy reduces the number of cross-shard transactions by about 5% compared to the transaction random placement strategy of the OmniLedger algorithm.

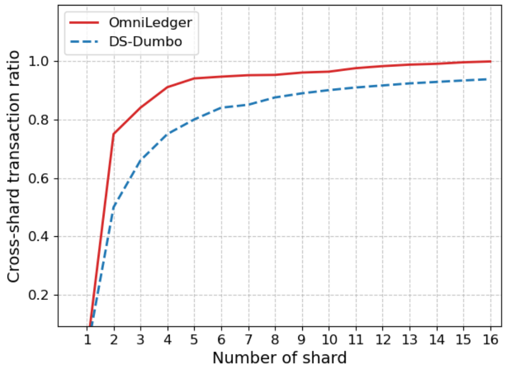

Figure 7 Comparison of the proportion of cross-shard transactions..

To sum up, this section mainly verifies the horizontal scalability of the DS-Dumbo algorithm and its performance indicators through experiments. By observing the results of Experiment 1 and Experiment 3, the following conclusions can be drawn. Since the DS-

Dumbo algorithm incorporates blockchain fragmentation technology, its throughput has been greatly improved compared with the DumboBFT algorithm, and as the number of nodes increases, Throughput also increases, demonstrating horizontal scalability. From the results of Experiment 2, it can be seen that because the DS-Dumbo algorithm optimizes the consensus execution process, its consensus delay is lower than that of the DumboBFT algorithm. Finally, this paper designs an intelligent placement strategy for transactions. Through Experiment 4, the following conclusions can be drawn. The number of cross-chip transactions of the DS-Dumbo algorithm is about 5% lower than that of the random strategy of the OmniLedger algorithm.

VII. Conclusions

In this paper, we first analyzed the pain points of combining sharding technology with asynchronous consensus, and discussed the most critical issue of node resharding. By designing a node weight and shard weight setting that comprehensively considers factors such as node physical capabilities and historical behavior, we proposed a dynamic re-sharding model for consensus nodes. A transaction intelligent placement strategy is also proposed for cross-shard transactions to reduce the number of cross-shard transactions in the system and improve consensus efficiency. Finally, the input buffer is optimized to enable the DumboBFT asynchronous consensus algorithm to execute two-stage tasks concurrently. In the end, this chapter compared the DS-Dumbo algorithm with the DumboBFT algorithm through experiments, demonstrating that DS-Dumbo algorithm has better concurrency ability and can better support electronic certificate business in the power system.

Acknowledgements

This work has been supported by State Grid Corporation of China science and technology project “Research and Application of Key Technologies for Trusted Issuance and Security Control of Electronic Licenses for Power Business ”(5700-202353318A-1-1-ZN).

References

[1] M. Andoni, V. Robu, D. Flynn, S. Abram, *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.

[2] G. Mirabelli and V. Solina, “Blockchain-based solutions for agri-food supply chains: A survey”, *International Journal of Simulation and Process Modelling*, vol.17, no.1, pp.1–15, 2021.

[3] A. Rot, M. Sobińska, M. Hernes, and B. Franczyk, “Digital transformation of public administration through blockchain technology”, *Towards Industry 4.0—current challenges in information systems*, in press, pp.111–126, 2020.

[4] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen, and A. Faxvaag, “Blockchain in healthcare and health sciences—a scoping review”, *International Journal of Medical Informatics*, vol.134, artilce no.104040, 2020.

[5] E. Syarief, “Electronic land certificates: Its goals and challenges”, *Research Horizon*, vol.1, no.4, pp.120–125, 2021.

[6] H. Guo and X. Yu, “A survey on blockchain technology and its security”, *Blockchain: research and applications*, vol.3, no.2, artilce no.100067, 2022.

[7] P. Zhang and J. Song, “Research advance on efficiency optimization of blockchain consensus algorithms”, *Comput. Sci*, vol.47, pp.296–303, 2020.

[8] A. I. Sanka and R. C. Cheung, “A systematic review of blockchain scalability: Issues, solutions, analysis and future research”, *Journal of Network and Computer Applications*, vol.195, artilce no.103232, 2021.

[9] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, *et al.*, “Towards scaling blockchain systems via sharding”, in *Proceedings of the 2019 international conference on management of data*, pp.123–140, 2019.

[10] X. Wu, W. Jiang, M. Song, Z. Jia, and J. Qin, “An efficient sharding consensus algorithm for consortium chains”, *Scientific Reports*, vol.13, no.1, artilce no.20, 2023.

[11] A. Kumar, A. Sangoi, S. Raj, and M. Kiran, “Shardcons-a sharding based consensus algorithm for blockchain”, in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, pp.1–6, 2021.

[12] Y. Niu, T. Yang, Y. Hou, S. Cai, *et al.*, “Consensus tracking-based clock synchronization for the internet of things”, *Soft Computing*, vol.26, no.13, pp.6415–6428, 2022.

[13] R. Pass, L. Seeman, and A. Shelat, “Analysis of the blockchain protocol in asynchronous networks”, in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, pp.643–673, 2017.

[14] S. Zhang, T. Xie, K. Gai, and L. Xu, “Arc: An asynchronous consensus and relay chain-based cross-chain solution to consortium blockchain”, in *2022 IEEE 9th International Conference on Cyber Security and Cloud Computing (CSCloud)/2022 IEEE 8th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, IEEE, pp.86–92, 2022.

[15] J. Wang and H. Wang, “Monoxide: Scale out blockchains with asynchronous consensus zones”, in *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, pp.95–112, 2019.

[16] Y. Xu, J. Shao, T. Slaats, and B. Düdder, “Mwpow+: A strong consensus protocol for intra-shard consensus in blockchain sharding”, *ACM Transactions on Internet Technology*, vol.23, no.2, pp.1–27, 2023.

[17] N. Chaudhry and M. M. Yousaf, “Consensus algorithms in blockchain: Comparative analysis, challenges and opportunities”, in *2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*, IEEE, pp.54–63, 2018.

[18] Y. Wang, Z. Song, and T. Cheng, “Improvement research of pbft consensus algorithm based on credit”, in *Blockchain and Trustworthy Systems: First International Conference, BlockSys 2019, Guangzhou, China, December 7–8, 2019, Proceedings 1*, Springer, pp.47–59, 2020.

[19] H. Xiong, M. Chen, C. Wu, Y. Zhao, and W. Yi, “Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms”, *Future Internet*, vol.14, no.2, artilce no.47, 2022.

[20] W. Yao, J. Ye, R. Murimi, and G. Wang, “A survey on consortium blockchain consensus mechanisms”, *arXiv preprint arXiv:2102.12058*, in press, 2021.

[21] M. Du, Q. Chen, and X. Ma, “Mbft: A new consensus algorithm for consortium blockchain”, *IEEE Access*, vol.8, pp.87665–87675, 2020.

[22] G. Wang, Z. J. Shi, M. Nixon, and S. Han, “Sok: Sharding on blockchain”, in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pp.41–61, 2019.

[23] L. Luu, V. Narayanan, C. Zheng, K. Baweja, *et al.*, "A secure sharding protocol for open blockchains", in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp.17–30, 2016.

[24] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, *et al.*, "Omniledger: A secure, scale-out, decentralized ledger via sharding", in *2018 IEEE symposium on security and privacy (SP)*, IEEE, pp.583–598, 2018.

[25] M. Ben-Or, B. Kelmer, and T. Rabin, "Asynchronous secure computations with optimal resilience", in *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*, pp.183–192, 1994.

[26] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols", in *Annual International Cryptology Conference*, Springer, pp.524–541, 2001.

[27] X. MillerA *et al.*, "Thehoneybadgerof bftprotocols//proceedingsofthe2016acmsigsac conferenceoncomputerandcommunicationssecurity", *Vienna, Austria*, vol.31, artilce no.42, 2016.

[28] B. Guo, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo: Faster asynchronous bft protocols", in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp.803–818, 2020.

Firstname1 Middlename1 Lastname1 With a frequency range of approximately 70–300 MHz, the MWA spans a number of Earth and space-based broadcast bands, including the ubiquitous FM band (approximately 88–108 MHz in Australia), constituting a primary source of RFI at the MRO. Likewise, the frequency range for SKA_low is 50–350 MHz, also encompassing the FM band. With a frequency range of approximately 70–300 MHz, the MWA spans a number of Earth and space-based broadcast bands. (Email: xxxxxxxx@xxx.xxx.xx)

Firstname2 Middlename2 Lastname2 With a frequency range of approximately 70–300 MHz, the MWA spans a number of Earth and space-based broadcast bands, including the ubiquitous FM band (approximately 88–108 MHz in Australia), constituting a primary source of RFI at the MRO. Likewise, the frequency range for SKA_low is 50–350 MHz, also encompassing the FM band. With a frequency range of approximately 70–300 MHz, the MWA spans a number of Earth and space-based broadcast bands. With a frequency range of approximately 70–300 MHz, the MWA spans a number of Earth and space-based broadcast bands, including the ubiquitous FM band (approximately 88–108 MHz in Australia), constituting a primary source of RFI at the MRO. Likewise, the frequency range for SKA_low is 50–350 MHz, also encompassing the FM band. With a frequency range of approximately 70–300 MHz, the MWA spans a number of Earth and space-based broadcast bands. (Email: xxxxxxxx@xxx.xxx.xx)

Firstname3 Middlename3 Lastname3 Xxxx xxx xxx xxx xxxxxxxx xxxxxxxxx xxxxxxxx xxxxx xxxxxxxx xxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxxxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxx xxxxxxxx xxxxx xxxxxx xxxxxxxx xxx (Email: xxxxxxxx@xxx.xxx.xx)

Firstname4 Middlename4 Lastname4 Xxxx xxx xxx xxx xxxxxxxx xxxxxxxxx xxxxxxxx xxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxx (Email: xxxxxxxx@xxx.xxx.xx)

Reviewer: 1

Comments to the Author
Comments for Chinese Journal of Electronics

#Contributions of the paper:
This manuscript addresses the issue of sharding in asynchronous consensus networks, presenting a forward-looking insight into the domain of blockchain consensus. However, it's pertinent to highlight the following potential concerns.

#Weakness of the paper:
1)The introductory section of the manuscript exhibits significant logical inconsistencies. First, the portion between references 8 and 9, which briefly touches on PBFT-style consensus, appears to be verbose given its two-sentence description. Secondly, the emphasis of this section should ideally be on the integration of asynchronous consensus with sharding. Therefore, I recommend restructuring this chapter to systematically address shard consensus, asynchronous consensus, and their combined implementation. Lastly, I question the appropriateness of citing references 10 and 14. There seems to be a paucity of references related to sharding in the asynchronous consensus domain. Is there a lack of research in this specific area?

Response: Thank you for your constructive suggestions.For reference 8 and 9, the contents of this part are simplified and redundant references are deleted. The specific modification is described in the last part of the first paragraph of Introduction. For the Introduction part, the logic has been readjust. Firstly, it is introduced that sharding technology can improve the performance of blockchain and is applied to high concurrent data processing requirements.Asynchronous consensus can achieve information synchronization in complex network environment and ensure network security. The combination of sharding technology and asynchronous consensus can solve the high concurrent data processing requirements of blockchain in complex networks, improve the performance of blockchain, and ensure information synchronization and security. The detailed description is shown in the modified Introduction section. In the original references 10 and 14, this paper deleted the references in this part. We refactored the Introduction part of the article and added some new references. See the introduction part of the revised article for details. For the literature that combines asynchronous consensus with sharding technology, relevant references are added in the Introduction section, such as references [15][16].

2)In the 'Related Work' , I noticed that the first paragraph commences with a mention of sharding followed by PBFT, the rationale for which is unclear. Furthermore, the subsequent paragraph revisits sharding and PBFT before delving into asynchronous consensus, raising concerns about the logical flow. The extensive length of the paragraph also poses challenges for reader comprehension. A more structured and concise presentation is recommended.

Response:Thank you for your constructive suggestions.The content of the related work chapter is modified. The first paragraph introduces the existing problems and principles of PBFT algorithm in detail, and then introduces the related technology of using fragmentation and asynchronous consensus. The second paragraph introduces the principle, research status and existing problems of sharding technology in detail. The third paragraph introduces the research status and existing problems of asynchronous consensus algorithms in detail. For the arrangement of paragraphs in this section, the paragraphs are re-divided and introduced in a more concise language. See the Related work section of the manuscript for details.

3)The primary focus of this manuscript appears to be on asynchronous consensus. However, I observed that the sections spanning from Section 3 to Section 5 heavily discuss potential issues,design goal and pertinent background knowledge. I recommend repositioning this content to the introductory sections to provide readers with foundational insights upfront. Furthermore, the inclusion of a 'Methodology' chapter would be beneficial for a coherent presentation of your proposed approaches.

Response:Thank you for your constructive suggestions.We adjusted the content of Section 3-5 in the article, deleted the content of the asynchronous method for node fragment selection in Section 3.2 and described it in the introduction section, deleted the relevant background knowledge and design goals in Section 4, and reconstructed the structure of Section 4. In order to control the length and conciseness of the whole paper, we have introduced the current related research technologies and methods in the introduction and related work chapters of this paper, and generally proposed the methods used in this paper, instead of using the 'Methodology' section to introduce the methods proposed in this paper.

4)In Section 2, specifically within the Node Weight Configuration Subsection, you introduced a time parameter 't' to represent the values of computation power value, historical performance value, and recommended reputation value during that time frame. However, you previously mentioned that time locks are not consistent in asynchronous consensus networks. Could you clarify the rationale behind using 't' in this context? Furthermore, as depicted in Figure 3, the asynchronous consensus model uses 'slot' as a unit of consensus round. Have you considered employing 'slot' instead of 't' for better consistency and relevance?

Response:Thank you for your constructive suggestions. This time t is an absolute time, which only represents the node performance index at that instant, and does not require t between different nodes to be aligned. At the same time, the several values related to t are similar to the physical indicators updated by the node in real time, which are an attribute of the consensus node and have nothing to do with the working status of the consensus node.

5)The process of re-sharding appears to be executed concurrently within the asynchronous consensus algorithm, which seems counterintuitive. This aspect doesn't seem to be sufficiently highlighted or addressed in the manuscript. Could you provide clarification on this particular design choice?

Response:Thank you for your constructive suggestions.The process of re-sharding is indeed concurrent and asynchronous, that is, each consensus node can immediately perform the operation of re-selecting shards after it reaches the conditions for re-sharding. The specific process of consensus node re-selecting shards is described in Section 3.3. This is to be compatible with asynchronous consensus, because in the case of asynchronous consensus, the state of each node is not uniform, and if the sharding operation is uniform for all nodes, the consensus work of some nodes will be canceled. Affect the overall work efficiency.

6)The manuscript encompasses a plethora of symbols, which can be challenging for readers to follow. Would it be possible to consider incorporating a table of symbols for clarity and ease of reference? Additionally, there are instances of symbol confusion; for example, on page 3, 't<n/3' denotes the number of nodes and the total format. Yet, in subsequent sections, 't' is utilized to represent a specific moment in time, and 'n' is designated for a shard. This ambiguity might confuse readers and could benefit from a more consistent usage or clear delineation.
Response:Thank you for your constructive suggestions. In our manuscript, we modify the notation that causes ambiguity. In related work, the original notation t< n/3 is changed to f< N/3, where f represents the number of Byzantine nodes in the network; N stands for the overall number of nodes, which is also modified in. For the symbols appearing in the article, we added

the symbol table to explain the symbols, which was added at the end of Section 5 of the article.

7)On page 5, it's mentioned that different nodes undergo a varying number of shardings. Could you elucidate how the specific number, denoted as \delta, is determined?

Response:Thank you for your constructive suggestions. δ is indeed a parameter, It's just a dynamic parameter. Will try to adjust according to the consensus network situation. If δ is chosen too small, the nodes will be fragmented frequently, which will cause network jitter; If the choice is large, it will reduce the anti-attack ability of the network. This parameter is similar to the difficulty factor in the Bitcoin network。 Suppose that the attack resilience function is f(δ), The network jitter function is g(δ), The value of δ is determined as the point of $f^{'}(\delta) = g^{'}(\delta)$.

8)On page 6, the notation G=(V, E) is introduced, which traditionally stems from the small-world model. Could you provide a rationale for its usage in this context? It seems there might be a lack of explanation and associated references to support its inclusion.

Response:Thank you for your constructive suggestions.Using the notation G=(V,E), G represents the graph of things, V is the node that represents the thing, and E is the edge that represents the reference relationship between transactions. The purpose of using this model in this paper is to associate transactions with graphs, which is not related to the size of the world model, but to facilitate the subsequent calculation of the correlation of transactions, according to the weight value of the calculated edges. The transaction is dynamic, and the transaction will be implemented more and more. The structure of the graph can intuitively reflect the state of the whole transaction network. The notation explanation for G=(V,E) is added in Section 4.2 of the paper.

9)Kindly ensure the accuracy of each mathematical expression, such as Equation (6).
Response:Thank you for your constructive suggestions.We re-checked the correctness of the expression of the formula in the article and made modifications to the formula. The problem of formula 6 is corrected. See the revised article for details.

10)In Equation (7), you present a sum expression involving three parameters to delineate varying weight values. However, the experimental section does not detail the specific allocation of these parameter values, nor does it depict the results under different values. Could you provide a comprehensive explanation regarding this?

Response:Thank you for your constructive suggestions.In the experimental simulation test of this paper, the experimental parameters in equation 7 are set as $\alpha=0.2$, $\beta=0.5$, and $\gamma=0.3$. This part of the content is supplementary explained in the results experiment section in Section VI.

11)The number of citations in the references section seems to be insufficient given the breadth of the topic covered. Furthermore, the manuscript could benefit from refinement in terms of language consistency and conciseness.

Response:Thank you for your constructive suggestions.According to the above modification suggestions, the introduction and related work sections are modified, and the references of sharding technology, asynchronous consensus technology and the combination of sharding and asynchronous consensus technology are added. The language expression of the manuscript has also been simplified and improved. The introduction part has been reconstructed, the related work part has been described in a more concise language, the original description content of Section 3.2 and Chapter 4 has been deleted, and the overall article has been rewritten with more concise statements.

Reviewer: 2

Comments to the Author
This paper proposes the DS-Dumbo algorithm, an asynchronous consensus algorithm that integrates dynamic sharding strategies. The algorithm aims to address the limitations of current consensus algorithms used in consortium chains, such as their inability to reach consensus in an asynchronous network and their efficiency decreasing with the number of nodes. The DS-Dumbo algorithm incorporates a node asynchronous sharding model, an intelligent transaction placement strategy, and an optimized execution process of the DumboBFT algorithm. Experimental results show that the DS-Dumbo algorithm has higher throughput and lower delay than the DumboBFT algorithm and has the ability of horizontal expansion. Some comments to improve this paper are listed as follows:

Paper Strength:
1) The paper addresses an issue in blockchain technology and proposes an algorithm that integrates dynamic sharding strategies to improve consensus in an asynchronous network.
2) The algorithm is well-designed and incorporates multiple components, including a node asynchronous sharding model, an intelligent transaction

placement strategy, and an optimized execution process.
3) The experimental evaluation provides evidence that the DS-Dumbo algorithm outperforms the DumboBFT algorithm in terms of throughput and delay.

Paper Weakness:
1) The paper lacks detailed implementation details, making it difficult to reproduce the study. More information on the implementation of the DS-Dumbo algorithm and the experimental setup would be helpful.

Response:Thank you for your constructive suggestions.This experiment was carried out under the parameter Settings of α= 0.2, β = 0.5 and γ=0.3. The Settings of experimental parameters have been described in the experiment section of Section VI.

2) The evaluation could benefit from more comprehensive comparisons with other existing consensus algorithms and scalability analysis with a larger number of nodes.

Response:Thank you for your constructive suggestions.In the node scalability experiment, the number of nodes is up to 250 nodes. Compared with DumboBFT algorithm, the proposed algorithm has good performance scalability. About a more comprehensive comparison with existing consensus algorithms, as well as experimental analysis when the number of nodes is set more, this part will be in our future work plan.

Suggestions to improve this paper:
1) Could you provide more details on the implementation of the DS-Dumbo algorithm and the experimental setup? This would help readers to better understand and reproduce the study.

Response:Thank you for your constructive suggestions.Thank you for your constructive suggestions.This experiment was carried out under the parameter Settings of α= 0.2, β = 0.5 and γ=0.3. The Settings of experimental parameters have been described in the experiment section of Section VI.

2) Have you considered comparing the DS-Dumbo algorithm with other existing consensus algorithms, such as PBFT and Tendermint? A comparison with widely-known baselines would strengthen the evaluation of the proposed algorithm.

Response:Thank you for your constructive suggestions.In this paper, the algorithm is compared with DumboBFT and OmnliLedger algorithm, and the proposed algorithm has good performance. For a more comprehensive comparison with existing consensus algorithms, this part will be in our future work plan.

3) Have you tested the scalability of the DS-Dumbo algorithm with a larger number of nodes? It would be interesting to see how the algorithm performs in a more realistic and scalable setting.

Response:Thank you for your constructive suggestions.In the node scalability experiment, the number of nodes is up to 250 nodes. Compared with DumboBFT algorithm, the proposed algorithm has good performance scalability. Regarding the scalability of the analysis algorithm with a larger number of nodes, this part will be in our future work plan.